

QEverCloud

3.0.3

Generated by Doxygen 1.8.17



<b>1 QEverCloud</b>	<b>1</b>
1.1 What's this	1
1.1.1 API breaks from 2.x to 3.0	1
1.1.2 C++11/14/17 features	1
1.2 Include files for applications using the library	2
1.3 Related projects	2
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy	5
<b>4 Class Index</b>	<b>9</b>
4.1 Class List	9
<b>5 File Index</b>	<b>11</b>
5.1 File List	11
<b>6 Namespace Documentation</b>	<b>13</b>
6.1 qevercloud Namespace Reference	13
6.1.1 Detailed Description	18
6.1.2 Typedef Documentation	18
6.1.2.1 Guid	18
6.1.2.2 Timestamp	18
6.1.2.3 UserID	19
6.1.3 Function Documentation	19
6.1.3.1 evernoteNetworkAccessManager()	19
6.1.3.2 libraryVersion()	19
6.1.3.3 setNonceGenerator()	19
6.1.4 Variable Documentation	19
6.1.4.1 CLASSIFICATION_RECIPE_SERVICE_RECIPE	20
6.1.4.2 CLASSIFICATION_RECIPE_USER_NON_RECIPE	20
6.1.4.3 CLASSIFICATION_RECIPE_USER_RECIPE	20
6.1.4.4 EDAM_APPLICATIONDATA_ENTRY_LEN_MAX	20
6.1.4.5 EDAM_APPLICATIONDATA_NAME_LEN_MAX	20
6.1.4.6 EDAM_APPLICATIONDATA_NAME_LEN_MIN	20
6.1.4.7 EDAM_APPLICATIONDATA_NAME_REGEX	20
6.1.4.8 EDAM_APPLICATIONDATA_VALUE_LEN_MAX	21
6.1.4.9 EDAM_APPLICATIONDATA_VALUE_LEN_MIN	21
6.1.4.10 EDAM_APPLICATIONDATA_VALUE_REGEX	21
6.1.4.11 EDAM_ATTRIBUTE_LEN_MAX	21
6.1.4.12 EDAM_ATTRIBUTE_LEN_MIN	21
6.1.4.13 EDAM_ATTRIBUTE_LIST_MAX	21

6.1.4.14 EDAM_ATTRIBUTE_MAP_MAX	21
6.1.4.15 EDAM_ATTRIBUTE_REGEX	22
6.1.4.16 EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LEN_MAX	22
6.1.4.17 EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LEN_MIN	22
6.1.4.18 EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_REGEX	22
6.1.4.19 EDAM_BUSINESS_NOTEBOOKS_MAX	22
6.1.4.20 EDAM_BUSINESS_NOTES_MAX	22
6.1.4.21 EDAM_BUSINESS_PHONE_NUMBER_LEN_MAX	22
6.1.4.22 EDAM_BUSINESS_TAGS_MAX	23
6.1.4.23 EDAM_BUSINESS_URI_LEN_MAX	23
6.1.4.24 EDAM_CONTENT_CLASS_FOOD_MEAL	23
6.1.4.25 EDAM_CONTENT_CLASS_HELLO_ENCOUNTER	23
6.1.4.26 EDAM_CONTENT_CLASS_HELLO_PROFILE	23
6.1.4.27 EDAM_CONTENT_CLASS_PENULTIMATE_NOTEBOOK	23
6.1.4.28 EDAM_CONTENT_CLASS_PENULTIMATE_PREFIX	23
6.1.4.29 EDAM_CONTENT_CLASS_SKITCH	24
6.1.4.30 EDAM_CONTENT_CLASS_SKITCH_PDF	24
6.1.4.31 EDAM_CONTENT_CLASS_SKITCH_PREFIX	24
6.1.4.32 EDAM_DEVICE_DESCRIPTION_LEN_MAX	24
6.1.4.33 EDAM_DEVICE_DESCRIPTION_REGEX	24
6.1.4.34 EDAM_DEVICE_ID_LEN_MAX	24
6.1.4.35 EDAM_DEVICE_ID_REGEX	24
6.1.4.36 EDAM_EMAIL_DOMAIN_REGEX	25
6.1.4.37 EDAM_EMAIL_LEN_MAX	25
6.1.4.38 EDAM_EMAIL_LEN_MIN	25
6.1.4.39 EDAM_EMAIL_LOCAL_REGEX	25
6.1.4.40 EDAM_EMAIL_REGEX	25
6.1.4.41 EDAM_FOOD_APP_CONTENT_CLASS_PREFIX	25
6.1.4.42 EDAM_GUID_LEN_MAX	25
6.1.4.43 EDAM_GUID_LEN_MIN	26
6.1.4.44 EDAM_GUID_REGEX	26
6.1.4.45 EDAM_HASH_LEN	26
6.1.4.46 EDAM_HELLO_APP_CONTENT_CLASS_PREFIX	26
6.1.4.47 EDAM_INDEXABLE_RESOURCE_MIME_TYPES	26
6.1.4.48 EDAM_MAX_PREFERENCES	26
6.1.4.49 EDAM_MAX_VALUES_PER_PREFERENCE	26
6.1.4.50 EDAM_MIME_LEN_MAX	27
6.1.4.51 EDAM_MIME_LEN_MIN	27
6.1.4.52 EDAM_MIME_REGEX	27
6.1.4.53 EDAM_MIME_TYPE_AAC	27
6.1.4.54 EDAM_MIME_TYPE_AMR	27
6.1.4.55 EDAM_MIME_TYPE_DEFAULT	27

6.1.4.56 EDAM_MIME_TYPE_GIF . . . . .	27
6.1.4.57 EDAM_MIME_TYPE_INK . . . . .	27
6.1.4.58 EDAM_MIME_TYPE_JPEG . . . . .	28
6.1.4.59 EDAM_MIME_TYPE_M4A . . . . .	28
6.1.4.60 EDAM_MIME_TYPE_MP3 . . . . .	28
6.1.4.61 EDAM_MIME_TYPE_MP4_VIDEO . . . . .	28
6.1.4.62 EDAM_MIME_TYPE_PDF . . . . .	28
6.1.4.63 EDAM_MIME_TYPE_PNG . . . . .	28
6.1.4.64 EDAM_MIME_TYPE_WAV . . . . .	28
6.1.4.65 EDAM_MIME_TYPES . . . . .	28
6.1.4.66 EDAM_NOTE_CONTENT_CLASS_LEN_MAX . . . . .	29
6.1.4.67 EDAM_NOTE_CONTENT_CLASS_LEN_MIN . . . . .	29
6.1.4.68 EDAM_NOTE_CONTENT_CLASS_REGEX . . . . .	29
6.1.4.69 EDAM_NOTE_CONTENT_LEN_MAX . . . . .	29
6.1.4.70 EDAM_NOTE_CONTENT_LEN_MIN . . . . .	29
6.1.4.71 EDAM_NOTE_RESOURCES_MAX . . . . .	29
6.1.4.72 EDAM_NOTE_SIZE_MAX_FREE . . . . .	29
6.1.4.73 EDAM_NOTE_SIZE_MAX_PREMIUM . . . . .	30
6.1.4.74 EDAM_NOTE_SOURCE_MAIL_CLIP . . . . .	30
6.1.4.75 EDAM_NOTE_SOURCE_MAIL_SMTP_GATEWAY . . . . .	30
6.1.4.76 EDAM_NOTE_SOURCE_WEB_CLIP . . . . .	30
6.1.4.77 EDAM_NOTE_TAGS_MAX . . . . .	30
6.1.4.78 EDAM_NOTE_TITLE_LEN_MAX . . . . .	30
6.1.4.79 EDAM_NOTE_TITLE_LEN_MIN . . . . .	30
6.1.4.80 EDAM_NOTE_TITLE_REGEX . . . . .	31
6.1.4.81 EDAM_NOTEBOOK_NAME_LEN_MAX . . . . .	31
6.1.4.82 EDAM_NOTEBOOK_NAME_LEN_MIN . . . . .	31
6.1.4.83 EDAM_NOTEBOOK_NAME_REGEX . . . . .	31
6.1.4.84 EDAM_NOTEBOOK_SHARED_NOTEBOOK_MAX . . . . .	31
6.1.4.85 EDAM_NOTEBOOK_STACK_LEN_MAX . . . . .	31
6.1.4.86 EDAM_NOTEBOOK_STACK_LEN_MIN . . . . .	31
6.1.4.87 EDAM_NOTEBOOK_STACK_REGEX . . . . .	32
6.1.4.88 EDAM_PREFERENCE_NAME_LEN_MAX . . . . .	32
6.1.4.89 EDAM_PREFERENCE_NAME_LEN_MIN . . . . .	32
6.1.4.90 EDAM_PREFERENCE_NAME_REGEX . . . . .	32
6.1.4.91 EDAM_PREFERENCE_SHORTCUTS . . . . .	32
6.1.4.92 EDAM_PREFERENCE_SHORTCUTS_MAX_VALUES . . . . .	32
6.1.4.93 EDAM_PREFERENCE_VALUE_LEN_MAX . . . . .	32
6.1.4.94 EDAM_PREFERENCE_VALUE_LEN_MIN . . . . .	33
6.1.4.95 EDAM_PREFERENCE_VALUE_REGEX . . . . .	33
6.1.4.96 EDAM_PUBLISHING_DESCRIPTION_LEN_MAX . . . . .	33
6.1.4.97 EDAM_PUBLISHING_DESCRIPTION_LEN_MIN . . . . .	33

6.1.4.98 EDAM_PUBLISHING_DESCRIPTION_REGEX . . . . .	33
6.1.4.99 EDAM_PUBLISHING_URI_LEN_MAX . . . . .	33
6.1.4.100 EDAM_PUBLISHING_URI_LEN_MIN . . . . .	33
6.1.4.101 EDAM_PUBLISHING_URI_PROHIBITED . . . . .	34
6.1.4.102 EDAM_PUBLISHING_URI_REGEX . . . . .	34
6.1.4.103 EDAM_RELATED_MAX_NOTEBOOKS . . . . .	34
6.1.4.104 EDAM_RELATED_MAX_NOTES . . . . .	34
6.1.4.105 EDAM_RELATED_MAX_TAGS . . . . .	34
6.1.4.106 EDAM_RELATED_PLAINTEXT_LEN_MAX . . . . .	34
6.1.4.107 EDAM_RELATED_PLAINTEXT_LEN_MIN . . . . .	34
6.1.4.108 EDAM_RESOURCE_SIZE_MAX_FREE . . . . .	34
6.1.4.109 EDAM_RESOURCE_SIZE_MAX_PREMIUM . . . . .	35
6.1.4.110 EDAM_SAVED_SEARCH_NAME_LEN_MAX . . . . .	35
6.1.4.111 EDAM_SAVED_SEARCH_NAME_LEN_MIN . . . . .	35
6.1.4.112 EDAM_SAVED_SEARCH_NAME_REGEX . . . . .	35
6.1.4.113 EDAM_SEARCH_QUERY_LEN_MAX . . . . .	35
6.1.4.114 EDAM_SEARCH_QUERY_LEN_MIN . . . . .	35
6.1.4.115 EDAM_SEARCH_QUERY_REGEX . . . . .	35
6.1.4.116 EDAM_SEARCH_SUGGESTIONS_MAX . . . . .	36
6.1.4.117 EDAM_SEARCH_SUGGESTIONS_PREFIX_LEN_MAX . . . . .	36
6.1.4.118 EDAM_SEARCH_SUGGESTIONS_PREFIX_LEN_MIN . . . . .	36
6.1.4.119 EDAM_TAG_NAME_LEN_MAX . . . . .	36
6.1.4.120 EDAM_TAG_NAME_LEN_MIN . . . . .	36
6.1.4.121 EDAM_TAG_NAME_REGEX . . . . .	36
6.1.4.122 EDAM_TIMEZONE_LEN_MAX . . . . .	36
6.1.4.123 EDAM_TIMEZONE_LEN_MIN . . . . .	37
6.1.4.124 EDAM_TIMEZONE_REGEX . . . . .	37
6.1.4.125 EDAM_USER_LINKED_NOTEBOOK_MAX . . . . .	37
6.1.4.126 EDAM_USER_LINKED_NOTEBOOK_MAX_PREMIUM . . . . .	37
6.1.4.127 EDAM_USER_MAIL_LIMIT_DAILY_FREE . . . . .	37
6.1.4.128 EDAM_USER_MAIL_LIMIT_DAILY_PREMIUM . . . . .	37
6.1.4.129 EDAM_USER_NAME_LEN_MAX . . . . .	38
6.1.4.130 EDAM_USER_NAME_LEN_MIN . . . . .	38
6.1.4.131 EDAM_USER_NAME_REGEX . . . . .	38
6.1.4.132 EDAM_USER_NOTEBOOKS_MAX . . . . .	38
6.1.4.133 EDAM_USER_NOTES_MAX . . . . .	38
6.1.4.134 EDAM_USER_PASSWORD_LEN_MAX . . . . .	38
6.1.4.135 EDAM_USER_PASSWORD_LEN_MIN . . . . .	38
6.1.4.136 EDAM_USER_PASSWORD_REGEX . . . . .	38
6.1.4.137 EDAM_USER_RECENT_MAILED_ADDRESSES_MAX . . . . .	39
6.1.4.138 EDAM_USER_SAVED_SEARCHES_MAX . . . . .	39
6.1.4.139 EDAM_USER_TAGS_MAX . . . . .	39

6.1.4.140 EDAM_USER_UPLOAD_LIMIT_BUSINESS	39
6.1.4.141 EDAM_USER_UPLOAD_LIMIT_FREE	39
6.1.4.142 EDAM_USER_UPLOAD_LIMIT_PREMIUM	39
6.1.4.143 EDAM_USER_USERNAME_LEN_MAX	39
6.1.4.144 EDAM_USER_USERNAME_LEN_MIN	40
6.1.4.145 EDAM_USER_USERNAME_REGEX	40
6.1.4.146 EDAM_VAT_REGEX	40
6.1.4.147 EDAM_VERSION_MAJOR	40
6.1.4.148 EDAM_VERSION_MINOR	40
6.1.4.149 EverCloudExceptionData	40
<b>7 Class Documentation</b>	<b>41</b>
7.1 qevercloud::Accounting Struct Reference	41
7.1.1 Detailed Description	42
7.1.2 Member Function Documentation	42
7.1.2.1 operator!=(())	42
7.1.2.2 operator==(())	42
7.1.3 Member Data Documentation	42
7.1.3.1 businessId	42
7.1.3.2 businessName	42
7.1.3.3 businessRole	42
7.1.3.4 currency	43
7.1.3.5 lastFailedCharge	43
7.1.3.6 lastFailedChargeReason	43
7.1.3.7 lastRequestedCharge	43
7.1.3.8 lastSuccessfulCharge	43
7.1.3.9 nextChargeDate	43
7.1.3.10 nextPaymentDue	43
7.1.3.11 premiumCommerceService	43
7.1.3.12 premiumLockUntil	44
7.1.3.13 premiumOrderNumber	44
7.1.3.14 premiumServiceSKU	44
7.1.3.15 premiumServiceStart	44
7.1.3.16 premiumServiceStatus	44
7.1.3.17 premiumSubscriptionNumber	44
7.1.3.18 unitDiscount	44
7.1.3.19 unitPrice	45
7.1.3.20 updated	45
7.1.3.21 uploadLimit	45
7.1.3.22 uploadLimitEnd	45
7.1.3.23 uploadLimitNextMonth	45
7.2 qevercloud::AsyncResult Class Reference	45

7.2.1 Detailed Description . . . . .	46
7.2.2 Member Typedef Documentation . . . . .	46
7.2.2.1 ReadFunctionType . . . . .	46
7.2.3 Constructor & Destructor Documentation . . . . .	46
7.2.3.1 AsyncResult() [1/2] . . . . .	47
7.2.3.2 AsyncResult() [2/2] . . . . .	47
7.2.3.3 ~AsyncResult() . . . . .	47
7.2.4 Member Function Documentation . . . . .	47
7.2.4.1 finished . . . . .	47
7.2.4.2 waitForFinished() . . . . .	47
7.3 qevercloud::AuthenticationResult Struct Reference . . . . .	48
7.3.1 Detailed Description . . . . .	48
7.3.2 Member Function Documentation . . . . .	48
7.3.2.1 operator!=(()) . . . . .	48
7.3.2.2 operator==(()) . . . . .	49
7.3.3 Member Data Documentation . . . . .	49
7.3.3.1 authenticationToken . . . . .	49
7.3.3.2 currentTime . . . . .	49
7.3.3.3 expiration . . . . .	49
7.3.3.4 noteStoreUrl . . . . .	49
7.3.3.5 publicUserInfo . . . . .	49
7.3.3.6 secondFactorDeliveryHint . . . . .	50
7.3.3.7 secondFactorRequired . . . . .	50
7.3.3.8 user . . . . .	50
7.3.3.9 webApiUrlPrefix . . . . .	50
7.4 qevercloud::BootstrapInfo Struct Reference . . . . .	50
7.4.1 Detailed Description . . . . .	51
7.4.2 Member Function Documentation . . . . .	51
7.4.2.1 operator!=(()) . . . . .	51
7.4.2.2 operator==(()) . . . . .	51
7.4.3 Member Data Documentation . . . . .	51
7.4.3.1 profiles . . . . .	51
7.5 qevercloud::BootstrapProfile Struct Reference . . . . .	51
7.5.1 Detailed Description . . . . .	52
7.5.2 Member Function Documentation . . . . .	52
7.5.2.1 operator!=(()) . . . . .	52
7.5.2.2 operator==(()) . . . . .	52
7.5.3 Member Data Documentation . . . . .	52
7.5.3.1 name . . . . .	52
7.5.3.2 settings . . . . .	52
7.6 qevercloud::BootstrapSettings Struct Reference . . . . .	52
7.6.1 Detailed Description . . . . .	53



7.6.2 Member Function Documentation . . . . .	53
7.6.2.1 operator"!=( ) . . . . .	53
7.6.2.2 operator==( ) . . . . .	53
7.6.3 Member Data Documentation . . . . .	53
7.6.3.1 accountEmailDomain . . . . .	53
7.6.3.2 enableFacebookSharing . . . . .	54
7.6.3.3 enableGiftSubscriptions . . . . .	54
7.6.3.4 enableLinkedInSharing . . . . .	54
7.6.3.5 enablePublicNotebooks . . . . .	54
7.6.3.6 enableSharedNotebooks . . . . .	54
7.6.3.7 enableSingleNoteSharing . . . . .	54
7.6.3.8 enableSponsoredAccounts . . . . .	54
7.6.3.9 enableSupportTickets . . . . .	54
7.6.3.10 enableTwitterSharing . . . . .	55
7.6.3.11 marketingUrl . . . . .	55
7.6.3.12 serviceHost . . . . .	55
7.6.3.13 supportUrl . . . . .	55
7.7 qevercloud::BusinessNotebook Struct Reference . . . . .	55
7.7.1 Detailed Description . . . . .	56
7.7.2 Member Function Documentation . . . . .	56
7.7.2.1 operator"!=( ) . . . . .	56
7.7.2.2 operator==( ) . . . . .	56
7.7.3 Member Data Documentation . . . . .	56
7.7.3.1 notebookDescription . . . . .	56
7.7.3.2 privilege . . . . .	56
7.7.3.3 recommended . . . . .	56
7.8 qevercloud::BusinessUserInfo Struct Reference . . . . .	57
7.8.1 Detailed Description . . . . .	57
7.8.2 Member Function Documentation . . . . .	57
7.8.2.1 operator"!=( ) . . . . .	57
7.8.2.2 operator==( ) . . . . .	57
7.8.3 Member Data Documentation . . . . .	57
7.8.3.1 businessId . . . . .	58
7.8.3.2 businessName . . . . .	58
7.8.3.3 email . . . . .	58
7.8.3.4 role . . . . .	58
7.9 qevercloud::BusinessUserRole Struct Reference . . . . .	58
7.9.1 Detailed Description . . . . .	58
7.9.2 Member Enumeration Documentation . . . . .	58
7.9.2.1 type . . . . .	58
7.10 qevercloud::ClientUsageMetrics Struct Reference . . . . .	59
7.10.1 Detailed Description . . . . .	59

7.10.2 Member Function Documentation	59
7.10.2.1 operator"!=()	59
7.10.2.2 operator==(())	59
7.10.3 Member Data Documentation	59
7.10.3.1 sessions	60
7.11 qevercloud::Data Struct Reference	60
7.11.1 Detailed Description	60
7.11.2 Member Function Documentation	60
7.11.2.1 operator"!=()	60
7.11.2.2 operator==(())	61
7.11.3 Member Data Documentation	61
7.11.3.1 body	61
7.11.3.2 bodyHash	61
7.11.3.3 size	61
7.12 qevercloud::EDAMErrorCode Struct Reference	61
7.12.1 Detailed Description	62
7.12.2 Member Enumeration Documentation	62
7.12.2.1 type	63
7.13 qevercloud::EDAMNotFoundException Class Reference	63
7.13.1 Detailed Description	64
7.13.2 Constructor & Destructor Documentation	64
7.13.2.1 EDAMNotFoundException()	64
7.13.2.2 ~EDAMNotFoundException()	64
7.13.3 Member Function Documentation	64
7.13.3.1 exceptionData()	64
7.13.3.2 what()	65
7.13.4 Member Data Documentation	65
7.13.4.1 identifier	65
7.13.4.2 key	65
7.14 qevercloud::EDAMNotFoundExceptionData Class Reference	65
7.14.1 Detailed Description	66
7.14.2 Constructor & Destructor Documentation	66
7.14.2.1 EDAMNotFoundExceptionData()	66
7.14.3 Member Function Documentation	66
7.14.3.1 throwException()	66
7.14.4 Member Data Documentation	66
7.14.4.1 m_identifier	66
7.14.4.2 m_key	66
7.15 qevercloud::EDAMSystemException Class Reference	67
7.15.1 Detailed Description	67
7.15.2 Constructor & Destructor Documentation	67
7.15.2.1 EDAMSystemException()	68

7.15.2.2 ~EDAMSystemException()	68
7.15.3 Member Function Documentation	68
7.15.3.1 exceptionData()	68
7.15.3.2 what()	68
7.15.4 Member Data Documentation	68
7.15.4.1 errorCode	68
7.15.4.2 message	68
7.15.4.3 rateLimitDuration	69
7.16 qevercloud::EDAMSystemExceptionAuthExpired Class Reference	69
7.16.1 Detailed Description	69
7.16.2 Member Function Documentation	69
7.16.2.1 exceptionData()	69
7.17 qevercloud::EDAMSystemExceptionAuthExpiredData Class Reference	70
7.17.1 Detailed Description	70
7.17.2 Constructor & Destructor Documentation	70
7.17.2.1 EDAMSystemExceptionAuthExpiredData()	70
7.17.3 Member Function Documentation	70
7.17.3.1 throwException()	71
7.18 qevercloud::EDAMSystemExceptionData Class Reference	71
7.18.1 Detailed Description	71
7.18.2 Constructor & Destructor Documentation	71
7.18.2.1 EDAMSystemExceptionData()	72
7.18.3 Member Function Documentation	72
7.18.3.1 throwException()	72
7.18.4 Member Data Documentation	72
7.18.4.1 m_errorCode	72
7.18.4.2 m_message	72
7.18.4.3 m_rateLimitDuration	72
7.19 qevercloud::EDAMSystemExceptionRateLimitReached Class Reference	73
7.19.1 Detailed Description	73
7.19.2 Member Function Documentation	73
7.19.2.1 exceptionData()	73
7.20 qevercloud::EDAMSystemExceptionRateLimitReachedData Class Reference	74
7.20.1 Detailed Description	74
7.20.2 Constructor & Destructor Documentation	74
7.20.2.1 EDAMSystemExceptionRateLimitReachedData()	74
7.20.3 Member Function Documentation	74
7.20.3.1 throwException()	75
7.21 qevercloud::EDAMUserException Class Reference	75
7.21.1 Detailed Description	75
7.21.2 Constructor & Destructor Documentation	76
7.21.2.1 EDAMUserException()	76

7.21.2.2 ~EDAMUserException()	76
7.21.3 Member Function Documentation	76
7.21.3.1 exceptionData()	76
7.21.3.2 what()	76
7.21.4 Member Data Documentation	76
7.21.4.1 errorCode	76
7.21.4.2 parameter	76
7.22 qevercloud::EDAMUserExceptionData Class Reference	77
7.22.1 Detailed Description	77
7.22.2 Constructor & Destructor Documentation	77
7.22.2.1 EDAMUserExceptionData()	77
7.22.3 Member Function Documentation	77
7.22.3.1 throwException()	78
7.22.4 Member Data Documentation	78
7.22.4.1 m_errorCode	78
7.22.4.2 m_parameter	78
7.23 qevercloud::EventLoopFinisher Class Reference	78
7.23.1 Constructor & Destructor Documentation	79
7.23.1.1 EventLoopFinisher()	79
7.23.1.2 ~EventLoopFinisher()	79
7.23.2 Member Function Documentation	79
7.23.2.1 stopEventLoop	79
7.24 qevercloud::EverCloudException Class Reference	79
7.24.1 Detailed Description	80
7.24.2 Constructor & Destructor Documentation	80
7.24.2.1 EverCloudException() [1/4]	80
7.24.2.2 EverCloudException() [2/4]	80
7.24.2.3 EverCloudException() [3/4]	80
7.24.2.4 EverCloudException() [4/4]	80
7.24.2.5 ~EverCloudException()	80
7.24.3 Member Function Documentation	81
7.24.3.1 exceptionData()	81
7.24.3.2 what()	81
7.24.4 Member Data Documentation	81
7.24.4.1 m_error	81
7.25 qevercloud::EverCloudExceptionData Class Reference	81
7.25.1 Detailed Description	82
7.25.2 Constructor & Destructor Documentation	82
7.25.2.1 EverCloudExceptionData()	82
7.25.3 Member Function Documentation	82
7.25.3.1 throwException()	83
7.25.4 Member Data Documentation	83

7.25.4.1 errorMessage	83
7.26 qevercloud::EvernoteException Class Reference	83
7.26.1 Detailed Description	83
7.26.2 Constructor & Destructor Documentation	84
7.26.2.1 EvernoteException() [1/4]	84
7.26.2.2 EvernoteException() [2/4]	84
7.26.2.3 EvernoteException() [3/4]	84
7.26.2.4 EvernoteException() [4/4]	84
7.26.3 Member Function Documentation	84
7.26.3.1 exceptionData()	84
7.27 qevercloud::EvernoteExceptionData Class Reference	85
7.27.1 Detailed Description	85
7.27.2 Constructor & Destructor Documentation	85
7.27.2.1 EvernoteExceptionData()	85
7.27.3 Member Function Documentation	85
7.27.3.1 throwException()	85
7.28 qevercloud::EvernoteOAuthDialog Class Reference	86
7.28.1 Detailed Description	86
7.28.2 Member Typedef Documentation	86
7.28.2.1 OAuthResult	87
7.28.3 Constructor & Destructor Documentation	87
7.28.3.1 EvernoteOAuthDialog()	87
7.28.3.2 ~EvernoteOAuthDialog()	87
7.28.4 Member Function Documentation	87
7.28.4.1 exec()	87
7.28.4.2 isSucceeded()	88
7.28.4.3 oauthError()	88
7.28.4.4 oauthResult()	88
7.28.4.5 open()	88
7.28.4.6 setWebViewSizeHint()	88
7.29 qevercloud::EvernoteOAuthWebView Class Reference	89
7.29.1 Detailed Description	89
7.29.2 Constructor & Destructor Documentation	89
7.29.2.1 EvernoteOAuthWebView()	90
7.29.3 Member Function Documentation	90
7.29.3.1 authenticate()	90
7.29.3.2 authenticationFailed	90
7.29.3.3 authenticationFinished	90
7.29.3.4 authenticationSucceeded	91
7.29.3.5 isSucceeded()	91
7.29.3.6 oauthError()	91
7.29.3.7 oauthResult()	91

7.29.3.8 setSizeHint()	91
7.29.3.9 sizeHint()	91
7.30 qevercloud::Thumbnail::ImageType Struct Reference	92
7.30.1 Detailed Description	92
7.30.2 Member Enumeration Documentation	92
7.30.2.1 type	92
7.31 qevercloud::LazyMap Struct Reference	92
7.31.1 Detailed Description	93
7.31.2 Member Function Documentation	93
7.31.2.1 operator!=(())	93
7.31.2.2 operator==(())	93
7.31.3 Member Data Documentation	93
7.31.3.1 fullMap	93
7.31.3.2 keysOnly	93
7.32 qevercloud::LinkedNotebook Struct Reference	94
7.32.1 Detailed Description	94
7.32.2 Member Function Documentation	94
7.32.2.1 operator!=(())	94
7.32.2.2 operator==(())	94
7.32.3 Member Data Documentation	94
7.32.3.1 businessId	95
7.32.3.2 guid	95
7.32.3.3 noteStoreUrl	95
7.32.3.4 shardId	95
7.32.3.5 shareKey	95
7.32.3.6 shareName	95
7.32.3.7 stack	96
7.32.3.8 updateSequenceNum	96
7.32.3.9 uri	96
7.32.3.10 username	96
7.32.3.11 webApiUrlPrefix	96
7.33 qevercloud::Note Struct Reference	96
7.33.1 Detailed Description	97
7.33.2 Member Function Documentation	97
7.33.2.1 operator!=(())	97
7.33.2.2 operator==(())	97
7.33.3 Member Data Documentation	97
7.33.3.1 active	97
7.33.3.2 attributes	98
7.33.3.3 content	98
7.33.3.4 contentHash	98
7.33.3.5 contentLength	98

7.33.3.6 created	98
7.33.3.7 deleted	98
7.33.3.8 guid	99
7.33.3.9 notebookGuid	99
7.33.3.10 resources	99
7.33.3.11 tagGuids	99
7.33.3.12 tagNames	99
7.33.3.13 title	99
7.33.3.14 updated	100
7.33.3.15 updateSequenceNum	100
7.34 qevercloud::NoteAttributes Struct Reference	100
7.34.1 Detailed Description	101
7.34.2 Member Function Documentation	101
7.34.2.1 operator!=(())	101
7.34.2.2 operator==(())	101
7.34.3 Member Data Documentation	101
7.34.3.1 altitude	101
7.34.3.2 applicationData	101
7.34.3.3 author	102
7.34.3.4 classifications	102
7.34.3.5 contentClass	102
7.34.3.6 creatorId	102
7.34.3.7 lastEditedBy	102
7.34.3.8 lastEditorId	103
7.34.3.9 latitude	103
7.34.3.10 longitude	103
7.34.3.11 placeName	103
7.34.3.12 reminderDoneTime	103
7.34.3.13 reminderOrder	103
7.34.3.14 reminderTime	104
7.34.3.15 shareDate	104
7.34.3.16 source	104
7.34.3.17 sourceApplication	104
7.34.3.18 sourceURL	104
7.34.3.19 subjectDate	104
7.35 qevercloud::Notebook Struct Reference	105
7.35.1 Detailed Description	105
7.35.2 Member Function Documentation	105
7.35.2.1 operator!=(())	105
7.35.2.2 operator==(())	105
7.35.3 Member Data Documentation	106
7.35.3.1 businessNotebook	106

7.35.3.2 contact	106
7.35.3.3 defaultNotebook	106
7.35.3.4 guid	106
7.35.3.5 name	106
7.35.3.6 published	107
7.35.3.7 publishing	107
7.35.3.8 restrictions	107
7.35.3.9 serviceCreated	107
7.35.3.10 serviceUpdated	107
7.35.3.11 sharedNotebookIds	107
7.35.3.12 sharedNotebooks	108
7.35.3.13 stack	108
7.35.3.14 updateSequenceNum	108
7.36 qevercloud::NotebookDescriptor Struct Reference	108
7.36.1 Detailed Description	108
7.36.2 Member Function Documentation	109
7.36.2.1 operator"!=()	109
7.36.2.2 operator==(())	109
7.36.3 Member Data Documentation	109
7.36.3.1 contactName	109
7.36.3.2 guid	109
7.36.3.3 hasSharedNotebook	109
7.36.3.4 joinedUserCount	109
7.36.3.5 notebookDisplayName	110
7.37 qevercloud::NotebookRestrictions Struct Reference	110
7.37.1 Detailed Description	110
7.37.2 Member Function Documentation	111
7.37.2.1 operator"!=()	111
7.37.2.2 operator==(())	111
7.37.3 Member Data Documentation	111
7.37.3.1 expungeWhichSharedNotebookRestrictions	111
7.37.3.2 noCreateNotes	111
7.37.3.3 noCreateSharedNotebooks	111
7.37.3.4 noCreateTags	111
7.37.3.5 noEmailNotes	112
7.37.3.6 noExpungeNotebook	112
7.37.3.7 noExpungeNotes	112
7.37.3.8 noExpungeTags	112
7.37.3.9 noPublishToBusinessLibrary	112
7.37.3.10 noPublishToPublic	112
7.37.3.11 noReadNotes	112
7.37.3.12 noSendMessageToRecipients	113



7.37.3.13 noSetDefaultNotebook	113
7.37.3.14 noSetNotebookStack	113
7.37.3.15 noSetParentTag	113
7.37.3.16 noShareNotes	113
7.37.3.17 noUpdateNotebook	113
7.37.3.18 noUpdateNotes	113
7.37.3.19 noUpdateTags	114
7.37.3.20 updateWhichSharedNotebookRestrictions	114
7.38 qevercloud::NoteCollectionCounts Struct Reference	114
7.38.1 Detailed Description	114
7.38.2 Member Function Documentation	114
7.38.2.1 operator"!=()	114
7.38.2.2 operator=="()	115
7.38.3 Member Data Documentation	115
7.38.3.1 notebookCounts	115
7.38.3.2 tagCounts	115
7.38.3.3 trashCount	115
7.39 qevercloud::NoteEmailParameters Struct Reference	115
7.39.1 Detailed Description	116
7.39.2 Member Function Documentation	116
7.39.2.1 operator"!=()	116
7.39.2.2 operator=="()	116
7.39.3 Member Data Documentation	116
7.39.3.1 ccAddresses	116
7.39.3.2 guid	116
7.39.3.3 message	116
7.39.3.4 note	117
7.39.3.5 subject	117
7.39.3.6 toAddresses	117
7.40 qevercloud::NoteFilter Struct Reference	117
7.40.1 Detailed Description	117
7.40.2 Member Function Documentation	118
7.40.2.1 operator"!=()	118
7.40.2.2 operator=="()	118
7.40.3 Member Data Documentation	118
7.40.3.1 ascending	118
7.40.3.2 emphasized	118
7.40.3.3 inactive	118
7.40.3.4 notebookGuid	118
7.40.3.5 order	119
7.40.3.6 tagGuids	119
7.40.3.7 timeZone	119

7.40.3.8 words	119
7.41 qevercloud::NoteList Struct Reference	119
7.41.1 Detailed Description	120
7.41.2 Member Function Documentation	120
7.41.2.1 operator"!=()	120
7.41.2.2 operator==(())	120
7.41.3 Member Data Documentation	120
7.41.3.1 notes	120
7.41.3.2 searchedWords	120
7.41.3.3 startIndex	120
7.41.3.4 stoppedWords	121
7.41.3.5 totalNotes	121
7.41.3.6 updateCount	121
7.42 qevercloud::NoteMetadata Struct Reference	121
7.42.1 Detailed Description	122
7.42.2 Member Function Documentation	122
7.42.2.1 operator"!=()	122
7.42.2.2 operator==(())	122
7.42.3 Member Data Documentation	122
7.42.3.1 attributes	122
7.42.3.2 contentLength	122
7.42.3.3 created	122
7.42.3.4 deleted	123
7.42.3.5 guid	123
7.42.3.6 largestResourceMime	123
7.42.3.7 largestResourceSize	123
7.42.3.8 notebookGuid	123
7.42.3.9 tagGuids	123
7.42.3.10 title	123
7.42.3.11 updated	124
7.42.3.12 updateSequenceNum	124
7.43 qevercloud::NotesMetadataList Struct Reference	124
7.43.1 Detailed Description	124
7.43.2 Member Function Documentation	124
7.43.2.1 operator"!=()	124
7.43.2.2 operator==(())	125
7.43.3 Member Data Documentation	125
7.43.3.1 notes	125
7.43.3.2 searchedWords	125
7.43.3.3 startIndex	125
7.43.3.4 stoppedWords	125
7.43.3.5 totalNotes	125

7.43.3.6 updateCount . . . . .	126
7.44 qevercloud::NotesMetadataResultSpec Struct Reference . . . . .	126
7.44.1 Detailed Description . . . . .	126
7.44.2 Member Function Documentation . . . . .	126
7.44.2.1 operator!=(()) . . . . .	127
7.44.2.2 operator==(()) . . . . .	127
7.44.3 Member Data Documentation . . . . .	127
7.44.3.1 includeAttributes . . . . .	127
7.44.3.2 includeContentLength . . . . .	127
7.44.3.3 includeCreated . . . . .	127
7.44.3.4 includeDeleted . . . . .	127
7.44.3.5 includeLargestResourceMime . . . . .	127
7.44.3.6 includeLargestResourceSize . . . . .	128
7.44.3.7 includeNotebookGuid . . . . .	128
7.44.3.8 includeTagGuids . . . . .	128
7.44.3.9 includeTitle . . . . .	128
7.44.3.10 includeUpdated . . . . .	128
7.44.3.11 includeUpdateSequenceNum . . . . .	128
7.45 qevercloud::NoteSortOrder Struct Reference . . . . .	128
7.45.1 Detailed Description . . . . .	129
7.45.2 Member Enumeration Documentation . . . . .	129
7.45.2.1 type . . . . .	129
7.46 qevercloud::NoteStore Class Reference . . . . .	129
7.46.1 Detailed Description . . . . .	133
7.46.2 Constructor & Destructor Documentation . . . . .	134
7.46.2.1 NoteStore() [1/2] . . . . .	134
7.46.2.2 NoteStore() [2/2] . . . . .	134
7.46.3 Member Function Documentation . . . . .	134
7.46.3.1 authenticateToSharedNote() . . . . .	134
7.46.3.2 authenticateToSharedNoteAsync() . . . . .	135
7.46.3.3 authenticateToSharedNotebook() . . . . .	135
7.46.3.4 authenticateToSharedNotebookAsync() . . . . .	136
7.46.3.5 authenticationToken() . . . . .	136
7.46.3.6 copyNote() . . . . .	136
7.46.3.7 copyNoteAsync() . . . . .	137
7.46.3.8 createLinkedNotebook() . . . . .	137
7.46.3.9 createLinkedNotebookAsync() . . . . .	138
7.46.3.10 createNote() . . . . .	138
7.46.3.11 createNoteAsync() . . . . .	139
7.46.3.12 createNotebook() . . . . .	140
7.46.3.13 createNotebookAsync() . . . . .	141
7.46.3.14 createSearch() . . . . .	141

7.46.3.15 createSearchAsync()	142
7.46.3.16 createSharedNotebook()	142
7.46.3.17 createSharedNotebookAsync()	143
7.46.3.18 createTag()	143
7.46.3.19 createTagAsync()	144
7.46.3.20 deleteNote()	144
7.46.3.21 deleteNoteAsync()	145
7.46.3.22 emailNote()	145
7.46.3.23 emailNoteAsync()	146
7.46.3.24 expungeInactiveNotes()	146
7.46.3.25 expungeInactiveNotesAsync()	147
7.46.3.26 expungeLinkedNotebook()	147
7.46.3.27 expungeLinkedNotebookAsync()	147
7.46.3.28 expungeNote()	147
7.46.3.29 expungeNoteAsync()	148
7.46.3.30 expungeNotebook()	148
7.46.3.31 expungeNotebookAsync()	149
7.46.3.32 expungeNotes()	149
7.46.3.33 expungeNotesAsync()	150
7.46.3.34 expungeSearch()	150
7.46.3.35 expungeSearchAsync()	150
7.46.3.36 expungeSharedNotebooks()	151
7.46.3.37 expungeSharedNotebooksAsync()	151
7.46.3.38 expungeTag()	151
7.46.3.39 expungeTagAsync()	152
7.46.3.40 findNoteCounts()	152
7.46.3.41 findNoteCountsAsync()	153
7.46.3.42 findNoteOffset()	153
7.46.3.43 findNoteOffsetAsync()	154
7.46.3.44 findNotes()	154
7.46.3.45 findNotesAsync()	154
7.46.3.46 findNotesMetadata()	155
7.46.3.47 findNotesMetadataAsync()	156
7.46.3.48 findRelated()	156
7.46.3.49 findRelatedAsync()	157
7.46.3.50 getDefaultNotebook()	157
7.46.3.51 getDefaultNotebookAsync()	157
7.46.3.52 getFilteredSyncChunk()	157
7.46.3.53 getFilteredSyncChunkAsync()	158
7.46.3.54 getLinkedNotebookSyncChunk()	158
7.46.3.55 getLinkedNotebookSyncChunkAsync()	159
7.46.3.56 getLinkedNotebookSyncState()	159

---

7.46.3.57 getLinkedNotebookSyncStateAsync()	160
7.46.3.58 getNote()	160
7.46.3.59 getNoteApplicationData()	160
7.46.3.60 getNoteApplicationDataAsync()	161
7.46.3.61 getNoteApplicationDataEntry()	161
7.46.3.62 getNoteApplicationDataEntryAsync()	161
7.46.3.63 getNoteAsync()	161
7.46.3.64 getNotebook()	162
7.46.3.65 getNotebookAsync()	162
7.46.3.66 getNoteContent()	162
7.46.3.67 getNoteContentAsync()	163
7.46.3.68 getNoteSearchText()	163
7.46.3.69 getNoteSearchTextAsync()	164
7.46.3.70 getNoteTagNames()	164
7.46.3.71 getNoteTagNamesAsync()	164
7.46.3.72 getNoteVersion()	165
7.46.3.73 getNoteVersionAsync()	165
7.46.3.74 getPublicNotebook()	166
7.46.3.75 getPublicNotebookAsync()	166
7.46.3.76 getResource()	166
7.46.3.77 getResourceAlternateData()	167
7.46.3.78 getResourceAlternateDataAsync()	168
7.46.3.79 getResourceApplicationData()	168
7.46.3.80 getResourceApplicationDataAsync()	168
7.46.3.81 getResourceApplicationDataEntry()	168
7.46.3.82 getResourceApplicationDataEntryAsync()	168
7.46.3.83 getResourceAsync()	169
7.46.3.84 getResourceAttributes()	169
7.46.3.85 getResourceAttributesAsync()	169
7.46.3.86 getResourceByHash()	170
7.46.3.87 getResourceByHashAsync()	170
7.46.3.88 getResourceData()	171
7.46.3.89 getResourceDataAsync()	171
7.46.3.90 getResourceRecognition()	171
7.46.3.91 getResourceRecognitionAsync()	172
7.46.3.92 getResourceSearchText()	172
7.46.3.93 getResourceSearchTextAsync()	173
7.46.3.94 getSearch()	173
7.46.3.95 getSearchAsync()	173
7.46.3.96 getSharedNotebookByAuth()	173
7.46.3.97 getSharedNotebookByAuthAsync()	174
7.46.3.98 getSyncChunk()	174

7.46.3.99	<a href="#">getSyncChunkAsync()</a>	174
7.46.3.100	<a href="#">getSyncState()</a>	174
7.46.3.101	<a href="#">getSyncStateAsync()</a>	175
7.46.3.102	<a href="#">getSyncStateWithMetrics()</a>	175
7.46.3.103	<a href="#">getSyncStateWithMetricsAsync()</a>	175
7.46.3.104	<a href="#">getTag()</a>	175
7.46.3.105	<a href="#">getTagAsync()</a>	176
7.46.3.106	<a href="#">listLinkedNotebooks()</a>	176
7.46.3.107	<a href="#">listLinkedNotebooksAsync()</a>	176
7.46.3.108	<a href="#">listNotebooks()</a>	176
7.46.3.109	<a href="#">listNotebooksAsync()</a>	176
7.46.3.110	<a href="#">listNoteVersions()</a>	177
7.46.3.111	<a href="#">listNoteVersionsAsync()</a>	178
7.46.3.112	<a href="#">listSearches()</a>	178
7.46.3.113	<a href="#">listSearchesAsync()</a>	178
7.46.3.114	<a href="#">listSharedNotebooks()</a>	178
7.46.3.115	<a href="#">listSharedNotebooksAsync()</a>	179
7.46.3.116	<a href="#">listTags()</a>	179
7.46.3.117	<a href="#">listTagsAsync()</a>	179
7.46.3.118	<a href="#">listTagsByNotebook()</a>	179
7.46.3.119	<a href="#">listTagsByNotebookAsync()</a>	179
7.46.3.120	<a href="#">noteStoreUrl()</a>	180
7.46.3.121	<a href="#">sendMessageToSharedNotebookMembers()</a>	180
7.46.3.122	<a href="#">sendMessageToSharedNotebookMembersAsync()</a>	180
7.46.3.123	<a href="#">setAuthenticationToken()</a>	181
7.46.3.124	<a href="#">setNoteApplicationDataEntry()</a>	181
7.46.3.125	<a href="#">setNoteApplicationDataEntryAsync()</a>	181
7.46.3.126	<a href="#">setNoteStoreUrl()</a>	181
7.46.3.127	<a href="#">setResourceApplicationDataEntry()</a>	181
7.46.3.128	<a href="#">setResourceApplicationDataEntryAsync()</a>	182
7.46.3.129	<a href="#">setSharedNotebookRecipientSettings()</a>	182
7.46.3.130	<a href="#">setSharedNotebookRecipientSettingsAsync()</a>	182
7.46.3.131	<a href="#">shareNote()</a>	183
7.46.3.132	<a href="#">shareNoteAsync()</a>	183
7.46.3.133	<a href="#">stopSharingNote()</a>	183
7.46.3.134	<a href="#">stopSharingNoteAsync()</a>	184
7.46.3.135	<a href="#">unsetNoteApplicationDataEntry()</a>	184
7.46.3.136	<a href="#">unsetNoteApplicationDataEntryAsync()</a>	184
7.46.3.137	<a href="#">unsetResourceApplicationDataEntry()</a>	184
7.46.3.138	<a href="#">unsetResourceApplicationDataEntryAsync()</a>	185
7.46.3.139	<a href="#">untagAll()</a>	185
7.46.3.140	<a href="#">untagAllAsync()</a>	185

7.46.3.141 updateLinkedNotebook()	185
7.46.3.142 updateLinkedNotebookAsync()	186
7.46.3.143 updateNote()	186
7.46.3.144 updateNoteAsync()	187
7.46.3.145 updateNotebook()	188
7.46.3.146 updateNotebookAsync()	188
7.46.3.147 updateResource()	188
7.46.3.148 updateResourceAsync()	189
7.46.3.149 updateSearch()	189
7.46.3.150 updateSearchAsync()	190
7.46.3.151 updateSharedNotebook()	190
7.46.3.152 updateSharedNotebookAsync()	191
7.46.3.153 updateTag()	191
7.46.3.154 updateTagAsync()	192
7.47 qevercloud::NoteVersionId Struct Reference	192
7.47.1 Detailed Description	192
7.47.2 Member Function Documentation	193
7.47.2.1 operator!=(())	193
7.47.2.2 operator==(())	193
7.47.3 Member Data Documentation	193
7.47.3.1 saved	193
7.47.3.2 title	193
7.47.3.3 updated	193
7.47.3.4 updateSequenceNum	193
7.48 qevercloud::EvernoteOAuthWebView::OAuthResult Struct Reference	194
7.48.1 Detailed Description	194
7.48.2 Member Data Documentation	194
7.48.2.1 authenticationToken	194
7.48.2.2 expires	194
7.48.2.3 noteStoreUrl	195
7.48.2.4 shardId	195
7.48.2.5 userId	195
7.48.2.6 webApiUrlPrefix	195
7.49 qevercloud::Optional< T > Class Template Reference	195
7.49.1 Detailed Description	196
7.49.2 Constructor & Destructor Documentation	197
7.49.2.1 Optional() [1/5]	197
7.49.2.2 Optional() [2/5]	197
7.49.2.3 Optional() [3/5]	197
7.49.2.4 Optional() [4/5]	197
7.49.2.5 Optional() [5/5]	197
7.49.3 Member Function Documentation	197

7.49.3.1 clear()	198
7.49.3.2 init()	198
7.49.3.3 isEqual()	198
7.49.3.4 isSet()	198
7.49.3.5 operator const T &()	199
7.49.3.6 operator T&()	199
7.49.3.7 operator->() [1/2]	199
7.49.3.8 operator->() [2/2]	199
7.49.3.9 operator=() [1/4]	200
7.49.3.10 operator=() [2/4]	200
7.49.3.11 operator=() [3/4]	200
7.49.3.12 operator=() [4/4]	200
7.49.3.13 ref() [1/2]	200
7.49.3.14 ref() [2/2]	201
7.49.3.15 value()	201
7.49.4 Friends And Related Function Documentation	201
7.49.4.1 Optional	201
7.49.4.2 swap	201
7.50 qevercloud::PremiumInfo Struct Reference	201
7.50.1 Detailed Description	202
7.50.2 Member Function Documentation	202
7.50.2.1 operator"!=( )	202
7.50.2.2 operator==( )	202
7.50.3 Member Data Documentation	202
7.50.3.1 canPurchaseUploadAllowance	203
7.50.3.2 currentTime	203
7.50.3.3 premium	203
7.50.3.4 premiumCancellationPending	203
7.50.3.5 premiumExpirationDate	203
7.50.3.6 premiumExtendable	203
7.50.3.7 premiumPending	203
7.50.3.8 premiumRecurring	204
7.50.3.9 premiumUpgradable	204
7.50.3.10 sponsoredGroupName	204
7.50.3.11 sponsoredGroupRole	204
7.51 qevercloud::PremiumOrderStatus Struct Reference	204
7.51.1 Detailed Description	204
7.51.2 Member Enumeration Documentation	205
7.51.2.1 type	205
7.52 qevercloud::PrivilegeLevel Struct Reference	206
7.52.1 Detailed Description	206
7.52.2 Member Enumeration Documentation	206



7.52.2.1 type	206
7.53 qevercloud::PublicUserInfo Struct Reference	207
7.53.1 Detailed Description	207
7.53.2 Member Function Documentation	207
7.53.2.1 operator!=(())	207
7.53.2.2 operator==(())	207
7.53.3 Member Data Documentation	207
7.53.3.1 noteStoreUrl	208
7.53.3.2 privilege	208
7.53.3.3 shardId	208
7.53.3.4 userId	208
7.53.3.5 username	208
7.53.3.6 webApiUrlPrefix	208
7.54 qevercloud::Publishing Struct Reference	208
7.54.1 Detailed Description	209
7.54.2 Member Function Documentation	209
7.54.2.1 operator!=(())	209
7.54.2.2 operator==(())	209
7.54.3 Member Data Documentation	209
7.54.3.1 ascending	209
7.54.3.2 order	210
7.54.3.3 publicDescription	210
7.54.3.4 uri	210
7.55 qevercloud::QueryFormat Struct Reference	210
7.55.1 Detailed Description	210
7.55.2 Member Enumeration Documentation	210
7.55.2.1 type	210
7.56 qevercloud::RelatedQuery Struct Reference	211
7.56.1 Detailed Description	211
7.56.2 Member Function Documentation	211
7.56.2.1 operator!=(())	211
7.56.2.2 operator==(())	211
7.56.3 Member Data Documentation	212
7.56.3.1 filter	212
7.56.3.2 noteGuid	212
7.56.3.3 plainText	212
7.56.3.4 referenceUri	212
7.57 qevercloud::RelatedResult Struct Reference	212
7.57.1 Detailed Description	213
7.57.2 Member Function Documentation	213
7.57.2.1 operator!=(())	213
7.57.2.2 operator==(())	213

7.57.3 Member Data Documentation	213
7.57.3.1 containingNotebooks	213
7.57.3.2 notebooks	213
7.57.3.3 notes	213
7.57.3.4 tags	214
7.58 qevercloud::RelatedResultSpec Struct Reference	214
7.58.1 Detailed Description	214
7.58.2 Member Function Documentation	214
7.58.2.1 operator"!=( )	214
7.58.2.2 operator==( )	214
7.58.3 Member Data Documentation	215
7.58.3.1 includeContainingNotebooks	215
7.58.3.2 maxNotebooks	215
7.58.3.3 maxNotes	215
7.58.3.4 maxTags	215
7.58.3.5 writableNotebooksOnly	215
7.59 qevercloud::ReminderEmailConfig Struct Reference	215
7.59.1 Detailed Description	216
7.59.2 Member Enumeration Documentation	216
7.59.2.1 type	216
7.60 qevercloud::Resource Struct Reference	216
7.60.1 Detailed Description	217
7.60.2 Member Function Documentation	217
7.60.2.1 operator"!=( )	217
7.60.2.2 operator==( )	217
7.60.3 Member Data Documentation	217
7.60.3.1 active	217
7.60.3.2 alternateData	217
7.60.3.3 attributes	218
7.60.3.4 data	218
7.60.3.5 duration	218
7.60.3.6 guid	218
7.60.3.7 height	218
7.60.3.8 mime	218
7.60.3.9 noteGuid	218
7.60.3.10 recognition	219
7.60.3.11 updateSequenceNum	219
7.60.3.12 width	219
7.61 qevercloud::ResourceAttributes Struct Reference	219
7.61.1 Detailed Description	219
7.61.2 Member Function Documentation	220
7.61.2.1 operator"!=( )	220

7.61.2.2 operator==( )	220
7.61.3 Member Data Documentation	220
7.61.3.1 altitude	220
7.61.3.2 applicationData	220
7.61.3.3 attachment	220
7.61.3.4 cameraMake	221
7.61.3.5 cameraModel	221
7.61.3.6 clientWillIndex	221
7.61.3.7 fileName	221
7.61.3.8 latitude	221
7.61.3.9 longitude	221
7.61.3.10 recoType	221
7.61.3.11 sourceURL	222
7.61.3.12 timestamp	222
7.62 qevercloud::SavedSearch Struct Reference	222
7.62.1 Detailed Description	222
7.62.2 Member Function Documentation	222
7.62.2.1 operator!=( )	222
7.62.2.2 operator==( )	223
7.62.3 Member Data Documentation	223
7.62.3.1 format	223
7.62.3.2 guid	223
7.62.3.3 name	223
7.62.3.4 query	223
7.62.3.5 scope	223
7.62.3.6 updateSequenceNum	224
7.63 qevercloud::SavedSearchScope Struct Reference	224
7.63.1 Detailed Description	224
7.63.2 Member Function Documentation	224
7.63.2.1 operator!=( )	224
7.63.2.2 operator==( )	224
7.63.3 Member Data Documentation	225
7.63.3.1 includeAccount	225
7.63.3.2 includeBusinessLinkedNotebooks	225
7.63.3.3 includePersonalLinkedNotebooks	225
7.64 qevercloud::SharedNotebook Struct Reference	225
7.64.1 Detailed Description	226
7.64.2 Member Function Documentation	226
7.64.2.1 operator!=( )	226
7.64.2.2 operator==( )	226
7.64.3 Member Data Documentation	226
7.64.3.1 allowPreview	226

7.64.3.2 email	226
7.64.3.3 id	226
7.64.3.4 notebookGuid	227
7.64.3.5 notebookModifiable	227
7.64.3.6 privilege	227
7.64.3.7 recipientSettings	227
7.64.3.8 requireLogin	227
7.64.3.9 serviceCreated	227
7.64.3.10 serviceUpdated	227
7.64.3.11 shareKey	228
7.64.3.12 userId	228
7.64.3.13 username	228
7.65 qevercloud::SharedNotebookInstanceRestrictions Struct Reference	228
7.65.1 Detailed Description	228
7.65.2 Member Enumeration Documentation	228
7.65.2.1 type	228
7.66 qevercloud::SharedNotebookPrivilegeLevel Struct Reference	229
7.66.1 Detailed Description	229
7.66.2 Member Enumeration Documentation	229
7.66.2.1 type	229
7.67 qevercloud::SharedNotebookRecipientSettings Struct Reference	230
7.67.1 Detailed Description	230
7.67.2 Member Function Documentation	230
7.67.2.1 operator"!=()	230
7.67.2.2 operator==( )	231
7.67.3 Member Data Documentation	231
7.67.3.1 reminderNotifyEmail	231
7.67.3.2 reminderNotifyInApp	231
7.68 qevercloud::SponsoredGroupRole Struct Reference	231
7.68.1 Detailed Description	231
7.68.2 Member Enumeration Documentation	231
7.68.2.1 type	231
7.69 qevercloud::SyncChunk Struct Reference	232
7.69.1 Detailed Description	232
7.69.2 Member Function Documentation	232
7.69.2.1 operator"!=()	233
7.69.2.2 operator==( )	233
7.69.3 Member Data Documentation	233
7.69.3.1 chunkHighUSN	233
7.69.3.2 currentTime	233
7.69.3.3 expungedLinkedNotebooks	233
7.69.3.4 expungedNotebooks	233

7.69.3.5 expungedNotes . . . . .	233
7.69.3.6 expungedSearches . . . . .	234
7.69.3.7 expungedTags . . . . .	234
7.69.3.8 linkedNotebooks . . . . .	234
7.69.3.9 notebooks . . . . .	234
7.69.3.10 notes . . . . .	234
7.69.3.11 resources . . . . .	234
7.69.3.12 searches . . . . .	234
7.69.3.13 tags . . . . .	235
7.69.3.14 updateCount . . . . .	235
7.70 qevercloud::SyncChunkFilter Struct Reference . . . . .	235
7.70.1 Detailed Description . . . . .	235
7.70.2 Member Function Documentation . . . . .	236
7.70.2.1 operator!=(()) . . . . .	236
7.70.2.2 operator==(()) . . . . .	236
7.70.3 Member Data Documentation . . . . .	236
7.70.3.1 includeExpunged . . . . .	236
7.70.3.2 includeLinkedNotebooks . . . . .	236
7.70.3.3 includeNoteApplicationDataFullMap . . . . .	236
7.70.3.4 includeNoteAttributes . . . . .	236
7.70.3.5 includeNotebooks . . . . .	237
7.70.3.6 includeNoteResourceApplicationDataFullMap . . . . .	237
7.70.3.7 includeNoteResources . . . . .	237
7.70.3.8 includeNotes . . . . .	237
7.70.3.9 includeResourceApplicationDataFullMap . . . . .	237
7.70.3.10 includeResources . . . . .	237
7.70.3.11 includeSearches . . . . .	237
7.70.3.12 includeTags . . . . .	238
7.70.3.13 requireNoteContentClass . . . . .	238
7.71 qevercloud::SyncState Struct Reference . . . . .	238
7.71.1 Detailed Description . . . . .	238
7.71.2 Member Function Documentation . . . . .	238
7.71.2.1 operator!=(()) . . . . .	238
7.71.2.2 operator==(()) . . . . .	239
7.71.3 Member Data Documentation . . . . .	239
7.71.3.1 currentTime . . . . .	239
7.71.3.2 fullSyncBefore . . . . .	239
7.71.3.3 updateCount . . . . .	239
7.71.3.4 uploaded . . . . .	239
7.72 qevercloud::Tag Struct Reference . . . . .	239
7.72.1 Detailed Description . . . . .	240
7.72.2 Member Function Documentation . . . . .	240

7.72.2.1 operator!=(())	240
7.72.2.2 operator==(())	240
7.72.3 Member Data Documentation	240
7.72.3.1 guid	240
7.72.3.2 name	241
7.72.3.3 parentGuid	241
7.72.3.4 updateSequenceNum	241
7.73 qevercloud::ThriftException Class Reference	241
7.73.1 Detailed Description	242
7.73.2 Constructor & Destructor Documentation	242
7.73.2.1 ThriftException() [1/3]	242
7.73.2.2 ThriftException() [2/3]	242
7.73.2.3 ThriftException() [3/3]	242
7.73.3 Member Function Documentation	242
7.73.3.1 exceptionData()	243
7.73.3.2 type()	243
7.73.3.3 what()	243
7.73.4 Member Data Documentation	243
7.73.4.1 m_type	243
7.74 qevercloud::ThriftExceptionData Class Reference	243
7.74.1 Detailed Description	244
7.74.2 Constructor & Destructor Documentation	244
7.74.2.1 ThriftExceptionData()	244
7.74.3 Member Function Documentation	244
7.74.3.1 throwException()	244
7.74.4 Member Data Documentation	244
7.74.4.1 m_type	244
7.75 qevercloud::Thumbnail Class Reference	245
7.75.1 Detailed Description	245
7.75.2 Constructor & Destructor Documentation	245
7.75.2.1 Thumbnail() [1/2]	246
7.75.2.2 Thumbnail() [2/2]	246
7.75.2.3 ~Thumbnail()	246
7.75.3 Member Function Documentation	246
7.75.3.1 createPostRequest()	246
7.75.3.2 download()	247
7.75.3.3 downloadAsync()	247
7.75.3.4 setAuthenticationToken()	247
7.75.3.5 setHost()	248
7.75.3.6 setImageType()	248
7.75.3.7 setShardId()	248
7.75.3.8 setSize()	248

7.76 qevercloud::ThriftException::Type Struct Reference . . . . .	249
7.76.1 Member Enumeration Documentation . . . . .	249
7.76.1.1 type . . . . .	249
7.77 qevercloud::User Struct Reference . . . . .	249
7.77.1 Detailed Description . . . . .	250
7.77.2 Member Function Documentation . . . . .	250
7.77.2.1 operator"!=( ) . . . . .	250
7.77.2.2 operator==( ) . . . . .	250
7.77.3 Member Data Documentation . . . . .	250
7.77.3.1 accounting . . . . .	250
7.77.3.2 active . . . . .	251
7.77.3.3 attributes . . . . .	251
7.77.3.4 businessUserInfo . . . . .	251
7.77.3.5 created . . . . .	251
7.77.3.6 deleted . . . . .	251
7.77.3.7 email . . . . .	251
7.77.3.8 id . . . . .	251
7.77.3.9 name . . . . .	252
7.77.3.10 premiumInfo . . . . .	252
7.77.3.11 privilege . . . . .	252
7.77.3.12 shardId . . . . .	252
7.77.3.13 timezone . . . . .	252
7.77.3.14 updated . . . . .	252
7.77.3.15 username . . . . .	253
7.78 qevercloud::UserAttributes Struct Reference . . . . .	253
7.78.1 Detailed Description . . . . .	254
7.78.2 Member Function Documentation . . . . .	254
7.78.2.1 operator"!=( ) . . . . .	254
7.78.2.2 operator==( ) . . . . .	254
7.78.3 Member Data Documentation . . . . .	254
7.78.3.1 businessAddress . . . . .	254
7.78.3.2 clipFullPage . . . . .	254
7.78.3.3 comments . . . . .	254
7.78.3.4 dailyEmailLimit . . . . .	255
7.78.3.5 dateAgreedToTermsOfService . . . . .	255
7.78.3.6 defaultLatitude . . . . .	255
7.78.3.7 defaultLocationName . . . . .	255
7.78.3.8 defaultLongitude . . . . .	255
7.78.3.9 educationalDiscount . . . . .	255
7.78.3.10 emailOptOutDate . . . . .	255
7.78.3.11 groupName . . . . .	256
7.78.3.12 hideSponsorBilling . . . . .	256

7.78.3.13 incomingEmailAddress . . . . .	256
7.78.3.14 maxReferrals . . . . .	256
7.78.3.15 partnerEmailOptInDate . . . . .	256
7.78.3.16 preactivation . . . . .	256
7.78.3.17 preferredCountry . . . . .	256
7.78.3.18 preferredLanguage . . . . .	257
7.78.3.19 recentMailedAddresses . . . . .	257
7.78.3.20 recognitionLanguage . . . . .	257
7.78.3.21 refererCode . . . . .	257
7.78.3.22 referralCount . . . . .	257
7.78.3.23 referralProof . . . . .	257
7.78.3.24 reminderEmailConfig . . . . .	257
7.78.3.25 sentEmailCount . . . . .	258
7.78.3.26 sentEmailDate . . . . .	258
7.78.3.27 taxExempt . . . . .	258
7.78.3.28 twitterId . . . . .	258
7.78.3.29 twitterUserName . . . . .	258
7.78.3.30 useEmailAutoFiling . . . . .	258
7.78.3.31 viewedPromotions . . . . .	258
7.79 qevercloud::UserStore Class Reference . . . . .	259
7.79.1 Detailed Description . . . . .	260
7.79.2 Constructor & Destructor Documentation . . . . .	260
7.79.2.1 UserStore() . . . . .	260
7.79.3 Member Function Documentation . . . . .	260
7.79.3.1 authenticate() . . . . .	260
7.79.3.2 authenticateAsync() . . . . .	261
7.79.3.3 authenticateLongSession() . . . . .	262
7.79.3.4 authenticateLongSessionAsync() . . . . .	263
7.79.3.5 authenticateToBusiness() . . . . .	263
7.79.3.6 authenticateToBusinessAsync() . . . . .	264
7.79.3.7 authenticationToken() . . . . .	264
7.79.3.8 checkVersion() . . . . .	264
7.79.3.9 checkVersionAsync() . . . . .	265
7.79.3.10 completeTwoFactorAuthentication() . . . . .	265
7.79.3.11 completeTwoFactorAuthenticationAsync() . . . . .	266
7.79.3.12 getBootstrapInfo() . . . . .	266
7.79.3.13 getBootstrapInfoAsync() . . . . .	267
7.79.3.14 getNoteStoreUrl() . . . . .	267
7.79.3.15 getNoteStoreUrlAsync() . . . . .	267
7.79.3.16 getPremiumInfo() . . . . .	267
7.79.3.17 getPremiumInfoAsync() . . . . .	267
7.79.3.18 getPublicUserInfo() . . . . .	267



7.79.3.19 getPublicUserInfoAsync()	268
7.79.3.20 getUser()	268
7.79.3.21 getUserAsync()	268
7.79.3.22 refreshAuthentication()	268
7.79.3.23 refreshAuthenticationAsync()	269
7.79.3.24 revokeLongSession()	269
7.79.3.25 revokeLongSessionAsync()	269
7.79.3.26 setAuthenticationToken()	270
<b>8 File Documentation</b>	<b>271</b>
8.1 AsyncResult.h File Reference	271
8.2 constants.h File Reference	271
8.3 EDAMErrorCode.h File Reference	274
8.4 EventLoopFinisher.h File Reference	275
8.5 EverCloudException.h File Reference	275
8.6 exceptions.h File Reference	276
8.7 export.h File Reference	276
8.7.1 Macro Definition Documentation	276
8.7.1.1 QEVERCLOUD_EXPORT	277
8.8 globals.h File Reference	277
8.9 oauth.h File Reference	277
8.10 Optional.h File Reference	278
8.11 QEverCloud.h File Reference	278
8.12 QEverCloudOAuth.h File Reference	278
8.13 qt4helpers.h File Reference	278
8.13.1 Macro Definition Documentation	279
8.13.1.1 QEC_SIGNAL	279
8.13.1.2 QEC_SLOT	279
8.14 README.md File Reference	279
8.15 services.h File Reference	279
8.16 thumbnail.h File Reference	280
8.17 types.h File Reference	280
<b>Index</b>	<b>283</b>



# Chapter 1

## QEverCloud

### Unofficial Evernote Cloud API for Qt

Travis CI (Linux, OS X):

AppVeyor CI (Windows):

### 1.1 What's this

This library presents the complete Evernote SDK for Qt. All the functionality that is described on [Evernote site](#) is implemented and ready to use. In particular OAuth authentication is implemented.

Read doxygen generated [documentation](#) for detailed info. The documentation is also provided by the [qevercloud-doc](#) package in Debian.

#### 1.1.1 API breaks from 2.x to 3.0

The API breaks only include the relocation of header files required in order to use the library: in 2.x one could simply do

```
#include <QEverCloud.h>
```

while since 3.0 the intended way to use the installed shared library is the following:

```
#if QT_VERSION < QT_VERSION_CHECK(5, 0, 0)
#include <qt4qevercloud/QEverCloud.h>
#else
#include <qt5qevercloud/QEverCloud.h>
#endif
```

#### 1.1.2 C++11/14/17 features

The library does not use any C++11/14/17 features directly but only through macros like `Q_DECL_OVERRIDE`, `Q_STATIC_ASSERT_X`, `QStringLiteral` and others. Some of these macros are also "backported" to Qt4 version of the library i.e. they are defined by the library itself for Qt4 version. So the library should be buildable even with not C++11/14/17-compliant compiler.

## 1.2 Include files for applications using the library

Two "cumulative" headers - [QEverCloud.h](#) or [QEverCloudOAuth.h](#) - include everything needed for the general and OAuth functionality correspondingly. More "fine-grained" headers are available within the same subfolder if needed.

## 1.3 Related projects

- [NotePoster](#) is an example app using QEverCloud library to post notes to Evernote.
- [QEverCloud packaging](#) repository contains various files and scripts required for building QEverCloud packages for various platforms and distributions.
- [QEverCloudGenerator](#) repository contains the parser of [Evernote Thrift IDL files](#) generating headers and sources for QEverCloud library.

## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">qevercloud</a>	.....	13
----------------------------	-------	----



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

qevercloud::Accounting . . . . .	41
qevercloud::AuthenticationResult . . . . .	48
qevercloud::BootstrapInfo . . . . .	50
qevercloud::BootstrapProfile . . . . .	51
qevercloud::BootstrapSettings . . . . .	52
qevercloud::BusinessNotebook . . . . .	55
qevercloud::BusinessUserInfo . . . . .	57
qevercloud::BusinessUserRole . . . . .	58
qevercloud::ClientUsageMetrics . . . . .	59
qevercloud::Data . . . . .	60
qevercloud::EDAMErrorCode . . . . .	61
exception	
qevercloud::EverCloudException . . . . .	79
qevercloud::EvernoteException . . . . .	83
qevercloud::EDAMNotFoundException . . . . .	63
qevercloud::EDAMSystemException . . . . .	67
qevercloud::EDAMSystemExceptionAuthExpired . . . . .	69
qevercloud::EDAMSystemExceptionRateLimitReached . . . . .	73
qevercloud::EDAMUserException . . . . .	75
qevercloud::ThriftException . . . . .	241
qevercloud::Thumbnail::ImageType . . . . .	92
qevercloud::LazyMap . . . . .	92
qevercloud::LinkedNotebook . . . . .	94
qevercloud::Note . . . . .	96
qevercloud::NoteAttributes . . . . .	100
qevercloud::Notebook . . . . .	105
qevercloud::NotebookDescriptor . . . . .	108
qevercloud::NotebookRestrictions . . . . .	110
qevercloud::NoteCollectionCounts . . . . .	114
qevercloud::NoteEmailParameters . . . . .	115
qevercloud::NoteFilter . . . . .	117
qevercloud::NoteList . . . . .	119
qevercloud::NoteMetadata . . . . .	121
qevercloud::NotesMetadataList . . . . .	124
qevercloud::NotesMetadataResultSpec . . . . .	126

qevercloud::NoteSortOrder	128
qevercloud::NoteVersionId	192
qevercloud::EvernoteOAuthWebView::OAuthResult	194
qevercloud::Optional< T >	195
qevercloud::Optional< bool >	195
qevercloud::Optional< BusinessUserRole::type >	195
qevercloud::Optional< double >	195
qevercloud::Optional< Guid >	195
qevercloud::Optional< NoteSortOrder::type >	195
qevercloud::Optional< PremiumOrderStatus::type >	195
qevercloud::Optional< PrivilegeLevel::type >	195
qevercloud::Optional< QByteArray >	195
qevercloud::Optional< qevercloud::Accounting >	195
qevercloud::Optional< qevercloud::BusinessNotebook >	195
qevercloud::Optional< qevercloud::BusinessUserInfo >	195
qevercloud::Optional< qevercloud::Data >	195
qevercloud::Optional< qevercloud::LazyMap >	195
qevercloud::Optional< qevercloud::Note >	195
qevercloud::Optional< qevercloud::NoteAttributes >	195
qevercloud::Optional< qevercloud::NotebookRestrictions >	195
qevercloud::Optional< qevercloud::NoteFilter >	195
qevercloud::Optional< qevercloud::PremiumInfo >	195
qevercloud::Optional< qevercloud::PublicUserInfo >	195
qevercloud::Optional< qevercloud::Publishing >	195
qevercloud::Optional< qevercloud::ResourceAttributes >	195
qevercloud::Optional< qevercloud::SavedSearchScope >	195
qevercloud::Optional< qevercloud::SharedNotebookRecipientSettings >	195
qevercloud::Optional< qevercloud::User >	195
qevercloud::Optional< qevercloud::UserAttributes >	195
qevercloud::Optional< qint16 >	195
qevercloud::Optional< qint32 >	195
qevercloud::Optional< qint64 >	195
qevercloud::Optional< QList< Guid > >	195
qevercloud::Optional< QList< qevercloud::LinkedNotebook > >	195
qevercloud::Optional< QList< qevercloud::Note > >	195
qevercloud::Optional< QList< qevercloud::Notebook > >	195
qevercloud::Optional< QList< qevercloud::NotebookDescriptor > >	195
qevercloud::Optional< QList< qevercloud::Resource > >	195
qevercloud::Optional< QList< qevercloud::SavedSearch > >	195
qevercloud::Optional< QList< qevercloud::SharedNotebook > >	195
qevercloud::Optional< QList< qevercloud::Tag > >	195
qevercloud::Optional< QList< qint64 > >	195
qevercloud::Optional< QMap< Guid, qint32 > >	195
qevercloud::Optional< QMap< QString, QString > >	195
qevercloud::Optional< QSet< QString > >	195
qevercloud::Optional< QString >	195
qevercloud::Optional< QStringList >	195
qevercloud::Optional< QueryFormat::type >	195
qevercloud::Optional< ReminderEmailConfig::type >	195
qevercloud::Optional< SharedNotebookInstanceRestrictions::type >	195
qevercloud::Optional< SharedNotebookPrivilegeLevel::type >	195
qevercloud::Optional< SponsoredGroupRole::type >	195
qevercloud::Optional< Timestamp >	195
qevercloud::Optional< UserID >	195
qevercloud::PremiumInfo	201
qevercloud::PremiumOrderStatus	204
qevercloud::PrivilegeLevel	206
qevercloud::PublicUserInfo	207



qevercloud::Publishing . . . . .	208
QDialog	
qevercloud::EvernoteOAuthDialog . . . . .	86
QObject	
qevercloud::AsyncResult . . . . .	45
qevercloud::EventLoopFinisher . . . . .	78
qevercloud::EverCloudExceptionData . . . . .	81
qevercloud::EvernoteExceptionData . . . . .	85
qevercloud::EDAMNotFoundExceptionData . . . . .	65
qevercloud::EDAMSystemExceptionData . . . . .	71
qevercloud::EDAMSystemExceptionAuthExpiredData . . . . .	70
qevercloud::EDAMSystemExceptionRateLimitReachedData . . . . .	74
qevercloud::EDAMUserExceptionData . . . . .	77
qevercloud::ThriftExceptionData . . . . .	243
qevercloud::NoteStore . . . . .	129
qevercloud::UserStore . . . . .	259
qevercloud::QueryFormat . . . . .	210
QWidget	
qevercloud::EvernoteOAuthWebView . . . . .	89
qevercloud::RelatedQuery . . . . .	211
qevercloud::RelatedResult . . . . .	212
qevercloud::RelatedResultSpec . . . . .	214
qevercloud::ReminderEmailConfig . . . . .	215
qevercloud::Resource . . . . .	216
qevercloud::ResourceAttributes . . . . .	219
qevercloud::SavedSearch . . . . .	222
qevercloud::SavedSearchScope . . . . .	224
qevercloud::SharedNotebook . . . . .	225
qevercloud::SharedNotebookInstanceRestrictions . . . . .	228
qevercloud::SharedNotebookPrivilegeLevel . . . . .	229
qevercloud::SharedNotebookRecipientSettings . . . . .	230
qevercloud::SponsoredGroupRole . . . . .	231
qevercloud::SyncChunk . . . . .	232
qevercloud::SyncChunkFilter . . . . .	235
qevercloud::SyncState . . . . .	238
qevercloud::Tag . . . . .	239
qevercloud::Thumbnail . . . . .	245
qevercloud::ThriftException::Type . . . . .	249
qevercloud::User . . . . .	249
qevercloud::UserAttributes . . . . .	253



## Chapter 4

# Class Index

### 4.1 Class List

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

qevercloud::Accounting . . . . .	41
qevercloud::AsyncResult	
Returned by asynchronous versions of functions . . . . .	45
qevercloud::AuthenticationResult . . . . .	48
qevercloud::BootstrapInfo . . . . .	50
qevercloud::BootstrapProfile . . . . .	51
qevercloud::BootstrapSettings . . . . .	52
qevercloud::BusinessNotebook . . . . .	55
qevercloud::BusinessUserInfo . . . . .	57
qevercloud::BusinessUserRole . . . . .	58
qevercloud::ClientUsageMetrics . . . . .	59
qevercloud::Data . . . . .	60
qevercloud::EDAMErrorCode . . . . .	61
qevercloud::EDAMNotFoundException . . . . .	63
qevercloud::EDAMNotFoundExceptionData . . . . .	65
qevercloud::EDAMSystemException . . . . .	67
qevercloud::EDAMSystemExceptionAuthExpired . . . . .	69
qevercloud::EDAMSystemExceptionAuthExpiredData . . . . .	70
qevercloud::EDAMSystemExceptionData . . . . .	71
qevercloud::EDAMSystemExceptionRateLimitReached . . . . .	73
qevercloud::EDAMSystemExceptionRateLimitReachedData . . . . .	74
qevercloud::EDAMUserException . . . . .	75
qevercloud::EDAMUserExceptionData . . . . .	77
qevercloud::EventLoopFinisher . . . . .	78
qevercloud::EverCloudException . . . . .	79
qevercloud::EverCloudExceptionData	
EverCloudException counterpart for asynchronous API . . . . .	81
qevercloud::EvernoteException . . . . .	83
qevercloud::EvernoteExceptionData . . . . .	85
qevercloud::EvernoteOAuthDialog	
Authorizes your app with the Evernote service by means of OAuth authentication . . . . .	86
qevercloud::EvernoteOAuthWebView	
The class is tailored specifically for OAuth authorization with Evernote . . . . .	89
qevercloud::Thumbnail::ImageType . . . . .	92
qevercloud::LazyMap . . . . .	92

qevercloud::LinkedNotebook	94
qevercloud::Note	96
qevercloud::NoteAttributes	100
qevercloud::Notebook	105
qevercloud::NotebookDescriptor	108
qevercloud::NotebookRestrictions	110
qevercloud::NoteCollectionCounts	114
qevercloud::NoteEmailParameters	115
qevercloud::NoteFilter	117
qevercloud::NoteList	119
qevercloud::NoteMetadata	121
qevercloud::NotesMetadataList	124
qevercloud::NotesMetadataResultSpec	126
qevercloud::NoteSortOrder	128
qevercloud::NoteStore	129
qevercloud::NoteVersionId	192
qevercloud::EvernoteOAuthWebView::OAuthResult	194
qevercloud::Optional< T >	195
qevercloud::PremiumInfo	201
qevercloud::PremiumOrderStatus	204
qevercloud::PrivilegeLevel	206
qevercloud::PublicUserInfo	207
qevercloud::Publishing	208
qevercloud::QueryFormat	210
qevercloud::RelatedQuery	211
qevercloud::RelatedResult	212
qevercloud::RelatedResultSpec	214
qevercloud::ReminderEmailConfig	215
qevercloud::Resource	216
qevercloud::ResourceAttributes	219
qevercloud::SavedSearch	222
qevercloud::SavedSearchScope	224
qevercloud::SharedNotebook	225
qevercloud::SharedNotebookInstanceRestrictions	228
qevercloud::SharedNotebookPrivilegeLevel	229
qevercloud::SharedNotebookRecipientSettings	230
qevercloud::SponsoredGroupRole	231
qevercloud::SyncChunk	232
qevercloud::SyncChunkFilter	235
qevercloud::SyncState	238
qevercloud::Tag	239
qevercloud::ThriftException	241
qevercloud::ThriftExceptionData	243
qevercloud::Thumbnail	
The class is for downloading thumbnails for notes and resources from Evernote servers	245
qevercloud::ThriftException::Type	249
qevercloud::User	249
qevercloud::UserAttributes	253
qevercloud::UserStore	259

## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AsyncResult.h</a>	271
<a href="#">constants.h</a>	271
<a href="#">EDAMErrorCode.h</a>	274
<a href="#">EventLoopFinisher.h</a>	275
<a href="#">EverCloudException.h</a>	275
<a href="#">exceptions.h</a>	276
<a href="#">export.h</a>	276
<a href="#">globals.h</a>	277
<a href="#">oauth.h</a>	277
<a href="#">Optional.h</a>	278
<a href="#">QEverCloud.h</a>	278
<a href="#">QEverCloudOAuth.h</a>	278
<a href="#">qt4helpers.h</a>	278
<a href="#">services.h</a>	279
<a href="#">thumbnail.h</a>	280
<a href="#">types.h</a>	280



## Chapter 6

# Namespace Documentation

### 6.1 qevercloud Namespace Reference

#### Classes

- struct [Accounting](#)
- class [AsyncResult](#)
  - Returned by asynchronous versions of functions.*
- struct [AuthenticationResult](#)
- struct [BootstrapInfo](#)
- struct [BootstrapProfile](#)
- struct [BootstrapSettings](#)
- struct [BusinessNotebook](#)
- struct [BusinessUserInfo](#)
- struct [BusinessUserRole](#)
- struct [ClientUsageMetrics](#)
- struct [Data](#)
- struct [EDAMErrorCode](#)
- class [EDAMNotFoundException](#)
- class [EDAMNotFoundExceptionData](#)
- class [EDAMSystemException](#)
- class [EDAMSystemExceptionAuthExpired](#)
- class [EDAMSystemExceptionAuthExpiredData](#)
- class [EDAMSystemExceptionData](#)
- class [EDAMSystemExceptionRateLimitReached](#)
- class [EDAMSystemExceptionRateLimitReachedData](#)
- class [EDAMUserException](#)
- class [EDAMUserExceptionData](#)
- class [EventLoopFinisher](#)
- class [EverCloudException](#)
- class [EverCloudExceptionData](#)
  - [EverCloudException](#) counterpart for asynchronous API.*
- class [EvernoteException](#)
- class [EvernoteExceptionData](#)
- class [EvernoteOAuthDialog](#)
  - Authorizes your app with the Evernote service by means of OAuth authentication.*
- class [EvernoteOAuthWebView](#)

*The class is tailored specifically for OAuth authorization with Evernote.*

- struct [LazyMap](#)
- struct [LinkedNotebook](#)
- struct [Note](#)
- struct [NoteAttributes](#)
- struct [Notebook](#)
- struct [NotebookDescriptor](#)
- struct [NotebookRestrictions](#)
- struct [NoteCollectionCounts](#)
- struct [NoteEmailParameters](#)
- struct [NoteFilter](#)
- struct [NoteList](#)
- struct [NoteMetadata](#)
- struct [NotesMetadataList](#)
- struct [NotesMetadataResultSpec](#)
- struct [NoteSortOrder](#)
- class [NoteStore](#)
- struct [NoteVersionId](#)
- class [Optional](#)
- struct [PremiumInfo](#)
- struct [PremiumOrderStatus](#)
- struct [PrivilegeLevel](#)
- struct [PublicUserInfo](#)
- struct [Publishing](#)
- struct [QueryFormat](#)
- struct [RelatedQuery](#)
- struct [RelatedResult](#)
- struct [RelatedResultSpec](#)
- struct [ReminderEmailConfig](#)
- struct [Resource](#)
- struct [ResourceAttributes](#)
- struct [SavedSearch](#)
- struct [SavedSearchScope](#)
- struct [SharedNotebook](#)
- struct [SharedNotebookInstanceRestrictions](#)
- struct [SharedNotebookPrivilegeLevel](#)
- struct [SharedNotebookRecipientSettings](#)
- struct [SponsoredGroupRole](#)
- struct [SyncChunk](#)
- struct [SyncChunkFilter](#)
- struct [SyncState](#)
- struct [Tag](#)
- class [ThriftException](#)
- class [ThriftExceptionData](#)
- class [Thumbnail](#)

*The class is for downloading thumbnails for notes and resources from Evernote servers.*

- struct [User](#)
- struct [UserAttributes](#)
- class [UserStore](#)

## Typedefs

- typedef quint32 [UserID](#)
- typedef QString [Guid](#)
- typedef quint64 [Timestamp](#)



## Functions

- `QEVERCLOUD_EXPORT QNetworkAccessManager * evernoteNetworkAccessManager ()`
- `QEVERCLOUD_EXPORT int libraryVersion ()`
- `void setNonceGenerator (quint64(*nonceGenerator)())`

*Sets the function to use for nonce generation for OAuth authentication.*

## Variables

- `class QEVERCLOUD_EXPORT EverCloudExceptionData`
- `const QEVERCLOUD_EXPORT qint32 EDAM_ATTRIBUTE_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_ATTRIBUTE_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_ATTRIBUTE_REGEX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_ATTRIBUTE_LIST_MAX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_ATTRIBUTE_MAP_MAX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_GUID_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_GUID_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_GUID_REGEX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_EMAIL_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_EMAIL_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_EMAIL_LOCAL_REGEX`
- `const QEVERCLOUD_EXPORT QString EDAM_EMAIL_DOMAIN_REGEX`
- `const QEVERCLOUD_EXPORT QString EDAM_EMAIL_REGEX`
- `const QEVERCLOUD_EXPORT QString EDAM_VAT_REGEX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_TIMEZONE_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_TIMEZONE_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_TIMEZONE_REGEX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_MIME_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_MIME_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_REGEX`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_GIF`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_JPEG`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_PNG`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_WAV`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_MP3`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_AMR`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_AAC`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_M4A`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_MP4_VIDEO`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_INK`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_PDF`
- `const QEVERCLOUD_EXPORT QString EDAM_MIME_TYPE_DEFAULT`
- `const QEVERCLOUD_EXPORT QSet< QString > EDAM_MIME_TYPES`
- `const QEVERCLOUD_EXPORT QSet< QString > EDAM_INDEXABLE_RESOURCE_MIME_TYPES`
- `const QEVERCLOUD_EXPORT qint32 EDAM_SEARCH_QUERY_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_SEARCH_QUERY_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_SEARCH_QUERY_REGEX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_HASH_LEN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_USER_USERNAME_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_USER_USERNAME_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_USER_USERNAME_REGEX`
- `const QEVERCLOUD_EXPORT qint32 EDAM_USER_NAME_LEN_MIN`
- `const QEVERCLOUD_EXPORT qint32 EDAM_USER_NAME_LEN_MAX`
- `const QEVERCLOUD_EXPORT QString EDAM_USER_NAME_REGEX`

- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_TAG\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_TAG\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_TAG\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_TITLE\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_TITLE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_NOTE\\_TITLE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_CONTENT\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_CONTENT\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_APPLICATIONDATA\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_APPLICATIONDATA\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_APPLICATIONDATA\\_VALUE\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_APPLICATIONDATA\\_VALUE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_APPLICATIONDATA\\_ENTRY\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_APPLICATIONDATA\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_APPLICATIONDATA\\_VALUE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTEBOOK\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTEBOOK\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_NOTEBOOK\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTEBOOK\\_STACK\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTEBOOK\\_STACK\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_NOTEBOOK\\_STACK\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PUBLISHING\\_URI\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PUBLISHING\\_URI\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_PUBLISHING\\_URI\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QSet< QString > [EDAM\\_PUBLISHING\\_URI\\_PROHIBITED](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PUBLISHING\\_DESCRIPTION\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PUBLISHING\\_DESCRIPTION\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_PUBLISHING\\_DESCRIPTION\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_SAVED\\_SEARCH\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_SAVED\\_SEARCH\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_SAVED\\_SEARCH\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_PASSWORD\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_PASSWORD\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_USER\\_PASSWORD\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_BUSINESS\\_URI\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_TAGS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_RESOURCES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_TAGS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_BUSINESS\\_TAGS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_SAVED\\_SEARCHES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_NOTES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_BUSINESS\\_NOTES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_NOTEBOOKS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_BUSINESS\\_NOTEBOOKS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_RECENT\\_MAILED\\_ADDRESSES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_MAIL\\_LIMIT\\_DAILY\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_MAIL\\_LIMIT\\_DAILY\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint64 [EDAM\\_USER\\_UPLOAD\\_LIMIT\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint64 [EDAM\\_USER\\_UPLOAD\\_LIMIT\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint64 [EDAM\\_USER\\_UPLOAD\\_LIMIT\\_BUSINESS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_SIZE\\_MAX\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_SIZE\\_MAX\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_RESOURCE\\_SIZE\\_MAX\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_RESOURCE\\_SIZE\\_MAX\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_LINKED\\_NOTEBOOK\\_MAX](#)

- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_USER\\_LINKED\\_NOTEBOOK\\_MAX\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTEBOOK\\_SHARED\\_NOTEBOOK\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_CONTENT\\_CLASS\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_NOTE\\_CONTENT\\_CLASS\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_NOTE\\_CONTENT\\_CLASS\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_HELLO\\_APP\\_CONTENT\\_CLASS\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_FOOD\\_APP\\_CONTENT\\_CLASS\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_HELLO\\_ENCOUNTER](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_HELLO\\_PROFILE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_FOOD\\_MEAL](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_SKITCH\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_SKITCH](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_SKITCH\\_PDF](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_PENULTIMATE\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_CONTENT\\_CLASS\\_PENULTIMATE\\_NOTEBOOK](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_RELATED\\_PLAINTEXT\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_RELATED\\_PLAINTEXT\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_RELATED\\_MAX\\_NOTES](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_RELATED\\_MAX\\_NOTEBOOKS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_RELATED\\_MAX\\_TAGS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_BUSINESS\\_PHONE\\_NUMBER\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PREFERENCE\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PREFERENCE\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PREFERENCE\\_VALUE\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PREFERENCE\\_VALUE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_MAX\\_PREFERENCES](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_MAX\\_VALUES\\_PER\\_PREFERENCE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_PREFERENCE\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_PREFERENCE\\_VALUE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_PREFERENCE\\_SHORTCUTS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_PREFERENCE\\_SHORTCUTS\\_MAX\\_VALUES](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_DEVICE\\_ID\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_DEVICE\\_ID\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_DEVICE\\_DESCRIPTION\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_DEVICE\\_DESCRIPTION\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_SEARCH\\_SUGGESTIONS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_SEARCH\\_SUGGESTIONS\\_PREFIX\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [EDAM\\_SEARCH\\_SUGGESTIONS\\_PREFIX\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [CLASSIFICATION\\_RECIPE\\_USER\\_NON\\_RECIPE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [CLASSIFICATION\\_RECIPE\\_USER\\_RECIPE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [CLASSIFICATION\\_RECIPE\\_SERVICE\\_RECIPE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_NOTE\\_SOURCE\\_WEB\\_CLIP](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_NOTE\\_SOURCE\\_MAIL\\_CLIP](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [EDAM\\_NOTE\\_SOURCE\\_MAIL\\_SMTP\\_GATEWAY](#)
- const [QEVERCLOUD\\_EXPORT](#) qint16 [EDAM\\_VERSION\\_MAJOR](#)
- const [QEVERCLOUD\\_EXPORT](#) qint16 [EDAM\\_VERSION\\_MINOR](#)

### 6.1.1 Detailed Description

Original work: Copyright (c) 2014 Sergey Skoblikov Modified work: Copyright (c) 2015-2016 Dmitry Ivanov

This file is a part of QEverCloud project and is distributed under the terms of MIT license: <https://opensource.org/licenses/MIT>

Original work: Copyright (c) 2014 Sergey Skoblikov Modified work: Copyright (c) 2015-2016 Dmitry Ivanov

This file is a part of QEverCloud project and is distributed under the terms of MIT license: <https://opensource.org/licenses/MIT>

This file was generated from Evernote Thrift API

Original work: Copyright (c) 2014 Sergey Skoblikov Modified work: Copyright (c) 2015-2016 Dmitry Ivanov

This file is a part of QEverCloud project and is distributed under the terms of MIT license: <https://opensource.org/licenses/MIT> All the library lives in this namespace.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 Guid

```
typedef QString qevercloud::Guid
```

Most data elements within a user's account (e.g. notebooks, notes, tags, resources, etc.) are internally referred to using a globally unique identifier that is written in a standard string format. For example:

```
"8743428c-ef91-4d05-9e7c-4a2e856e813a"
```

The internal components of the GUID are not given any particular meaning: only the entire string is relevant as a unique identifier.

#### 6.1.2.2 Timestamp

```
typedef quint64 qevercloud::Timestamp
```

An Evernote Timestamp is the date and time of an event in UTC time. This is expressed as a specific number of milliseconds since the standard base "epoch" of:

January 1, 1970, 00:00:00 GMT

NOTE: the time is expressed at the resolution of milliseconds, but the value is only precise to the level of seconds. This means that the last three (decimal) digits of the timestamp will be '000'.

The Thrift IDL specification does not include a native date/time type, so this value is used instead.

The service will accept timestamp values (e.g. for [Note](#) created and update times) between 1000-01-01 and 9999-12-31

### 6.1.2.3 UserID

```
typedef quint32 qevercloud::UserID
```

Every Evernote account is assigned a unique numeric identifier which will not change for the life of the account. This is independent of the (string-based) "username" which is known by the user for login purposes. The user should have no reason to know their UserID.

## 6.1.3 Function Documentation

### 6.1.3.1 evernoteNetworkAccessManager()

```
QEVERCLOUD_EXPORT QNetworkAccessManager* qevercloud::evernoteNetworkAccessManager ( )
```

All network request made by QEverCloud - including OAuth - are served by this NetworkAccessManager.

Use this function to handle proxy authentication requests etc.

### 6.1.3.2 libraryVersion()

```
QEVERCLOUD_EXPORT int qevercloud::libraryVersion ( )
```

qevercloud library version.

### 6.1.3.3 setNonceGenerator()

```
void qevercloud::setNonceGenerator (
    quint64 (*)() nonceGenerator )
```

Sets the function to use for nonce generation for OAuth authentication.

The default algorithm uses qrand() so do not forget to call qsrand() in your application!

qrand() is not guaranteed to be cryptographically strong. I try to amend the fact by using QUuid::createUuid() which uses /dev/urandom if it's available. But this is no guarantee either. So if you want total control over nonce generation you can write your own algorithm.

setNonceGenerator is NOT thread safe.

## 6.1.4 Variable Documentation

#### 6.1.4.1 CLASSIFICATION\_RECIPE\_SERVICE\_RECIPE

```
const QEVERCLOUD_EXPORT QString qevercloud::CLASSIFICATION_RECIPE_SERVICE_RECIPE
```

A value for the "recipe" key in the "classifications" map in [NoteAttributes](#) that indicates the Evernote service has classified a note as being a recipe.

#### 6.1.4.2 CLASSIFICATION\_RECIPE\_USER\_NON\_RECIPE

```
const QEVERCLOUD_EXPORT QString qevercloud::CLASSIFICATION_RECIPE_USER_NON_RECIPE
```

A value for the "recipe" key in the "classifications" map in [NoteAttributes](#) that indicates the user has classified a note as being a non-recipe.

#### 6.1.4.3 CLASSIFICATION\_RECIPE\_USER\_RECIPE

```
const QEVERCLOUD_EXPORT QString qevercloud::CLASSIFICATION_RECIPE_USER_RECIPE
```

A value for the "recipe" key in the "classifications" map in [NoteAttributes](#) that indicates the user has classified a note as being a recipe.

#### 6.1.4.4 EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX

```
const QEVERCLOUD_EXPORT quint32 qevercloud::EDAM_APPLICATIONDATA_ENTRY_LEN_MAX
```

The total length of an entry in an applicationData [LazyMap](#), which is the sum of the length of the key and the value for the entry.

#### 6.1.4.5 EDAM\_APPLICATIONDATA\_NAME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT quint32 qevercloud::EDAM_APPLICATIONDATA_NAME_LEN_MAX
```

Maximum length of an application name, which is the key in an applicationData [LazyMap](#) found in entities such as Resources and Notes.

#### 6.1.4.6 EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT quint32 qevercloud::EDAM_APPLICATIONDATA_NAME_LEN_MIN
```

Minimum length of an application name, which is the key in an applicationData [LazyMap](#) found in entities such as Resources and Notes.

#### 6.1.4.7 EDAM\_APPLICATIONDATA\_NAME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_APPLICATIONDATA_NAME_REGEX
```

An application name must match this regex. An application name is the key portion of an entry in an applicationData map as found in entities such as Resources and Notes. [Note](#) that even if both the name and value regexes match, it is still necessary to check the sum of the lengths against EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX.

#### 6.1.4.8 EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_APPLICATIONDATA_VALUE_LEN_MAX
```

Maximum length of an applicationData value in a [LazyMap](#), found in entities such as Resources and Notes. [Note](#), however, that the sum of the size of the key and value is constrained by EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX, so the maximum length, in practice, depends upon the key value being used.

#### 6.1.4.9 EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_APPLICATIONDATA_VALUE_LEN_MIN
```

Minimum length of an applicationData value in a [LazyMap](#), found in entities such as Resources and Notes.

#### 6.1.4.10 EDAM\_APPLICATIONDATA\_VALUE\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_APPLICATIONDATA_VALUE_REGEX
```

An applicationData map value must match this regex. [Note](#) that even if both the name and value regexes match, it is still necessary to check the sum of the lengths against EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX.

#### 6.1.4.11 EDAM\_ATTRIBUTE\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_ATTRIBUTE_LEN_MAX
```

Maximum length of any string-based attribute, in Unicode chars

#### 6.1.4.12 EDAM\_ATTRIBUTE\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_ATTRIBUTE_LEN_MIN
```

Minimum length of any string-based attribute, in Unicode chars

#### 6.1.4.13 EDAM\_ATTRIBUTE\_LIST\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_ATTRIBUTE_LIST_MAX
```

The maximum number of values that can be stored in a list-based attribute (e.g. see [UserAttributes.recentMailedAddresses](#))

#### 6.1.4.14 EDAM\_ATTRIBUTE\_MAP\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_ATTRIBUTE_MAP_MAX
```

The maximum number of entries that can be stored in a map-based attribute such as applicationData fields in Resources and Notes.

#### 6.1.4.15 EDAM\_ATTRIBUTE\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_ATTRIBUTE_REGEX
```

Any string-based attribute must match the provided regular expression. This excludes all Unicode line endings and control characters.

#### 6.1.4.16 EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LEN_MAX
```

The maximum length, in Unicode characters, of a description for a business notebook.

#### 6.1.4.17 EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LEN_MIN
```

The minimum length, in Unicode characters, of a description for a business notebook.

#### 6.1.4.18 EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_REGEX
```

All business notebook descriptions must match this pattern. This excludes control chars or line/paragraph separators. The string may not begin or end with whitespace.

#### 6.1.4.19 EDAM\_BUSINESS\_NOTEBOOKS\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_BUSINESS_NOTEBOOKS_MAX
```

Maximum number of Notebooks in a business account

#### 6.1.4.20 EDAM\_BUSINESS\_NOTES\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_BUSINESS_NOTES_MAX
```

Maximum number of Notes per business account

#### 6.1.4.21 EDAM\_BUSINESS\_PHONE\_NUMBER\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_BUSINESS_PHONE_NUMBER_LEN_MAX
```

The maximum length of a business phone number.



#### 6.1.4.22 EDAM\_BUSINESS\_TAGS\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_BUSINESS_TAGS_MAX
```

Maximum number of Tags per business account.

#### 6.1.4.23 EDAM\_BUSINESS\_URI\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_BUSINESS_URI_LEN_MAX
```

The maximum length of an Evernote Business URI

#### 6.1.4.24 EDAM\_CONTENT\_CLASS\_FOOD\_MEAL

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_FOOD_MEAL
```

The content class prefix used for structured notes created by Evernote Food that captures the experience of a particular meal. When performing a wildcard search via filtered sync chunks or search strings, the \* character must be appended to this constant.

#### 6.1.4.25 EDAM\_CONTENT\_CLASS\_HELLO\_ENCOUNTER

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_HELLO_ENCOUNTER
```

The content class prefix used for structured notes created by Evernote Hello that represents an encounter with a person. When performing a wildcard search via filtered sync chunks or search strings, the \* character must be appended to this constant.

#### 6.1.4.26 EDAM\_CONTENT\_CLASS\_HELLO\_PROFILE

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_HELLO_PROFILE
```

The content class prefix used for structured notes created by Evernote Hello that represents the user's profile. When performing a wildcard search via filtered sync chunks or search strings, the \* character must be appended to this constant.

#### 6.1.4.27 EDAM\_CONTENT\_CLASS\_PENULTIMATE\_NOTEBOOK

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_PENULTIMATE_NOTEBOOK
```

The content class value used for structured notes created by Evernote Penultimate that represents a Penultimate notebook.

#### 6.1.4.28 EDAM\_CONTENT\_CLASS\_PENULTIMATE\_PREFIX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_PENULTIMATE_PREFIX
```

The content class prefix used for structured notes created by Evernote Penultimate. When performing a wildcard search via filtered sync chunks or search strings, the \* character must be appended to this constant.

#### 6.1.4.29 EDAM\_CONTENT\_CLASS\_SKITCH

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_SKITCH
```

The content class value used for structured image notes created by Evernote Skitch.

#### 6.1.4.30 EDAM\_CONTENT\_CLASS\_SKITCH\_PDF

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_SKITCH_PDF
```

The content class value used for structured PDF notes created by Evernote Skitch.

#### 6.1.4.31 EDAM\_CONTENT\_CLASS\_SKITCH\_PREFIX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_CONTENT_CLASS_SKITCH_PREFIX
```

The content class prefix used for structured notes created by Evernote Skitch. When performing a wildcard search via filtered sync chunks or search strings, the \* character must be appended to this constant.

#### 6.1.4.32 EDAM\_DEVICE\_DESCRIPTION\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_DEVICE_DESCRIPTION_LEN_MAX
```

Maximum length of the device description string associated with long sessions.

#### 6.1.4.33 EDAM\_DEVICE\_DESCRIPTION\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_DEVICE_DESCRIPTION_REGEX
```

Regular expression for device description strings associated with long sessions.

#### 6.1.4.34 EDAM\_DEVICE\_ID\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_DEVICE_ID_LEN_MAX
```

Maximum length of the device identifier string associated with long sessions.

#### 6.1.4.35 EDAM\_DEVICE\_ID\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_DEVICE_ID_REGEX
```

Regular expression for device identifier strings associated with long sessions.

#### 6.1.4.36 EDAM\_EMAIL\_DOMAIN\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_EMAIL_DOMAIN_REGEX
```

A regular expression that matches the part of an email address after the '@' symbol.

#### 6.1.4.37 EDAM\_EMAIL\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_EMAIL_LEN_MAX
```

The maximum length of any email address

#### 6.1.4.38 EDAM\_EMAIL\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_EMAIL_LEN_MIN
```

The minimum length of any email address

#### 6.1.4.39 EDAM\_EMAIL\_LOCAL\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_EMAIL_LOCAL_REGEX
```

A regular expression that matches the part of an email address before the '@' symbol.

#### 6.1.4.40 EDAM\_EMAIL\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_EMAIL_REGEX
```

A regular expression that must match any email address given to Evernote. Email addresses must comply with RFC 2821 and 2822.

#### 6.1.4.41 EDAM\_FOOD\_APP\_CONTENT\_CLASS\_PREFIX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_FOOD_APP_CONTENT_CLASS_PREFIX
```

The content class prefix used for all notes created by Evernote Food. This prefix can be used to assemble individual content class strings, or can be used to create a wildcard search to get all notes created by Food. When performing a wildcard search via filtered sync chunks or search strings, the \* character must be appended to this constant.

#### 6.1.4.42 EDAM\_GUID\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_GUID_LEN_MAX
```

The maximum length of a GUID generated by the Evernote service

#### 6.1.4.43 EDAM\_GUID\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_GUID_LEN_MIN
```

The minimum length of a GUID generated by the Evernote service

#### 6.1.4.44 EDAM\_GUID\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_GUID_REGEX
```

GUIDs generated by the Evernote service will match the provided pattern

#### 6.1.4.45 EDAM\_HASH\_LEN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_HASH_LEN
```

The exact length of a MD5 hash checksum, in binary bytes. This is the exact length that must be matched for any binary hash value.

#### 6.1.4.46 EDAM\_HELLO\_APP\_CONTENT\_CLASS\_PREFIX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_HELLO_APP_CONTENT_CLASS_PREFIX
```

The content class prefix used for all notes created by Evernote Hello. This prefix can be used to assemble individual content class strings, or can be used to create a wildcard search to get all notes created by Hello. When performing a wildcard search via filtered sync chunks or search strings, the \* character must be appended to this constant.

#### 6.1.4.47 EDAM\_INDEXABLE\_RESOURCE\_MIME\_TYPES

```
const QEVERCLOUD_EXPORT QSet< QString > qevercloud::EDAM_INDEXABLE_RESOURCE_MIME_TYPES
```

The set of MIME types that Evernote will parse and index for searching. With exception of images, and PDFs, which are handled in a different way.

#### 6.1.4.48 EDAM\_MAX\_PREFERENCES

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_MAX_PREFERENCES
```

Maximum number of name/value pairs allowed

#### 6.1.4.49 EDAM\_MAX\_VALUES\_PER\_PREFERENCE

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_MAX_VALUES_PER_PREFERENCE
```

Maximum number of values per preference name

#### 6.1.4.50 EDAM\_MIME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_MIME_LEN_MAX
```

The maximum length of any MIME type string given to Evernote

#### 6.1.4.51 EDAM\_MIME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_MIME_LEN_MIN
```

The minimum length of any MIME type string given to Evernote

#### 6.1.4.52 EDAM\_MIME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_REGEX
```

Any MIME type string given to Evernote must match the provided pattern. E.g.: image/gif

#### 6.1.4.53 EDAM\_MIME\_TYPE\_AAC

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_AAC
```

Canonical MIME type string for AAC audio resources

#### 6.1.4.54 EDAM\_MIME\_TYPE\_AMR

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_AMR
```

Canonical MIME type string for AMR audio resources

#### 6.1.4.55 EDAM\_MIME\_TYPE\_DEFAULT

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_DEFAULT
```

MIME type used for attachments of an unspecified type

#### 6.1.4.56 EDAM\_MIME\_TYPE\_GIF

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_GIF
```

Canonical MIME type string for GIF image resources

#### 6.1.4.57 EDAM\_MIME\_TYPE\_INK

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_INK
```

Canonical MIME type string for Evernote Ink resources

#### 6.1.4.58 EDAM\_MIME\_TYPE\_JPEG

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_JPEG
```

Canonical MIME type string for JPEG image resources

#### 6.1.4.59 EDAM\_MIME\_TYPE\_M4A

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_M4A
```

Canonical MIME type string for MP4 audio resources

#### 6.1.4.60 EDAM\_MIME\_TYPE\_MP3

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_MP3
```

Canonical MIME type string for MP3 audio resources

#### 6.1.4.61 EDAM\_MIME\_TYPE\_MP4\_VIDEO

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_MP4_VIDEO
```

Canonical MIME type string for MP4 video resources

#### 6.1.4.62 EDAM\_MIME\_TYPE\_PDF

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_PDF
```

Canonical MIME type string for PDF resources

#### 6.1.4.63 EDAM\_MIME\_TYPE\_PNG

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_PNG
```

Canonical MIME type string for PNG image resources

#### 6.1.4.64 EDAM\_MIME\_TYPE\_WAV

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_MIME_TYPE_WAV
```

Canonical MIME type string for WAV audio resources

#### 6.1.4.65 EDAM\_MIME\_TYPES

```
const QEVERCLOUD_EXPORT QSet< QString > qevercloud::EDAM_MIME_TYPES
```

The set of resource MIME types that are expected to be handled correctly by all of the major Evernote client applications.

#### 6.1.4.66 EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_CONTENT_CLASS_LEN_MAX
```

The maximum length of the content class attribute of a note.

#### 6.1.4.67 EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_CONTENT_CLASS_LEN_MIN
```

The minimum length of the content class attribute of a note.

#### 6.1.4.68 EDAM\_NOTE\_CONTENT\_CLASS\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_NOTE_CONTENT_CLASS_REGEX
```

The regular expression that the content class of a note must match to be valid.

#### 6.1.4.69 EDAM\_NOTE\_CONTENT\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_CONTENT_LEN_MAX
```

Maximum length of a [Note.content](#) field. [Note.content](#) fields must comply with the ENML DTD.

#### 6.1.4.70 EDAM\_NOTE\_CONTENT\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_CONTENT_LEN_MIN
```

Minimum length of a [Note.content](#) field. [Note.content](#) fields must comply with the ENML DTD.

#### 6.1.4.71 EDAM\_NOTE\_RESOURCES\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_RESOURCES_MAX
```

The maximum number of Resources per [Note](#)

#### 6.1.4.72 EDAM\_NOTE\_SIZE\_MAX\_FREE

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_SIZE_MAX_FREE
```

Maximum total size of a [Note](#) that can be added to a Free account. The size of a note is calculated as: ENML content length (in Unicode characters) plus the sum of all resource sizes (in bytes).

#### 6.1.4.73 EDAM\_NOTE\_SIZE\_MAX\_PREMIUM

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_SIZE_MAX_PREMIUM
```

Maximum total size of a [Note](#) that can be added to a Premium account. The size of a note is calculated as: ENML content length (in Unicode characters) plus the sum of all resource sizes (in bytes).

#### 6.1.4.74 EDAM\_NOTE\_SOURCE\_MAIL\_CLIP

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_NOTE_SOURCE_MAIL_CLIP
```

Standardized value for the 'source' NoteAttribute for notes that were clipped from an email message.

#### 6.1.4.75 EDAM\_NOTE\_SOURCE\_MAIL\_SMTP\_GATEWAY

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_NOTE_SOURCE_MAIL_SMTP_GATEWAY
```

Standardized value for the 'source' NoteAttribute for notes that were created via email sent to Evernote's email interface.

#### 6.1.4.76 EDAM\_NOTE\_SOURCE\_WEB\_CLIP

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_NOTE_SOURCE_WEB_CLIP
```

Standardized value for the 'source' NoteAttribute for notes that were clipped from the web in some manner.

#### 6.1.4.77 EDAM\_NOTE\_TAGS\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_TAGS_MAX
```

The maximum number of Tags per [Note](#)

#### 6.1.4.78 EDAM\_NOTE\_TITLE\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_TITLE_LEN_MAX
```

The maximum length of a [Note.title](#), in Unicode characters

#### 6.1.4.79 EDAM\_NOTE\_TITLE\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTE_TITLE_LEN_MIN
```

The minimum length of a [Note.title](#), in Unicode characters



#### 6.1.4.80 EDAM\_NOTE\_TITLE\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_NOTE_TITLE_REGEX
```

All [Note.title](#) fields must match this pattern. This excludes control chars or line/paragraph separators. The string may not begin or end with whitespace.

#### 6.1.4.81 EDAM\_NOTEBOOK\_NAME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTEBOOK_NAME_LEN_MAX
```

The maximum length of a [Notebook.name](#), in Unicode characters

#### 6.1.4.82 EDAM\_NOTEBOOK\_NAME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTEBOOK_NAME_LEN_MIN
```

The minimum length of a [Notebook.name](#), in Unicode characters

#### 6.1.4.83 EDAM\_NOTEBOOK\_NAME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_NOTEBOOK_NAME_REGEX
```

All [Notebook.name](#) fields must match this pattern. This excludes control chars or line/paragraph separators. The string may not begin or end with whitespace.

#### 6.1.4.84 EDAM\_NOTEBOOK\_SHARED\_NOTEBOOK\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTEBOOK_SHARED_NOTEBOOK_MAX
```

Maximum number of shared notebooks per notebook

#### 6.1.4.85 EDAM\_NOTEBOOK\_STACK\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTEBOOK_STACK_LEN_MAX
```

The maximum length of a [Notebook.stack](#), in Unicode characters

#### 6.1.4.86 EDAM\_NOTEBOOK\_STACK\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_NOTEBOOK_STACK_LEN_MIN
```

The minimum length of a [Notebook.stack](#), in Unicode characters

#### 6.1.4.87 EDAM\_NOTEBOOK\_STACK\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_NOTEBOOK_STACK_REGEX
```

All [Notebook.stack](#) fields must match this pattern. This excludes control chars or line/paragraph separators. The string may not begin or end with whitespace.

#### 6.1.4.88 EDAM\_PREFERENCE\_NAME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PREFERENCE_NAME_LEN_MAX
```

Maximum length of a preference name

#### 6.1.4.89 EDAM\_PREFERENCE\_NAME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PREFERENCE_NAME_LEN_MIN
```

Minimum length of a preference name

#### 6.1.4.90 EDAM\_PREFERENCE\_NAME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_PREFERENCE_NAME_REGEX
```

A preference name must match this regex.

#### 6.1.4.91 EDAM\_PREFERENCE\_SHORTCUTS

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_PREFERENCE_SHORTCUTS
```

The name of the preferences entry that contains shortcuts.

#### 6.1.4.92 EDAM\_PREFERENCE\_SHORTCUTS\_MAX\_VALUES

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PREFERENCE_SHORTCUTS_MAX_VALUES
```

The maximum number of shortcuts that a user may have.

#### 6.1.4.93 EDAM\_PREFERENCE\_VALUE\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PREFERENCE_VALUE_LEN_MAX
```

Maximum length of a preference value

#### 6.1.4.94 EDAM\_PREFERENCE\_VALUE\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PREFERENCE_VALUE_LEN_MIN
```

Minimum length of a preference value

#### 6.1.4.95 EDAM\_PREFERENCE\_VALUE\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_PREFERENCE_VALUE_REGEX
```

A preference value must match this regex.

#### 6.1.4.96 EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PUBLISHING_DESCRIPTION_LEN_MAX
```

The maximum length of a [Publishing.publicDescription](#) field.

#### 6.1.4.97 EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PUBLISHING_DESCRIPTION_LEN_MIN
```

The minimum length of a [Publishing.publicDescription](#) field.

#### 6.1.4.98 EDAM\_PUBLISHING\_DESCRIPTION\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_PUBLISHING_DESCRIPTION_REGEX
```

Any public notebook's [Publishing.publicDescription](#) field must match this pattern. No control chars or line/paragraph separators, and can't start or end with whitespace.

#### 6.1.4.99 EDAM\_PUBLISHING\_URI\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PUBLISHING_URI_LEN_MAX
```

The maximum length of a public notebook URI component

#### 6.1.4.100 EDAM\_PUBLISHING\_URI\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_PUBLISHING_URI_LEN_MIN
```

The minimum length of a public notebook URI component

#### 6.1.4.101 EDAM\_PUBLISHING\_URI\_PROHIBITED

```
const QEVERCLOUD_EXPORT QSet< QString > qevercloud::EDAM_PUBLISHING_URI_PROHIBITED
```

The set of strings that may not be used as a publishing URI

#### 6.1.4.102 EDAM\_PUBLISHING\_URI\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_PUBLISHING_URI_REGEX
```

A public notebook URI component must match the provided pattern

#### 6.1.4.103 EDAM\_RELATED\_MAX\_NOTEBOOKS

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_RELATED_MAX_NOTEBOOKS
```

The maximum number of notebooks that will be returned from a findRelated() query.

#### 6.1.4.104 EDAM\_RELATED\_MAX\_NOTES

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_RELATED_MAX_NOTES
```

The maximum number of notes that will be returned from a findRelated() query.

#### 6.1.4.105 EDAM\_RELATED\_MAX\_TAGS

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_RELATED_MAX_TAGS
```

The maximum number of tags that will be returned from a findRelated() query.

#### 6.1.4.106 EDAM\_RELATED\_PLAINTEXT\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_RELATED_PLAINTEXT_LEN_MAX
```

The maximum length of the plain text in a findRelated query, assuming that plaintext is being provided.

#### 6.1.4.107 EDAM\_RELATED\_PLAINTEXT\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_RELATED_PLAINTEXT_LEN_MIN
```

The minimum length of the plain text in a findRelated query, assuming that plaintext is being provided.

#### 6.1.4.108 EDAM\_RESOURCE\_SIZE\_MAX\_FREE

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_RESOURCE_SIZE_MAX_FREE
```

Maximum size of a resource, in bytes, for Free accounts

#### 6.1.4.109 EDAM\_RESOURCE\_SIZE\_MAX\_PREMIUM

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_RESOURCE_SIZE_MAX_PREMIUM
```

Maximum size of a resource, in bytes, for Premium accounts

#### 6.1.4.110 EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_SAVED_SEARCH_NAME_LEN_MAX
```

The maximum length of a [SavedSearch.name](#) field

#### 6.1.4.111 EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_SAVED_SEARCH_NAME_LEN_MIN
```

The minimum length of a [SavedSearch.name](#) field

#### 6.1.4.112 EDAM\_SAVED\_SEARCH\_NAME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_SAVED_SEARCH_NAME_REGEX
```

[SavedSearch.name](#) fields must match this pattern. No control chars or line/paragraph separators, and can't start or end with whitespace.

#### 6.1.4.113 EDAM\_SEARCH\_QUERY\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_SEARCH_QUERY_LEN_MAX
```

The maximum length of a user search query string in Unicode chars

#### 6.1.4.114 EDAM\_SEARCH\_QUERY\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_SEARCH_QUERY_LEN_MIN
```

The minimum length of a user search query string in Unicode chars

#### 6.1.4.115 EDAM\_SEARCH\_QUERY\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_SEARCH_QUERY_REGEX
```

Search queries must match the provided pattern. This is used for both ad-hoc queries and [SavedSearch.query](#) fields. This excludes all control characters and line/paragraph separators.

#### 6.1.4.116 EDAM\_SEARCH\_SUGGESTIONS\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_SEARCH_SUGGESTIONS_MAX
```

Maximum number of search suggestions that can be returned

#### 6.1.4.117 EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_SEARCH_SUGGESTIONS_PREFIX_LEN_MAX
```

Maximum length of the search suggestion prefix

#### 6.1.4.118 EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_SEARCH_SUGGESTIONS_PREFIX_LEN_MIN
```

Minimum length of the search suggestion prefix

#### 6.1.4.119 EDAM\_TAG\_NAME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_TAG_NAME_LEN_MAX
```

The maximum length of a [Tag.name](#), in Unicode characters

#### 6.1.4.120 EDAM\_TAG\_NAME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_TAG_NAME_LEN_MIN
```

The minimum length of a [Tag.name](#), in Unicode characters

#### 6.1.4.121 EDAM\_TAG\_NAME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_TAG_NAME_REGEX
```

All [Tag.name](#) fields must match this pattern. This excludes control chars, commas or line/paragraph separators. The string may not begin or end with whitespace.

#### 6.1.4.122 EDAM\_TIMEZONE\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_TIMEZONE_LEN_MAX
```

The maximum length of a timezone specification string

#### 6.1.4.123 EDAM\_TIMEZONE\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_TIMEZONE_LEN_MIN
```

The minimum length of a timezone specification string

#### 6.1.4.124 EDAM\_TIMEZONE\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_TIMEZONE_REGEX
```

Any timezone string given to Evernote must match the provided pattern. This permits either a locale-based standard timezone or a GMT offset. E.g.:

- America/Los\_Angeles
- GMT+08:00

#### 6.1.4.125 EDAM\_USER\_LINKED\_NOTEBOOK\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_LINKED_NOTEBOOK_MAX
```

Maximum number of linked notebooks per account, for a free account.

#### 6.1.4.126 EDAM\_USER\_LINKED\_NOTEBOOK\_MAX\_PREMIUM

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_LINKED_NOTEBOOK_MAX_PREMIUM
```

Maximum number of linked notebooks per account, for a premium account. Users who are part of an active business are also covered under "premium".

#### 6.1.4.127 EDAM\_USER\_MAIL\_LIMIT\_DAILY\_FREE

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_MAIL_LIMIT_DAILY_FREE
```

The number of emails of any type that can be sent by a user with a Free account from the service per day. If an email is sent to two different recipients, this counts as two emails.

#### 6.1.4.128 EDAM\_USER\_MAIL\_LIMIT\_DAILY\_PREMIUM

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_MAIL_LIMIT_DAILY_PREMIUM
```

The number of emails of any type that can be sent by a user with a Premium account from the service per day. If an email is sent to two different recipients, this counts as two emails.

#### 6.1.4.129 EDAM\_USER\_NAME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_NAME_LEN_MAX
```

Maximum length of the [User.name](#) field

#### 6.1.4.130 EDAM\_USER\_NAME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_NAME_LEN_MIN
```

Minimum length of the [User.name](#) field

#### 6.1.4.131 EDAM\_USER\_NAME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_USER_NAME_REGEX
```

The [User.name](#) field must match this pattern, which excludes line endings and control characters.

#### 6.1.4.132 EDAM\_USER\_NOTEBOOKS\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_NOTEBOOKS_MAX
```

Maximum number of Notebooks per user

#### 6.1.4.133 EDAM\_USER\_NOTES\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_NOTES_MAX
```

Maximum number of Notes per user

#### 6.1.4.134 EDAM\_USER\_PASSWORD\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_PASSWORD_LEN_MAX
```

The maximum length of an Evernote user password

#### 6.1.4.135 EDAM\_USER\_PASSWORD\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_PASSWORD_LEN_MIN
```

The minimum length of an Evernote user password

#### 6.1.4.136 EDAM\_USER\_PASSWORD\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_USER_PASSWORD_REGEX
```

Evernote user passwords must match this regular expression



#### 6.1.4.137 EDAM\_USER\_RECENT\_MAILED\_ADDRESSES\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_RECENT_MAILED_ADDRESSES_MAX
```

Maximum number of recent email addresses that are maintained (see [UserAttributes.recentMailedAddresses](#))

#### 6.1.4.138 EDAM\_USER\_SAVED\_SEARCHES\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_SAVED_SEARCHES_MAX
```

Maximum number of SavedSearches per account

#### 6.1.4.139 EDAM\_USER\_TAGS\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_TAGS_MAX
```

Maximum number of Tags per account

#### 6.1.4.140 EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS

```
const QEVERCLOUD_EXPORT qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS
```

The number of bytes of new data that may be uploaded to a Business user's personal account each month. [Note](#) that content uploaded into the Business notebooks by the user does not count against this limit.

#### 6.1.4.141 EDAM\_USER\_UPLOAD\_LIMIT\_FREE

```
const QEVERCLOUD_EXPORT qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_FREE
```

The number of bytes of new data that may be uploaded to a Free user's account each month.

#### 6.1.4.142 EDAM\_USER\_UPLOAD\_LIMIT\_PREMIUM

```
const QEVERCLOUD_EXPORT qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_PREMIUM
```

The number of bytes of new data that may be uploaded to a Premium user's account each month.

#### 6.1.4.143 EDAM\_USER\_USERNAME\_LEN\_MAX

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_USERNAME_LEN_MAX
```

The maximum length of an Evernote username

#### 6.1.4.144 EDAM\_USER\_USERNAME\_LEN\_MIN

```
const QEVERCLOUD_EXPORT qint32 qevercloud::EDAM_USER_USERNAME_LEN_MIN
```

The minimum length of an Evernote username

#### 6.1.4.145 EDAM\_USER\_USERNAME\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_USER_USERNAME_REGEX
```

Any Evernote [User.username](#) field must match this pattern. This restricts usernames to a format that could permit use as a domain name component. E.g. "username.whatever.evernote.com"

#### 6.1.4.146 EDAM\_VAT\_REGEX

```
const QEVERCLOUD_EXPORT QString qevercloud::EDAM_VAT_REGEX
```

A regular expression that must match any VAT ID given to Evernote. [ref http://en.wikipedia.org/wiki/VAT\\_identification\\_number](http://en.wikipedia.org/wiki/VAT_identification_number) [ref http://my.safaribooksonline.com/book/programming/r](http://my.safaribooksonline.com/book/programming/r)

#### 6.1.4.147 EDAM\_VERSION\_MAJOR

```
const QEVERCLOUD_EXPORT qint16 qevercloud::EDAM_VERSION_MAJOR
```

The major version number for the current revision of the EDAM protocol. Clients pass this to the service using [UserStore.checkVersion](#) at the beginning of a session to confirm that they are not out of date.

#### 6.1.4.148 EDAM\_VERSION\_MINOR

```
const QEVERCLOUD_EXPORT qint16 qevercloud::EDAM_VERSION_MINOR
```

The minor version number for the current revision of the EDAM protocol. Clients pass this to the service using [UserStore.checkVersion](#) at the beginning of a session to confirm that they are not out of date.

#### 6.1.4.149 EverCloudExceptionData

```
class QEVERCLOUD_EXPORT qevercloud::EverCloudExceptionData
```

## Chapter 7

# Class Documentation

### 7.1 qevercloud::Accounting Struct Reference

```
#include <types.h>
```

#### Public Member Functions

- bool `operator==` (const [Accounting](#) &other) const
- bool `operator!=` (const [Accounting](#) &other) const

#### Public Attributes

- [Optional](#)< qint64 > `uploadLimit`
- [Optional](#)< [Timestamp](#) > `uploadLimitEnd`
- [Optional](#)< qint64 > `uploadLimitNextMonth`
- [Optional](#)< [PremiumOrderStatus::type](#) > `premiumServiceStatus`
- [Optional](#)< [QString](#) > `premiumOrderNumber`
- [Optional](#)< [QString](#) > `premiumCommerceService`
- [Optional](#)< [Timestamp](#) > `premiumServiceStart`
- [Optional](#)< [QString](#) > `premiumServiceSKU`
- [Optional](#)< [Timestamp](#) > `lastSuccessfulCharge`
- [Optional](#)< [Timestamp](#) > `lastFailedCharge`
- [Optional](#)< [QString](#) > `lastFailedChargeReason`
- [Optional](#)< [Timestamp](#) > `nextPaymentDue`
- [Optional](#)< [Timestamp](#) > `premiumLockUntil`
- [Optional](#)< [Timestamp](#) > `updated`
- [Optional](#)< [QString](#) > `premiumSubscriptionNumber`
- [Optional](#)< [Timestamp](#) > `lastRequestedCharge`
- [Optional](#)< [QString](#) > `currency`
- [Optional](#)< qint32 > `unitPrice`
- [Optional](#)< qint32 > `businessId`
- [Optional](#)< [QString](#) > `businessName`
- [Optional](#)< [BusinessUserRole::type](#) > `businessRole`
- [Optional](#)< qint32 > `unitDiscount`
- [Optional](#)< [Timestamp](#) > `nextChargeDate`

### 7.1.1 Detailed Description

This represents the bookkeeping information for the user's subscription.

### 7.1.2 Member Function Documentation

#### 7.1.2.1 `operator!=(())`

```
bool qevercloud::Accounting::operator!= (
    const Accounting & other ) const [inline]
```

#### 7.1.2.2 `operator==(())`

```
bool qevercloud::Accounting::operator== (
    const Accounting & other ) const [inline]
```

### 7.1.3 Member Data Documentation

#### 7.1.3.1 `businessId`

[Optional](#)< qint32 > qevercloud::Accounting::businessId

*DEPRECATED*: See [BusinessUserInfo](#).

#### 7.1.3.2 `businessName`

[Optional](#)< QString > qevercloud::Accounting::businessName

*DEPRECATED*: See [BusinessUserInfo](#).

#### 7.1.3.3 `businessRole`

[Optional](#)< [BusinessUserRole::type](#) > qevercloud::Accounting::businessRole

*DEPRECATED*: See [BusinessUserInfo](#).

#### 7.1.3.4 currency

`Optional< QString > qevercloud::Accounting::currency`

ISO 4217 currency code

#### 7.1.3.5 lastFailedCharge

`Optional< Timestamp > qevercloud::Accounting::lastFailedCharge`

Date the last time a charge was attempted and failed.

#### 7.1.3.6 lastFailedChargeReason

`Optional< QString > qevercloud::Accounting::lastFailedChargeReason`

Reason provided for the charge failure

#### 7.1.3.7 lastRequestedCharge

`Optional< Timestamp > qevercloud::Accounting::lastRequestedCharge`

Date charge last attempted

#### 7.1.3.8 lastSuccessfulCharge

`Optional< Timestamp > qevercloud::Accounting::lastSuccessfulCharge`

Date the last time the user was charged. Null if never charged.

#### 7.1.3.9 nextChargeDate

`Optional< Timestamp > qevercloud::Accounting::nextChargeDate`

The next time the user will be charged, may or may not be the same as nextPaymentDue

#### 7.1.3.10 nextPaymentDue

`Optional< Timestamp > qevercloud::Accounting::nextPaymentDue`

The end of the billing cycle. This could be in the past if there are failed charges.

#### 7.1.3.11 premiumCommerceService

`Optional< QString > qevercloud::Accounting::premiumCommerceService`

The commerce system used (paypal, Google checkout, etc)

#### 7.1.3.12 premiumLockUntil

```
Optional< Timestamp > qevercloud::Accounting::premiumLockUntil
```

An internal variable to manage locking operations on the commerce variables.

#### 7.1.3.13 premiumOrderNumber

```
Optional< QString > qevercloud::Accounting::premiumOrderNumber
```

The order number used by the commerce system to process recurring payments

#### 7.1.3.14 premiumServiceSKU

```
Optional< QString > qevercloud::Accounting::premiumServiceSKU
```

The code associated with the purchase eg. monthly or annual purchase. Clients should interpret this value and localize it.

#### 7.1.3.15 premiumServiceStart

```
Optional< Timestamp > qevercloud::Accounting::premiumServiceStart
```

The start date when this premium promotion began (this number will get overwritten if a premium service is canceled and then re-activated).

#### 7.1.3.16 premiumServiceStatus

```
Optional< PremiumOrderStatus::type > qevercloud::Accounting::premiumServiceStatus
```

Indicates the phases of a premium account during the billing process.

#### 7.1.3.17 premiumSubscriptionNumber

```
Optional< QString > qevercloud::Accounting::premiumSubscriptionNumber
```

The number number identifying the recurring subscription used to make the recurring charges.

#### 7.1.3.18 unitDiscount

```
Optional< qint32 > qevercloud::Accounting::unitDiscount
```

discount per seat in negative amount and smallest unit of the currency (e.g. cents for USD)

**7.1.3.19 unitPrice**

`Optional< qint32 > qevercloud::Accounting::unitPrice`

charge in the smallest unit of the currency (e.g. cents for USD)

**7.1.3.20 updated**

`Optional< Timestamp > qevercloud::Accounting::updated`

The date any modification where made to this record.

**7.1.3.21 uploadLimit**

`Optional< qint64 > qevercloud::Accounting::uploadLimit`

The number of bytes that can be uploaded to the account in the current month. For new notes that are created, this is the length of the note content (in Unicode characters) plus the size of each resource (in bytes). For edited notes, this is the the difference between the old length and the new length (if this is greater than 0) plus the size of each new resource.

**7.1.3.22 uploadLimitEnd**

`Optional< Timestamp > qevercloud::Accounting::uploadLimitEnd`

The date and time when the current upload limit expires. At this time, the monthly upload count reverts to 0 and a new limit is imposed. This date and time is exclusive, so this is effectively the start of the new month.

**7.1.3.23 uploadLimitNextMonth**

`Optional< qint64 > qevercloud::Accounting::uploadLimitNextMonth`

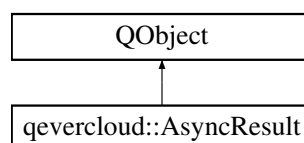
When uploadLimitEnd is reached, the service will change uploadLimit to uploadLimitNextMonth. If a premium account is canceled, this mechanism will reset the quota appropriately.

**7.2 qevercloud::AsyncResult Class Reference**

Returned by asynchronous versions of functions.

```
#include <AsyncResult.h>
```

Inheritance diagram for qevercloud::AsyncResult:



## Public Types

- typedef QVariant(\* [ReadFunctionType](#)) (QByteArray replyData)

## Signals

- void [finished](#) (QVariant result, QSharedPointer< [EverCloudExceptionData](#) > error)

*Emitted upon asynchronous call completion.*

## Public Member Functions

- [AsyncResult](#) (QString url, QByteArray postData, [ReadFunctionType](#) readFunction=AsyncResult::asIs, bool autoDelete=true, QObject \*parent=Q\_NULLPTR)
- [AsyncResult](#) (QNetworkRequest request, QByteArray postData, [ReadFunctionType](#) readFunction=AsyncResult::asIs, bool autoDelete=true, QObject \*parent=Q\_NULLPTR)
- [~AsyncResult](#) ()
- bool [waitForFinished](#) (int timeout=-1)

*Wait for asynchronous operation to complete.*

### 7.2.1 Detailed Description

Returned by asynchronous versions of functions.

Wait for [AsyncResult::finished](#) signal.

Intended usage is something like this:

```
NoteStore* ns;
Note note;
...
QObject::connect(ns->createNoteAsync(note), &AsyncResult::finished, [ns](QVariant result,
    QSharedPointer<EverCloudExceptionData> error) {
    if(!error.isNull()) {
        // do something in case of an error
    } else {
        Note note = result.value<Note>();
        // process returned result
    }
});
```

### 7.2.2 Member Typedef Documentation

#### 7.2.2.1 ReadFunctionType

```
typedef QVariant(* qevercloud::AsyncResult::ReadFunctionType) (QByteArray replyData)
```

### 7.2.3 Constructor & Destructor Documentation



### 7.2.3.1 AsyncResult() [1/2]

```
qevercloud::AsyncResult::AsyncResult (
    QString url,
    QByteArray postData,
    ReadFunctionType readFunction = AsyncResult::asIs,
    bool autoDelete = true,
    QObject * parent = Q_NULLPTR )
```

### 7.2.3.2 AsyncResult() [2/2]

```
qevercloud::AsyncResult::AsyncResult (
    QNetworkRequest request,
    QByteArray postData,
    ReadFunctionType readFunction = AsyncResult::asIs,
    bool autoDelete = true,
    QObject * parent = Q_NULLPTR )
```

### 7.2.3.3 ~AsyncResult()

```
qevercloud::AsyncResult::~AsyncResult ( )
```

## 7.2.4 Member Function Documentation

### 7.2.4.1 finished

```
void qevercloud::AsyncResult::finished (
    QVariant result,
    QSharedPointer< EverCloudExceptionData > error ) [signal]
```

Emitted upon asynchronous call completion.

#### Parameters

<i>result</i>	
<i>error</i>	error.isNull() != true in case of an error. See <a href="#">EverCloudExceptionData</a> for more details.

[AsyncResult](#) deletes itself after emitting this signal. You don't have to manage it's lifetime explicitly.

### 7.2.4.2 waitForFinished()

```
bool qevercloud::AsyncResult::waitForFinished (
```

```
int timeout = -1 )
```

Wait for asynchronous operation to complete.

#### Parameters

<i>timeout</i>	Maximum time to wait in milliseconds. Forever if < 0.
----------------	---

#### Returns

true if finished succesfully, false in case of the timeout

## 7.3 qevercloud::AuthenticationResult Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [AuthenticationResult](#) &other) const
- bool `operator!=` (const [AuthenticationResult](#) &other) const

### Public Attributes

- [Timestamp](#) `currentTime`
- [QString](#) `authenticationToken`
- [Timestamp](#) `expiration`
- [Optional](#)< [User](#) > `user`
- [Optional](#)< [PublicUserInfo](#) > `publicUserInfo`
- [Optional](#)< [QString](#) > `noteStoreUrl`
- [Optional](#)< [QString](#) > `webApiUrlPrefix`
- [Optional](#)< bool > `secondFactorRequired`
- [Optional](#)< [QString](#) > `secondFactorDeliveryHint`

### 7.3.1 Detailed Description

When an authentication (or re-authentication) is performed, this structure provides the result to the client.

### 7.3.2 Member Function Documentation

#### 7.3.2.1 `operator!=()`

```
bool qevercloud::AuthenticationResult::operator!= (
    const AuthenticationResult & other ) const [inline]
```

### 7.3.2.2 operator==( )

```
bool qevercloud::AuthenticationResult::operator==(
    const AuthenticationResult & other ) const [inline]
```

## 7.3.3 Member Data Documentation

### 7.3.3.1 authenticationToken

```
QString qevercloud::AuthenticationResult::authenticationToken
```

Holds an opaque, ASCII-encoded token that can be used by the client to perform actions on a [NoteStore](#).

### 7.3.3.2 currentTime

```
Timestamp qevercloud::AuthenticationResult::currentTime
```

The server-side date and time when this result was generated.

### 7.3.3.3 expiration

```
Timestamp qevercloud::AuthenticationResult::expiration
```

Holds the server-side date and time when the authentication token will expire. This time can be compared to "currentTime" to produce an expiration time that can be reconciled with the client's local clock.

### 7.3.3.4 noteStoreUrl

```
Optional< QString > qevercloud::AuthenticationResult::noteStoreUrl
```

This field will contain the full URL that clients should use to make [NoteStore](#) requests to the server shard that contains that user's data. I.e. this is the URL that should be used to create the Thrift HTTP client transport to send messages to the [NoteStore](#) service for the account.

### 7.3.3.5 publicUserInfo

```
Optional< PublicUserInfo > qevercloud::AuthenticationResult::publicUserInfo
```

If this authentication result was achieved without full permissions to access the full [User](#) structure, this field may be set to give back a more limited public set of data.

### 7.3.3.6 secondFactorDeliveryHint

```
Optional< QString > qevercloud::AuthenticationResult::secondFactorDeliveryHint
```

When secondFactorRequired is set to true, this field may contain a string describing the second factor delivery method that the user has configured. This will typically be an obfuscated mobile device number, such as "(xxx) xxx-x095". This string can be displayed to the user to remind them how to obtain the required second factor. TODO do we need to differentiate between SMS and voice delivery?

### 7.3.3.7 secondFactorRequired

```
Optional< bool > qevercloud::AuthenticationResult::secondFactorRequired
```

If set to true, this field indicates that the user has enabled two-factor authentication and must enter their second factor in order to complete authentication. In this case the value of authenticationResult will be a short-lived authentication token that may only be used to make a subsequent call to completeTwoFactorAuthentication.

### 7.3.3.8 user

```
Optional< User > qevercloud::AuthenticationResult::user
```

Holds the information about the account which was authenticated if this was a full authentication. May be absent if this particular authentication did not require user information.

### 7.3.3.9 webApiUrlPrefix

```
Optional< QString > qevercloud::AuthenticationResult::webApiUrlPrefix
```

This field will contain the initial part of the URLs that should be used to make requests to Evernote's thin client "web API", which provide optimized operations for clients that aren't capable of manipulating the full contents of accounts via the full Thrift data model. Clients should concatenate the relative path for the various servlets onto the end of this string to construct the full URL, as documented on our developer web site.

## 7.4 qevercloud::BootstrapInfo Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool operator== (const BootstrapInfo &other) const
- bool operator!= (const BootstrapInfo &other) const

### Public Attributes

- QList< BootstrapProfile > profiles

### 7.4.1 Detailed Description

This structure describes a collection of bootstrap profiles.

### 7.4.2 Member Function Documentation

#### 7.4.2.1 operator"!="()

```
bool qevercloud::BootstrapInfo::operator!= (
    const BootstrapInfo & other ) const [inline]
```

#### 7.4.2.2 operator==( )

```
bool qevercloud::BootstrapInfo::operator== (
    const BootstrapInfo & other ) const [inline]
```

### 7.4.3 Member Data Documentation

#### 7.4.3.1 profiles

```
QList< BootstrapProfile > qevercloud::BootstrapInfo::profiles
```

List of one or more bootstrap profiles, in descending preference order.

## 7.5 qevercloud::BootstrapProfile Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [BootstrapProfile](#) &other) const
- bool [operator!=](#) (const [BootstrapProfile](#) &other) const

### Public Attributes

- QString [name](#)
- [BootstrapSettings](#) [settings](#)

### 7.5.1 Detailed Description

This structure describes a collection of bootstrap settings.

### 7.5.2 Member Function Documentation

#### 7.5.2.1 operator!=(())

```
bool qevercloud::BootstrapProfile::operator!= (
    const BootstrapProfile & other ) const [inline]
```

#### 7.5.2.2 operator==(())

```
bool qevercloud::BootstrapProfile::operator== (
    const BootstrapProfile & other ) const [inline]
```

### 7.5.3 Member Data Documentation

#### 7.5.3.1 name

```
QString qevercloud::BootstrapProfile::name
```

The unique name of the profile, which is guaranteed to remain consistent across calls to getBootstrapInfo.

#### 7.5.3.2 settings

```
BootstrapSettings qevercloud::BootstrapProfile::settings
```

The settings for this profile.

## 7.6 qevercloud::BootstrapSettings Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [BootstrapSettings](#) &other) const
- bool [operator!=](#) (const [BootstrapSettings](#) &other) const

## Public Attributes

- QString [serviceHost](#)
- QString [marketingUrl](#)
- QString [supportUrl](#)
- QString [accountEmailDomain](#)
- Optional< bool > [enableFacebookSharing](#)
- Optional< bool > [enableGiftSubscriptions](#)
- Optional< bool > [enableSupportTickets](#)
- Optional< bool > [enableSharedNotebooks](#)
- Optional< bool > [enableSingleNoteSharing](#)
- Optional< bool > [enableSponsoredAccounts](#)
- Optional< bool > [enableTwitterSharing](#)
- Optional< bool > [enableLinkedInSharing](#)
- Optional< bool > [enablePublicNotebooks](#)

### 7.6.1 Detailed Description

This structure describes a collection of bootstrap settings.

### 7.6.2 Member Function Documentation

#### 7.6.2.1 operator"!="()

```
bool qevercloud::BootstrapSettings::operator!= (
    const BootstrapSettings & other ) const [inline]
```

#### 7.6.2.2 operator==( )

```
bool qevercloud::BootstrapSettings::operator== (
    const BootstrapSettings & other ) const [inline]
```

### 7.6.3 Member Data Documentation

#### 7.6.3.1 accountEmailDomain

```
QString qevercloud::BootstrapSettings::accountEmailDomain
```

The domain used for an Evernote user's incoming email address, which allows notes to be emailed into an account. E.g. m.evernote.com.

### 7.6.3.2 enableFacebookSharing

`Optional< bool > qevercloud::BootstrapSettings::enableFacebookSharing`

Whether the client application should enable sharing of notes on Facebook.

### 7.6.3.3 enableGiftSubscriptions

`Optional< bool > qevercloud::BootstrapSettings::enableGiftSubscriptions`

Whether the client application should enable gift subscriptions.

### 7.6.3.4 enableLinkedInSharing

`Optional< bool > qevercloud::BootstrapSettings::enableLinkedInSharing`

NOT DOCUMENTED

### 7.6.3.5 enablePublicNotebooks

`Optional< bool > qevercloud::BootstrapSettings::enablePublicNotebooks`

NOT DOCUMENTED

### 7.6.3.6 enableSharedNotebooks

`Optional< bool > qevercloud::BootstrapSettings::enableSharedNotebooks`

Whether the client application should enable shared notebooks.

### 7.6.3.7 enableSingleNoteSharing

`Optional< bool > qevercloud::BootstrapSettings::enableSingleNoteSharing`

Whether the client application should enable single note sharing.

### 7.6.3.8 enableSponsoredAccounts

`Optional< bool > qevercloud::BootstrapSettings::enableSponsoredAccounts`

Whether the client application should enable sponsored accounts.

### 7.6.3.9 enableSupportTickets

`Optional< bool > qevercloud::BootstrapSettings::enableSupportTickets`

Whether the client application should enable in-client creation of support tickets.



### 7.6.3.10 enableTwitterSharing

`Optional< bool > qevercloud::BootstrapSettings::enableTwitterSharing`

Whether the client application should enable sharing of notes on Twitter.

### 7.6.3.11 marketingUrl

`QString qevercloud::BootstrapSettings::marketingUrl`

The URL stem for the Evernote corporate marketing website, e.g. <http://www.evernote.com>. This stem can be used to compose website URLs. For example, the URL of the Evernote Trunk is composed by appending `"/about/trunk/"` to the value of `marketingUrl`.

### 7.6.3.12 serviceHost

`QString qevercloud::BootstrapSettings::serviceHost`

The hostname and optional port for composing Evernote web service URLs. This URL can be used to access the [UserStore](#) and related services, but must not be used to compose the [NoteStore](#) URL. Client applications must handle `serviceHost` values that include only the hostname (e.g. `www.evernote.com`) or both the hostname and port (e.g. `www.evernote.com:8080`). If no port is specified, or if port 443 is specified, client applications must use the scheme `"https"` when composing URLs. Otherwise, a client must use the scheme `"http"`.

### 7.6.3.13 supportUrl

`QString qevercloud::BootstrapSettings::supportUrl`

The full URL for the Evernote customer support website, e.g. <https://support.evernote.com>.

## 7.7 qevercloud::BusinessNotebook Struct Reference

```
#include <types.h>
```

### Public Member Functions

- `bool operator==` (const [BusinessNotebook](#) &other) const
- `bool operator!=` (const [BusinessNotebook](#) &other) const

### Public Attributes

- `Optional< QString > notebookDescription`
- `Optional< SharedNotebookPrivilegeLevel::type > privilege`
- `Optional< bool > recommended`

### 7.7.1 Detailed Description

If a [Notebook](#) contained in an Evernote Business account has been published to the business library, the [Notebook](#) will have a reference to one of these structures, which specifies how the [Notebook](#) will be represented in the library.

### 7.7.2 Member Function Documentation

#### 7.7.2.1 `operator!=(())`

```
bool qevercloud::BusinessNotebook::operator!= (
    const BusinessNotebook & other ) const [inline]
```

#### 7.7.2.2 `operator==(())`

```
bool qevercloud::BusinessNotebook::operator== (
    const BusinessNotebook & other ) const [inline]
```

### 7.7.3 Member Data Documentation

#### 7.7.3.1 `notebookDescription`

```
Optional< QString > qevercloud::BusinessNotebook::notebookDescription
```

A short description of the notebook's content that will be displayed in the business library user interface. The description may not begin or end with whitespace.

Length: EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MIN - EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MAX

Regex: EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_REGEX

#### 7.7.3.2 `privilege`

```
Optional< SharedNotebookPrivilegeLevel::type > qevercloud::BusinessNotebook::privilege
```

The privileges that will be granted to users who join the notebook through the business library.

#### 7.7.3.3 `recommended`

```
Optional< bool > qevercloud::BusinessNotebook::recommended
```

Whether the notebook should be "recommended" when displayed in the business library user interface.

## 7.8 qevercloud::BusinessUserInfo Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [BusinessUserInfo](#) &other) const
- bool `operator!=` (const [BusinessUserInfo](#) &other) const

### Public Attributes

- [Optional](#)< qint32 > `businessId`
- [Optional](#)< QString > `businessName`
- [Optional](#)< [BusinessUserRole::type](#) > `role`
- [Optional](#)< QString > `email`

### 7.8.1 Detailed Description

This structure is used to provide information about an Evernote Business membership, for members who are part of a business.

### 7.8.2 Member Function Documentation

#### 7.8.2.1 `operator"!=()`

```
bool qevercloud::BusinessUserInfo::operator!= (
    const BusinessUserInfo & other ) const [inline]
```

#### 7.8.2.2 `operator==(`

```
bool qevercloud::BusinessUserInfo::operator== (
    const BusinessUserInfo & other ) const [inline]
```

### 7.8.3 Member Data Documentation

### 7.8.3.1 businessId

`Optional< qint32 > qevercloud::BusinessUserInfo::businessId`

The ID of the Evernote Business account that the user is a member of.

`businessName` The human-readable name of the Evernote Business account that the user is a member of.

### 7.8.3.2 businessName

`Optional< QString > qevercloud::BusinessUserInfo::businessName`

NOT DOCUMENTED

### 7.8.3.3 email

`Optional< QString > qevercloud::BusinessUserInfo::email`

An e-mail address that will be used by the service in the context of your Evernote Business activities. For example, this e-mail address will be used when you e-mail a business note, when you update notes in the account of your business, etc. The business e-mail cannot be used for identification purposes such as for logging into the service.

### 7.8.3.4 role

`Optional< BusinessUserRole::type > qevercloud::BusinessUserInfo::role`

The role of the user within the Evernote Business account that they are a member of.

## 7.9 qevercloud::BusinessUserRole Struct Reference

```
#include <types.h>
```

### Public Types

- enum `type` { `ADMIN` = 1, `NORMAL` = 2 }

### 7.9.1 Detailed Description

Enumeration of the roles that a `User` can have within an Evernote Business account.

`ADMIN`: The user is an administrator of the Evernote Business account.

`NORMAL`: The user is a regular user within the Evernote Business account.

### 7.9.2 Member Enumeration Documentation

#### 7.9.2.1 type

```
enum qevercloud::BusinessUserRole::type
```

## Enumerator

ADMIN	
NORMAL	

## 7.10 qevercloud::ClientUsageMetrics Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [ClientUsageMetrics](#) &other) const
- bool [operator!=](#) (const [ClientUsageMetrics](#) &other) const

### Public Attributes

- [Optional](#)< quint32 > [sessions](#)

#### 7.10.1 Detailed Description

This structure is passed from clients to the Evernote service when they wish to relay coarse-grained usage metrics to the service to help improve products.

#### 7.10.2 Member Function Documentation

##### 7.10.2.1 [operator"!=\(\)](#)

```
bool qevercloud::ClientUsageMetrics::operator!= (
    const ClientUsageMetrics & other ) const [inline]
```

##### 7.10.2.2 [operator==\(\)](#)

```
bool qevercloud::ClientUsageMetrics::operator== (
    const ClientUsageMetrics & other ) const [inline]
```

#### 7.10.3 Member Data Documentation

### 7.10.3.1 sessions

```
Optional< qint32 > qevercloud::ClientUsageMetrics::sessions
```

This field contains a count of the number of usage "sessions" that have occurred with this client which have not previously been reported to the service. A "session" is defined as one of the 96 fifteen-minute intervals of the day when someone used Evernote's interface at least once. So if a user interacts with an Evernote client at 12:18, 12:24, and 12:36, and then the client synchronizes at 12:39, it would report that there were two previously-unreported sessions (one session for the 12:15-12:30 time period, and one for the 12:30-12:45 period). If the user used Evernote again at 12:41 and synchronized at 12:43, it would not report any new sessions, because the 12:30-12:45 session had already been reported.

## 7.11 qevercloud::Data Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [Data](#) &other) const
- bool `operator!=` (const [Data](#) &other) const

### Public Attributes

- [Optional](#)< QByteArray > `bodyHash`
- [Optional](#)< qint32 > `size`
- [Optional](#)< QByteArray > `body`

### 7.11.1 Detailed Description

In several places, EDAM exchanges blocks of bytes of data for a component which may be relatively large. For example: the contents of a clipped HTML note, the bytes of an embedded image, or the recognition XML for a large image. This structure is used in the protocol to represent any of those large blocks of data when they are transmitted or when they are only referenced their metadata.

### 7.11.2 Member Function Documentation

#### 7.11.2.1 `operator!=()`

```
bool qevercloud::Data::operator!= (
    const Data & other ) const [inline]
```

### 7.11.2.2 operator==( )

```
bool qevercloud::Data::operator== (
    const Data & other ) const [inline]
```

## 7.11.3 Member Data Documentation

### 7.11.3.1 body

```
Optional< QByteArray > qevercloud::Data::body
```

This field is set to contain the binary contents of the data whenever the resource is being transferred. If only metadata is being exchanged, this field will be empty. For example, a client could notify the service about the change to an attribute for a resource without transmitting the binary resource contents.

### 7.11.3.2 bodyHash

```
Optional< QByteArray > qevercloud::Data::bodyHash
```

This field carries a one-way hash of the contents of the data body, in binary form. The hash function is MD5  
Length: EDAM\_HASH\_LEN (exactly)

### 7.11.3.3 size

```
Optional< qint32 > qevercloud::Data::size
```

The length, in bytes, of the data body.

## 7.12 qevercloud::EDAMErrorCode Struct Reference

```
#include <EDAMErrorCode.h>
```

### Public Types

- enum `type` {  
`UNKNOWN` = 1, `BAD_DATA_FORMAT` = 2, `PERMISSION_DENIED` = 3, `INTERNAL_ERROR` = 4,  
`DATA_REQUIRED` = 5, `LIMIT_REACHED` = 6, `QUOTA_REACHED` = 7, `INVALID_AUTH` = 8,  
`AUTH_EXPIRED` = 9, `DATA_CONFLICT` = 10, `ENML_VALIDATION` = 11, `SHARD_UNAVAILABLE` = 12,  
`LEN_TOO_SHORT` = 13, `LEN_TOO_LONG` = 14, `TOO_FEW` = 15, `TOO_MANY` = 16,  
`UNSUPPORTED_OPERATION` = 17, `TAKEN_DOWN` = 18, `RATE_LIMIT_REACHED` = 19 }

### 7.12.1 Detailed Description

Numeric codes indicating the type of error that occurred on the service.

**UNKNOWN** No information available about the error

**BAD\_DATA\_FORMAT** The format of the request data was incorrect

**PERMISSION\_DENIED** Not permitted to perform action

**INTERNAL\_ERROR** Unexpected problem with the service

**DATA\_REQUIRED** A required parameter/field was absent

**LIMIT\_REACHED** Operation denied due to data model limit

**QUOTA\_REACHED** Operation denied due to user storage limit

**INVALID\_AUTH** Username and/or password incorrect

**AUTH\_EXPIRED** Authentication token expired

**DATA\_CONFLICT** Change denied due to data model conflict

**ENML\_VALIDATION** Content of submitted note was malformed

**SHARD\_UNAVAILABLE** Service shard with account data is temporarily down

**LEN\_TOO\_SHORT** Operation denied due to data model limit, where something such as a string length was too short

**LEN\_TOO\_LONG** Operation denied due to data model limit, where something such as a string length was too long

**TOO\_FEW** Operation denied due to data model limit, where there were too few of something.

**TOO\_MANY** Operation denied due to data model limit, where there were too many of something.

**UNSUPPORTED\_OPERATION** Operation denied because it is currently unsupported.

**TAKEN\_DOWN** Operation denied because access to the corresponding object is prohibited in response to a take-down notice.

**RATE\_LIMIT\_REACHED** Operation denied because the calling application has reached its hourly API call limit for this user.

### 7.12.2 Member Enumeration Documentation



## Enumerator

---

### 7.12.2.1 type

```
enum qevercloud::EDAMErrorCode::type
```

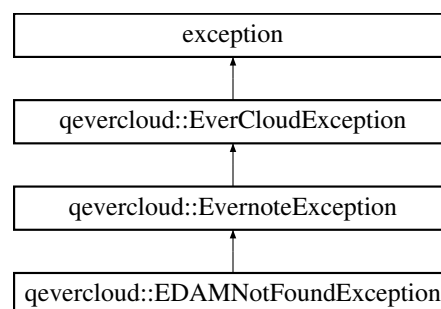
## Enumerator

UNKNOWN	
BAD_DATA_FORMAT	
PERMISSION_DENIED	
INTERNAL_ERROR	
DATA_REQUIRED	
LIMIT_REACHED	
QUOTA_REACHED	
INVALID_AUTH	
AUTH_EXPIRED	
DATA_CONFLICT	
ENML_VALIDATION	
SHARD_UNAVAILABLE	
LEN_TOO_SHORT	
LEN_TOO_LONG	
TOO_FEW	
TOO_MANY	
UNSUPPORTED_OPERATION	
TAKEN_DOWN	
RATE_LIMIT_REACHED	

## 7.13 qevercloud::EDAMNotFoundException Class Reference

```
#include <types.h>
```

Inheritance diagram for qevercloud::EDAMNotFoundException:



## Public Member Functions

- [EDAMNotFoundException](#) ()
- [~EDAMNotFoundException](#) () throw ()
- const char \* [what](#) () const Q\_DECL\_OVERRIDE throw ()
- virtual QSharedPointer< [EverCloudExceptionData](#) > [exceptionData](#) () const Q\_DECL\_OVERRIDE

## Public Attributes

- [Optional](#)< QString > [identifier](#)
- [Optional](#)< QString > [key](#)

## Additional Inherited Members

### 7.13.1 Detailed Description

This exception is thrown by EDAM procedures when a caller asks to perform an operation on an object that does not exist. This may be thrown based on an invalid primary identifier (e.g. a bad GUID), or when the caller refers to an object by another unique identifier (e.g. a [User](#)'s email address).

**identifier:** A description of the object that was not found on the server. For example, "Note.notebookGuid" when a caller attempts to create a note in a notebook that does not exist in the user's account.

**key:** The value passed from the client in the identifier, which was not found. For example, the GUID that was not found.

### 7.13.2 Constructor & Destructor Documentation

#### 7.13.2.1 EDAMNotFoundException()

```
qevercloud::EDAMNotFoundException::EDAMNotFoundException ( ) [inline]
```

#### 7.13.2.2 ~EDAMNotFoundException()

```
qevercloud::EDAMNotFoundException::~~EDAMNotFoundException ( ) throw ( ) [inline]
```

### 7.13.3 Member Function Documentation

#### 7.13.3.1 exceptionData()

```
virtual QSharedPointer<EverCloudExceptionData> qevercloud::EDAMNotFoundException::exceptionData ( ) const [virtual]
```

Reimplemented from [qevercloud::EvernoteException](#).

### 7.13.3.2 what()

```
const char* qevercloud::EDAMNotFoundException::what ( ) const throw ( )
```

## 7.13.4 Member Data Documentation

### 7.13.4.1 identifier

```
Optional< QString > qevercloud::EDAMNotFoundException::identifier
```

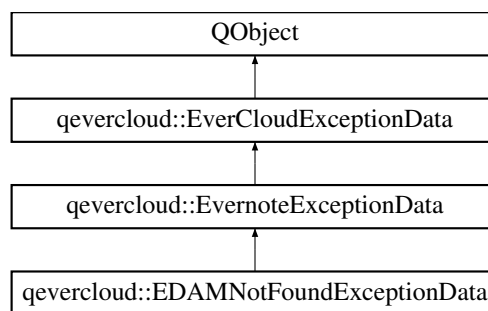
### 7.13.4.2 key

```
Optional< QString > qevercloud::EDAMNotFoundException::key
```

## 7.14 qevercloud::EDAMNotFoundExceptionData Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::EDAMNotFoundExceptionData:



### Public Member Functions

- [EDAMNotFoundExceptionData](#) (QString error, [Optional< QString >](#) identifier, [Optional< QString >](#) key)
- virtual void [throwException](#) () const Q\_DECL\_OVERRIDE

### Protected Attributes

- [Optional< QString >](#) [m\\_identifier](#)
- [Optional< QString >](#) [m\\_key](#)

## Additional Inherited Members

### 7.14.1 Detailed Description

Asynchronous API counterpart of [EDAMNotFoundException](#). See [EverCloudExceptionData](#) for more details.

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 EDAMNotFoundExceptionData()

```
qevercloud::EDAMNotFoundExceptionData::EDAMNotFoundExceptionData (
    QString error,
    Optional< QString > identifier,
    Optional< QString > key ) [explicit]
```

### 7.14.3 Member Function Documentation

#### 7.14.3.1 throwException()

```
virtual void qevercloud::EDAMNotFoundExceptionData::throwException ( ) const [virtual]
```

If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented from [qevercloud::EvernoteExceptionData](#).

### 7.14.4 Member Data Documentation

#### 7.14.4.1 m\_identifier

```
Optional<QString> qevercloud::EDAMNotFoundExceptionData::m_identifier [protected]
```

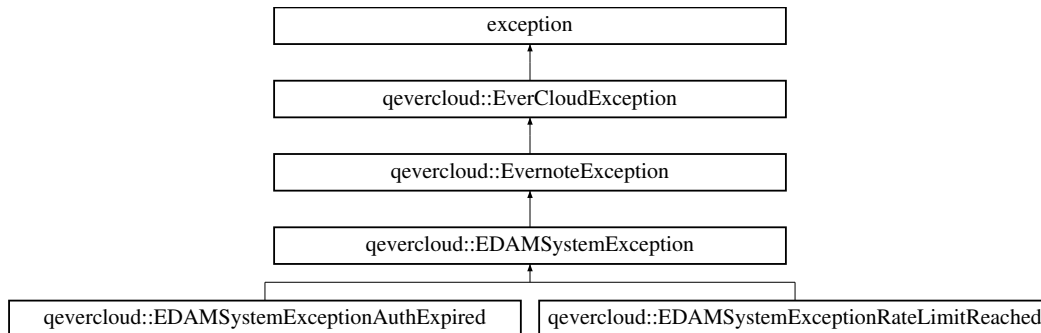
#### 7.14.4.2 m\_key

```
Optional<QString> qevercloud::EDAMNotFoundExceptionData::m_key [protected]
```

## 7.15 qevercloud::EDAMSystemException Class Reference

```
#include <types.h>
```

Inheritance diagram for qevercloud::EDAMSystemException:



### Public Member Functions

- [EDAMSystemException](#) ()
- [~EDAMSystemException](#) () throw ()
- const char \* [what](#) () const Q\_DECL\_OVERRIDE throw ()
- virtual QSharedPointer< [EverCloudExceptionData](#) > [exceptionData](#) () const Q\_DECL\_OVERRIDE

### Public Attributes

- [EDAMErrorCode::type](#) [errorCode](#)
- [Optional](#)< QString > [message](#)
- [Optional](#)< qint32 > [rateLimitDuration](#)

### Additional Inherited Members

#### 7.15.1 Detailed Description

This exception is thrown by EDAM procedures when a call fails as a result of a problem in the service that could not be changed through caller action.

**errorCode:** The numeric code indicating the type of error that occurred. must be one of the values of [EDAMErrorCode](#).

**message:** This may contain additional information about the error

**rateLimitDuration:** Indicates the minimum number of seconds that an application should expect subsequent API calls for this user to fail. The application should not retry API requests for the user until at least this many seconds have passed. Present only when **errorCode** is **RATE\_LIMIT\_REACHED**,

#### 7.15.2 Constructor & Destructor Documentation

### 7.15.2.1 EDAMSystemException()

```
qevercloud::EDAMSystemException::EDAMSystemException ( ) [inline]
```

### 7.15.2.2 ~EDAMSystemException()

```
qevercloud::EDAMSystemException::~~EDAMSystemException ( ) throw ( ) [inline]
```

## 7.15.3 Member Function Documentation

### 7.15.3.1 exceptionData()

```
virtual QSharedPointer<EverCloudExceptionData> qevercloud::EDAMSystemException::exceptionData  
( ) const [virtual]
```

Reimplemented from [qevercloud::EvernoteException](#).

Reimplemented in [qevercloud::EDAMSystemExceptionAuthExpired](#), and [qevercloud::EDAMSystemExceptionRateLimitReached](#).

### 7.15.3.2 what()

```
const char* qevercloud::EDAMSystemException::what ( ) const throw ( )
```

## 7.15.4 Member Data Documentation

### 7.15.4.1 errorCode

```
EDAMErrorCode::type qevercloud::EDAMSystemException::errorCode
```

### 7.15.4.2 message

```
Optional< QString > qevercloud::EDAMSystemException::message
```

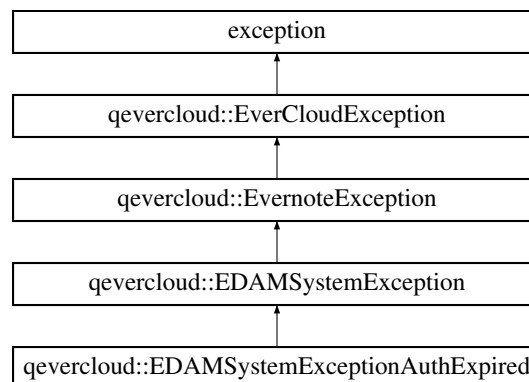
### 7.15.4.3 rateLimitDuration

`Optional< quint32 > qevercloud::EDAMSystemException::rateLimitDuration`

## 7.16 qevercloud::EDAMSystemExceptionAuthExpired Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionAuthExpired:



### Public Member Functions

- virtual `QSharedPointer< EverCloudExceptionData > exceptionData ()` const `Q_DECL_OVERRIDE`

### Additional Inherited Members

#### 7.16.1 Detailed Description

[EDAMSystemException](#) for `errorCode = AUTH_EXPIRED`

#### 7.16.2 Member Function Documentation

##### 7.16.2.1 exceptionData()

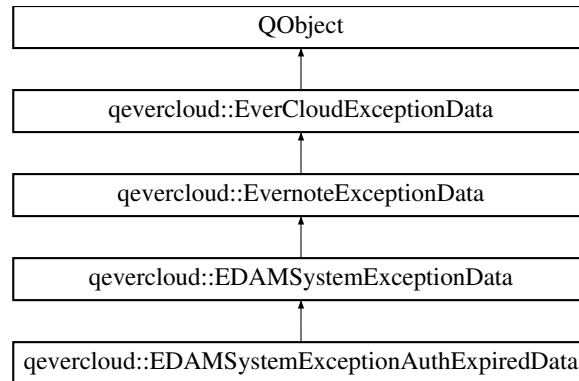
```
virtual QSharedPointer<EverCloudExceptionData> qevercloud::EDAMSystemExceptionAuthExpired←  
::exceptionData ( ) const [virtual]
```

Reimplemented from [qevercloud::EDAMSystemException](#).

## 7.17 qevercloud::EDAMSystemExceptionAuthExpiredData Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionAuthExpiredData:



### Public Member Functions

- [EDAMSystemExceptionAuthExpiredData](#) (QString error, [EDAMErrorCode::type](#) errorCode, [Optional](#)< Q↔String > message, [Optional](#)< qint32 > rateLimitDuration)
- virtual void [throwException](#) () const Q\_DECL\_OVERRIDE

### Additional Inherited Members

#### 7.17.1 Detailed Description

Asynchronous API counterpart of [EDAMSystemExceptionAuthExpired](#). See [EverCloudExceptionData](#) for more details.

#### 7.17.2 Constructor & Destructor Documentation

##### 7.17.2.1 EDAMSystemExceptionAuthExpiredData()

```

qevercloud::EDAMSystemExceptionAuthExpiredData::EDAMSystemExceptionAuthExpiredData (
    QString error,
    EDAMErrorCode::type errorCode,
    Optional< QString > message,
    Optional< qint32 > rateLimitDuration ) [explicit]
  
```

#### 7.17.3 Member Function Documentation



### 7.17.3.1 throwException()

```
virtual void qevercloud::EDAMSystemExceptionAuthExpiredData::throwException ( ) const [virtual]
```

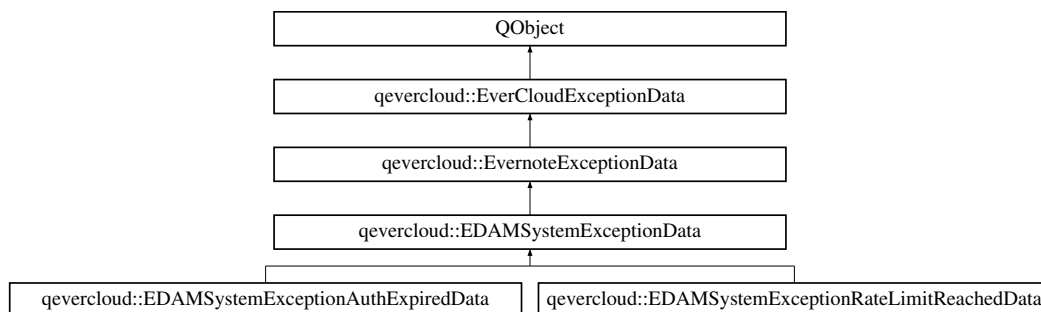
If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented from [qevercloud::EDAMSystemExceptionData](#).

## 7.18 qevercloud::EDAMSystemExceptionData Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for `qevercloud::EDAMSystemExceptionData`:



### Public Member Functions

- [EDAMSystemExceptionData](#) (QString err, [EDAMErrorCode::type](#) errorCode, [Optional](#)< QString > message, [Optional](#)< qint32 > rateLimitDuration)
- virtual void [throwException](#) () const Q\_DECL\_OVERRIDE

### Protected Attributes

- [EDAMErrorCode::type](#) m\_errorCode
- [Optional](#)< QString > m\_message
- [Optional](#)< qint32 > m\_rateLimitDuration

### Additional Inherited Members

#### 7.18.1 Detailed Description

Asynchronous API counterpart of [EDAMSystemException](#). See [EverCloudExceptionData](#) for more details.

#### 7.18.2 Constructor & Destructor Documentation

### 7.18.2.1 EDAMSystemExceptionData()

```
qevercloud::EDAMSystemExceptionData::EDAMSystemExceptionData (
    QString err,
    EDAMErrorCode::type errorCode,
    Optional< QString > message,
    Optional< qint32 > rateLimitDuration ) [explicit]
```

## 7.18.3 Member Function Documentation

### 7.18.3.1 throwException()

```
virtual void qevercloud::EDAMSystemExceptionData::throwException ( ) const [virtual]
```

If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented from [qevercloud::EvernoteExceptionData](#).

Reimplemented in [qevercloud::EDAMSystemExceptionAuthExpiredData](#), and [qevercloud::EDAMSystemExceptionRateLimitReachedData](#).

## 7.18.4 Member Data Documentation

### 7.18.4.1 m\_errorCode

```
EDAMErrorCode::type qevercloud::EDAMSystemExceptionData::m_errorCode [protected]
```

### 7.18.4.2 m\_message

```
Optional<QString> qevercloud::EDAMSystemExceptionData::m_message [protected]
```

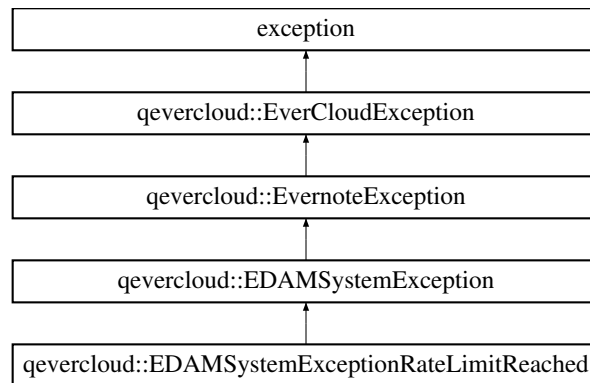
### 7.18.4.3 m\_rateLimitDuration

```
Optional<qint32> qevercloud::EDAMSystemExceptionData::m_rateLimitDuration [protected]
```

## 7.19 qevercloud::EDAMSystemExceptionRateLimitReached Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionRateLimitReached:



### Public Member Functions

- virtual QSharedPointer< [EverCloudExceptionData](#) > [exceptionData](#) () const Q\_DECL\_OVERRIDE

### Additional Inherited Members

#### 7.19.1 Detailed Description

[EDAMSystemException](#) for `errorCode = RATE_LIMIT_REACHED`

#### 7.19.2 Member Function Documentation

##### 7.19.2.1 exceptionData()

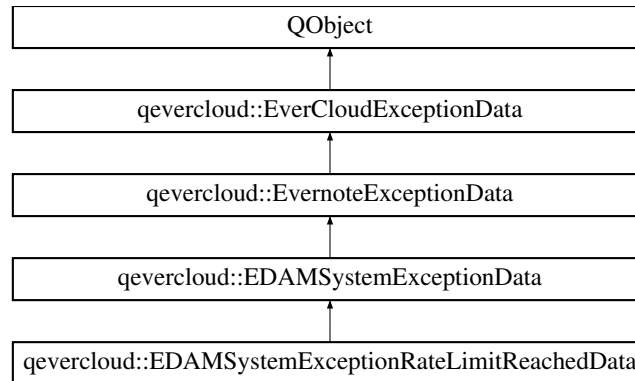
```
virtual QSharedPointer<EverCloudExceptionData> qevercloud::EDAMSystemExceptionRateLimitReached::exceptionData ( ) const [virtual]
```

Reimplemented from [qevercloud::EDAMSystemException](#).

## 7.20 qevercloud::EDAMSystemExceptionRateLimitReachedData Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionRateLimitReachedData:



### Public Member Functions

- [EDAMSystemExceptionRateLimitReachedData](#) (QString error, [EDAMErrorCode::type](#) errorCode, [Optional](#)< QString > message, [Optional](#)< qint32 > rateLimitDuration)
- virtual void [throwException](#) () const Q\_DECL\_OVERRIDE

### Additional Inherited Members

#### 7.20.1 Detailed Description

Asynchronous API counterpart of [EDAMSystemExceptionRateLimitReached](#). See [EverCloudExceptionData](#) for more details.

#### 7.20.2 Constructor & Destructor Documentation

##### 7.20.2.1 EDAMSystemExceptionRateLimitReachedData()

```

qevercloud::EDAMSystemExceptionRateLimitReachedData::EDAMSystemExceptionRateLimitReachedData (
    QString error,
    EDAMErrorCode::type errorCode,
    Optional< QString > message,
    Optional< qint32 > rateLimitDuration ) [explicit]
  
```

#### 7.20.3 Member Function Documentation

### 7.20.3.1 throwException()

```
virtual void qevercloud::EDAMSystemExceptionRateLimitReachedData::throwException ( ) const
[virtual]
```

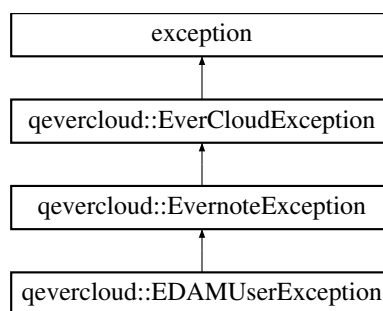
If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented from [qevercloud::EDAMSystemExceptionData](#).

## 7.21 qevercloud::EDAMUserException Class Reference

```
#include <types.h>
```

Inheritance diagram for qevercloud::EDAMUserException:



### Public Member Functions

- [EDAMUserException](#) ()
- [~EDAMUserException](#) () throw ()
- const char \* [what](#) () const Q\_DECL\_OVERRIDE throw ()
- virtual QSharedPointer< [EverCloudExceptionData](#) > [exceptionData](#) () const Q\_DECL\_OVERRIDE

### Public Attributes

- [EDAMErrorCode::type](#) [errorCode](#)
- [Optional](#)< QString > [parameter](#)

### Additional Inherited Members

#### 7.21.1 Detailed Description

This exception is thrown by EDAM procedures when a call fails as a result of a problem that a caller may be able to resolve. For example, if the user attempts to add a note to their account which would exceed their storage quota, this type of exception may be thrown to indicate the source of the error so that they can choose an alternate action.

This exception would not be used for internal system errors that do not reflect user actions, but rather reflect a problem within the service that the user cannot resolve.

**errorCode:** The numeric code indicating the type of error that occurred. must be one of the values of [EDAMErrorCode](#).

**parameter:** If the error applied to a particular input parameter, this will indicate which parameter.

## 7.21.2 Constructor & Destructor Documentation

### 7.21.2.1 EDAMUserException()

```
qevercloud::EDAMUserException::EDAMUserException ( ) [inline]
```

### 7.21.2.2 ~EDAMUserException()

```
qevercloud::EDAMUserException::~~EDAMUserException ( ) throw ( ) [inline]
```

## 7.21.3 Member Function Documentation

### 7.21.3.1 exceptionData()

```
virtual QSharedPointer<EverCloudExceptionData> qevercloud::EDAMUserException::exceptionData (
) const [virtual]
```

Reimplemented from [qevercloud::EvernoteException](#).

### 7.21.3.2 what()

```
const char* qevercloud::EDAMUserException::what ( ) const throw ( )
```

## 7.21.4 Member Data Documentation

### 7.21.4.1 errorCode

```
EDAMErrorCode::type qevercloud::EDAMUserException::errorCode
```

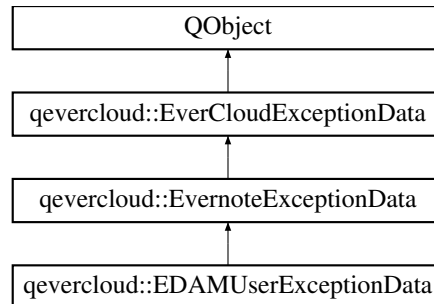
### 7.21.4.2 parameter

```
Optional< QString > qevercloud::EDAMUserException::parameter
```

## 7.22 qevercloud::EDAMUserExceptionData Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::EDAMUserExceptionData:



### Public Member Functions

- [EDAMUserExceptionData](#) (QString error, [EDAMErrorCode::type](#) errorCode, [Optional](#)< QString > parameter)
- virtual void [throwException](#) () const Q\_DECL\_OVERRIDE

### Protected Attributes

- [EDAMErrorCode::type](#) m\_errorCode
- [Optional](#)< QString > m\_parameter

### Additional Inherited Members

#### 7.22.1 Detailed Description

Asynchronous API counterpart of [EDAMUserException](#). See [EverCloudExceptionData](#) for more details.

#### 7.22.2 Constructor & Destructor Documentation

##### 7.22.2.1 EDAMUserExceptionData()

```
qevercloud::EDAMUserExceptionData::EDAMUserExceptionData (
    QString error,
    EDAMErrorCode::type errorCode,
    Optional< QString > parameter ) [explicit]
```

#### 7.22.3 Member Function Documentation

### 7.22.3.1 `throwException()`

```
virtual void qevercloud::EDAMUserExceptionData::throwException ( ) const [virtual]
```

If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented from [qevercloud::EvernoteExceptionData](#).

## 7.22.4 Member Data Documentation

### 7.22.4.1 `m_errorCode`

```
EDAMErrorCode::type qevercloud::EDAMUserExceptionData::m_errorCode [protected]
```

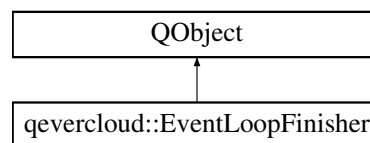
### 7.22.4.2 `m_parameter`

```
Optional<QString> qevercloud::EDAMUserExceptionData::m_parameter [protected]
```

## 7.23 `qevercloud::EventLoopFinisher` Class Reference

```
#include <EventLoopFinisher.h>
```

Inheritance diagram for `qevercloud::EventLoopFinisher`:



### Public Slots

- void [stopEventLoop](#) ()

### Public Member Functions

- [EventLoopFinisher](#) (QEventLoop \*loop, int exitCode, QObject \*parent=Q\_NULLPTR)
- [~EventLoopFinisher](#) ()



## 7.23.1 Constructor & Destructor Documentation

### 7.23.1.1 EventLoopFinisher()

```
qevercloud::EventLoopFinisher::EventLoopFinisher (
    QEventLoop * loop,
    int exitCode,
    QObject * parent = Q_NULLPTR ) [explicit]
```

### 7.23.1.2 ~EventLoopFinisher()

```
qevercloud::EventLoopFinisher::~~EventLoopFinisher ( )
```

## 7.23.2 Member Function Documentation

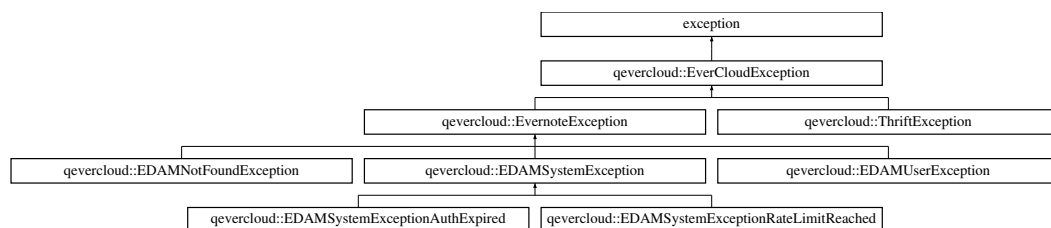
### 7.23.2.1 stopEventLoop

```
void qevercloud::EventLoopFinisher::stopEventLoop ( ) [slot]
```

## 7.24 qevercloud::EverCloudException Class Reference

```
#include <EverCloudException.h>
```

Inheritance diagram for qevercloud::EverCloudException:



## Public Member Functions

- [EverCloudException](#) ()
- [EverCloudException](#) (QString error)
- [EverCloudException](#) (const std::string &error)
- [EverCloudException](#) (const char \*error)
- [~EverCloudException](#) () throw ()
- const char \* [what](#) () const throw ()
- virtual QSharedPointer< [EverCloudExceptionData](#) > [exceptionData](#) () const

## Protected Attributes

- QByteArray [m\\_error](#)

### 7.24.1 Detailed Description

All exceptions throws by the library are of this class or its descendants.

### 7.24.2 Constructor & Destructor Documentation

#### 7.24.2.1 EverCloudException() [1/4]

```
qevercloud::EverCloudException::EverCloudException ( ) [explicit]
```

#### 7.24.2.2 EverCloudException() [2/4]

```
qevercloud::EverCloudException::EverCloudException (
    QString error ) [explicit]
```

#### 7.24.2.3 EverCloudException() [3/4]

```
qevercloud::EverCloudException::EverCloudException (
    const std::string & error ) [explicit]
```

#### 7.24.2.4 EverCloudException() [4/4]

```
qevercloud::EverCloudException::EverCloudException (
    const char * error ) [explicit]
```

#### 7.24.2.5 ~EverCloudException()

```
qevercloud::EverCloudException::~EverCloudException ( ) throw ( )
```

### 7.24.3 Member Function Documentation

#### 7.24.3.1 exceptionData()

```
virtual QSharedPointer<EverCloudExceptionData> qevercloud::EverCloudException::exceptionData (
) const [virtual]
```

Reimplemented in [qevercloud::EDAMNotFoundException](#), [qevercloud::EDAMSystemException](#), [qevercloud::EDAMUserException](#), [qevercloud::EDAMSystemExceptionAuthExpired](#), [qevercloud::EDAMSystemExceptionRateLimitReached](#), [qevercloud::EvernoteException](#) and [qevercloud::ThriftException](#).

#### 7.24.3.2 what()

```
const char* qevercloud::EverCloudException::what ( ) const throw ( )
```

### 7.24.4 Member Data Documentation

#### 7.24.4.1 m\_error

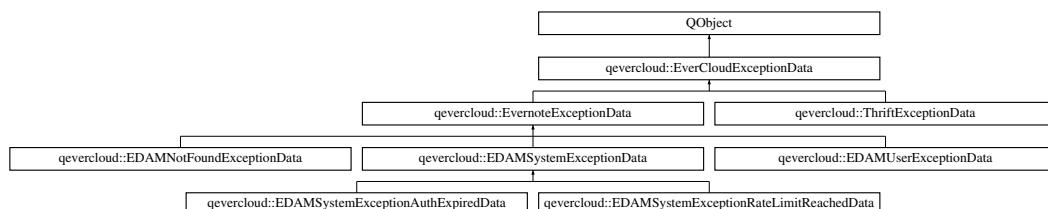
```
QByteArray qevercloud::EverCloudException::m_error [mutable], [protected]
```

## 7.25 qevercloud::EverCloudExceptionData Class Reference

[EverCloudException](#) counterpart for asynchronous API.

```
#include <EverCloudException.h>
```

Inheritance diagram for `qevercloud::EverCloudExceptionData`:



### Public Member Functions

- [EverCloudExceptionData](#) (QString error)
- virtual void [throwException](#) ( ) const

## Public Attributes

- QString [errorMessage](#)

### 7.25.1 Detailed Description

[EverCloudException](#) counterpart for asynchronous API.

Asynchronous functions cannot throw exceptions so descendants of [EverCloudExceptionData](#) are returned instead in case of an error. Every exception class has its own counterpart. The [EverCloudExceptionData](#) descendants hierarchy is a copy of the [EverCloudException](#) descendants hierarchy.

The main reason not to use exception classes directly is that `dynamic_cast` does not work across module (exe, dll, etc) boundaries in general, while `qobject_cast` do work as expected. That's why I decided to inherit my error classes from `QObject`.

In general error checking in asynchronous API look like this:

```
NoteStore* ns;
...
QObject::connect(ns->getNotebook(notebookGuid), &AsyncResult::finished, [](QVariant result,
    QSharedPointer<EverCloudExceptionData> error) {
    if(!error.isNull()) {
        QSharedPointer<EDAMNotFoundExceptionData> errorNotFound =
            error.objectCast<EDAMNotFoundExceptionData>();
        QSharedPointer<EDAMUserExceptionData> errorUser = error.objectCast<EDAMUserExceptionData>();
        QSharedPointer<EDAMSystemExceptionData> errorSystem = error.objectCast<EDAMSystemExceptionData>();
        if(!errorNotFound.isNull()) {
            qDebug() << "notebook not found" << errorNotFound.identifier << errorNotFound.key;
        } else if(!errorUser.isNull()) {
            qDebug() << errorUser.errorMessage;
        } else if(!errorSystem.isNull()) {
            if(errorSystem.errorCode == EDAMErrorCode::RATE_LIMIT_REACHED) {
                qDebug() << "Evernote API rate limits are reached";
            } else if(errorSystem.errorCode == EDAMErrorCode::AUTH_EXPIRED) {
                qDebug() << "Authorization token is inspired";
            } else {
                // some other Evernote trouble
                qDebug() << errorSystem.errorMessage;
            }
        } else {
            // some unexpected error
            qDebug() << error.errorMessage;
        }
    } else {
        // success
    }
});
```

### 7.25.2 Constructor & Destructor Documentation

#### 7.25.2.1 EverCloudExceptionData()

```
qevercloud::EverCloudExceptionData::EverCloudExceptionData (
    QString error ) [explicit]
```

### 7.25.3 Member Function Documentation

### 7.25.3.1 throwException()

```
virtual void qevercloud::EverCloudExceptionData::throwException ( ) const [virtual]
```

If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented in [qevercloud::EDAMSystemExceptionAuthExpiredData](#), [qevercloud::EDAMSystemExceptionRateLimitReachedData](#), [qevercloud::EvernoteExceptionData](#), [qevercloud::EDAMNotFoundExceptionData](#), [qevercloud::EDAMSystemExceptionData](#), [qevercloud::EDAMUserExceptionData](#), and [qevercloud::ThriftExceptionData](#).

## 7.25.4 Member Data Documentation

### 7.25.4.1 errorMessage

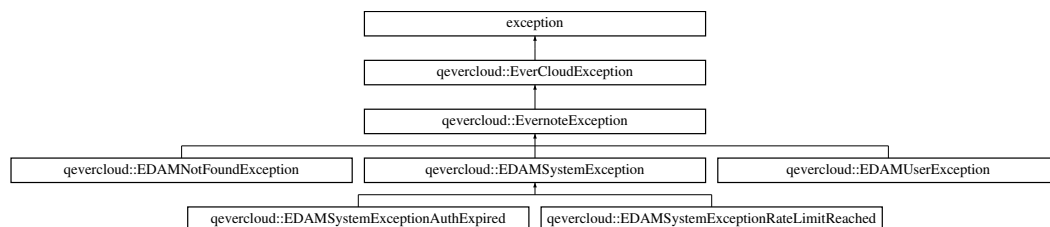
```
QString qevercloud::EverCloudExceptionData::errorMessage
```

Contains an error message. It's the `std::exception::what()` counterpart.

## 7.26 qevercloud::EvernoteException Class Reference

```
#include <EverCloudException.h>
```

Inheritance diagram for `qevercloud::EvernoteException`:



## Public Member Functions

- [EvernoteException](#) ()
- [EvernoteException](#) (QString error)
- [EvernoteException](#) (const std::string &error)
- [EvernoteException](#) (const char \*error)
- virtual QSharedPointer< [EverCloudExceptionData](#) > [exceptionData](#) () const Q\_DECL\_OVERRIDE

## Additional Inherited Members

### 7.26.1 Detailed Description

All exception sent by Evernote servers (as opposed to other error conditions, for example http errors) are descendants of this class.

## 7.26.2 Constructor & Destructor Documentation

### 7.26.2.1 `EvernoteException()` [1/4]

```
qevercloud::EvernoteException::EvernoteException ( ) [explicit]
```

### 7.26.2.2 `EvernoteException()` [2/4]

```
qevercloud::EvernoteException::EvernoteException (
    QString error ) [explicit]
```

### 7.26.2.3 `EvernoteException()` [3/4]

```
qevercloud::EvernoteException::EvernoteException (
    const std::string & error ) [explicit]
```

### 7.26.2.4 `EvernoteException()` [4/4]

```
qevercloud::EvernoteException::EvernoteException (
    const char * error ) [explicit]
```

## 7.26.3 Member Function Documentation

### 7.26.3.1 `exceptionData()`

```
virtual QSharedPointer<EverCloudExceptionData> qevercloud::EvernoteException::exceptionData (
) const [virtual]
```

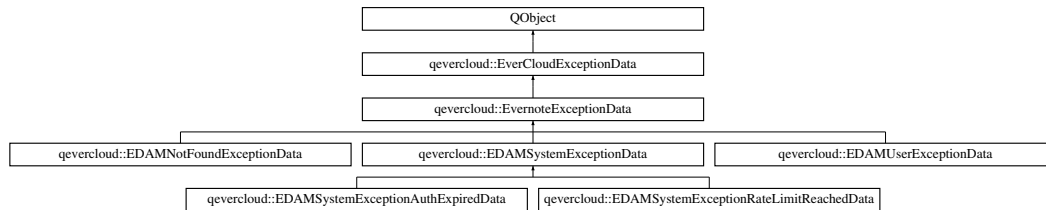
Reimplemented from [qevercloud::EverCloudException](#).

Reimplemented in [qevercloud::EDAMNotFoundException](#), [qevercloud::EDAMSystemException](#), [qevercloud::EDAMUserException](#), [qevercloud::EDAMSystemExceptionAuthExpired](#), and [qevercloud::EDAMSystemExceptionRateLimitReached](#).

## 7.27 qevercloud::EvernoteExceptionData Class Reference

```
#include <EverCloudException.h>
```

Inheritance diagram for qevercloud::EvernoteExceptionData:



### Public Member Functions

- [EvernoteExceptionData](#) (QString error)
- virtual void [throwException](#) () const Q\_DECL\_OVERRIDE

### Additional Inherited Members

#### 7.27.1 Detailed Description

Asynchronous API counterpart of [EvernoteException](#). See [EverCloudExceptionData](#) for more details.

#### 7.27.2 Constructor & Destructor Documentation

##### 7.27.2.1 EvernoteExceptionData()

```
qevercloud::EvernoteExceptionData::EvernoteExceptionData (
    QString error ) [explicit]
```

#### 7.27.3 Member Function Documentation

##### 7.27.3.1 throwException()

```
virtual void qevercloud::EvernoteExceptionData::throwException ( ) const [virtual]
```

If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented from [qevercloud::EverCloudExceptionData](#).

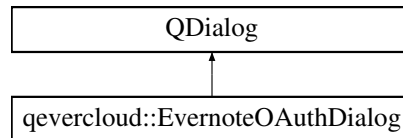
Reimplemented in [qevercloud::EDAMSystemExceptionAuthExpiredData](#), [qevercloud::EDAMSystemExceptionRateLimitReachedData](#), [qevercloud::EDAMNotFoundExceptionData](#), [qevercloud::EDAMSystemExceptionData](#), and [qevercloud::EDAMUserExceptionData](#).

## 7.28 qevercloud::EvernoteOAuthDialog Class Reference

Authorizes your app with the Evernote service by means of OAuth authentication.

```
#include <oauth.h>
```

Inheritance diagram for qevercloud::EvernoteOAuthDialog:



### Public Types

- typedef [EvernoteOAuthWebView::OAuthResult](#) [OAuthResult](#)

### Public Member Functions

- [EvernoteOAuthDialog](#) (QString consumerKey, QString consumerSecret, QString host=QString::Literal("www.evernote.com"), QWidget \*parent=Q\_NULLPTR)
- [~EvernoteOAuthDialog](#) ()
- void [setWebViewSizeHint](#) (QSize sizeHint)
- bool [isSucceeded](#) () const
- QString [oauthError](#) () const
- [OAuthResult](#) [oauthResult](#) () const
- int [exec](#) ()
- void [open](#) ()

### 7.28.1 Detailed Description

Authorizes your app with the Evernote service by means of OAuth authentication.

Intended usage:

```

#include <QEverCloudOAuth.h>
OAuthDialog d(myConsumerKey, myConsumerSecret);
if(d.exec() == QDialog::Accepted) {
    OAuthDialog::OAuthResult res = d.oauthResult();
    // Connect to Evernote
    ...
} else {
    QString errorText = d.oauthError();
    // handle an authentication error
    ...
}

```

Note that you have to include [QEverCloudOAuth.h](#) header.

By default [EvernoteOAuthDialog](#) uses `qrand()` for generating nonce so do not forget to call `qsrand()` in your application. See [setNonceGenerator](#) if you want more control over nonce generation.

### 7.28.2 Member Typedef Documentation



### 7.28.2.1 OAuthResult

```
typedef EvernoteOAuthWebView::OAuthResult qevercloud::EvernoteOAuthDialog::OAuthResult
```

## 7.28.3 Constructor & Destructor Documentation

### 7.28.3.1 EvernoteOAuthDialog()

```
qevercloud::EvernoteOAuthDialog::EvernoteOAuthDialog (
    QString consumerKey,
    QString consumerSecret,
    QString host = QStringLiteral("www.evernote.com"),
    QWidget * parent = Q_NULLPTR )
```

Constructs the dialog.

#### Parameters

<i>host</i>	Evernote host to authorize with. You need one of this: <ul style="list-style-type: none"><li>• "www.evernote.com" - the production service. It's the default value.</li><li>• "sandbox.evernote.com" - the developers "sandbox" service</li></ul>
<i>consumerKey</i>	get it <a href="#">from the Evernote</a>
<i>consumerSecret</i>	along with this

### 7.28.3.2 ~EvernoteOAuthDialog()

```
qevercloud::EvernoteOAuthDialog::~EvernoteOAuthDialog ( )
```

## 7.28.4 Member Function Documentation

### 7.28.4.1 exec()

```
int qevercloud::EvernoteOAuthDialog::exec ( )
```

#### Returns

QDialog::Accepted on a succesful authentication.

#### 7.28.4.2 isSucceeded()

```
bool qevercloud::EvernoteOAuthDialog::isSucceeded ( ) const
```

##### Returns

true in case of a succesful authentication. You probably better chech `exec()` return value instead.

#### 7.28.4.3 oauthError()

```
QString qevercloud::EvernoteOAuthDialog::oauthError ( ) const
```

##### Returns

In case of an authentication error may return some information about the error.

#### 7.28.4.4 oauthResult()

```
OAuthResult qevercloud::EvernoteOAuthDialog::oauthResult ( ) const
```

##### Returns

the result of a succesful authentication.

#### 7.28.4.5 open()

```
void qevercloud::EvernoteOAuthDialog::open ( )
```

Shows the dialog as a window modal dialog, returning immediately.

#### 7.28.4.6 setWebViewSizeHint()

```
void qevercloud::EvernoteOAuthDialog::setWebViewSizeHint (
    QSize sizeHint )
```

The dialog adjusts its initial size automatically based on the conatined QWebView preffered size. Use this method to set the size.

##### Parameters

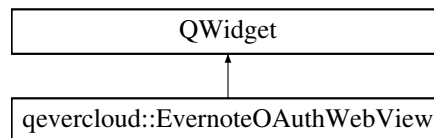
<i>sizeHint</i>	will be used as the preffered size of the contained QWebView.
-----------------	---

## 7.29 qevercloud::EvernoteOAuthWebView Class Reference

The class is tailored specifically for OAuth authorization with Evernote.

```
#include <oauth.h>
```

Inheritance diagram for qevercloud::EvernoteOAuthWebView:



### Classes

- struct [OAuthResult](#)

### Signals

- void [authenticationFinished](#) (bool success)
- void [authenticationSucceeded](#) ()
- void [authenticationFailed](#) ()

### Public Member Functions

- [EvernoteOAuthWebView](#) (QWidget \*parent=Q\_NULLPTR)
- void [authenticate](#) (QString host, QString consumerKey, QString consumerSecret)
- bool [isSucceeded](#) () const
- QString [oauthError](#) () const
- [OAuthResult](#) [oauthResult](#) () const
- void [setSizeHint](#) (QSize [sizeHint](#))
- virtual QSize [sizeHint](#) () const Q\_DECL\_OVERRIDE

#### 7.29.1 Detailed Description

The class is tailored specifically for OAuth authorization with Evernote.

While it is functional by itself you probably will prefer to use [EvernoteOAuthDialog](#).

Note that you have to include [QEverCloudOAuth.h](#) header.

By default [EvernoteOAuthWebView](#) uses `qrand()` for generating nonce so do not forget to call `qrand()` in your application. See [setNonceGenerator](#) If you want more control over nonce generation.

#### 7.29.2 Constructor & Destructor Documentation

### 7.29.2.1 EvernoteOAuthWebView()

```
qevercloud::EvernoteOAuthWebView::EvernoteOAuthWebView (
    QWidget * parent = Q_NULLPTR )
```

## 7.29.3 Member Function Documentation

### 7.29.3.1 authenticate()

```
void qevercloud::EvernoteOAuthWebView::authenticate (
    QString host,
    QString consumerKey,
    QString consumerSecret )
```

This function starts the OAuth sequence. In the end of the sequence will be emitted one of the signals: authenticationSucceeded or authenticationFailed.

Do not call the function while its call is in effect, i.e. one of the signals is not emitted.

#### Parameters

<i>host</i>	Evernote host to authorize with. You need one of this: <ul style="list-style-type: none"> <li>"www.evernote.com" - the production service. It's the default value.</li> <li>"sandex.evernote.com" - the developers "sandbox" service</li> </ul>
<i>consumerKey</i>	get it <a href="#">from the Evernote</a>
<i>consumerSecret</i>	along with this

### 7.29.3.2 authenticationFailed

```
void qevercloud::EvernoteOAuthWebView::authenticationFailed ( ) [signal]
```

Emitted when the OAuth sequence is finished with a failure. Some error info may be available with `errorText()`.

### 7.29.3.3 authenticationFinished

```
void qevercloud::EvernoteOAuthWebView::authenticationFinished (
    bool success ) [signal]
```

Emitted when the OAuth sequence started with [authenticate\(\)](#) call is finished

#### 7.29.3.4 authenticationSucceeded

```
void qevercloud::EvernoteOAuthWebView::authenticationSucceeded ( ) [signal]
```

Emitted when the OAuth sequence is successfully finished. Call [oauthResult\(\)](#) to get the data.

#### 7.29.3.5 isSucceeded()

```
bool qevercloud::EvernoteOAuthWebView::isSucceeded ( ) const
```

##### Returns

true if the last call to authenticate resulted in a successful authentication.

#### 7.29.3.6 oauthError()

```
QString qevercloud::EvernoteOAuthWebView::oauthError ( ) const
```

##### Returns

error message resulted from the last call to authenticate

#### 7.29.3.7 oauthResult()

```
OAuthResult qevercloud::EvernoteOAuthWebView::oauthResult ( ) const
```

##### Returns

the result of the last authentication, i.e. [authenticate\(\)](#) call.

#### 7.29.3.8 setSizeHint()

```
void qevercloud::EvernoteOAuthWebView::setSizeHint (
    QSize sizeHint )
```

The method is useful to specify default size for a EverOAuthWebView.

#### 7.29.3.9 sizeHint()

```
virtual QSize qevercloud::EvernoteOAuthWebView::sizeHint ( ) const [virtual]
```

## 7.30 qevercloud::Thumbnail::ImageType Struct Reference

```
#include <thumbnail.h>
```

### Public Types

- enum [type](#) { [PNG](#), [JPEG](#), [GIF](#), [BMP](#) }

#### 7.30.1 Detailed Description

Specifies image type of the returned thumbnail.

Can be PNG, JPEG, GIF or BMP.

#### 7.30.2 Member Enumeration Documentation

##### 7.30.2.1 type

```
enum qevercloud::Thumbnail::ImageType::type
```

##### Enumerator

PNG	
JPEG	
GIF	
BMP	

## 7.31 qevercloud::LazyMap Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [LazyMap](#) &other) const
- bool [operator!=](#) (const [LazyMap](#) &other) const

### Public Attributes

- [Optional](#)< QSet< QString > > [keysOnly](#)
- [Optional](#)< QMap< QString, QString > > [fullMap](#)

### 7.31.1 Detailed Description

A structure that wraps a map of name/value pairs whose values are not always present in the structure in order to reduce space when obtaining batches of entities that contain the map.

When the server provides the client with a [LazyMap](#), it will fill in either the `keysOnly` field or the `fullMap` field, but never both, based on the API and parameters.

When a client provides a [LazyMap](#) to the server as part of an update to an object, the server will only update the [LazyMap](#) if the `fullMap` field is set. If the `fullMap` field is not set, the server will not make any changes to the map.

Check the API documentation of the individual calls involving the [LazyMap](#) for full details including the constraints of the names and values of the map.

### 7.31.2 Member Function Documentation

#### 7.31.2.1 `operator!=()`

```
bool qevercloud::LazyMap::operator!= (
    const LazyMap & other ) const [inline]
```

#### 7.31.2.2 `operator==()`

```
bool qevercloud::LazyMap::operator== (
    const LazyMap & other ) const [inline]
```

### 7.31.3 Member Data Documentation

#### 7.31.3.1 `fullMap`

```
Optional< QMap< QString, QString > > qevercloud::LazyMap::fullMap
```

The complete map, including all keys and values.

#### 7.31.3.2 `keysOnly`

```
Optional< QSet< QString > > qevercloud::LazyMap::keysOnly
```

The set of keys for the map. This field is ignored by the server when set.

## 7.32 qevercloud::LinkedNotebook Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [LinkedNotebook](#) &other) const
- bool `operator!=` (const [LinkedNotebook](#) &other) const

### Public Attributes

- [Optional](#)< [QString](#) > [shareName](#)
- [Optional](#)< [QString](#) > [username](#)
- [Optional](#)< [QString](#) > [shardId](#)
- [Optional](#)< [QString](#) > [shareKey](#)
- [Optional](#)< [QString](#) > [uri](#)
- [Optional](#)< [Guid](#) > [guid](#)
- [Optional](#)< [qint32](#) > [updateSequenceNum](#)
- [Optional](#)< [QString](#) > [noteStoreUrl](#)
- [Optional](#)< [QString](#) > [webApiUrlPrefix](#)
- [Optional](#)< [QString](#) > [stack](#)
- [Optional](#)< [qint32](#) > [businessId](#)

### 7.32.1 Detailed Description

A link in an users account that refers them to a public or individual share in another user's account.

### 7.32.2 Member Function Documentation

#### 7.32.2.1 `operator"!=()`

```
bool qevercloud::LinkedNotebook::operator!= (
    const LinkedNotebook & other ) const [inline]
```

#### 7.32.2.2 `operator==(())`

```
bool qevercloud::LinkedNotebook::operator==(
    const LinkedNotebook & other ) const [inline]
```

### 7.32.3 Member Data Documentation



### 7.32.3.1 businessId

`Optional< qint32 > qevercloud::LinkedNotebook::businessId`

If set, this will be the unique identifier for the business that owns the notebook to which the linked notebook refers.

### 7.32.3.2 guid

`Optional< Guid > qevercloud::LinkedNotebook::guid`

The unique identifier of this linked notebook. Will be set whenever a linked notebook is retrieved from the service, but may be null when a client is creating a linked notebook.

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX

### 7.32.3.3 noteStoreUrl

`Optional< QString > qevercloud::LinkedNotebook::noteStoreUrl`

This field will contain the full URL that clients should use to make [NoteStore](#) requests to the server shard that contains that notebook's data. I.e. this is the URL that should be used to create the Thrift HTTP client transport to send messages to the [NoteStore](#) service for the account.

### 7.32.3.4 shardId

`Optional< QString > qevercloud::LinkedNotebook::shardId`

the shard ID of the notebook if the notebook is not public

shareKey the secret key that provides access to the shared notebook

### 7.32.3.5 shareKey

`Optional< QString > qevercloud::LinkedNotebook::shareKey`

NOT DOCUMENTED

### 7.32.3.6 shareName

`Optional< QString > qevercloud::LinkedNotebook::shareName`

the display name of the shared notebook. The link owner can change this.

### 7.32.3.7 stack

```
Optional< QString > qevercloud::LinkedNotebook::stack
```

If this is set, then the notebook is visually contained within a stack of notebooks with this name. All notebooks in the same account with the same 'stack' field are considered to be in the same stack. Notebooks with no stack set are "top level" and not contained within a stack. The link owner can change this and this field is for the benefit of the link owner.

### 7.32.3.8 updateSequenceNum

```
Optional< qint32 > qevercloud::LinkedNotebook::updateSequenceNum
```

A number identifying the last transaction to modify the state of this object. The USN values are sequential within an account, and can be used to compare the order of modifications within the service.

### 7.32.3.9 uri

```
Optional< QString > qevercloud::LinkedNotebook::uri
```

the identifier of the public notebook

### 7.32.3.10 username

```
Optional< QString > qevercloud::LinkedNotebook::username
```

the username of the user who owns the shared or public notebook

### 7.32.3.11 webApiUrlPrefix

```
Optional< QString > qevercloud::LinkedNotebook::webApiUrlPrefix
```

This field will contain the initial part of the URLs that should be used to make requests to Evernote's thin client "web API", which provide optimized operations for clients that aren't capable of manipulating the full contents of accounts via the full Thrift data model. Clients should concatenate the relative path for the various servlets onto the end of this string to construct the full URL, as documented on our developer web site.

## 7.33 qevercloud::Note Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [Note](#) &other) const
- bool `operator!=` (const [Note](#) &other) const

## Public Attributes

- [Optional](#)< [Guid](#) > [guid](#)
- [Optional](#)< [QString](#) > [title](#)
- [Optional](#)< [QString](#) > [content](#)
- [Optional](#)< [QByteArray](#) > [contentHash](#)
- [Optional](#)< [qint32](#) > [contentLength](#)
- [Optional](#)< [Timestamp](#) > [created](#)
- [Optional](#)< [Timestamp](#) > [updated](#)
- [Optional](#)< [Timestamp](#) > [deleted](#)
- [Optional](#)< [bool](#) > [active](#)
- [Optional](#)< [qint32](#) > [updateSequenceNum](#)
- [Optional](#)< [QString](#) > [notebookGuid](#)
- [Optional](#)< [QList](#)< [Guid](#) > > [tagGuids](#)
- [Optional](#)< [QList](#)< [Resource](#) > > [resources](#)
- [Optional](#)< [NoteAttributes](#) > [attributes](#)
- [Optional](#)< [QStringList](#) > [tagNames](#)

### 7.33.1 Detailed Description

Represents a single note in the user's account.

### 7.33.2 Member Function Documentation

#### 7.33.2.1 operator"!="()

```
bool qevercloud::Note::operator!= (
    const Note & other ) const [inline]
```

#### 7.33.2.2 operator=="()

```
bool qevercloud::Note::operator== (
    const Note & other ) const [inline]
```

### 7.33.3 Member Data Documentation

#### 7.33.3.1 active

```
Optional< bool > qevercloud::Note::active
```

If the note is available for normal actions and viewing, this flag will be set to true.

### 7.33.3.2 attributes

```
Optional< NoteAttributes > qevercloud::Note::attributes
```

A list of the attributes for this note. If the list of attributes are omitted on a call to `updateNote()`, then the server will assume that no changes have been made to the resources.

### 7.33.3.3 content

```
Optional< QString > qevercloud::Note::content
```

The XHTML block that makes up the note. This is the canonical form of the note's contents, so will include abstract Evernote tags for internal resource references. A client may create a separate transformed version of this content for internal presentation, but the same canonical bytes should be used for transmission and comparison unless the user chooses to modify their content.

Length: EDAM\_NOTE\_CONTENT\_LEN\_MIN - EDAM\_NOTE\_CONTENT\_LEN\_MAX

### 7.33.3.4 contentHash

```
Optional< QByteArray > qevercloud::Note::contentHash
```

The binary MD5 checksum of the UTF-8 encoded content body. This will always be set by the server, but clients may choose to omit this when they submit a note with content.

Length: EDAM\_HASH\_LEN (exactly)

### 7.33.3.5 contentLength

```
Optional< qint32 > qevercloud::Note::contentLength
```

The number of Unicode characters in the content of the note. This will always be set by the service, but clients may choose to omit this value when they submit a [Note](#).

### 7.33.3.6 created

```
Optional< Timestamp > qevercloud::Note::created
```

The date and time when the note was created in one of the clients. In most cases, this will match the user's sense of when the note was created, and ordering between notes will be based on ordering of this field. However, this is not a "reliable" timestamp if a client has an incorrect clock, so it cannot provide a true absolute ordering between notes. Notes created directly through the service (e.g. via the web GUI) will have an absolutely ordered "created" value.

### 7.33.3.7 deleted

```
Optional< Timestamp > qevercloud::Note::deleted
```

If present, the note is considered "deleted", and this stores the date and time when the note was deleted by one of the clients. In most cases, this will match the user's sense of when the note was deleted, but this field may be unreliable due to the possibility of client clock errors.

### 7.33.3.8 guid

`Optional< Guid > qevercloud::Note::guid`

The unique identifier of this note. Will be set by the server, but will be omitted by clients calling [NoteStore.createNote\(\)](#)

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX

### 7.33.3.9 notebookGuid

`Optional< QString > qevercloud::Note::notebookGuid`

The unique identifier of the notebook that contains this note. If no notebookGuid is provided on a call to [createNote\(\)](#), the default notebook will be used instead.

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX

### 7.33.3.10 resources

`Optional< QList< Resource > > qevercloud::Note::resources`

The list of resources that are embedded within this note. If the list of resources are omitted on a call to [updateNote\(\)](#), then the server will assume that no changes have been made to the resources. The binary contents of the resources must be provided when the resource is first sent to the service, but it will be omitted by the service when the [Note](#) is returned in the future. Maximum: EDAM\_NOTE\_RESOURCES\_MAX resources per note

### 7.33.3.11 tagGuids

`Optional< QList< Guid > > qevercloud::Note::tagGuids`

A list of the GUID identifiers for tags that are applied to this note. This may be provided in a call to [createNote\(\)](#) to unambiguously declare the tags that should be assigned to the new note. Alternately, clients may pass the names of desired tags via the 'tagNames' field during note creation. If the list of tags are omitted on a call to [createNote\(\)](#), then the server will assume that no changes have been made to the resources. Maximum: EDAM\_NOTE\_TAGS\_MAX tags per note

### 7.33.3.12 tagNames

`Optional< QStringList > qevercloud::Note::tagNames`

May be provided by clients during calls to [createNote\(\)](#) as an alternative to providing the tagGuids of existing tags. If any tagNames are provided during [createNote\(\)](#), these will be found, or created if they don't already exist. Created tags will have no parent (they will be at the top level of the tag panel).

### 7.33.3.13 title

`Optional< QString > qevercloud::Note::title`

The subject of the note. Can't begin or end with a space.

Length: EDAM\_NOTE\_TITLE\_LEN\_MIN - EDAM\_NOTE\_TITLE\_LEN\_MAX

Regex: EDAM\_NOTE\_TITLE\_REGEX

### 7.33.3.14 updated

```
Optional< Timestamp > qevercloud::Note::updated
```

The date and time when the note was last modified in one of the clients. In most cases, this will match the user's sense of when the note was modified, but this field may not be absolutely reliable due to the possibility of client clock errors.

### 7.33.3.15 updateSequenceNum

```
Optional< qint32 > qevercloud::Note::updateSequenceNum
```

A number identifying the last transaction to modify the state of this note (including changes to the note's attributes or resources). The USN values are sequential within an account, and can be used to compare the order of modifications within the service.

## 7.34 qevercloud::NoteAttributes Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [NoteAttributes](#) &other) const
- bool `operator!=` (const [NoteAttributes](#) &other) const

### Public Attributes

- [Optional< Timestamp >](#) `subjectDate`
- [Optional< double >](#) `latitude`
- [Optional< double >](#) `longitude`
- [Optional< double >](#) `altitude`
- [Optional< QString >](#) `author`
- [Optional< QString >](#) `source`
- [Optional< QString >](#) `sourceURL`
- [Optional< QString >](#) `sourceApplication`
- [Optional< Timestamp >](#) `shareDate`
- [Optional< qint64 >](#) `reminderOrder`
- [Optional< Timestamp >](#) `reminderDoneTime`
- [Optional< Timestamp >](#) `reminderTime`
- [Optional< QString >](#) `placeName`
- [Optional< QString >](#) `contentClass`
- [Optional< LazyMap >](#) `applicationData`
- [Optional< QString >](#) `lastEditedBy`
- [Optional< QMap< QString, QString > >](#) `classifications`
- [Optional< UserID >](#) `creatorId`
- [Optional< UserID >](#) `lastEditorId`

### 7.34.1 Detailed Description

The list of optional attributes that can be stored on a note.

### 7.34.2 Member Function Documentation

#### 7.34.2.1 operator!=(())

```
bool qevercloud::NoteAttributes::operator!= (
    const NoteAttributes & other ) const [inline]
```

#### 7.34.2.2 operator==(())

```
bool qevercloud::NoteAttributes::operator== (
    const NoteAttributes & other ) const [inline]
```

### 7.34.3 Member Data Documentation

#### 7.34.3.1 altitude

```
Optional< double > qevercloud::NoteAttributes::altitude
```

the altitude where the note was taken

#### 7.34.3.2 applicationData

```
Optional< LazyMap > qevercloud::NoteAttributes::applicationData
```

Provides a location for applications to store a relatively small (4kb) blob of data that is not meant to be visible to the user and that is opaque to the Evernote service. A single application may use at most one entry in this map, using its API consumer key as the map key. See the documentation for [LazyMap](#) for a description of when the actual map values are returned by the service.

To safely add or modify your application's entry in the map, use [NoteStore.setNoteApplicationDataEntry](#). To safely remove your application's entry from the map, use [NoteStore.unsetNoteApplicationDataEntry](#).

Minimum length of a name (key): EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN  
Sum max size of key and value: EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX  
Syntax regex for name (key): EDAM\_APPLICATIONDATA\_NAME\_REGEX

### 7.34.3.3 author

```
Optional< QString > qevercloud::NoteAttributes::author
```

the author of the content of the note

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

### 7.34.3.4 classifications

```
Optional< QMap< QString, QString > > qevercloud::NoteAttributes::classifications
```

A map of classifications applied to the note by clients or by the Evernote service. The key is the string name of the classification type, and the value is a constant that begins with CLASSIFICATION\_.

### 7.34.3.5 contentClass

```
Optional< QString > qevercloud::NoteAttributes::contentClass
```

The class (or type) of note. This field is used to indicate to clients that special structured information is represented within the note such that special rules apply when making modifications. If contentClass is set and the client application does not specifically support the specified class, the client MUST treat the note as read-only. In this case, the client MAY modify the note's notebook and tags via the [Note.notebookGuid](#) and [Note.tagGuids](#) fields. The client MAY also modify the reminderOrder field as well as the reminderTime and reminderDoneTime fields.

Applications should set contentClass only when they are creating notes that contain structured information that needs to be maintained in order for the user to be able to use the note within that application. Setting contentClass makes a note read-only in other applications, so there is a trade-off when an application chooses to use contentClass. Applications that set contentClass when creating notes must use a contentClass string of the form *CompanyName.ApplicationName* to ensure uniqueness.

Length restrictions: EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MIN, EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MAX  
Regex: EDAM\_NOTE\_CONTENT\_CLASS\_REGEX

### 7.34.3.6 creatorId

```
Optional< UserID > qevercloud::NoteAttributes::creatorId
```

The numeric user ID of the user who originally created the note.

### 7.34.3.7 lastEditedBy

```
Optional< QString > qevercloud::NoteAttributes::lastEditedBy
```

An indication of who made the last change to the note. If you are accessing the note via a shared notebook to which you have modification rights, or if you are the owner of the notebook to which the note belongs, then you have access to the value. In this case, the value will be unset if the owner of the notebook containing the note was the last to make the modification, else it will be a string describing the guest who made the last edit. If you do not have access to this value, it will be left unset. This field is read-only by clients. The server will ignore all values set by clients into this field.



#### 7.34.3.8 lastEditorId

`Optional< UserID > qevercloud::NoteAttributes::lastEditorId`

The numeric user ID of the user described in lastEditedBy.

#### 7.34.3.9 latitude

`Optional< double > qevercloud::NoteAttributes::latitude`

the latitude where the note was taken

#### 7.34.3.10 longitude

`Optional< double > qevercloud::NoteAttributes::longitude`

the longitude where the note was taken

#### 7.34.3.11 placeName

`Optional< QString > qevercloud::NoteAttributes::placeName`

Allows the user to assign a human-readable location name associated with a note. Users may assign values like 'Home' and 'Work'. Place names may also be populated with values from geonames database (e.g., a restaurant name). Applications are encouraged to normalize values so that grouping values by place name provides a useful result. Applications MUST NOT automatically add place name values based on geolocation without confirmation from the user; that is, the value in this field should be more useful than a simple automated lookup based on the note's latitude and longitude.

#### 7.34.3.12 reminderDoneTime

`Optional< Timestamp > qevercloud::NoteAttributes::reminderDoneTime`

The date and time when a user dismissed/"marked done" the reminder on the note. Users typically do not manually set this value directly as it is set to the time when the user dismissed/"marked done" the reminder.

#### 7.34.3.13 reminderOrder

`Optional< qint64 > qevercloud::NoteAttributes::reminderOrder`

The set of notes with this parameter set are considered "reminders" and are to be treated specially by clients to give them higher UI prominence within a notebook. The value is used to sort the reminder notes within the notebook with higher values representing greater prominence. Outside of the context of a notebook, the value of this parameter is undefined. The value is not intended to be compared to the values of reminder notes in other notebooks. In order to allow clients to place a note at a higher precedence than other notes, you should never set a value greater than the current time (as defined for a Timestamp). To place a note at higher precedence than existing notes, set the value to the current time as defined for a timestamp (milliseconds since the epoch). Synchronizing clients must remember the time when the update was performed, using the local clock on the client, and use that value when they later upload the note to the service. Clients must not set the reminderOrder to the reminderTime as the reminderTime could be in the future. Those two fields are never intended to be related. The correct value for reminderOrder field for new notes is the "current" time when the user indicated that the note is a reminder. Clients may implement a separate "sort by date" feature to show notes ordered by reminderTime. Whenever a reminderDoneTime or reminderTime is set but a reminderOrder is not set, the server will fill in the current server time for the reminderOrder field.

#### 7.34.3.14 reminderTime

`Optional< Timestamp > qevercloud::NoteAttributes::reminderTime`

The date and time a user has selected to be reminded of the note. A note with this value set is known as a "reminder" and the user can be reminded, via e-mail or client-specific notifications, of the note when the time is reached or about to be reached. When a user sets a reminder time on a note that has a reminder done time, and that reminder time is in the future, then the reminder done time should be cleared. This should happen regardless of any existing reminder time that may have previously existed on the note.

#### 7.34.3.15 shareDate

`Optional< Timestamp > qevercloud::NoteAttributes::shareDate`

The date and time when this note was directly shared via its own URL. This is only set on notes that were individually shared - it is independent of any notebook-level sharing of the containing notebook. This field is treated as "read-only" for clients; the server will ignore changes to this field from an external client.

#### 7.34.3.16 source

`Optional< QString > qevercloud::NoteAttributes::source`

the method that the note was added to the account, if the note wasn't directly authored in an Evernote desktop client.

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.34.3.17 sourceApplication

`Optional< QString > qevercloud::NoteAttributes::sourceApplication`

an identifying string for the application that created this note. This string does not have a guaranteed syntax or structure – it is intended for human inspection and tracking.

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.34.3.18 sourceURL

`Optional< QString > qevercloud::NoteAttributes::sourceURL`

the original location where the resource was hosted. For web clips, this will be the URL of the page that was clipped.

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.34.3.19 subjectDate

`Optional< Timestamp > qevercloud::NoteAttributes::subjectDate`

time that the note refers to

## 7.35 qevercloud::Notebook Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [Notebook](#) &other) const
- bool [operator!=](#) (const [Notebook](#) &other) const

### Public Attributes

- [Optional](#)< [Guid](#) > [guid](#)
- [Optional](#)< [QString](#) > [name](#)
- [Optional](#)< [qint32](#) > [updateSequenceNum](#)
- [Optional](#)< [bool](#) > [defaultNotebook](#)
- [Optional](#)< [Timestamp](#) > [serviceCreated](#)
- [Optional](#)< [Timestamp](#) > [serviceUpdated](#)
- [Optional](#)< [Publishing](#) > [publishing](#)
- [Optional](#)< [bool](#) > [published](#)
- [Optional](#)< [QString](#) > [stack](#)
- [Optional](#)< [QList](#)< [qint64](#) > > [sharedNotebookIds](#)
- [Optional](#)< [QList](#)< [SharedNotebook](#) > > [sharedNotebooks](#)
- [Optional](#)< [BusinessNotebook](#) > [businessNotebook](#)
- [Optional](#)< [User](#) > [contact](#)
- [Optional](#)< [NotebookRestrictions](#) > [restrictions](#)

### 7.35.1 Detailed Description

A unique container for a set of notes.

### 7.35.2 Member Function Documentation

#### 7.35.2.1 [operator"!=\(\)](#)

```
bool qevercloud::Notebook::operator!= (
    const Notebook & other ) const [inline]
```

#### 7.35.2.2 [operator==\(\)](#)

```
bool qevercloud::Notebook::operator== (
    const Notebook & other ) const [inline]
```

### 7.35.3 Member Data Documentation

#### 7.35.3.1 businessNotebook

`Optional< BusinessNotebook > qevercloud::Notebook::businessNotebook`

If the notebook is part of a business account and has been published to the business library, this will contain information for the library listing. The presence or absence of this field is not a reliable test of whether a given notebook is in fact a business notebook - the field is only used when a notebook is or has been published to the business library.

#### 7.35.3.2 contact

`Optional< User > qevercloud::Notebook::contact`

Intended for use with Business accounts, this field identifies the user who has been designated as the "contact". For notebooks created in business accounts, the server will automatically set this value to the user who created the notebook unless Notebook.contact.username has been set, in which that value will be used. When updating a notebook, it is common to leave Notebook.contact field unset, indicating that no change to the value is being requested and that the existing value, if any, should be preserved.

#### 7.35.3.3 defaultNotebook

`Optional< bool > qevercloud::Notebook::defaultNotebook`

If true, this notebook should be used for new notes whenever the user has not (or cannot) specify a desired target notebook. For example, if a note is submitted via SMTP email. The service will maintain at most one default Notebook per account. If a second notebook is created or updated with defaultNotebook set to true, the service will automatically update the prior notebook's defaultNotebook field to false. If the default notebook is deleted (i.e. "active" set to false), the "defaultNotebook" field will be set to false by the service. If the account has no default notebook set, the service will use the most recent notebook as the default.

#### 7.35.3.4 guid

`Optional< Guid > qevercloud::Notebook::guid`

The unique identifier of this notebook.

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX

#### 7.35.3.5 name

`Optional< QString > qevercloud::Notebook::name`

A sequence of characters representing the name of the notebook. May be changed by clients, but the account may not contain two notebooks with names that are equal via a case-insensitive comparison. Can't begin or end with a space.

Length: EDAM\_NOTEBOOK\_NAME\_LEN\_MIN - EDAM\_NOTEBOOK\_NAME\_LEN\_MAX

Regex: EDAM\_NOTEBOOK\_NAME\_REGEX

### 7.35.3.6 published

`Optional< bool > qevercloud::Notebook::published`

If this is set to true, then the [Notebook](#) will be accessible either to the public, or for business users to their business, via the 'publishing' specification, which must also be set. If this is set to false, the [Notebook](#) will not be available to the public (or business). Clients that do not wish to change the publishing behavior of a [Notebook](#) should not set this value when calling [NoteStore.updateNotebook\(\)](#).

### 7.35.3.7 publishing

`Optional< Publishing > qevercloud::Notebook::publishing`

If the [Notebook](#) has been opened for public access, or business users shared with their business (i.e. if 'published' is set to true), then this will point to the set of publishing information for the [Notebook](#) (URI, description, etc.). A [Notebook](#) cannot be published without providing this information, but it will persist for later use if publishing is ever disabled on the [Notebook](#). Clients that do not wish to change the publishing behavior of a [Notebook](#) should not set this value when calling [NoteStore.updateNotebook\(\)](#).

### 7.35.3.8 restrictions

`Optional< NotebookRestrictions > qevercloud::Notebook::restrictions`

NOT DOCUMENTED

### 7.35.3.9 serviceCreated

`Optional< Timestamp > qevercloud::Notebook::serviceCreated`

The time when this notebook was created on the service. This will be set on the service during creation, and the service will provide this value when it returns a [Notebook](#) to a client. The service will ignore this value if it is sent by clients.

### 7.35.3.10 serviceUpdated

`Optional< Timestamp > qevercloud::Notebook::serviceUpdated`

The time when this notebook was last modified on the service. This will be set on the service during creation, and the service will provide this value when it returns a [Notebook](#) to a client. The service will ignore this value if it is sent by clients.

### 7.35.3.11 sharedNotebookIds

`Optional< QList< qint64 > > qevercloud::Notebook::sharedNotebookIds`

*DEPRECATED* - replaced by sharedNotebooks.

### 7.35.3.12 sharedNotebooks

```
Optional< QList< SharedNotebook > > qevercloud::Notebook::sharedNotebooks
```

The list of recipients to whom this notebook has been shared (one [SharedNotebook](#) object per recipient email address). This field will be unset if you do not have permission to access this data. If you are accessing the notebook as the owner or via a shared notebook that is modifiable, then you have access to this data and the value will be set. This field is read-only. Clients may not make changes to shared notebooks via this field.

### 7.35.3.13 stack

```
Optional< QString > qevercloud::Notebook::stack
```

If this is set, then the notebook is visually contained within a stack of notebooks with this name. All notebooks in the same account with the same 'stack' field are considered to be in the same stack. Notebooks with no stack set are "top level" and not contained within a stack.

### 7.35.3.14 updateSequenceNum

```
Optional< qint32 > qevercloud::Notebook::updateSequenceNum
```

A number identifying the last transaction to modify the state of this object. The USN values are sequential within an account, and can be used to compare the order of modifications within the service.

## 7.36 qevercloud::NotebookDescriptor Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [NotebookDescriptor](#) &other) const
- bool [operator!=](#) (const [NotebookDescriptor](#) &other) const

### Public Attributes

- [Optional< Guid >](#) [guid](#)
- [Optional< QString >](#) [notebookDisplayName](#)
- [Optional< QString >](#) [contactName](#)
- [Optional< bool >](#) [hasSharedNotebook](#)
- [Optional< qint32 >](#) [joinedUserCount](#)

### 7.36.1 Detailed Description

A structure that describes a notebook or a user's relationship with a notebook. [NotebookDescriptor](#) is expected to remain a lighter-weight structure when compared to [Notebook](#).

## 7.36.2 Member Function Documentation

### 7.36.2.1 operator!=(())

```
bool qevercloud::NotebookDescriptor::operator!= (
    const NotebookDescriptor & other ) const [inline]
```

### 7.36.2.2 operator==(())

```
bool qevercloud::NotebookDescriptor::operator== (
    const NotebookDescriptor & other ) const [inline]
```

## 7.36.3 Member Data Documentation

### 7.36.3.1 contactName

[Optional](#)< [QString](#) > qevercloud::NotebookDescriptor::contactName

The [User.name](#) value of the notebook's "contact".

### 7.36.3.2 guid

[Optional](#)< [Guid](#) > qevercloud::NotebookDescriptor::guid

The unique identifier of the notebook.

### 7.36.3.3 hasSharedNotebook

[Optional](#)< bool > qevercloud::NotebookDescriptor::hasSharedNotebook

Whether a [SharedNotebook](#) record exists between the calling user and this notebook.

### 7.36.3.4 joinedUserCount

[Optional](#)< [qint32](#) > qevercloud::NotebookDescriptor::joinedUserCount

The number of users who have joined this notebook.

### 7.36.3.5 notebookDisplayName

`Optional< QString > qevercloud::NotebookDescriptor::notebookDisplayName`

A sequence of characters representing the name of the notebook.

## 7.37 qevercloud::NotebookRestrictions Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [NotebookRestrictions](#) &other) const
- bool `operator!=` (const [NotebookRestrictions](#) &other) const

### Public Attributes

- `Optional< bool > noReadNotes`
- `Optional< bool > noCreateNotes`
- `Optional< bool > noUpdateNotes`
- `Optional< bool > noExpungeNotes`
- `Optional< bool > noShareNotes`
- `Optional< bool > noEmailNotes`
- `Optional< bool > noSendMessageToRecipients`
- `Optional< bool > noUpdateNotebook`
- `Optional< bool > noExpungeNotebook`
- `Optional< bool > noSetDefaultNotebook`
- `Optional< bool > noSetNotebookStack`
- `Optional< bool > noPublishToPublic`
- `Optional< bool > noPublishToBusinessLibrary`
- `Optional< bool > noCreateTags`
- `Optional< bool > noUpdateTags`
- `Optional< bool > noExpungeTags`
- `Optional< bool > noSetParentTag`
- `Optional< bool > noCreateSharedNotebooks`
- `Optional< SharedNotebookInstanceRestrictions::type > updateWhichSharedNotebookRestrictions`
- `Optional< SharedNotebookInstanceRestrictions::type > expungeWhichSharedNotebookRestrictions`

### 7.37.1 Detailed Description

This structure captures information about the types of operations that cannot be performed on a given notebook with a type of authenticated access and credentials. The values filled into this structure are based on then-current values in the server database for shared notebooks and notebook publishing records, as well as information related to the authentication token. Information from the authentication token includes the application that is accessing the server, as defined by the permissions granted by consumer (api) key, and the method used to obtain the token, for example via `authenticateToSharedNotebook`, `authenticateToBusiness`, etc. **Note** that changes to values in this structure that are the result of shared notebook or publishing record changes are communicated to the client via a change in the notebook USN during sync. It is important to use the same access method, parameters, and consumer key in order to obtain correct results from the sync engine.

The server has the final say on what is allowed as values may change between calls to obtain [NotebookRestrictions](#) instances and to operate on data on the service.

If the following are set and true, then the given restriction is in effect, as accessed by the same authentication token from which the values were obtained.



## 7.37.2 Member Function Documentation

### 7.37.2.1 operator!=(())

```
bool qevercloud::NotebookRestrictions::operator!= (
    const NotebookRestrictions & other ) const [inline]
```

### 7.37.2.2 operator==(())

```
bool qevercloud::NotebookRestrictions::operator== (
    const NotebookRestrictions & other ) const [inline]
```

## 7.37.3 Member Data Documentation

### 7.37.3.1 expungeWhichSharedNotebookRestrictions

```
Optional< SharedNotebookInstanceRestrictions::type > qevercloud::NotebookRestrictions::expunge↵
WhichSharedNotebookRestrictions
```

Restrictions on which shared notebook instances can be expunged. If the value is not set or null, then the client can expunge any of the shared notebooks associated with the notebook on which the [NotebookRestrictions](#) are defined. See the enumeration for further details.

### 7.37.3.2 noCreateNotes

```
Optional< bool > qevercloud::NotebookRestrictions::noCreateNotes
```

The client may not create new notes in the notebook.

### 7.37.3.3 noCreateSharedNotebooks

```
Optional< bool > qevercloud::NotebookRestrictions::noCreateSharedNotebooks
```

The client is unable to create shared notebooks for the notebook.

### 7.37.3.4 noCreateTags

```
Optional< bool > qevercloud::NotebookRestrictions::noCreateTags
```

The client may not complete an operation that results in a new tag being created in the owner's account.

### 7.37.3.5 noEmailNotes

```
Optional< bool > qevercloud::NotebookRestrictions::noEmailNotes
```

The client may not e-mail notes via the Evernote service by using the emailNote method.

### 7.37.3.6 noExpungeNotebook

```
Optional< bool > qevercloud::NotebookRestrictions::noExpungeNotebook
```

The client may not expunge the [Notebook](#) object itself, for example, via the expungeNotebook method.

### 7.37.3.7 noExpungeNotes

```
Optional< bool > qevercloud::NotebookRestrictions::noExpungeNotes
```

The client may not expunge notes currently in the notebook.

### 7.37.3.8 noExpungeTags

```
Optional< bool > qevercloud::NotebookRestrictions::noExpungeTags
```

The client may not expunge tags in the owner's account.

### 7.37.3.9 noPublishToBusinessLibrary

```
Optional< bool > qevercloud::NotebookRestrictions::noPublishToBusinessLibrary
```

The client may not publish the notebook to the business library.

### 7.37.3.10 noPublishToPublic

```
Optional< bool > qevercloud::NotebookRestrictions::noPublishToPublic
```

The client may not change the publish the notebook to the public. For example, business notebooks may not be shared publicly.

### 7.37.3.11 noReadNotes

```
Optional< bool > qevercloud::NotebookRestrictions::noReadNotes
```

The client is not able to read notes from the service and the notebook is write-only.

### 7.37.3.12 noSendMessageToRecipients

`Optional< bool > qevercloud::NotebookRestrictions::noSendMessageToRecipients`

The client may not send messages to the share recipients of the notebook.

### 7.37.3.13 noSetDefaultNotebook

`Optional< bool > qevercloud::NotebookRestrictions::noSetDefaultNotebook`

The client may not set this notebook to be the default notebook. The caller should leave [Notebook.defaultNotebook](#) unset.

### 7.37.3.14 noSetNotebookStack

`Optional< bool > qevercloud::NotebookRestrictions::noSetNotebookStack`

If the client is able to update the [Notebook](#), the [Notebook.stack](#) value may not be set.

### 7.37.3.15 noSetParentTag

`Optional< bool > qevercloud::NotebookRestrictions::noSetParentTag`

If the client is able to create or update tags in the owner's account, then they will not be able to set the parent tag. Leave the value unset.

### 7.37.3.16 noShareNotes

`Optional< bool > qevercloud::NotebookRestrictions::noShareNotes`

The client may not share notes in the notebook via the `shareNote` method.

### 7.37.3.17 noUpdateNotebook

`Optional< bool > qevercloud::NotebookRestrictions::noUpdateNotebook`

The client may not update the [Notebook](#) object itself, for example, via the `updateNotebook` method.

### 7.37.3.18 noUpdateNotes

`Optional< bool > qevercloud::NotebookRestrictions::noUpdateNotes`

The client may not update notes currently in the notebook.

### 7.37.3.19 noUpdateTags

`Optional< bool > qevercloud::NotebookRestrictions::noUpdateTags`

The client may not update tags in the owner's account.

### 7.37.3.20 updateWhichSharedNotebookRestrictions

`Optional< SharedNotebookInstanceRestrictions::type > qevercloud::NotebookRestrictions::updateWhichSharedNotebookRestrictions`

Restrictions on which shared notebook instances can be updated. If the value is not set or null, then the client can update any of the shared notebooks associated with the notebook on which the [NotebookRestrictions](#) are defined. See the enumeration for further details.

## 7.38 qevercloud::NoteCollectionCounts Struct Reference

```
#include <types.h>
```

### Public Member Functions

- `bool operator==(const NoteCollectionCounts &other) const`
- `bool operator!=(const NoteCollectionCounts &other) const`

### Public Attributes

- `Optional< QMap< Guid, qint32 > > notebookCounts`
- `Optional< QMap< Guid, qint32 > > tagCounts`
- `Optional< qint32 > trashCount`

### 7.38.1 Detailed Description

A data structure representing the number of notes for each notebook and tag with a non-zero set of applicable notes.

### 7.38.2 Member Function Documentation

#### 7.38.2.1 operator"!="()

```
bool qevercloud::NoteCollectionCounts::operator!=(
    const NoteCollectionCounts & other ) const [inline]
```

### 7.38.2.2 operator==( )

```
bool qevercloud::NoteCollectionCounts::operator== (
    const NoteCollectionCounts & other ) const [inline]
```

## 7.38.3 Member Data Documentation

### 7.38.3.1 notebookCounts

```
Optional< QMap< Guid, qint32 > > qevercloud::NoteCollectionCounts::notebookCounts
```

A mapping from the [Notebook](#) GUID to the number of notes (from some selection) that are in the corresponding notebook.

### 7.38.3.2 tagCounts

```
Optional< QMap< Guid, qint32 > > qevercloud::NoteCollectionCounts::tagCounts
```

A mapping from the [Tag](#) GUID to the number of notes (from some selection) that have the corresponding tag.

### 7.38.3.3 trashCount

```
Optional< qint32 > qevercloud::NoteCollectionCounts::trashCount
```

If this is set, then this is the number of notes that are in the trash. If this is not set, then the number of notes in the trash hasn't been reported. (I.e. if there are no notes in the trash, this will be set to 0.)

## 7.39 qevercloud::NoteEmailParameters Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [NoteEmailParameters](#) &other) const
- bool [operator!=](#) (const [NoteEmailParameters](#) &other) const

### Public Attributes

- [Optional](#)< QString > [guid](#)
- [Optional](#)< [Note](#) > [note](#)
- [Optional](#)< QStringList > [toAddresses](#)
- [Optional](#)< QStringList > [ccAddresses](#)
- [Optional](#)< QString > [subject](#)
- [Optional](#)< QString > [message](#)

### 7.39.1 Detailed Description

Parameters that must be given to the [NoteStore](#) emailNote call. These allow the caller to specify the note to send, the recipient addresses, etc.

### 7.39.2 Member Function Documentation

#### 7.39.2.1 operator!=(())

```
bool qevercloud::NoteEmailParameters::operator!= (
    const NoteEmailParameters & other ) const [inline]
```

#### 7.39.2.2 operator==(())

```
bool qevercloud::NoteEmailParameters::operator== (
    const NoteEmailParameters & other ) const [inline]
```

### 7.39.3 Member Data Documentation

#### 7.39.3.1 ccAddresses

```
Optional< QStringList > qevercloud::NoteEmailParameters::ccAddresses
```

If provided, this should contain a list of the SMTP email addresses that should be included in the "Cc:" line of the email. Callers must specify at least one "to" or "cc" email address.

#### 7.39.3.2 guid

```
Optional< QString > qevercloud::NoteEmailParameters::guid
```

If set, this must be the GUID of a note within the user's account that should be retrieved from the service and sent as email. If not set, the 'note' field must be provided instead.

#### 7.39.3.3 message

```
Optional< QString > qevercloud::NoteEmailParameters::message
```

If provided, this is additional personal text that should be included into the email as a message from the owner to the recipient(s).

#### 7.39.3.4 note

`Optional< Note > qevercloud::NoteEmailParameters::note`

If the 'guid' field is not set, this field must be provided, including the full contents of the note note (and all of its Resources) to send. This can be used for a [Note](#) that has not been created in the service, for example by a local client with local notes.

#### 7.39.3.5 subject

`Optional< QString > qevercloud::NoteEmailParameters::subject`

If provided, this should contain the subject line of the email that will be sent. If not provided, the title of the note will be used as the subject of the email.

#### 7.39.3.6 toAddresses

`Optional< QStringList > qevercloud::NoteEmailParameters::toAddresses`

If provided, this should contain a list of the SMTP email addresses that should be included in the "To:" line of the email. Callers must specify at least one "to" or "cc" email address.

## 7.40 qevercloud::NoteFilter Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [NoteFilter](#) &other) const
- bool `operator!=` (const [NoteFilter](#) &other) const

### Public Attributes

- `Optional< qint32 > order`
- `Optional< bool > ascending`
- `Optional< QString > words`
- `Optional< Guid > notebookGuid`
- `Optional< QList< Guid > > tagGuids`
- `Optional< QString > timeZone`
- `Optional< bool > inactive`
- `Optional< QString > emphasized`

#### 7.40.1 Detailed Description

A list of criteria that are used to indicate which notes are desired from the account. This is used in queries to the [NoteStore](#) to determine which notes should be retrieved.

## 7.40.2 Member Function Documentation

### 7.40.2.1 operator!=(())

```
bool qevercloud::NoteFilter::operator!= (
    const NoteFilter & other ) const [inline]
```

### 7.40.2.2 operator==(())

```
bool qevercloud::NoteFilter::operator== (
    const NoteFilter & other ) const [inline]
```

## 7.40.3 Member Data Documentation

### 7.40.3.1 ascending

```
Optional< bool > qevercloud::NoteFilter::ascending
```

If true, the results will be ascending in the requested sort order. If false, the results will be descending.

### 7.40.3.2 emphasized

```
Optional< QString > qevercloud::NoteFilter::emphasized
```

If present, a search query string that may or may not influence the notes to be returned, both in terms of coverage as well as of order. Think of it as a wish list, not a requirement. Accepts the full search grammar documented in the Evernote API Overview.

### 7.40.3.3 inactive

```
Optional< bool > qevercloud::NoteFilter::inactive
```

If true, then only notes that are not active (i.e. notes in the Trash) will be returned. Otherwise, only active notes will be returned. There is no way to find both active and inactive notes in a single query.

### 7.40.3.4 notebookGuid

```
Optional< Guid > qevercloud::NoteFilter::notebookGuid
```

If present, the Guid of the notebook that must contain the notes.



#### 7.40.3.5 order

```
Optional< qint32 > qevercloud::NoteFilter::order
```

The [NoteSortOrder](#) value indicating what criterion should be used to sort the results of the filter.

#### 7.40.3.6 tagGuids

```
Optional< QList< Guid > > qevercloud::NoteFilter::tagGuids
```

If present, the list of tags (by GUID) that must be present on the notes.

#### 7.40.3.7 timeZone

```
Optional< QString > qevercloud::NoteFilter::timeZone
```

The zone ID for the user, which will be used to interpret any dates or times in the queries that do not include their desired zone information. For example, if a query requests notes created "yesterday", this will be evaluated from the provided time zone, if provided. The format must be encoded as a standard zone ID such as "America/Los\_↵Angeles".

#### 7.40.3.8 words

```
Optional< QString > qevercloud::NoteFilter::words
```

If present, a search query string that will filter the set of notes to be returned. Accepts the full search grammar documented in the Evernote API Overview.

## 7.41 qevercloud::NoteList Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [NoteList](#) &other) const
- bool [operator!=](#) (const [NoteList](#) &other) const

### Public Attributes

- qint32 [startIndex](#)
- qint32 [totalNotes](#)
- QList< [Note](#) > [notes](#)
- Optional< QStringList > [stoppedWords](#)
- Optional< QStringList > [searchedWords](#)
- Optional< qint32 > [updateCount](#)

### 7.41.1 Detailed Description

A small structure for returning a list of notes out of a larger set.

### 7.41.2 Member Function Documentation

#### 7.41.2.1 operator!=(())

```
bool qevercloud::NoteList::operator!= (
    const NoteList & other ) const [inline]
```

#### 7.41.2.2 operator==(())

```
bool qevercloud::NoteList::operator== (
    const NoteList & other ) const [inline]
```

### 7.41.3 Member Data Documentation

#### 7.41.3.1 notes

```
QList< Note > qevercloud::NoteList::notes
```

The list of notes from this range. The Notes will include all metadata (attributes, resources, etc.), but will not include the ENML content of the note or the binary contents of any resources.

#### 7.41.3.2 searchedWords

```
Optional< QStringList > qevercloud::NoteList::searchedWords
```

If the [NoteList](#) was produced using a text based search query that included viable search words or quoted expressions, this will include a list of those words. Any stopped words will not be included in this list.

#### 7.41.3.3 startIndex

```
qint32 qevercloud::NoteList::startIndex
```

The starting index within the overall set of notes. This is also the number of notes that are "before" this list in the set.

### 7.41.3.4 stoppedWords

```
Optional< QStringList > qevercloud::NoteList::stoppedWords
```

If the [NoteList](#) was produced using a text based search query that included words that are not indexed or searched by the service, this will include a list of those ignored words.

### 7.41.3.5 totalNotes

```
qint32 qevercloud::NoteList::totalNotes
```

The number of notes in the larger set. This can be used to calculate how many notes are "after" this note in the set. (I.e. remaining = totalNotes - (startIndex + notes.length) )

### 7.41.3.6 updateCount

```
Optional< qint32 > qevercloud::NoteList::updateCount
```

Indicates the total number of transactions that have been committed within the account. This reflects (for example) the number of discrete additions or modifications that have been made to the data in this account (tags, notes, resources, etc.). This number is the "high water mark" for Update Sequence Numbers (USN) within the account.

## 7.42 qevercloud::NoteMetadata Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [NoteMetadata](#) &other) const
- bool [operator!=](#) (const [NoteMetadata](#) &other) const

### Public Attributes

- [Guid](#) [guid](#)
- [Optional](#)< [QString](#) > [title](#)
- [Optional](#)< [qint32](#) > [contentLength](#)
- [Optional](#)< [Timestamp](#) > [created](#)
- [Optional](#)< [Timestamp](#) > [updated](#)
- [Optional](#)< [Timestamp](#) > [deleted](#)
- [Optional](#)< [qint32](#) > [updateSequenceNum](#)
- [Optional](#)< [QString](#) > [notebookGuid](#)
- [Optional](#)< [QList](#)< [Guid](#) > > [tagGuids](#)
- [Optional](#)< [NoteAttributes](#) > [attributes](#)
- [Optional](#)< [QString](#) > [largestResourceMime](#)
- [Optional](#)< [qint32](#) > [largestResourceSize](#)

### 7.42.1 Detailed Description

This structure is used in the set of results returned by the `findNotesMetadata` function. It represents the high-level information about a single [Note](#), without some of the larger deep structure. This allows for the information about a list of Notes to be returned relatively quickly with less marshalling and data transfer to remote clients. Most fields in this structure are identical to the corresponding field in the [Note](#) structure, with the exception of:

### 7.42.2 Member Function Documentation

#### 7.42.2.1 `operator!=()`

```
bool qevercloud::NoteMetadata::operator!= (
    const NoteMetadata & other ) const [inline]
```

#### 7.42.2.2 `operator==()`

```
bool qevercloud::NoteMetadata::operator== (
    const NoteMetadata & other ) const [inline]
```

### 7.42.3 Member Data Documentation

#### 7.42.3.1 `attributes`

```
Optional< NoteAttributes > qevercloud::NoteMetadata::attributes
```

NOT DOCUMENTED

#### 7.42.3.2 `contentLength`

```
Optional< qint32 > qevercloud::NoteMetadata::contentLength
```

NOT DOCUMENTED

#### 7.42.3.3 `created`

```
Optional< Timestamp > qevercloud::NoteMetadata::created
```

NOT DOCUMENTED

#### 7.42.3.4 deleted

`Optional< Timestamp > qevercloud::NoteMetadata::deleted`

NOT DOCUMENTED

#### 7.42.3.5 guid

`Guid qevercloud::NoteMetadata::guid`

NOT DOCUMENTED

#### 7.42.3.6 largestResourceMime

`Optional< QString > qevercloud::NoteMetadata::largestResourceMime`

If set, then this will contain the MIME type of the largest [Resource](#) (in bytes) within the [Note](#). This may be useful, for example, to choose an appropriate icon or thumbnail to represent the [Note](#).

#### 7.42.3.7 largestResourceSize

`Optional< qint32 > qevercloud::NoteMetadata::largestResourceSize`

If set, this will contain the size of the largest [Resource](#) file, in bytes, within the [Note](#). This may be useful, for example, to decide whether to ask the server for a thumbnail to represent the [Note](#).

#### 7.42.3.8 notebookGuid

`Optional< QString > qevercloud::NoteMetadata::notebookGuid`

NOT DOCUMENTED

#### 7.42.3.9 tagGuids

`Optional< QList< Guid > > qevercloud::NoteMetadata::tagGuids`

NOT DOCUMENTED

#### 7.42.3.10 title

`Optional< QString > qevercloud::NoteMetadata::title`

NOT DOCUMENTED

#### 7.42.3.11 updated

`Optional< Timestamp > qevercloud::NoteMetadata::updated`

NOT DOCUMENTED

#### 7.42.3.12 updateSequenceNum

`Optional< qint32 > qevercloud::NoteMetadata::updateSequenceNum`

NOT DOCUMENTED

### 7.43 qevercloud::NotesMetadataList Struct Reference

```
#include <types.h>
```

#### Public Member Functions

- bool `operator==` (const [NotesMetadataList](#) &other) const
- bool `operator!=` (const [NotesMetadataList](#) &other) const

#### Public Attributes

- qint32 [startIndex](#)
- qint32 [totalNotes](#)
- QList< [NoteMetadata](#) > [notes](#)
- Optional< QStringList > [stoppedWords](#)
- Optional< QStringList > [searchedWords](#)
- Optional< qint32 > [updateCount](#)

#### 7.43.1 Detailed Description

This structure is returned from calls to the `findNotesMetadata` function to give the high-level metadata about a subset of Notes that are found to match a specified [NoteFilter](#) in a search.

#### 7.43.2 Member Function Documentation

##### 7.43.2.1 operator"!=()

```
bool qevercloud::NotesMetadataList::operator!= (  
    const NotesMetadataList & other ) const [inline]
```

### 7.43.2.2 operator==( )

```
bool qevercloud::NotesMetadataList::operator== (
    const NotesMetadataList & other ) const [inline]
```

## 7.43.3 Member Data Documentation

### 7.43.3.1 notes

```
QList< NoteMetadata > qevercloud::NotesMetadataList::notes
```

The list of metadata for Notes in this range. The set of optional fields that are set in each metadata structure will depend on the [NotesMetadataResultSpec](#) provided by the caller when the search was performed. Only the 'guid' field will be guaranteed to be set in each [Note](#).

### 7.43.3.2 searchedWords

```
Optional< QStringList > qevercloud::NotesMetadataList::searchedWords
```

If the [NoteList](#) was produced using a text based search query that included viable search words or quoted expressions, this will include a list of those words. Any stopped words will not be included in this list.

### 7.43.3.3 startIndex

```
qint32 qevercloud::NotesMetadataList::startIndex
```

The starting index within the overall set of notes. This is also the number of notes that are "before" this list in the set.

### 7.43.3.4 stoppedWords

```
Optional< QStringList > qevercloud::NotesMetadataList::stoppedWords
```

If the [NoteList](#) was produced using a text based search query that included words that are not indexed or searched by the service, this will include a list of those ignored words.

### 7.43.3.5 totalNotes

```
qint32 qevercloud::NotesMetadataList::totalNotes
```

The number of notes in the larger set. This can be used to calculate how many notes are "after" this note in the set. (I.e. remaining = totalNotes - (startIndex + notes.length) )

### 7.43.3.6 updateCount

`Optional< qint32 > qevercloud::NotesMetadataList::updateCount`

Indicates the total number of transactions that have been committed within the account. This reflects (for example) the number of discrete additions or modifications that have been made to the data in this account (tags, notes, resources, etc.). This number is the "high water mark" for Update Sequence Numbers (USN) within the account.

## 7.44 qevercloud::NotesMetadataResultSpec Struct Reference

```
#include <types.h>
```

### Public Member Functions

- `bool operator==` (const [NotesMetadataResultSpec](#) &other) const
- `bool operator!=` (const [NotesMetadataResultSpec](#) &other) const

### Public Attributes

- `Optional< bool > includeTitle`
- `Optional< bool > includeContentLength`
- `Optional< bool > includeCreated`
- `Optional< bool > includeUpdated`
- `Optional< bool > includeDeleted`
- `Optional< bool > includeUpdateSequenceNum`
- `Optional< bool > includeNotebookGuid`
- `Optional< bool > includeTagGuids`
- `Optional< bool > includeAttributes`
- `Optional< bool > includeLargestResourceMime`
- `Optional< bool > includeLargestResourceSize`

### 7.44.1 Detailed Description

This structure is provided to the `findNotesMetadata` function to specify the subset of fields that should be included in each [NoteMetadata](#) element that is returned in the [NotesMetadataList](#). Each field on this structure is a boolean flag that indicates whether the corresponding field should be included in the [NoteMetadata](#) structure when it is returned. For example, if the 'includeTitle' field is set on this structure when calling `findNotesMetadata`, then each [NoteMetadata](#) in the list should have its 'title' field set. If one of the fields in this spec is not set, then it will be treated as 'false' by the server, so the default behavior is to include nothing in replies (but the mandatory GUID)

### 7.44.2 Member Function Documentation



#### 7.44.2.1 operator!=(())

```
bool qevercloud::NotesMetadataResultSpec::operator!= (
    const NotesMetadataResultSpec & other ) const [inline]
```

#### 7.44.2.2 operator==(())

```
bool qevercloud::NotesMetadataResultSpec::operator== (
    const NotesMetadataResultSpec & other ) const [inline]
```

### 7.44.3 Member Data Documentation

#### 7.44.3.1 includeAttributes

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeAttributes`

NOT DOCUMENTED

#### 7.44.3.2 includeContentLength

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeContentLength`

NOT DOCUMENTED

#### 7.44.3.3 includeCreated

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeCreated`

NOT DOCUMENTED

#### 7.44.3.4 includeDeleted

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeDeleted`

NOT DOCUMENTED

#### 7.44.3.5 includeLargestResourceMime

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeLargestResourceMime`

NOT DOCUMENTED

#### 7.44.3.6 includeLargestResourceSize

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeLargestResourceSize`

NOT DOCUMENTED

#### 7.44.3.7 includeNotebookGuid

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeNotebookGuid`

NOT DOCUMENTED

#### 7.44.3.8 includeTagGuids

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeTagGuids`

NOT DOCUMENTED

#### 7.44.3.9 includeTitle

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeTitle`

NOT DOCUMENTED

#### 7.44.3.10 includeUpdated

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeUpdated`

NOT DOCUMENTED

#### 7.44.3.11 includeUpdateSequenceNum

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeUpdateSequenceNum`

NOT DOCUMENTED

### 7.45 qevercloud::NoteSortOrder Struct Reference

```
#include <types.h>
```

#### Public Types

- enum `type` {  
    `CREATED` = 1, `UPDATED` = 2, `RELEVANCE` = 3, `UPDATE_SEQUENCE_NUMBER` = 4,  
    `TITLE` = 5 }

### 7.45.1 Detailed Description

This enumeration defines the possible sort ordering for notes when they are returned from a search result.

### 7.45.2 Member Enumeration Documentation

#### 7.45.2.1 type

```
enum qevercloud::NoteSortOrder::type
```

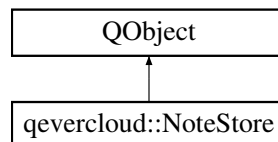
Enumerator

CREATED	
UPDATED	
RELEVANCE	
UPDATE_SEQUENCE_NUMBER	
TITLE	

## 7.46 qevercloud::NoteStore Class Reference

```
#include <services.h>
```

Inheritance diagram for qevercloud::NoteStore:



### Public Member Functions

- [NoteStore](#) (QString [noteStoreUrl](#)=QString(), QString [authenticationToken](#)=QString(), QObject \*parent=0)
- [NoteStore](#) (QObject \*parent)
- void [setNoteStoreUrl](#) (QString [noteStoreUrl](#))
- QString [noteStoreUrl](#) ()
- void [setAuthenticationToken](#) (QString [authenticationToken](#))
- QString [authenticationToken](#) ()
- [SyncState](#) [getSyncState](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getSyncStateAsync](#) (QString [authenticationToken](#)=QString())
- [SyncState](#) [getSyncStateWithMetrics](#) (const [ClientUsageMetrics](#) &clientMetrics, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getSyncStateWithMetricsAsync](#) (const [ClientUsageMetrics](#) &clientMetrics, QString [authenticationToken](#)=QString())

- [SyncChunk](#) [getSyncChunk](#) (qint32 afterUSN, qint32 maxEntries, bool fullSyncOnly, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getSyncChunkAsync](#) (qint32 afterUSN, qint32 maxEntries, bool fullSyncOnly, QString [authenticationToken](#)=QString())
- [SyncChunk](#) [getFilteredSyncChunk](#) (qint32 afterUSN, qint32 maxEntries, const [SyncChunkFilter](#) &filter, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getFilteredSyncChunkAsync](#) (qint32 afterUSN, qint32 maxEntries, const [SyncChunkFilter](#) &filter, QString [authenticationToken](#)=QString())
- [SyncState](#) [getLinkedNotebookSyncState](#) (const [LinkedNotebook](#) &linkedNotebook, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getLinkedNotebookSyncStateAsync](#) (const [LinkedNotebook](#) &linkedNotebook, QString [authenticationToken](#)=QString())
- [SyncChunk](#) [getLinkedNotebookSyncChunk](#) (const [LinkedNotebook](#) &linkedNotebook, qint32 afterUSN, qint32 maxEntries, bool fullSyncOnly, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getLinkedNotebookSyncChunkAsync](#) (const [LinkedNotebook](#) &linkedNotebook, qint32 afterUSN, qint32 maxEntries, bool fullSyncOnly, QString [authenticationToken](#)=QString())
- [QList](#)< [Notebook](#) > [listNotebooks](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [listNotebooksAsync](#) (QString [authenticationToken](#)=QString())
- [Notebook](#) [getNotebook](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getNotebookAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [Notebook](#) [getDefaultNotebook](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getDefaultNotebookAsync](#) (QString [authenticationToken](#)=QString())
- [Notebook](#) [createNotebook](#) (const [Notebook](#) &notebook, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [createNotebookAsync](#) (const [Notebook](#) &notebook, QString [authenticationToken](#)=QString())
- qint32 [updateNotebook](#) (const [Notebook](#) &notebook, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [updateNotebookAsync](#) (const [Notebook](#) &notebook, QString [authenticationToken](#)=QString())
- qint32 [expungeNotebook](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [expungeNotebookAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [QList](#)< [Tag](#) > [listTags](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [listTagsAsync](#) (QString [authenticationToken](#)=QString())
- [QList](#)< [Tag](#) > [listTagsByNotebook](#) (Guid notebookGuid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [listTagsByNotebookAsync](#) (Guid notebookGuid, QString [authenticationToken](#)=QString())
- [Tag](#) [getTag](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getTagAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [Tag](#) [createTag](#) (const [Tag](#) &tag, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [createTagAsync](#) (const [Tag](#) &tag, QString [authenticationToken](#)=QString())
- qint32 [updateTag](#) (const [Tag](#) &tag, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [updateTagAsync](#) (const [Tag](#) &tag, QString [authenticationToken](#)=QString())
- void [untagAll](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [untagAllAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- qint32 [expungeTag](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [expungeTagAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [QList](#)< [SavedSearch](#) > [listSearches](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [listSearchesAsync](#) (QString [authenticationToken](#)=QString())
- [SavedSearch](#) [getSearch](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getSearchAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [SavedSearch](#) [createSearch](#) (const [SavedSearch](#) &search, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [createSearchAsync](#) (const [SavedSearch](#) &search, QString [authenticationToken](#)=QString())
- qint32 [updateSearch](#) (const [SavedSearch](#) &search, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [updateSearchAsync](#) (const [SavedSearch](#) &search, QString [authenticationToken](#)=QString())
- qint32 [expungeSearch](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [expungeSearchAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [NoteList](#) [findNotes](#) (const [NoteFilter](#) &filter, qint32 offset, qint32 maxNotes, QString [authenticationToken](#)=QString())

- [AsyncResult](#) \* [findNotesAsync](#) (const [NoteFilter](#) &filter, qint32 offset, qint32 maxNotes, [QString authenticationToken=QString\(\)](#))
- qint32 [findNoteOffset](#) (const [NoteFilter](#) &filter, [Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [findNoteOffsetAsync](#) (const [NoteFilter](#) &filter, [Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [NotesMetadataList](#) [findNotesMetadata](#) (const [NoteFilter](#) &filter, qint32 offset, qint32 maxNotes, const [NotesMetadataResultSpec](#) &resultSpec, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [findNotesMetadataAsync](#) (const [NoteFilter](#) &filter, qint32 offset, qint32 maxNotes, const [NotesMetadataResultSpec](#) &resultSpec, [QString authenticationToken=QString\(\)](#))
- [NoteCollectionCounts](#) [findNoteCounts](#) (const [NoteFilter](#) &filter, bool withTrash, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [findNoteCountsAsync](#) (const [NoteFilter](#) &filter, bool withTrash, [QString authenticationToken=QString\(\)](#))
- [Note](#) [getNote](#) ([Guid](#) guid, bool withContent, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [getNoteAsync](#) ([Guid](#) guid, bool withContent, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [QString authenticationToken=QString\(\)](#))
- [LazyMap](#) [getNoteApplicationData](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [getNoteApplicationDataAsync](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [QString](#) [getNoteApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [getNoteApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString authenticationToken=QString\(\)](#))
- qint32 [setNoteApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [setNoteApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [QString authenticationToken=QString\(\)](#))
- qint32 [unsetNoteApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [unsetNoteApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString authenticationToken=QString\(\)](#))
- [QString](#) [getNoteContent](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [getNoteContentAsync](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [QString](#) [getNoteSearchText](#) ([Guid](#) guid, bool noteOnly, bool tokenizeForIndexing, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [getNoteSearchTextAsync](#) ([Guid](#) guid, bool noteOnly, bool tokenizeForIndexing, [QString authenticationToken=QString\(\)](#))
- [QString](#) [getResourceSearchText](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [getResourceSearchTextAsync](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [QStringList](#) [getNoteTagNames](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [getNoteTagNamesAsync](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [Note](#) [createNote](#) (const [Note](#) &note, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [createNoteAsync](#) (const [Note](#) &note, [QString authenticationToken=QString\(\)](#))
- [Note](#) [updateNote](#) (const [Note](#) &note, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [updateNoteAsync](#) (const [Note](#) &note, [QString authenticationToken=QString\(\)](#))
- qint32 [deleteNote](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [deleteNoteAsync](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- qint32 [expungeNote](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [expungeNoteAsync](#) ([Guid](#) guid, [QString authenticationToken=QString\(\)](#))
- qint32 [expungeNotes](#) ([QList< Guid >](#) noteGuids, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [expungeNotesAsync](#) ([QList< Guid >](#) noteGuids, [QString authenticationToken=QString\(\)](#))
- qint32 [expungeInactiveNotes](#) ([QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [expungeInactiveNotesAsync](#) ([QString authenticationToken=QString\(\)](#))
- [Note](#) [copyNote](#) ([Guid](#) noteGuid, [Guid](#) toNotebookGuid, [QString authenticationToken=QString\(\)](#))
- [AsyncResult](#) \* [copyNoteAsync](#) ([Guid](#) noteGuid, [Guid](#) toNotebookGuid, [QString authenticationToken=QString\(\)](#))
- [QList< NoteVersionId >](#) [listNoteVersions](#) ([Guid](#) noteGuid, [QString authenticationToken=QString\(\)](#))

- [AsyncResult](#) \* [listNoteVersionsAsync](#) ([Guid](#) noteGuid, [QString](#) authenticationToken=[QString](#)())
- [Note](#) [getNoteVersion](#) ([Guid](#) noteGuid, [qint32](#) updateSequenceNum, [bool](#) withResourcesData, [bool](#) withResourcesRecognition, [bool](#) withResourcesAlternateData, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getNoteVersionAsync](#) ([Guid](#) noteGuid, [qint32](#) updateSequenceNum, [bool](#) withResourcesData, [bool](#) withResourcesRecognition, [bool](#) withResourcesAlternateData, [QString](#) authenticationToken=[QString](#)())
- [Resource](#) [getResource](#) ([Guid](#) guid, [bool](#) withData, [bool](#) withRecognition, [bool](#) withAttributes, [bool](#) withAlternateData, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceAsync](#) ([Guid](#) guid, [bool](#) withData, [bool](#) withRecognition, [bool](#) withAttributes, [bool](#) withAlternateData, [QString](#) authenticationToken=[QString](#)())
- [LazyMap](#) [getResourceApplicationData](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceApplicationDataAsync](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [QString](#) [getResourceApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString](#) authenticationToken=[QString](#)())
- [qint32](#) [setResourceApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [setResourceApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [QString](#) authenticationToken=[QString](#)())
- [qint32](#) [unsetResourceApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [unsetResourceApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString](#) authenticationToken=[QString](#)())
- [qint32](#) [updateResource](#) (const [Resource](#) &resource, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [updateResourceAsync](#) (const [Resource](#) &resource, [QString](#) authenticationToken=[QString](#)())
- [QByteArray](#) [getResourceData](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceDataAsync](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [Resource](#) [getResourceByHash](#) ([Guid](#) noteGuid, [QByteArray](#) contentHash, [bool](#) withData, [bool](#) withRecognition, [bool](#) withAlternateData, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceByHashAsync](#) ([Guid](#) noteGuid, [QByteArray](#) contentHash, [bool](#) withData, [bool](#) withRecognition, [bool](#) withAlternateData, [QString](#) authenticationToken=[QString](#)())
- [QByteArray](#) [getResourceRecognition](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceRecognitionAsync](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [QByteArray](#) [getResourceAlternateData](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceAlternateDataAsync](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [ResourceAttributes](#) [getResourceAttributes](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [getResourceAttributesAsync](#) ([Guid](#) guid, [QString](#) authenticationToken=[QString](#)())
- [Notebook](#) [getPublicNotebook](#) ([UserID](#) userId, [QString](#) publicUri)
- [AsyncResult](#) \* [getPublicNotebookAsync](#) ([UserID](#) userId, [QString](#) publicUri)
- [SharedNotebook](#) [createSharedNotebook](#) (const [SharedNotebook](#) &sharedNotebook, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [createSharedNotebookAsync](#) (const [SharedNotebook](#) &sharedNotebook, [QString](#) authenticationToken=[QString](#)())
- [qint32](#) [updateSharedNotebook](#) (const [SharedNotebook](#) &sharedNotebook, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [updateSharedNotebookAsync](#) (const [SharedNotebook](#) &sharedNotebook, [QString](#) authenticationToken=[QString](#)())
- [qint32](#) [setSharedNotebookRecipientSettings](#) ([qint64](#) sharedNotebookId, const [SharedNotebookRecipientSettings](#) &recipientSettings, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [setSharedNotebookRecipientSettingsAsync](#) ([qint64](#) sharedNotebookId, const [SharedNotebookRecipientSettings](#) &recipientSettings, [QString](#) authenticationToken=[QString](#)())
- [qint32](#) [sendMessageToSharedNotebookMembers](#) ([Guid](#) notebookGuid, [QString](#) messageText, [QStringList](#) recipients, [QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [sendMessageToSharedNotebookMembersAsync](#) ([Guid](#) notebookGuid, [QString](#) messageText, [QStringList](#) recipients, [QString](#) authenticationToken=[QString](#)())
- [QList](#)< [SharedNotebook](#) > [listSharedNotebooks](#) ([QString](#) authenticationToken=[QString](#)())
- [AsyncResult](#) \* [listSharedNotebooksAsync](#) ([QString](#) authenticationToken=[QString](#)())

- qint32 [expungeSharedNotebooks](#) (QList< qint64 > sharedNotebookIds, QString [authenticationToken](#)=Q←String())
- [AsyncResult](#) \* [expungeSharedNotebooksAsync](#) (QList< qint64 > sharedNotebookIds, QString [authenticationToken](#)=Q←String())
- [LinkedNotebook](#) [createLinkedNotebook](#) (const [LinkedNotebook](#) &linkedNotebook, QString [authenticationToken](#)=Q←String())
- [AsyncResult](#) \* [createLinkedNotebookAsync](#) (const [LinkedNotebook](#) &linkedNotebook, QString [authenticationToken](#)=Q←String())
- qint32 [updateLinkedNotebook](#) (const [LinkedNotebook](#) &linkedNotebook, QString [authenticationToken](#)=Q←String())
- [AsyncResult](#) \* [updateLinkedNotebookAsync](#) (const [LinkedNotebook](#) &linkedNotebook, QString [authenticationToken](#)=Q←String())
- QList< [LinkedNotebook](#) > [listLinkedNotebooks](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [listLinkedNotebooksAsync](#) (QString [authenticationToken](#)=QString())
- qint32 [expungeLinkedNotebook](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [expungeLinkedNotebookAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AuthenticationResult](#) [authenticateToSharedNotebook](#) (QString shareKey, QString [authenticationToken](#)=Q←String())
- [AsyncResult](#) \* [authenticateToSharedNotebookAsync](#) (QString shareKey, QString [authenticationToken](#)=Q←String())
- [SharedNotebook](#) [getSharedNotebookByAuth](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getSharedNotebookByAuthAsync](#) (QString [authenticationToken](#)=QString())
- void [emailNote](#) (const [NoteEmailParameters](#) &parameters, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [emailNoteAsync](#) (const [NoteEmailParameters](#) &parameters, QString [authenticationToken](#)=Q←String())
- QString [shareNote](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [shareNoteAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- void [stopSharingNote](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [stopSharingNoteAsync](#) (Guid guid, QString [authenticationToken](#)=QString())
- [AuthenticationResult](#) [authenticateToSharedNote](#) (QString guid, QString noteKey, QString [authenticationToken](#)=Q←String())
- [AsyncResult](#) \* [authenticateToSharedNoteAsync](#) (QString guid, QString noteKey, QString [authenticationToken](#)=Q←String())
- [RelatedResult](#) [findRelated](#) (const [RelatedQuery](#) &query, const [RelatedResultSpec](#) &resultSpec, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [findRelatedAsync](#) (const [RelatedQuery](#) &query, const [RelatedResultSpec](#) &resultSpec, Q←String [authenticationToken](#)=QString())

### 7.46.1 Detailed Description

Service: [NoteStore](#)

The [NoteStore](#) service is used by EDAM clients to exchange information about the collection of notes in an account. This is primarily used for synchronization, but could also be used by a "thin" client without a full local cache.

All functions take an "authenticationToken" parameter, which is the value returned by the [UserStore](#) which permits access to the account. This parameter is mandatory for all functions.

Calls which require an authenticationToken may throw an [EDAMUserException](#) for the following reasons:

- AUTH\_EXPIRED "authenticationToken" - token has expired
- BAD\_DATA\_FORMAT "authenticationToken" - token is malformed
- DATA\_REQUIRED "authenticationToken" - token is empty
- INVALID\_AUTH "authenticationToken" - token signature is invalid

## 7.46.2 Constructor & Destructor Documentation

### 7.46.2.1 NoteStore() [1/2]

```
qevercloud::NoteStore::NoteStore (
    QString noteStoreUrl = QString(),
    QString authenticationToken = QString(),
    QObject * parent = 0 ) [explicit]
```

### 7.46.2.2 NoteStore() [2/2]

```
qevercloud::NoteStore::NoteStore (
    QObject * parent ) [explicit]
```

## 7.46.3 Member Function Documentation

### 7.46.3.1 authenticateToSharedNote()

```
AuthenticationResult qevercloud::NoteStore::authenticateToSharedNote (
    QString guid,
    QString noteKey,
    QString authenticationToken = QString() )
```

Asks the service to produce an authentication token that can be used to access the contents of a single [Note](#) which was individually shared from someone's account. This authenticationToken can be used with the various other [NoteStore](#) calls to find and retrieve the [Note](#) and its directly-referenced children.

#### Parameters

<i>guid</i>	The GUID identifying this <a href="#">Note</a> on this shard.
<i>noteKey</i>	The 'noteKey' identifier from the <a href="#">Note</a> that was originally created via a call to <a href="#">shareNote()</a> and then given to a recipient to access.
<i>authenticationToken</i>	An optional authenticationToken that identifies the user accessing the shared note. This parameter may be required to access some shared notes.



## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "Note" - the <a href="#">Note</a> with that GUID is either not shared, or the noteKey doesn't match the current key for this note</li> <li>• PERMISSION_DENIED "authenticationToken" - an authentication token is required to access this <a href="#">Note</a>, but either no authentication token or a "non-owner" authentication token was provided.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "guid" - the note with that GUID is not found</li> </ul>
<a href="#"><i>EDAMSystemException</i></a>	<ul style="list-style-type: none"> <li>• TAKEN_DOWN "Note" - The specified shared note is taken down (for all requesters).</li> <li>• TAKEN_DOWN "Country" - The specified shared note is taken down for the requester because of an IP-based country lookup.</li> </ul>

## 7.46.3.2 authenticateToSharedNoteAsync()

```
AsyncResult* qevercloud::NoteStore::authenticateToSharedNoteAsync (
    QString guid,
    QString noteKey,
    QString authenticationToken = QString() )
```

Asynchronous version of [authenticateToSharedNote](#)

## 7.46.3.3 authenticateToSharedNotebook()

```
AuthenticationResult qevercloud::NoteStore::authenticateToSharedNotebook (
    QString shareKey,
    QString authenticationToken = QString() )
```

Asks the service to produce an authentication token that can be used to access the contents of a shared notebook from someone else's account. This authenticationToken can be used with the various other [NoteStore](#) calls to find and retrieve notes, and if the permissions in the shared notebook are sufficient, to make changes to the contents of the notebook.

## Parameters

<i>shareKey</i>	The 'shareKey' identifier from the <a href="#">SharedNotebook</a> that was granted to some recipient. This string internally encodes the notebook identifier and a security signature.
<i>authenticationToken</i>	If a non-empty string is provided, this is the full user-based authentication token that identifies the user who is currently logged in and trying to access the shared notebook. This may be required if the notebook was created with 'requireLogin'. If this string is empty, the service will attempt to authenticate to the shared notebook without any logged in user.

## Exceptions

<a href="#"><i>EDAMSystemException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "shareKey" - invalid shareKey string</li> <li>• INVALID_AUTH "shareKey" - bad signature on shareKey string</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "SharedNotebook.id" - the shared notebook no longer exists</li> </ul>
<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "authenticationToken" - the share requires login, and no valid authentication token was provided.</li> <li>• PERMISSION_DENIED "SharedNotebook.username" - share requires login, and another username has already been bound to this notebook.</li> </ul>

7.46.3.4 `authenticateToSharedNotebookAsync()`

```
AsyncResult* qevercloud::NoteStore::authenticateToSharedNotebookAsync (
    QString shareKey,
    QString authenticationToken = QString() )
```

Asynchronous version of [authenticateToSharedNotebook](#)

7.46.3.5 `authenticationToken()`

```
QString qevercloud::NoteStore::authenticationToken ( ) [inline]
```

7.46.3.6 `copyNote()`

```
Note qevercloud::NoteStore::copyNote (
    Guid noteGuid,
    Guid toNotebookGuid,
    QString authenticationToken = QString() )
```

Performs a deep copy of the [Note](#) with the provided GUID 'noteGuid' into the [Notebook](#) with the provided GUID 'toNotebookGuid'. The caller must be the owner of both the [Note](#) and the [Notebook](#). This creates a new [Note](#) in the destination [Notebook](#) with new content and Resources that match all of the content and Resources from the original [Note](#), but with new GUID identifiers. The original [Note](#) is not modified by this operation. The copied note is considered as an "upload" for the purpose of upload transfer limit calculation, so its size is added to the upload count for the owner.

## Parameters

<i>noteGuid</i>	The GUID of the <a href="#">Note</a> to copy.
<i>toNotebookGuid</i>	The GUID of the <a href="#">Notebook</a> that should receive the new <a href="#">Note</a> .

## Returns

The metadata for the new [Note](#) that was created. This will include the new GUID for this [Note](#) (and any copied Resources), but will not include the content body or the binary bodies of any Resources.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>LIMIT_REACHED "Note" - at max number per account</li> <li>PERMISSION_DENIED "Notebook.guid" - destination not owned by user</li> <li>PERMISSION_DENIED "Note" - user doesn't own</li> <li>QUOTA_REACHED "Accounting.uploadLimit" - note exceeds upload quota</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Notebook.guid" - not found, by GUID</li> </ul>

## 7.46.3.7 copyNoteAsync()

```
AsyncResult* qevercloud::NoteStore::copyNoteAsync (
    Guid noteGuid,
    Guid toNotebookGuid,
    QString authenticationToken = QString() )
```

Asynchronous version of [copyNote](#)

## 7.46.3.8 createLinkedNotebook()

```
LinkedNotebook qevercloud::NoteStore::createLinkedNotebook (
    const LinkedNotebook & linkedNotebook,
    QString authenticationToken = QString() )
```

Asks the service to make a linked notebook with the provided name, username of the owner and identifiers provided. A linked notebook can be either a link to a public notebook or to a private shared notebook.

## Parameters

<i>linkedNotebook</i>	The desired fields for the linked notebook must be provided on this object. The name of the linked notebook must be set. Either a username uri or a shard id and share key must be provided otherwise a <a href="#">EDAMUserException</a> is thrown.
-----------------------	--

## Returns

The newly created [LinkedNotebook](#). The server-side id will be saved in this object's 'id' field.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "LinkedNotebook.name" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "LinkedNotebook.username" - bad username format</li> <li>• BAD_DATA_FORMAT "LinkedNotebook.uri" - if public notebook set but bad uri</li> <li>• BAD_DATA_FORMAT "LinkedNotebook.shareKey" - if private notebook set but bad shareKey</li> <li>• DATA_REQUIRED "LinkedNotebook.shardId" - if private notebook but shard id not provided</li> </ul>
-----------------------------------	---

**7.46.3.9 createLinkedNotebookAsync()**

```
AsyncResult* qevercloud::NoteStore::createLinkedNotebookAsync (
    const LinkedNotebook & linkedNotebook,
    QString authenticationToken = QString() )
```

Asynchronous version of [createLinkedNotebook](#)

**7.46.3.10 createNote()**

```
Note qevercloud::NoteStore::createNote (
    const Note & note,
    QString authenticationToken = QString() )
```

Asks the service to make a note with the provided set of information.

## Parameters

<i>note</i>	A <a href="#">Note</a> object containing the desired fields to be populated on the service.
-------------	---

## Returns

The newly created [Note](#) from the service. The server-side GUIDs for the [Note](#) and any Resources will be saved in this object.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Note.title" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "Note.content" - invalid length for ENML content</li> <li>• BAD_DATA_FORMAT "Resource.mime" - invalid resource MIME type</li> <li>• BAD_DATA_FORMAT "NoteAttributes.*" - bad resource string</li> <li>• BAD_DATA_FORMAT "ResourceAttributes.*" - bad resource string</li> <li>• DATA_CONFLICT "Note.deleted" - deleted time set on active note</li> <li>• DATA_REQUIRED "Resource.data" - resource data body missing</li> <li>• ENML_VALIDATION "*" - note content doesn't validate against DTD</li> <li>• LIMIT_REACHED "Note" - at max number per account</li> <li>• LIMIT_REACHED "Note.size" - total note size too large</li> <li>• LIMIT_REACHED "Note.resources" - too many resources on <a href="#">Note</a></li> <li>• LIMIT_REACHED "Note.tagGuids" - too many Tags on <a href="#">Note</a></li> <li>• LIMIT_REACHED "Resource.data.size" - resource too large</li> <li>• LIMIT_REACHED "NoteAttribute.*" - attribute string too long</li> <li>• LIMIT_REACHED "ResourceAttribute.*" - attribute string too long</li> <li>• PERMISSION_DENIED "Note.notebookGuid" - NB not owned by user</li> <li>• QUOTA_REACHED "Accounting.uploadLimit" - note exceeds upload quota</li> <li>• BAD_DATA_FORMAT "Tag.name" - <a href="#">Note.tagNames</a> was provided, and one of the specified tags had an invalid length or pattern</li> <li>• LIMIT_REACHED "Tag" - <a href="#">Note.tagNames</a> was provided, and the required new tags would exceed the maximum number per account</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.notebookGuid" - not found, by GUID</li> </ul>

## 7.46.3.11 createNoteAsync()

```
AsyncResult* qevercloud::NoteStore::createNoteAsync (
    const Note & note,
    QString authenticationToken = QString() )
```

Asynchronous version of [createNote](#)

#### 7.46.3.12 createNotebook()

```

Notebook qevercloud::NoteStore::createNotebook (
    const Notebook & notebook,
    QString authenticationToken = QString() )

```

Asks the service to make a notebook with the provided name.

## Parameters

<i>notebook</i>	The desired fields for the notebook must be provided on this object. The name of the notebook must be set, and either the 'active' or 'defaultNotebook' fields may be set by the client at creation. If a notebook exists in the account with the same name (via case-insensitive compare), this will throw an <a href="#">EDAMUserException</a> .
-----------------	--

## Returns

The newly created [Notebook](#). The server-side GUID will be saved in this object's 'guid' field.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Notebook.name" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "Notebook.stack" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "Publishing.uri" - if publishing set but bad uri</li> <li>• BAD_DATA_FORMAT "Publishing.publicDescription" - if too long</li> <li>• DATA_CONFLICT "Notebook.name" - name already in use</li> <li>• DATA_CONFLICT "Publishing.uri" - if URI already in use</li> <li>• DATA_REQUIRED "Publishing.uri" - if publishing set but uri missing</li> <li>• LIMIT_REACHED "Notebook" - at max number of notebooks</li> </ul>
-----------------------------------	---

## 7.46.3.13 createNotebookAsync()

```
AsyncResult* qevercloud::NoteStore::createNotebookAsync (
    const Notebook & notebook,
    QString authenticationToken = QString() )
```

Asynchronous version of [createNotebook](#)

## 7.46.3.14 createSearch()

```
SavedSearch qevercloud::NoteStore::createSearch (
    const SavedSearch & search,
    QString authenticationToken = QString() )
```

Asks the service to make a saved search with a set of information.

## Parameters

<i>search</i>	The desired list of fields for the search are specified in this object. The caller must specify the name and query for the search, and may optionally specify a search scope. The <a href="#">SavedSearch.format</a> field is ignored by the service.
---------------	---

**Returns**

The newly created [SavedSearch](#). The server-side GUID will be saved in this object.

**Exceptions**

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "SavedSearch.name" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "SavedSearch.query" - invalid length</li> <li>• DATA_CONFLICT "SavedSearch.name" - name already in use</li> <li>• LIMIT_REACHED "SavedSearch" - at max number of searches</li> </ul>
-----------------------------------	---

**7.46.3.15 createSearchAsync()**

```
AsyncResult* qevercloud::NoteStore::createSearchAsync (
    const SavedSearch & search,
    QString authenticationToken = QString() )
```

Asynchronous version of [createSearch](#)

**7.46.3.16 createSharedNotebook()**

```
SharedNotebook qevercloud::NoteStore::createSharedNotebook (
    const SharedNotebook & sharedNotebook,
    QString authenticationToken = QString() )
```

Used to construct a shared notebook object. The constructed notebook will contain a "share key" which serve as a unique identifier and access token for a user to access the notebook of the shared notebook owner.

**Parameters**

<i>sharedNotebook</i>	A shared notebook object populated with the email address of the share recipient, the notebook guid and the access permissions. All other attributes of the shared object are ignored. The <a href="#">SharedNotebook.allowPreview</a> field must be explicitly set with either a true or false value.
-----------------------	--

**Returns**

The fully populated [SharedNotebook](#) object including the server assigned share id and shareKey which can both be used to uniquely identify the [SharedNotebook](#).



## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "SharedNotebook.email" - if the email was not valid</li> <li>• BAD_DATA_FORMAT "requireLogin" - if the <a href="#">SharedNotebook.allowPreview</a> field was not set, and the <a href="#">SharedNotebook.requireLogin</a> was also not set or was set to false.</li> <li>• PERMISSION_DENIED "SharedNotebook.recipientSettings" - if recipientSettings is set in the sharedNotebook. Only the recipient can set these values via the setSharedNotebookRecipientSettings method.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• <a href="#">Notebook.guid</a> - if the notebookGuid is not a valid GUID for the user.</li> </ul>

## 7.46.3.17 createSharedNotebookAsync()

```
AsyncResult* qevercloud::NoteStore::createSharedNotebookAsync (
    const SharedNotebook & sharedNotebook,
    QString authenticationToken = QString() )
```

Asynchronous version of [createSharedNotebook](#)

## 7.46.3.18 createTag()

```
Tag qevercloud::NoteStore::createTag (
    const Tag & tag,
    QString authenticationToken = QString() )
```

Asks the service to make a tag with a set of information.

## Parameters

<i>tag</i>	The desired list of fields for the tag are specified in this object. The caller must specify the tag name, and may provide the parentGUID.
------------	--

## Returns

The newly created [Tag](#). The server-side GUID will be saved in this object.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Tag.name" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "Tag.parentGuid" - malformed GUID</li> <li>• DATA_CONFLICT "Tag.name" - name already in use</li> <li>• LIMIT_REACHED "Tag" - at max number of tags</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Tag.parentGuid" - not found, by GUID</li> </ul>

**7.46.3.19 createTagAsync()**

```
AsyncResult* qevercloud::NoteStore::createTagAsync (
    const Tag & tag,
    QString authenticationToken = QString() )
```

Asynchronous version of [createTag](#)

**7.46.3.20 deleteNote()**

```
qint32 qevercloud::NoteStore::deleteNote (
    Guid guid,
    QString authenticationToken = QString() )
```

Moves the note into the trash. The note may still be undeleted, unless it is expunged. This is equivalent to calling [updateNote\(\)](#) after setting [Note.active](#) = false

## Parameters

<i>guid</i>	The GUID of the note to delete.
-------------	---------------------------------

## Returns

The Update Sequence Number for this change within the account.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "Note" - user doesn't have permission to update the note.</li> </ul>
<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• DATA_CONFLICT "Note.guid" - the note is already deleted</li> </ul>

## Exceptions

<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"><li>• "Note.guid" - not found, by GUID</li></ul>
---------------------------------------	--

**7.46.3.21 deleteNoteAsync()**

```
AsyncResult* qevercloud::NoteStore::deleteNoteAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [deleteNote](#)

**7.46.3.22 emailNote()**

```
void qevercloud::NoteStore::emailNote (
    const NoteEmailParameters & parameters,
    QString authenticationToken = QString() )
```

Attempts to send a single note to one or more email recipients.

NOTE: This function is generally not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

## Parameters

<i>authenticationToken</i>	The note will be sent as the user logged in via this token, using that user's registered email address. If the authenticated user doesn't have permission to read that note, the emailing will fail.
<i>parameters</i>	The note must be specified either by GUID (in which case it will be sent using the existing data in the service), or else the full <a href="#">Note</a> must be passed to this call. This also specifies the additional email fields that will be used in the email.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• LIMIT_REACHED "NoteEmailParameters.toAddresses" - The email can't be sent because this would exceed the user's daily email limit.</li> <li>• BAD_DATA_FORMAT "(email address)" - email address malformed</li> <li>• DATA_REQUIRED "NoteEmailParameters.toAddresses" - if there are no To: or Cc: addresses provided.</li> <li>• DATA_REQUIRED "Note.title" - if the caller provides a <a href="#">Note</a> parameter with no title</li> <li>• DATA_REQUIRED "Note.content" - if the caller provides a <a href="#">Note</a> parameter with no content</li> <li>• ENML_VALIDATION "*" - note content doesn't validate against DTD</li> <li>• DATA_REQUIRED "NoteEmailParameters.note" - if no guid or note provided</li> <li>• PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

**7.46.3.23 emailNoteAsync()**

```
AsyncResult* qevercloud::NoteStore::emailNoteAsync (
    const NoteEmailParameters & parameters,
    QString authenticationToken = QString() )
```

Asynchronous version of [emailNote](#)

**7.46.3.24 expungeInactiveNotes()**

```
qint32 qevercloud::NoteStore::expungeInactiveNotes (
    QString authenticationToken = QString() )
```

Permanently removes all of the Notes that are currently marked as inactive. This is equivalent to "emptying the trash", and these Notes will be gone permanently.

This operation may be relatively slow if the account contains a large number of inactive Notes.

NOTE: This function is not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

**Returns**

The number of notes that were expunged.

### 7.46.3.25 expungeInactiveNotesAsync()

```
AsyncResult* qevercloud::NoteStore::expungeInactiveNotesAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeInactiveNotes](#)

### 7.46.3.26 expungeLinkedNotebook()

```
qint32 qevercloud::NoteStore::expungeLinkedNotebook (
    Guid guid,
    QString authenticationToken = QString() )
```

Permanently expunges the linked notebook from the account.

NOTE: This function is generally not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

#### Parameters

<i>guid</i>	The <a href="#">LinkedNotebook.guid</a> field of the <a href="#">LinkedNotebook</a> to permanently remove from the account.
-------------	---

### 7.46.3.27 expungeLinkedNotebookAsync()

```
AsyncResult* qevercloud::NoteStore::expungeLinkedNotebookAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeLinkedNotebook](#)

### 7.46.3.28 expungeNote()

```
qint32 qevercloud::NoteStore::expungeNote (
    Guid guid,
    QString authenticationToken = QString() )
```

Permanently removes a [Note](#), and all of its Resources, from the service.

NOTE: This function is not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

#### Parameters

<i>guid</i>	The GUID of the note to delete.
-------------	---------------------------------

**Returns**

The Update Sequence Number for this change within the account.

**Exceptions**

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "Note" - user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

**7.46.3.29 expungeNoteAsync()**

```
AsyncResult* qevercloud::NoteStore::expungeNoteAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeNote](#)

**7.46.3.30 expungeNotebook()**

```
qint32 qevercloud::NoteStore::expungeNotebook (
    Guid guid,
    QString authenticationToken = QString() )
```

Permanently removes the notebook from the user's account. After this action, the notebook is no longer available for undeletion, etc. If the notebook contains any Notes, they will be moved to the current default notebook and moved into the trash (i.e. [Note.active=false](#)).

NOTE: This function is generally not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

**Parameters**

<i>guid</i>	The GUID of the notebook to delete.
-------------	-------------------------------------

**Returns**

The Update Sequence Number for this change within the account.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Notebook.guid" - if the parameter is missing</li> <li>• LIMIT_REACHED "Notebook" - trying to expunge the last <a href="#">Notebook</a></li> <li>• PERMISSION_DENIED "Notebook" - private notebook, user doesn't own</li> </ul>
--	--

## 7.46.3.31 expungeNotebookAsync()

```
AsyncResult* qevercloud::NoteStore::expungeNotebookAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeNotebook](#)

## 7.46.3.32 expungeNotes()

```
qint32 qevercloud::NoteStore::expungeNotes (
    QList< Guid > noteGuids,
    QString authenticationToken = QString() )
```

Permanently removes a list of Notes, and all of their Resources, from the service. This should be invoked with a small number of [Note](#) GUIDs (e.g. 100 or less) on each call. To expunge a larger number of notes, call this method multiple times. This should also be used to reduce the number of Notes in a notebook before calling [expungeNotebook\(\)](#) or in the trash before calling [expungeInactiveNotes\(\)](#), since these calls may be prohibitively slow if there are more than a few hundred notes. If an exception is thrown for any of the GUIDs, then none of the notes will be deleted. I.e. this call can be treated as an atomic transaction.

NOTE: This function is not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

## Parameters

<i>noteGuids</i>	The list of GUIDs for the Notes to remove.
------------------	--

## Returns

The account's updateCount at the end of this operation

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "Note" - user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

### 7.46.3.33 expungeNotesAsync()

```
AsyncResult* qevercloud::NoteStore::expungeNotesAsync (
    QList< Guid > noteGuids,
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeNotes](#)

### 7.46.3.34 expungeSearch()

```
qint32 qevercloud::NoteStore::expungeSearch (
    Guid guid,
    QString authenticationToken = QString() )
```

Permanently deletes the saved search with the provided GUID, if present.

NOTE: This function is generally not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

#### Parameters

<i>guid</i>	The GUID of the search to delete.
-------------	-----------------------------------

#### Returns

The Update Sequence Number for this change within the account.

#### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"><li>BAD_DATA_FORMAT "SavedSearch.guid" - if the guid parameter is empty</li><li>PERMISSION_DENIED "SavedSearch" - user doesn't own</li></ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"><li>"SavedSearch.guid" - not found, by GUID</li></ul>

### 7.46.3.35 expungeSearchAsync()

```
AsyncResult* qevercloud::NoteStore::expungeSearchAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeSearch](#)



### 7.46.3.36 expungeSharedNotebooks()

```
qint32 qevercloud::NoteStore::expungeSharedNotebooks (
    QList< qint64 > sharedNotebookIds,
    QString authenticationToken = QString() )
```

Expunges the SharedNotebooks in the user's account using the [SharedNotebook.id](#) as the identifier.

NOTE: This function is generally not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

#### Parameters

<i>sharedNotebookIds</i>	- a list of ShardNotebook.id longs identifying the objects to delete permanently.
--------------------------	---

#### Returns

The account's update sequence number.

### 7.46.3.37 expungeSharedNotebooksAsync()

```
AsyncResult* qevercloud::NoteStore::expungeSharedNotebooksAsync (
    QList< qint64 > sharedNotebookIds,
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeSharedNotebooks](#)

### 7.46.3.38 expungeTag()

```
qint32 qevercloud::NoteStore::expungeTag (
    Guid guid,
    QString authenticationToken = QString() )
```

Permanently deletes the tag with the provided GUID, if present.

NOTE: This function is generally not available to third party applications. Calls will result in an [EDAMUserException](#) with the error code PERMISSION\_DENIED.

#### Parameters

<i>guid</i>	The GUID of the tag to delete.
-------------	--------------------------------

#### Returns

The Update Sequence Number for this change within the account.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Tag.guid" - if the guid parameter is missing</li> <li>• PERMISSION_DENIED "Tag" - user doesn't own tag</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Tag.guid" - tag not found, by GUID</li> </ul>

**7.46.3.39 expungeTagAsync()**

```
AsyncResult* qevercloud::NoteStore::expungeTagAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [expungeTag](#)

**7.46.3.40 findNoteCounts()**

```
NoteCollectionCounts qevercloud::NoteStore::findNoteCounts (
    const NoteFilter & filter,
    bool withTrash,
    QString authenticationToken = QString() )
```

This function is used to determine how many notes are found for each notebook and tag in the user's account, given a current set of filter parameters that determine the current selection. This function will return a structure that gives the note count for each notebook and tag that has at least one note under the requested filter. Any notebook or tag that has zero notes in the filtered set will not be listed in the reply to this function (so they can be assumed to be 0).

## Parameters

<i>authenticationToken</i>	Must be a valid token for the user's account unless the <a href="#">NoteFilter</a> 'notebookGuid' is the GUID of a public notebook.
<i>filter</i>	The note selection filter that is currently being applied. The note counts are to be calculated with this filter applied to the total set of notes in the user's account.
<i>withTrash</i>	If true, then the <a href="#">NoteCollectionCounts.trashCount</a> will be calculated and supplied in the reply. Otherwise, the trash value will be omitted.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "NoteFilter.notebookGuid" - if malformed</li> <li>• BAD_DATA_FORMAT "NoteFilter.notebookGuids" - if any are malformed</li> <li>• BAD_DATA_FORMAT "NoteFilter.words" - if search string too long</li> </ul>
--	---

## Exceptions

<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"><li>• "Notebook.guid" - not found, by GUID</li></ul>
---------------------------------------	--

## 7.46.3.41 findNoteCountsAsync()

```
AsyncResult* qevercloud::NoteStore::findNoteCountsAsync (
    const NoteFilter & filter,
    bool withTrash,
    QString authenticationToken = QString() )
```

Asynchronous version of [findNoteCounts](#)

## 7.46.3.42 findNoteOffset()

```
qint32 qevercloud::NoteStore::findNoteOffset (
    const NoteFilter & filter,
    Guid guid,
    QString authenticationToken = QString() )
```

Finds the position of a note within a sorted subset of all of the user's notes. This may be useful for thin clients that are displaying a paginated listing of a large account, which need to know where a particular note sits in the list without retrieving all notes first.

## Parameters

<i>authenticationToken</i>	Must be a valid token for the user's account unless the <a href="#">NoteFilter</a> 'notebookGuid' is the GUID of a public notebook.
<i>filter</i>	The list of criteria that will constrain the notes to be returned.
<i>guid</i>	The GUID of the note to be retrieved.

## Returns

If the note with the provided GUID is found within the matching note list, this will return the offset of that note within that list (where the first offset is 0). If the note is not found within the set of notes, this will return -1.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "offset" - not between 0 and EDAM_USER_NOTES_MAX</li> <li>• BAD_DATA_FORMAT "maxNotes" - not between 0 and EDAM_USER_NOTES_MAX</li> <li>• BAD_DATA_FORMAT "NoteFilter.notebookGuid" - if malformed</li> <li>• BAD_DATA_FORMAT "NoteFilter.tagGuids" - if any are malformed</li> <li>• BAD_DATA_FORMAT "NoteFilter.words" - if search string too long</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Notebook.guid" - not found, by GUID</li> <li>• "Note.guid" - not found, by GUID</li> </ul>

**7.46.3.43 findNoteOffsetAsync()**

```
AsyncResult* qevercloud::NoteStore::findNoteOffsetAsync (
    const NoteFilter & filter,
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [findNoteOffset](#)

**7.46.3.44 findNotes()**

```
NoteList qevercloud::NoteStore::findNotes (
    const NoteFilter & filter,
    quint32 offset,
    quint32 maxNotes,
    QString authenticationToken = QString() )
```

DEPRECATED. Use findNotesMetadata.

**7.46.3.45 findNotesAsync()**

```
AsyncResult* qevercloud::NoteStore::findNotesAsync (
    const NoteFilter & filter,
    quint32 offset,
    quint32 maxNotes,
    QString authenticationToken = QString() )
```

Asynchronous version of [findNotes](#)

## 7.46.3.46 findNotesMetadata()

```
NotesMetadataList qevercloud::NoteStore::findNotesMetadata (
    const NoteFilter & filter,
    qint32 offset,
    qint32 maxNotes,
    const NotesMetadataResultSpec & resultSpec,
    QString authenticationToken = QString() )
```

Used to find the high-level information about a set of the notes from a user's account based on various criteria specified via a [NoteFilter](#) object.

Web applications that wish to periodically check for new content in a user's Evernote account should consider using webhooks instead of polling this API. See [http://dev.evernote.com/documentation/cloud/chapters/polling\\_notification.php](http://dev.evernote.com/documentation/cloud/chapters/polling_notification.php) for more information.

## Parameters

<i>authenticationToken</i>	Must be a valid token for the user's account unless the <a href="#">NoteFilter</a> 'notebookGuid' is the GUID of a public notebook.
<i>filter</i>	The list of criteria that will constrain the notes to be returned.
<i>offset</i>	The numeric index of the first note to show within the sorted results. The numbering scheme starts with "0". This can be used for pagination.
<i>maxNotes</i>	The maximum notes to return in this query. The service will return a set of notes that is no larger than this number, but may return fewer notes if needed. The <a href="#">NoteList.totalNotes</a> field in the return value will indicate whether there are more values available after the returned set.
<i>resultSpec</i>	This specifies which information should be returned for each matching <a href="#">Note</a> . The fields on this structure can be used to eliminate data that the client doesn't need, which will reduce the time and bandwidth to receive and process the reply.

## Returns

The list of notes that match the criteria.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "offset" - not between 0 and EDAM_USER_NOTES_MAX</li> <li>BAD_DATA_FORMAT "maxNotes" - not between 0 and EDAM_USER_NOTES_MAX</li> <li>BAD_DATA_FORMAT "NoteFilter.notebookGuid" - if malformed</li> <li>BAD_DATA_FORMAT "NoteFilter.tagGuids" - if any are malformed</li> <li>BAD_DATA_FORMAT "NoteFilter.words" - if search string too long</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Notebook.guid" - not found, by GUID</li> </ul>

**7.46.3.47 findNotesMetadataAsync()**

```
AsyncResult* qevercloud::NoteStore::findNotesMetadataAsync (
    const NoteFilter & filter,
    qint32 offset,
    qint32 maxNotes,
    const NotesMetadataResultSpec & resultSpec,
    QString authenticationToken = QString() )
```

Asynchronous version of [findNotesMetadata](#)

**7.46.3.48 findRelated()**

```
RelatedResult qevercloud::NoteStore::findRelated (
    const RelatedQuery & query,
    const RelatedResultSpec & resultSpec,
    QString authenticationToken = QString() )
```

Identify related entities on the service, such as notes, notebooks, and tags related to notes or content.

**Parameters**

<i>query</i>	The information about which we are finding related entities.
<i>resultSpec</i>	Allows the client to indicate the type and quantity of information to be returned, allowing a saving of time and bandwidth.

**Returns**

The result of the query, with information considered to likely be relevantly related to the information described by the query.

**Exceptions**

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "RelatedQuery.plainText" - If you provided a a zero-length plain text value.</li> <li>BAD_DATA_FORMAT "RelatedQuery.noteGuid" - If you provided an invalid <a href="#">Note</a> GUID, that is, one that does not match the constraints defined by EDAM_GUID_LEN_MIN, EDAM_GUID_LEN_MAX, EDAM_GUID_REGEX.</li> <li>BAD_DATA_FORMAT "NoteFilter.notebookGuid" - if malformed</li> <li>BAD_DATA_FORMAT "NoteFilter.tagGuids" - if any are malformed</li> <li>BAD_DATA_FORMAT "NoteFilter.words" - if search string too long</li> <li>PERMISSION_DENIED "Note" - If the caller does not have access to the note identified by <a href="#">RelatedQuery.noteGuid</a>.</li> <li>DATA_REQUIRED "RelatedResultSpec" - If you did not not set any values in the result spec.</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"RelatedQuery.noteGuid" - the note with that GUID is not found, if that field has been set in the query.</li> </ul>

**7.46.3.49 findRelatedAsync()**

```
AsyncResult* qevercloud::NoteStore::findRelatedAsync (
    const RelatedQuery & query,
    const RelatedResultSpec & resultSpec,
    QString authenticationToken = QString() )
```

Asynchronous version of [findRelated](#)

**7.46.3.50 getDefaultNotebook()**

```
Notebook qevercloud::NoteStore::getDefaultNotebook (
    QString authenticationToken = QString() )
```

Returns the notebook that should be used to store new notes in the user's account when no other notebooks are specified.

**7.46.3.51 getDefaultNotebookAsync()**

```
AsyncResult* qevercloud::NoteStore::getDefaultNotebookAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [getDefaultNotebook](#)

**7.46.3.52 getFilteredSyncChunk()**

```
SyncChunk qevercloud::NoteStore::getFilteredSyncChunk (
    qint32 afterUSN,
    qint32 maxEntries,
    const SyncChunkFilter & filter,
    QString authenticationToken = QString() )
```

Asks the [NoteStore](#) to provide the state of the account in order of last modification. This request retrieves one block of the server's state so that a client can make several small requests against a large account rather than getting the entire state in one big message. This call gives fine-grained control of the data that will be received by a client by omitting data elements that a client doesn't need. This may reduce network traffic and sync times.

**Parameters**

<i>afterUSN</i>	The client can pass this value to ask only for objects that have been updated after a certain point. This allows the client to receive updates after its last checkpoint rather than doing a full synchronization on every pass. The default value of "0" indicates that the client wants to get objects from the start of the account.
<i>maxEntries</i>	The maximum number of modified objects that should be returned in the result <a href="#">SyncChunk</a> . This can be used to limit the size of each individual message to be friendly for network transfer.
<i>filter</i>	The caller must set some of the flags in this structure to specify which data types should be returned during the synchronization. See the <a href="#">SyncChunkFilter</a> structure for information on each flag.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "afterUSN" - if negative</li> <li>• BAD_DATA_FORMAT "maxEntries" - if less than 1</li> </ul>
-----------------------------------	---

7.46.3.53 `getFilteredSyncChunkAsync()`

```
AsyncResult* qevercloud::NoteStore::getFilteredSyncChunkAsync (
    qint32 afterUSN,
    qint32 maxEntries,
    const SyncChunkFilter & filter,
    QString authenticationToken = QString() )
```

Asynchronous version of [getFilteredSyncChunk](#)

7.46.3.54 `getLinkedNotebookSyncChunk()`

```
SyncChunk qevercloud::NoteStore::getLinkedNotebookSyncChunk (
    const LinkedNotebook & linkedNotebook,
    qint32 afterUSN,
    qint32 maxEntries,
    bool fullSyncOnly,
    QString authenticationToken = QString() )
```

Asks the [NoteStore](#) to provide information about the contents of a linked notebook that has been shared with the caller, or that is public to the world. This will return a result that is similar to `getSyncChunk`, but will only contain entries that are visible to the caller. I.e. only that particular [Notebook](#) will be visible, along with its Notes, and Tags on those Notes.

This function must be called on the shard that owns the referenced notebook. (I.e. the `shardId` in `/shard/shardId/edam/note` must be the same as [LinkedNotebook.shardId](#).)

## Parameters

<i>authenticationToken</i>	This should be an authenticationToken for the guest who has received the invitation to the share. (I.e. this should not be the result of <a href="#">NoteStore.authenticateToSharedNotebook</a> )
<i>linkedNotebook</i>	This structure should contain identifying information and permissions to access the notebook in question. This must contain the valid fields for either a shared notebook (e.g. <code>shareKey</code> ) or a public notebook (e.g. <code>username</code> , <code>uri</code> )
<i>afterUSN</i>	The client can pass this value to ask only for objects that have been updated after a certain point. This allows the client to receive updates after its last checkpoint rather than doing a full synchronization on every pass. The default value of "0" indicates that the client wants to get objects from the start of the account.
<i>maxEntries</i>	The maximum number of modified objects that should be returned in the result <a href="#">SyncChunk</a> . This can be used to limit the size of each individual message to be friendly for network transfer. Applications should not request more than 256 objects at a time, and must handle the case where the service returns less than the requested number of objects in a given request even though more objects are available on the service.
<i>fullSyncOnly</i>	If true, then the client only wants initial data for a full sync. In this case, the service will not return any expunged objects, and will not return any Resources, since these are also provided in their corresponding Notes.



## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "afterUSN" - if negative</li> <li>• BAD_DATA_FORMAT "maxEntries" - if less than 1</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "LinkedNotebook" - if the provided information doesn't match any valid notebook</li> <li>• "LinkedNotebook.uri" - if the provided public URI doesn't match any valid notebook</li> <li>• "SharedNotebook.id" - if the provided information indicates a shared notebook that no longer exists</li> </ul>

## 7.46.3.55 getLinkedNotebookSyncChunkAsync()

```
AsyncResult* qevercloud::NoteStore::getLinkedNotebookSyncChunkAsync (
    const LinkedNotebook & linkedNotebook,
    qint32 afterUSN,
    qint32 maxEntries,
    bool fullSyncOnly,
    QString authenticationToken = QString() )
```

Asynchronous version of [getLinkedNotebookSyncChunk](#)

## 7.46.3.56 getLinkedNotebookSyncState()

```
SyncState qevercloud::NoteStore::getLinkedNotebookSyncState (
    const LinkedNotebook & linkedNotebook,
    QString authenticationToken = QString() )
```

Asks the [NoteStore](#) to provide information about the status of a linked notebook that has been shared with the caller, or that is public to the world. This will return a result that is similar to [getSyncState](#), but may omit [SyncState.uploaded](#) if the caller doesn't have permission to write to the linked notebook.

This function must be called on the shard that owns the referenced notebook. (I.e. the shardId in /shard/shardId/edam/note must be the same as [LinkedNotebook.shardId](#).)

## Parameters

<i>authenticationToken</i>	This should be an authenticationToken for the guest who has received the invitation to the share. (I.e. this should not be the result of <a href="#">NoteStore.authenticateToSharedNotebook</a> )
<i>linkedNotebook</i>	This structure should contain identifying information and permissions to access the notebook in question.

### 7.46.3.57 `getLinkedNotebookSyncStateAsync()`

```
AsyncResult* qevercloud::NoteStore::getLinkedNotebookSyncStateAsync (
    const LinkedNotebook & linkedNotebook,
    QString authenticationToken = QString() )
```

Asynchronous version of [getLinkedNotebookSyncState](#)

### 7.46.3.58 `getNote()`

```
Note qevercloud::NoteStore::getNote (
    Guid guid,
    bool withContent,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    QString authenticationToken = QString() )
```

Returns the current state of the note in the service with the provided GUID. The ENML contents of the note will only be provided if the 'withContent' parameter is true. The service will include the meta-data for each resource in the note, but the binary contents of the resources and their recognition data will be omitted. If the [Note](#) is found in a public notebook, the authenticationToken will be ignored (so it could be an empty string). The applicationData fields are returned as keysOnly.

#### Parameters

<i>guid</i>	The GUID of the note to be retrieved.
<i>withContent</i>	If true, the note will include the ENML contents of its 'content' field.
<i>withResourcesData</i>	If true, any <a href="#">Resource</a> elements in this <a href="#">Note</a> will include the binary contents of their 'data' field's body.
<i>withResourcesRecognition</i>	If true, any <a href="#">Resource</a> elements will include the binary contents of the 'recognition' field's body if recognition data is present.
<i>withResourcesAlternateData</i>	If true, any <a href="#">Resource</a> elements in this <a href="#">Note</a> will include the binary contents of their 'alternateData' fields' body, if an alternate form is present.

#### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Note.guid" - not found, by GUID</li> </ul>

### 7.46.3.59 `getNoteApplicationData()`

```
LazyMap qevercloud::NoteStore::getNoteApplicationData (
    Guid guid,
    QString authenticationToken = QString() )
```

Get all of the application data for the note identified by GUID, with values returned within the [LazyMap](#) fullMap field. If there are no applicationData entries, then a [LazyMap](#) with an empty fullMap will be returned. If your application only needs to fetch its own applicationData entry, use [getNoteApplicationDataEntry](#) instead.

#### 7.46.3.60 getNoteApplicationDataAsync()

```
AsyncResult* qevercloud::NoteStore::getNoteApplicationDataAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getNoteApplicationData](#)

#### 7.46.3.61 getNoteApplicationDataEntry()

```
QString qevercloud::NoteStore::getNoteApplicationDataEntry (
    Guid guid,
    QString key,
    QString authenticationToken = QString() )
```

Get the value of a single entry in the applicationData map for the note identified by GUID.

##### Exceptions

<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"><li>• "Note.guid" - note not found, by GUID</li><li>• "NoteAttributes.applicationData.key" - note not found, by key</li></ul>
---------------------------------------	---

#### 7.46.3.62 getNoteApplicationDataEntryAsync()

```
AsyncResult* qevercloud::NoteStore::getNoteApplicationDataEntryAsync (
    Guid guid,
    QString key,
    QString authenticationToken = QString() )
```

Asynchronous version of [getNoteApplicationDataEntry](#)

#### 7.46.3.63 getNoteAsync()

```
AsyncResult* qevercloud::NoteStore::getNoteAsync (
    Guid guid,
    bool withContent,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    QString authenticationToken = QString() )
```

Asynchronous version of [getNote](#)

**7.46.3.64 getNotebook()**

```

Notebook qevercloud::NoteStore::getNotebook (
    Guid guid,
    QString authenticationToken = QString() )

```

Returns the current state of the notebook with the provided GUID. The notebook may be active or deleted (but not expunged).

**Parameters**

<i>guid</i>	The GUID of the notebook to be retrieved.
-------------	---

**Exceptions**

<i>EDAMUserException</i>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "Notebook.guid" - if the parameter is missing</li> <li>PERMISSION_DENIED "Notebook" - private notebook, user doesn't own</li> </ul>
<i>EDAMNotFoundException</i>	<ul style="list-style-type: none"> <li>"Notebook.guid" - tag not found, by GUID</li> </ul>

**7.46.3.65 getNotebookAsync()**

```

AsyncResult* qevercloud::NoteStore::getNotebookAsync (
    Guid guid,
    QString authenticationToken = QString() )

```

Asynchronous version of [getNotebook](#)

**7.46.3.66 getNoteContent()**

```

QString qevercloud::NoteStore::getNoteContent (
    Guid guid,
    QString authenticationToken = QString() )

```

Returns XHTML contents of the note with the provided GUID. If the [Note](#) is found in a public notebook, the authenticationToken will be ignored (so it could be an empty string).

**Parameters**

<i>guid</i>	The GUID of the note to be retrieved.
-------------	---------------------------------------

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

7.46.3.67 `getNoteContentAsync()`

```
AsyncResult* qevercloud::NoteStore::getNoteContentAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getNoteContent](#)

7.46.3.68 `getNoteSearchText()`

```
QString qevercloud::NoteStore::getNoteSearchText (
    Guid guid,
    bool noteOnly,
    bool tokenizeForIndexing,
    QString authenticationToken = QString() )
```

Returns a block of the extracted plain text contents of the note with the provided GUID. This text can be indexed for search purposes by a light client that doesn't have capabilities to extract all of the searchable text content from the note and its resources.

If the [Note](#) is found in a public notebook, the authenticationToken will be ignored (so it could be an empty string).

## Parameters

<i>guid</i>	The GUID of the note to be retrieved.
<i>noteOnly</i>	If true, this will only return the text extracted from the ENML contents of the note itself. If false, this will also include the extracted text from any text-bearing resources (PDF, recognized images)
<i>tokenizeForIndexing</i>	If true, this will break the text into cleanly separated and sanitized tokens. If false, this will return the more raw text extraction, with its original punctuation, capitalization, spacing, etc.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
--	--

## Exceptions

<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>
--	--

**7.46.3.69 getNoteSearchTextAsync()**

```
AsyncResult* qevercloud::NoteStore::getNoteSearchTextAsync (
    Guid guid,
    bool noteOnly,
    bool tokenizeForIndexing,
    QString authenticationToken = QString() )
```

Asynchronous version of [getNoteSearchText](#)

**7.46.3.70 getNoteTagNames()**

```
QStringList qevercloud::NoteStore::getNoteTagNames (
    Guid guid,
    QString authenticationToken = QString() )
```

Returns a list of the names of the tags for the note with the provided guid. This can be used with authentication to get the tags for a user's own note, or can be used without valid authentication to retrieve the names of the tags for a note in a public notebook.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

**7.46.3.71 getNoteTagNamesAsync()**

```
AsyncResult* qevercloud::NoteStore::getNoteTagNamesAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getNoteTagNames](#)

### 7.46.3.72 getNoteVersion()

```

Note qevercloud::NoteStore::getNoteVersion (
    Guid noteGuid,
    qint32 updateSequenceNum,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    QString authenticationToken = QString() )

```

This can be used to retrieve a previous version of a [Note](#) after it has been updated within the service. The caller must identify the note (via its guid) and the version (via the updateSequenceNumber of that version). to find a listing of the stored version USNs for a note, call `listNoteVersions`. This call is only available for notes in Premium accounts. (I.e. access to past versions of Notes is a Premium-only feature.)

#### Parameters

<i>noteGuid</i>	The GUID of the note to be retrieved.
<i>updateSequenceNum</i>	The USN of the version of the note that is being retrieved
<i>withResourcesData</i>	If true, any <a href="#">Resource</a> elements in this <a href="#">Note</a> will include the binary contents of their 'data' field's body.
<i>withResourcesRecognition</i>	If true, any <a href="#">Resource</a> elements will include the binary contents of the 'recognition' field's body if recognition data is present.
<i>withResourcesAlternateData</i>	If true, any <a href="#">Resource</a> elements in this <a href="#">Note</a> will include the binary contents of their 'alternateData' fields' body, if an alternate form is present.

#### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>PERMISSION_DENIED "Note" - private note, user doesn't own</li> <li>PERMISSION_DENIED "updateSequenceNum" - The account isn't permitted to access previous versions of notes. (i.e. this is a Free account.)</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Note.guid" - not found, by GUID</li> <li>"Note.updateSequenceNumber" - the <a href="#">Note</a> doesn't have a version with the corresponding USN.</li> </ul>

### 7.46.3.73 getNoteVersionAsync()

```

AsyncResult* qevercloud::NoteStore::getNoteVersionAsync (
    Guid noteGuid,
    qint32 updateSequenceNum,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    QString authenticationToken = QString() )

```

Asynchronous version of [getNoteVersion](#)

### 7.46.3.74 getPublicNotebook()

```

Notebook qevercloud::NoteStore::getPublicNotebook (
    UserID userId,
    QString publicUri )

```

Looks for a user account with the provided `userId` on this [NoteStore](#) shard and determines whether that account contains a public notebook with the given URI. If the account is not found, or no public notebook exists with this URI, this will throw an [EDAMNotFoundException](#), otherwise this will return the information for that [Notebook](#).

If a notebook is visible on the web with a full URL like <http://www.evernote.com/pub/sethdemo/api> Then 'sethdemo' is the username that can be used to look up the `userId`, and 'api' is the `publicUri`.

#### Parameters

<i>userId</i>	The numeric identifier for the user who owns the public notebook. To find this value based on a username string, you can invoke <a href="#">UserStore.getPublicUserInfo</a>
<i>publicUri</i>	The uri string for the public notebook, from <code>Notebook.publishing.uri</code> .

#### Exceptions

<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Publishing.uri" - not found, by URI</li> </ul>
<a href="#">EDAMSystemException</a>	<ul style="list-style-type: none"> <li>TAKEN_DOWN "PublicNotebook" - The specified public notebook is taken down (for all requesters).</li> <li>TAKEN_DOWN "Country" - The specified public notebook is taken down for the requester because of an IP-based country lookup.</li> </ul>

### 7.46.3.75 getPublicNotebookAsync()

```

AsyncResult* qevercloud::NoteStore::getPublicNotebookAsync (
    UserID userId,
    QString publicUri )

```

Asynchronous version of [getPublicNotebook](#)

### 7.46.3.76 getResource()

```

Resource qevercloud::NoteStore::getResource (
    Guid guid,
    bool withData,
    bool withRecognition,
    bool withAttributes,
    bool withAlternateData,
    QString authenticationToken = QString() )

```

Returns the current state of the resource in the service with the provided GUID. If the [Resource](#) is found in a public notebook, the `authenticationToken` will be ignored (so it could be an empty string). Only the keys for the `applicationData` will be returned.



## Parameters

<i>guid</i>	The GUID of the resource to be retrieved.
<i>withData</i>	If true, the <a href="#">Resource</a> will include the binary contents of the 'data' field's body.
<i>withRecognition</i>	If true, the <a href="#">Resource</a> will include the binary contents of the 'recognition' field's body if recognition data is present.
<i>withAttributes</i>	If true, the <a href="#">Resource</a> will include the attributes
<i>withAlternateData</i>	If true, the <a href="#">Resource</a> will include the binary contents of the 'alternateData' field's body, if an alternate form is present.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Resource.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Resource.guid" - not found, by GUID</li> </ul>

## 7.46.3.77 getResourceAlternateData()

```
QByteArray qevercloud::NoteStore::getResourceAlternateData (
    Guid guid,
    QString authenticationToken = QString() )
```

If the [Resource](#) with the provided GUID has an alternate data representation (indicated via the [Resource.alternateData](#) field), then this request can be used to retrieve the binary contents of that alternate data file. If the caller asks about a resource that has no alternate data form, this will throw [EDAMNotFoundException](#).

## Parameters

<i>guid</i>	The GUID of the resource whose recognition data should be retrieved.
-------------	--

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Resource.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Resource.guid" - not found, by GUID</li> <li>• "Resource.alternateData" - resource has no recognition</li> </ul>

### 7.46.3.78 getResourceAlternateDataAsync()

```
AsyncResult* qevercloud::NoteStore::getResourceAlternateDataAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResourceAlternateData](#)

### 7.46.3.79 getResourceApplicationData()

```
LazyMap qevercloud::NoteStore::getResourceApplicationData (
    Guid guid,
    QString authenticationToken = QString() )
```

Get all of the application data for the [Resource](#) identified by GUID, with values returned within the [LazyMap](#) fullMap field. If there are no applicationData entries, then a [LazyMap](#) with an empty fullMap will be returned. If your application only needs to fetch its own applicationData entry, use [getResourceApplicationDataEntry](#) instead.

### 7.46.3.80 getResourceApplicationDataAsync()

```
AsyncResult* qevercloud::NoteStore::getResourceApplicationDataAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResourceApplicationData](#)

### 7.46.3.81 getResourceApplicationDataEntry()

```
QString qevercloud::NoteStore::getResourceApplicationDataEntry (
    Guid guid,
    QString key,
    QString authenticationToken = QString() )
```

Get the value of a single entry in the applicationData map for the [Resource](#) identified by GUID.

#### Exceptions

<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Resource.guid" - <a href="#">Resource</a> not found, by GUID</li> <li>• "ResourceAttributes.applicationData.key" - <a href="#">Resource</a> not found, by key</li> </ul>
---------------------------------------	--

### 7.46.3.82 getResourceApplicationDataEntryAsync()

```
AsyncResult* qevercloud::NoteStore::getResourceApplicationDataEntryAsync (
    Guid guid,
```

```
QString key,
QString authenticationToken = QString() )
```

Asynchronous version of [getResourceApplicationDataEntry](#)

#### 7.46.3.83 getResourceAsync()

```
AsyncResult* qevercloud::NoteStore::getResourceAsync (
    Guid guid,
    bool withData,
    bool withRecognition,
    bool withAttributes,
    bool withAlternateData,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResource](#)

#### 7.46.3.84 getResourceAttributes()

```
ResourceAttributes qevercloud::NoteStore::getResourceAttributes (
    Guid guid,
    QString authenticationToken = QString() )
```

Returns the set of attributes for the [Resource](#) with the provided GUID. If the [Resource](#) is found in a public notebook, the authenticationToken will be ignored (so it could be an empty string).

##### Parameters

<i>guid</i>	The GUID of the resource whose attributes should be retrieved.
-------------	--

##### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "Resource.guid" - if the parameter is missing</li> <li>PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Resource.guid" - not found, by GUID</li> </ul>

#### 7.46.3.85 getResourceAttributesAsync()

```
AsyncResult* qevercloud::NoteStore::getResourceAttributesAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResourceAttributes](#)

### 7.46.3.86 getResourceByHash()

```
Resource qevercloud::NoteStore::getResourceByHash (
    Guid noteGuid,
    QByteArray contentHash,
    bool withData,
    bool withRecognition,
    bool withAlternateData,
    QString authenticationToken = QString() )
```

Returns the current state of a resource, referenced by containing note GUID and resource content hash.

#### Parameters

<i>noteGuid</i>	The GUID of the note that holds the resource to be retrieved.
<i>contentHash</i>	The MD5 checksum of the resource within that note. <a href="#">Note</a> that this is the binary checksum, for example from Resource.data.bodyHash, and not the hex-encoded checksum that is used within an en-media tag in a note body.
<i>withData</i>	If true, the <a href="#">Resource</a> will include the binary contents of the 'data' field's body.
<i>withRecognition</i>	If true, the <a href="#">Resource</a> will include the binary contents of the 'recognition' field's body.
<i>withAlternateData</i>	If true, the <a href="#">Resource</a> will include the binary contents of the 'alternateData' field's body, if an alternate form is present.

#### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "Note.guid" - noteGuid param missing</li> <li>• DATA_REQUIRED "Note.contentHash" - contentHash param missing</li> <li>• PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Note" - not found, by guid</li> <li>• "Resource" - not found, by hash</li> </ul>

### 7.46.3.87 getResourceByHashAsync()

```
AsyncResult* qevercloud::NoteStore::getResourceByHashAsync (
    Guid noteGuid,
    QByteArray contentHash,
    bool withData,
    bool withRecognition,
    bool withAlternateData,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResourceByHash](#)

**7.46.3.88 getResourceData()**

```
QByteArray qevercloud::NoteStore::getResourceData (
    Guid guid,
    QString authenticationToken = QString() )
```

Returns binary data of the resource with the provided GUID. For example, if this were an image resource, this would contain the raw bits of the image. If the [Resource](#) is found in a public notebook, the authenticationToken will be ignored (so it could be an empty string).

**Parameters**

<i>guid</i>	The GUID of the resource to be retrieved.
-------------	---

**Exceptions**

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Resource.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Resource.guid" - not found, by GUID</li> </ul>

**7.46.3.89 getResourceDataAsync()**

```
AsyncResult* qevercloud::NoteStore::getResourceDataAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResourceData](#)

**7.46.3.90 getResourceRecognition()**

```
QByteArray qevercloud::NoteStore::getResourceRecognition (
    Guid guid,
    QString authenticationToken = QString() )
```

Returns the binary contents of the recognition index for the resource with the provided GUID. If the caller asks about a resource that has no recognition data, this will throw [EDAMNotFoundException](#). If the [Resource](#) is found in a public notebook, the authenticationToken will be ignored (so it could be an empty string).

**Parameters**

<i>guid</i>	The GUID of the resource whose recognition data should be retrieved.
-------------	--

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Resource.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Resource.guid" - not found, by GUID</li> <li>• "Resource.recognition" - resource has no recognition</li> </ul>

**7.46.3.91 getResourceRecognitionAsync()**

```
AsyncResult* qevercloud::NoteStore::getResourceRecognitionAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResourceRecognition](#)

**7.46.3.92 getResourceSearchText()**

```
QString qevercloud::NoteStore::getResourceSearchText (
    Guid guid,
    QString authenticationToken = QString() )
```

Returns a block of the extracted plain text contents of the resource with the provided GUID. This text can be indexed for search purposes by a light client that doesn't have capability to extract all of the searchable text content from a resource.

If the [Resource](#) is found in a public notebook, the authenticationToken will be ignored (so it could be an empty string).

## Parameters

<i>guid</i>	The GUID of the resource to be retrieved.
-------------	---

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Resource.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Resource.guid" - not found, by GUID</li> </ul>

**7.46.3.93 getResourceSearchTextAsync()**

```
AsyncResult* qevercloud::NoteStore::getResourceSearchTextAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getResourceSearchText](#)

**7.46.3.94 getSearch()**

```
SavedSearch qevercloud::NoteStore::getSearch (
    Guid guid,
    QString authenticationToken = QString() )
```

Returns the current state of the search with the provided GUID.

**Parameters**

<i>guid</i>	The GUID of the search to be retrieved.
-------------	---

**Exceptions**

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "SavedSearch.guid" - if the parameter is missing</li> <li>PERMISSION_DENIED "SavedSearch" - private <a href="#">Tag</a>, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"SavedSearch.guid" - not found, by GUID</li> </ul>

**7.46.3.95 getSearchAsync()**

```
AsyncResult* qevercloud::NoteStore::getSearchAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getSearch](#)

**7.46.3.96 getSharedNotebookByAuth()**

```
SharedNotebook qevercloud::NoteStore::getSharedNotebookByAuth (
    QString authenticationToken = QString() )
```

This function is used to retrieve extended information about a shared notebook by a guest who has already authenticated to access that notebook. This requires an 'authenticationToken' parameter which should be the result of a call to `authenticateToSharedNotebook(...)`. I.e. this is the token that gives access to the particular shared notebook in someone else's account – it's not the authenticationToken for the owner of the notebook itself.

## Parameters

<i>authenticationToken</i>	Should be the authentication token retrieved from the reply of <a href="#">authenticateToSharedNotebook()</a> , proving access to a particular shared notebook.
----------------------------	---

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "authenticationToken" - authentication token doesn't correspond to a valid shared notebook</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "SharedNotebook.id" - the shared notebook no longer exists</li> </ul>

**7.46.3.97 getSharedNotebookByAuthAsync()**

```
AsyncResult* qevercloud::NoteStore::getSharedNotebookByAuthAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [getSharedNotebookByAuth](#)

**7.46.3.98 getSyncChunk()**

```
SyncChunk qevercloud::NoteStore::getSyncChunk (
    quint32 afterUSN,
    quint32 maxEntries,
    bool fullSyncOnly,
    QString authenticationToken = QString() )
```

DEPRECATED - use [getFilteredSyncChunk](#).

**7.46.3.99 getSyncChunkAsync()**

```
AsyncResult* qevercloud::NoteStore::getSyncChunkAsync (
    quint32 afterUSN,
    quint32 maxEntries,
    bool fullSyncOnly,
    QString authenticationToken = QString() )
```

Asynchronous version of [getSyncChunk](#)

**7.46.3.100 getSyncState()**

```
SyncState qevercloud::NoteStore::getSyncState (
    QString authenticationToken = QString() )
```

Asks the [NoteStore](#) to provide information about the status of the user account corresponding to the provided authentication token.



### 7.46.3.101 getSyncStateAsync()

```
AsyncResult* qevercloud::NoteStore::getSyncStateAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [getSyncState](#)

### 7.46.3.102 getSyncStateWithMetrics()

```
SyncState qevercloud::NoteStore::getSyncStateWithMetrics (
    const ClientUsageMetrics & clientMetrics,
    QString authenticationToken = QString() )
```

Asks the [NoteStore](#) to provide information about the status of the user account corresponding to the provided authentication token. This version of 'getSyncState' allows the client to upload coarse- grained usage metrics to the service.

#### Parameters

<i>clientMetrics</i>	see the documentation of the <a href="#">ClientUsageMetrics</a> structure for an explanation of the fields that clients can pass to the service.
----------------------	--

### 7.46.3.103 getSyncStateWithMetricsAsync()

```
AsyncResult* qevercloud::NoteStore::getSyncStateWithMetricsAsync (
    const ClientUsageMetrics & clientMetrics,
    QString authenticationToken = QString() )
```

Asynchronous version of [getSyncStateWithMetrics](#)

### 7.46.3.104 getTag()

```
Tag qevercloud::NoteStore::getTag (
    Guid guid,
    QString authenticationToken = QString() )
```

Returns the current state of the [Tag](#) with the provided GUID.

#### Parameters

<i>guid</i>	The GUID of the tag to be retrieved.
-------------	--------------------------------------

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Tag.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Tag" - private <a href="#">Tag</a>, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Tag.guid" - tag not found, by GUID</li> </ul>

**7.46.3.105 getTagAsync()**

```
AsyncResult* qevercloud::NoteStore::getTagAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [getTag](#)

**7.46.3.106 listLinkedNotebooks()**

```
QList< LinkedNotebook > qevercloud::NoteStore::listLinkedNotebooks (
    QString authenticationToken = QString() )
```

Returns a list of linked notebooks

**7.46.3.107 listLinkedNotebooksAsync()**

```
AsyncResult* qevercloud::NoteStore::listLinkedNotebooksAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [listLinkedNotebooks](#)

**7.46.3.108 listNotebooks()**

```
QList< Notebook > qevercloud::NoteStore::listNotebooks (
    QString authenticationToken = QString() )
```

Returns a list of all of the notebooks in the account.

**7.46.3.109 listNotebooksAsync()**

```
AsyncResult* qevercloud::NoteStore::listNotebooksAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [listNotebooks](#)

### 7.46.3.110 listNoteVersions()

```
QList< NoteVersionId > qevercloud::NoteStore::listNoteVersions (
    Guid noteGuid,
    QString authenticationToken = QString() )
```

Returns a list of the prior versions of a particular note that are saved within the service. These prior versions are stored to provide a recovery from unintentional removal of content from a note. The identifiers that are returned by this call can be used with `getNoteVersion` to retrieve the previous note. The identifiers will be listed from the most recent versions to the oldest.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

**7.46.3.111 listNoteVersionsAsync()**

```
AsyncResult* qevercloud::NoteStore::listNoteVersionsAsync (
    Guid noteGuid,
    QString authenticationToken = QString() )
```

Asynchronous version of [listNoteVersions](#)

**7.46.3.112 listSearches()**

```
QList< SavedSearch > qevercloud::NoteStore::listSearches (
    QString authenticationToken = QString() )
```

Returns a list of the searches in the account. Evernote does not support the undeletion of searches, so this will only include active searches.

**7.46.3.113 listSearchesAsync()**

```
AsyncResult* qevercloud::NoteStore::listSearchesAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [listSearches](#)

**7.46.3.114 listSharedNotebooks()**

```
QList< SharedNotebook > qevercloud::NoteStore::listSharedNotebooks (
    QString authenticationToken = QString() )
```

Lists the collection of shared notebooks for all notebooks in the users account.

**Returns**

The list of all SharedNotebooks for the user

### 7.46.3.115 listSharedNotebooksAsync()

```
AsyncResult* qevercloud::NoteStore::listSharedNotebooksAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [listSharedNotebooks](#)

### 7.46.3.116 listTags()

```
QList< Tag > qevercloud::NoteStore::listTags (
    QString authenticationToken = QString() )
```

Returns a list of the tags in the account. Evernote does not support the undeletion of tags, so this will only include active tags.

### 7.46.3.117 listTagsAsync()

```
AsyncResult* qevercloud::NoteStore::listTagsAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [listTags](#)

### 7.46.3.118 listTagsByNotebook()

```
QList< Tag > qevercloud::NoteStore::listTagsByNotebook (
    Guid notebookGuid,
    QString authenticationToken = QString() )
```

Returns a list of the tags that are applied to at least one note within the provided notebook. If the notebook is public, the authenticationToken may be ignored.

#### Parameters

<i>notebookGuid</i>	the GUID of the notebook to use to find tags
---------------------	--

#### Exceptions

<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"><li>"Notebook.guid" - notebook not found by GUID</li></ul>
---------------------------------------	--

### 7.46.3.119 listTagsByNotebookAsync()

```
AsyncResult* qevercloud::NoteStore::listTagsByNotebookAsync (
    Guid notebookGuid,
    QString authenticationToken = QString() )
```

Asynchronous version of [listTagsByNotebook](#)

**7.46.3.120 noteStoreUrl()**

```
QString qevercloud::NoteStore::noteStoreUrl ( ) [inline]
```

**7.46.3.121 sendMessageToSharedNotebookMembers()**

```
qint32 qevercloud::NoteStore::sendMessageToSharedNotebookMembers (
    Guid notebookGuid,
    QString messageText,
    QStringList recipients,
    QString authenticationToken = QString() )
```

Send a reminder message to some or all of the email addresses that a notebook has been shared with. The message includes the current link to view the notebook.

**Parameters**

<i>authenticationToken</i>	The auth token of the user with permissions to share the notebook
<i>notebookGuid</i>	The guid of the shared notebook
<i>messageText</i>	<a href="#">User</a> provided text to include in the email
<i>recipients</i>	The email addresses of the recipients. If this list is empty then all of the users that the notebook has been shared with are emailed. If an email address doesn't correspond to share invite members then that address is ignored.

**Returns**

The number of messages sent

**Exceptions**

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>LIMIT_REACHED "(recipients)" - The email can't be sent because this would exceed the user's daily email limit.</li> <li>PERMISSION_DENIED "Notebook.guid" - The user doesn't have permission to send a message for the specified notebook.</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Notebook.guid" - not found, by GUID</li> </ul>

**7.46.3.122 sendMessageToSharedNotebookMembersAsync()**

```
AsyncResult* qevercloud::NoteStore::sendMessageToSharedNotebookMembersAsync (
    Guid notebookGuid,
    QString messageText,
```

```
QStringList recipients,  
QString authenticationToken = QString() )
```

Asynchronous version of [sendMessageToSharedNotebookMembers](#)

#### 7.46.3.123 setAuthenticationToken()

```
void qevercloud::NoteStore::setAuthenticationToken (   
    QString authenticationToken ) [inline]
```

#### 7.46.3.124 setNoteApplicationDataEntry()

```
qint32 qevercloud::NoteStore::setNoteApplicationDataEntry (   
    Guid guid,  
    QString key,  
    QString value,  
    QString authenticationToken = QString() )
```

Update, or create, an entry in the applicationData map for the note identified by guid.

#### 7.46.3.125 setNoteApplicationDataEntryAsync()

```
AsyncResult* qevercloud::NoteStore::setNoteApplicationDataEntryAsync (   
    Guid guid,  
    QString key,  
    QString value,  
    QString authenticationToken = QString() )
```

Asynchronous version of [setNoteApplicationDataEntry](#)

#### 7.46.3.126 setNoteStoreUrl()

```
void qevercloud::NoteStore::setNoteStoreUrl (   
    QString noteStoreUrl ) [inline]
```

#### 7.46.3.127 setResourceApplicationDataEntry()

```
qint32 qevercloud::NoteStore::setResourceApplicationDataEntry (   
    Guid guid,  
    QString key,  
    QString value,  
    QString authenticationToken = QString() )
```

Update, or create, an entry in the applicationData map for the [Resource](#) identified by guid.

**7.46.3.128 setResourceApplicationDataEntryAsync()**

```
AsyncResult* qevercloud::NoteStore::setResourceApplicationDataEntryAsync (
    Guid guid,
    QString key,
    QString value,
    QString authenticationToken = QString() )
```

Asynchronous version of [setResourceApplicationDataEntry](#)

**7.46.3.129 setSharedNotebookRecipientSettings()**

```
qint32 qevercloud::NoteStore::setSharedNotebookRecipientSettings (
    qint64 sharedNotebookId,
    const SharedNotebookRecipientSettings & recipientSettings,
    QString authenticationToken = QString() )
```

Set values for the recipient settings associated with a shared notebook. Having update rights to the shared notebook record itself has no effect on this call; only the recipient of the shared notebook can can the recipient settings.

If you do *not* wish to, or cannot, change one of the reminderNotifyEmail or reminderNotifyInApp fields, you must leave that field unset in recipientSettings. This method will skip that field for updates and leave the existing state as it is.

**Returns**

The update sequence number of the account to which the shared notebook belongs, which is the account from which we are sharing a notebook.

**Exceptions**

<a href="#">EDAMNotFoundException</a>	"sharedNotebookId" - Thrown if the service does not have a shared notebook record for the sharedNotebookId on the given shard. If you receive this exception, it is probable that the shared notebook record has been revoked or expired, or that you accessed the wrong shard.
<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "authenticationToken" - If you do not have permission to set the recipient settings for the shared notebook. Only the recipient has permission to do this.</li> <li>• DATA_CONFLICT "recipientSettings.reminderNotifyEmail" - Setting whether or not you want to receive reminder e-mail notifications is possible on a business notebook in the business to which the user belongs. All others can safely unset the reminderNotifyEmail field from the recipientSettings parameter.</li> </ul>

**7.46.3.130 setSharedNotebookRecipientSettingsAsync()**

```
AsyncResult* qevercloud::NoteStore::setSharedNotebookRecipientSettingsAsync (
    qint64 sharedNotebookId,
```



```
const SharedNotebookRecipientSettings & recipientSettings,
QString authenticationToken = QString() )
```

Asynchronous version of [setSharedNotebookRecipientSettings](#)

#### 7.46.3.131 shareNote()

```
QString qevercloud::NoteStore::shareNote (
    Guid guid,
    QString authenticationToken = QString() )
```

If this note is not already shared (via its own direct URL), then this will start sharing that note. This will return the secret "Note Key" for this note that can currently be used in conjunction with the [Note](#)'s GUID to gain direct read-only access to the [Note](#). If the note is already shared, then this won't make any changes to the note, and the existing "Note Key" will be returned. The only way to change the [Note](#) Key for an existing note is to [stopSharingNote](#) first, and then call this function.

##### Parameters

<i>guid</i>	The GUID of the note to be shared.
-------------	------------------------------------

##### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Note.guid" - not found, by GUID</li> </ul>

#### 7.46.3.132 shareNoteAsync()

```
AsyncResult* qevercloud::NoteStore::shareNoteAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [shareNote](#)

#### 7.46.3.133 stopSharingNote()

```
void qevercloud::NoteStore::stopSharingNote (
    Guid guid,
    QString authenticationToken = QString() )
```

If this note is not already shared then this will stop sharing that note and invalidate its "Note Key", so any existing URLs to access that [Note](#) will stop working. If the [Note](#) is not shared, then this function will do nothing.

## Parameters

<i>guid</i>	The GUID of the note to be un-shared.
-------------	---------------------------------------

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Note.guid" - if the parameter is missing</li> <li>• PERMISSION_DENIED "Note" - private note, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

**7.46.3.134 stopSharingNoteAsync()**

```
AsyncResult* qevercloud::NoteStore::stopSharingNoteAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [stopSharingNote](#)

**7.46.3.135 unsetNoteApplicationDataEntry()**

```
qint32 qevercloud::NoteStore::unsetNoteApplicationDataEntry (
    Guid guid,
    QString key,
    QString authenticationToken = QString() )
```

Remove an entry identified by 'key' from the applicationData map for the note identified by 'guid'. Silently ignores an unset of a non-existing key.

**7.46.3.136 unsetNoteApplicationDataEntryAsync()**

```
AsyncResult* qevercloud::NoteStore::unsetNoteApplicationDataEntryAsync (
    Guid guid,
    QString key,
    QString authenticationToken = QString() )
```

Asynchronous version of [unsetNoteApplicationDataEntry](#)

**7.46.3.137 unsetResourceApplicationDataEntry()**

```
qint32 qevercloud::NoteStore::unsetResourceApplicationDataEntry (
    Guid guid,
    QString key,
    QString authenticationToken = QString() )
```

Remove an entry identified by 'key' from the applicationData map for the [Resource](#) identified by 'guid'.

**7.46.3.138 unsetResourceApplicationDataEntryAsync()**

```
AsyncResult* qevercloud::NoteStore::unsetResourceApplicationDataEntryAsync (
    Guid guid,
    QString key,
    QString authenticationToken = QString() )
```

Asynchronous version of [unsetResourceApplicationDataEntry](#)

**7.46.3.139 untagAll()**

```
void qevercloud::NoteStore::untagAll (
    Guid guid,
    QString authenticationToken = QString() )
```

Removes the provided tag from every note that is currently tagged with this tag. If this operation is successful, the tag will still be in the account, but it will not be tagged on any notes.

This function is not intended for use by full synchronizing clients, since it does not provide enough result information to the client to reconcile the local state without performing a follow-up sync from the service. This is intended for "thin clients" that need to efficiently support this as a UI operation.

**Parameters**

<i>guid</i>	The GUID of the tag to remove from all notes.
-------------	---

**Exceptions**

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "Tag.guid" - if the guid parameter is missing</li> <li>PERMISSION_DENIED "Tag" - user doesn't own tag</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"Tag.guid" - tag not found, by GUID</li> </ul>

**7.46.3.140 untagAllAsync()**

```
AsyncResult* qevercloud::NoteStore::untagAllAsync (
    Guid guid,
    QString authenticationToken = QString() )
```

Asynchronous version of [untagAll](#)

**7.46.3.141 updateLinkedNotebook()**

```
qint32 qevercloud::NoteStore::updateLinkedNotebook (
    const LinkedNotebook & linkedNotebook,
    QString authenticationToken = QString() )
```

## Parameters

<i>linkedNotebook</i>	Updates the name of a linked notebook.
-----------------------	--

## Returns

The Update Sequence Number for this change within the account.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "LinkedNotebook.name" - invalid length or pattern</li> </ul>
-----------------------------------	---

**7.46.3.142 updateLinkedNotebookAsync()**

```
AsyncResult* qevercloud::NoteStore::updateLinkedNotebookAsync (
    const LinkedNotebook & linkedNotebook,
    QString authenticationToken = QString() )
```

Asynchronous version of [updateLinkedNotebook](#)

**7.46.3.143 updateNote()**

```
Note qevercloud::NoteStore::updateNote (
    const Note & note,
    QString authenticationToken = QString() )
```

Submit a set of changes to a note to the service. The provided data must include the note's guid field for identification. The note's title must also be set.

## Parameters

<i>note</i>	A <a href="#">Note</a> object containing the desired fields to be populated on the service. With the exception of the note's title and guid, fields that are not being changed do not need to be set. If the content is not being modified, note.content should be left unset. If the list of resources is not being modified, note.resources should be left unset.
-------------	---

## Returns

The metadata (no contents) for the [Note](#) on the server after the update

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Note.title" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "Note.content" - invalid length for ENML body</li> <li>• BAD_DATA_FORMAT "NoteAttributes.*" - bad resource string</li> <li>• BAD_DATA_FORMAT "ResourceAttributes.*" - bad resource string</li> <li>• BAD_DATA_FORMAT "Resource.mime" - invalid resource MIME type</li> <li>• DATA_CONFLICT "Note.deleted" - deleted time set on active note</li> <li>• DATA_REQUIRED "Resource.data" - resource data body missing</li> <li>• ENML_VALIDATION "*" - note content doesn't validate against DTD</li> <li>• LIMIT_REACHED "Note.tagGuids" - too many Tags on <a href="#">Note</a></li> <li>• LIMIT_REACHED "Note.resources" - too many resources on <a href="#">Note</a></li> <li>• LIMIT_REACHED "Note.size" - total note size too large</li> <li>• LIMIT_REACHED "Resource.data.size" - resource too large</li> <li>• LIMIT_REACHED "NoteAttribute.*" - attribute string too long</li> <li>• LIMIT_REACHED "ResourceAttribute.*" - attribute string too long</li> <li>• PERMISSION_DENIED "Note" - user doesn't own</li> <li>• PERMISSION_DENIED "Note.notebookGuid" - user doesn't own destination</li> <li>• QUOTA_REACHED "Accounting.uploadLimit" - note exceeds upload quota</li> <li>• BAD_DATA_FORMAT "Tag.name" - <a href="#">Note.tagNames</a> was provided, and one of the specified tags had an invalid length or pattern</li> <li>• LIMIT_REACHED "Tag" - <a href="#">Note.tagNames</a> was provided, and the required new tags would exceed the maximum number per account</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - note not found, by GUID</li> <li>• "Note.notebookGuid" - if notebookGuid provided, but not found</li> </ul>

## 7.46.3.144 updateNoteAsync()

```
AsyncResult* qevercloud::NoteStore::updateNoteAsync (
    const Note & note,
    QString authenticationToken = QString() )
```

Asynchronous version of [updateNote](#)

**7.46.3.145 updateNotebook()**

```
qint32 qevercloud::NoteStore::updateNotebook (
    const Notebook & notebook,
    QString authenticationToken = QString() )
```

Submits notebook changes to the service. The provided data must include the notebook's guid field for identification.

**Parameters**

<i>notebook</i>	The notebook object containing the requested changes.
-----------------	---

**Returns**

The Update Sequence Number for this change within the account.

**Exceptions**

<i>EDAMUserException</i>	<ul style="list-style-type: none"> <li>BAD_DATA_FORMAT "Notebook.name" - invalid length or pattern</li> <li>BAD_DATA_FORMAT "Notebook.stack" - invalid length or pattern</li> <li>BAD_DATA_FORMAT "Publishing.uri" - if publishing set but bad uri</li> <li>BAD_DATA_FORMAT "Publishing.publicDescription" - if too long</li> <li>DATA_CONFLICT "Notebook.name" - name already in use</li> <li>DATA_CONFLICT "Publishing.uri" - if URI already in use</li> <li>DATA_REQUIRED "Publishing.uri" - if publishing set but uri missing</li> </ul>
<i>EDAMNotFoundException</i>	<ul style="list-style-type: none"> <li>"Notebook.guid" - not found, by GUID</li> </ul>

**7.46.3.146 updateNotebookAsync()**

```
AsyncResult* qevercloud::NoteStore::updateNotebookAsync (
    const Notebook & notebook,
    QString authenticationToken = QString() )
```

Asynchronous version of [updateNotebook](#)

**7.46.3.147 updateResource()**

```
qint32 qevercloud::NoteStore::updateResource (
    const Resource & resource,
    QString authenticationToken = QString() )
```

Submit a set of changes to a resource to the service. This can be used to update the meta-data about the resource, but cannot be used to change the binary contents of the resource (including the length and hash). These cannot be changed directly without creating a new resource and removing the old one via `updateNote`.

## Parameters

<i>resource</i>	<p>A <a href="#">Resource</a> object containing the desired fields to be populated on the service. The service will attempt to update the resource with the following fields from the client:</p> <ul style="list-style-type: none"> <li>• guid: must be provided to identify the resource</li> <li>• mime</li> <li>• width</li> <li>• height</li> <li>• duration</li> <li>• attributes: optional. if present, the set of attributes will be replaced.</li> </ul>
-----------------	---

## Returns

The Update Sequence Number of the resource after the changes have been applied.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Resource.guid" - if the parameter is missing</li> <li>• BAD_DATA_FORMAT "Resource.mime" - invalid resource MIME type</li> <li>• BAD_DATA_FORMAT "ResourceAttributes.*" - bad resource string</li> <li>• LIMIT_REACHED "ResourceAttribute.*" - attribute string too long</li> <li>• PERMISSION_DENIED "Resource" - private resource, user doesn't own</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Resource.guid" - not found, by GUID</li> </ul>

## 7.46.3.148 updateResourceAsync()

```
AsyncResult* qevercloud::NoteStore::updateResourceAsync (
    const Resource & resource,
    QString authenticationToken = QString() )
```

Asynchronous version of [updateResource](#)

## 7.46.3.149 updateSearch()

```
qint32 qevercloud::NoteStore::updateSearch (
    const SavedSearch & search,
    QString authenticationToken = QString() )
```

Submits search changes to the service. The provided data must include the search's guid field for identification. The service will apply updates to the following search fields: name, query, and scope.

## Parameters

<i>search</i>	The search object containing the requested changes.
---------------	---

## Returns

The Update Sequence Number for this change within the account.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "SavedSearch.name" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "SavedSearch.query" - invalid length</li> <li>• DATA_CONFLICT "SavedSearch.name" - name already in use</li> <li>• PERMISSION_DENIED "SavedSearch" - user doesn't own tag</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "SavedSearch.guid" - not found, by GUID</li> </ul>

7.46.3.150 `updateSearchAsync()`

```
AsyncResult* qevercloud::NoteStore::updateSearchAsync (
    const SavedSearch & search,
    QString authenticationToken = QString() )
```

Asynchronous version of [updateSearch](#)

7.46.3.151 `updateSharedNotebook()`

```
qint32 qevercloud::NoteStore::updateSharedNotebook (
    const SharedNotebook & sharedNotebook,
    QString authenticationToken = QString() )
```

Update a [SharedNotebook](#) object.

## Parameters

<i>authenticationToken</i>	Must be an authentication token from the owner or a shared notebook authentication token or business authentication token with sufficient permissions to change invitations for a notebook.
<i>sharedNotebook</i>	The <a href="#">SharedNotebook</a> object containing the requested changes. The "id" of the shared notebook must be set to allow the service to identify the <a href="#">SharedNotebook</a> to be updated. In addition, you MUST set the email, permission, and allowPreview fields to the desired values. All other fields will be ignored if set.



## Returns

The Update Serial Number for this change within the account.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• UNSUPPORTED_OPERATION "updateSharedNotebook" - if this service instance does not support shared notebooks.</li> <li>• BAD_DATA_FORMAT "SharedNotebook.email" - if the email was not valid.</li> <li>• DATA_REQUIRED "SharedNotebook.id" - if the id field was not set.</li> <li>• DATA_REQUIRED "SharedNotebook.privilege" - if the privilege field was not set.</li> <li>• DATA_REQUIRED "SharedNotebook.allowPreview" - if the allowPreview field was not set.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• <a href="#">SharedNotebook.id</a> - if no shared notebook with the specified ID was found.</li> </ul>

## 7.46.3.152 updateSharedNotebookAsync()

```
AsyncResult* qevercloud::NoteStore::updateSharedNotebookAsync (
    const SharedNotebook & sharedNotebook,
    QString authenticationToken = QString() )
```

Asynchronous version of [updateSharedNotebook](#)

## 7.46.3.153 updateTag()

```
qint32 qevercloud::NoteStore::updateTag (
    const Tag & tag,
    QString authenticationToken = QString() )
```

Submits tag changes to the service. The provided data must include the tag's guid field for identification. The service will apply updates to the following tag fields: name, parentGuid

## Parameters

<i>tag</i>	The tag object containing the requested changes.
------------	--

## Returns

The Update Sequence Number for this change within the account.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "Tag.name" - invalid length or pattern</li> <li>• BAD_DATA_FORMAT "Tag.parentGuid" - malformed GUID</li> <li>• DATA_CONFLICT "Tag.name" - name already in use</li> <li>• DATA_CONFLICT "Tag.parentGuid" - can't set parent: circular</li> <li>• PERMISSION_DENIED "Tag" - user doesn't own tag</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Tag.guid" - tag not found, by GUID</li> <li>• "Tag.parentGuid" - parent not found, by GUID</li> </ul>

**7.46.3.154 updateTagAsync()**

```
AsyncResult* qevercloud::NoteStore::updateTagAsync (
    const Tag & tag,
    QString authenticationToken = QString() )
```

Asynchronous version of [updateTag](#)

**7.47 qevercloud::NoteVersionId Struct Reference**

```
#include <types.h>
```

**Public Member Functions**

- bool [operator==](#) (const [NoteVersionId](#) &other) const
- bool [operator!=](#) (const [NoteVersionId](#) &other) const

**Public Attributes**

- qint32 [updateSequenceNum](#)
- [Timestamp](#) [updated](#)
- [Timestamp](#) [saved](#)
- QString [title](#)

**7.47.1 Detailed Description**

Identifying information about previous versions of a note that are backed up within Evernote's servers. Used in the return value of the `listNoteVersions` call.

## 7.47.2 Member Function Documentation

### 7.47.2.1 operator!=(())

```
bool qevercloud::NoteVersionId::operator!= (
    const NoteVersionId & other ) const [inline]
```

### 7.47.2.2 operator==(())

```
bool qevercloud::NoteVersionId::operator== (
    const NoteVersionId & other ) const [inline]
```

## 7.47.3 Member Data Documentation

### 7.47.3.1 saved

`Timestamp qevercloud::NoteVersionId::saved`

A timestamp that holds the date and time when this version of the note was backed up by Evernote's servers. This

### 7.47.3.2 title

`QString qevercloud::NoteVersionId::title`

The title of the note when this particular version was saved. (The current title of the note may differ from this value.)

### 7.47.3.3 updated

`Timestamp qevercloud::NoteVersionId::updated`

The 'updated' time that was set on the [Note](#) when it had this version of the content. This is the user-modifiable modification time on the note, so it's not reliable for guaranteeing the order of various versions. (E.g. if someone modifies the note, then changes this time manually into the past and then updates the note again.)

### 7.47.3.4 updateSequenceNum

`qint32 qevercloud::NoteVersionId::updateSequenceNum`

The update sequence number for the [Note](#) when it last had this content. This serves to uniquely identify each version of the note, since USN values are unique within an account for each update.

## 7.48 qevercloud::EvernoteOAuthWebView::OAuthResult Struct Reference

```
#include <oauth.h>
```

### Public Attributes

- QString [noteStoreUrl](#)  
*note store url for the user; no need to question [UserStore::getNoteStoreUrl](#) for it.*
- [Timestamp](#) [expires](#)  
*authenticationToken time of expiration.*
- QString [shardId](#)  
*usually is not used*
- [UserID](#) [userId](#)  
*same as [PublicUserInfo::userId](#)*
- QString [webApiUrlPrefix](#)  
*see [PublicUserInfo::webApiUrlPrefix](#)*
- QString [authenticationToken](#)  
*This is what this all was for!*

### 7.48.1 Detailed Description

Holds data that is returned by Evernote on a succesful authentication

### 7.48.2 Member Data Documentation

#### 7.48.2.1 authenticationToken

```
QString qevercloud::EvernoteOAuthWebView::OAuthResult::authenticationToken
```

This is what this all was for!

#### 7.48.2.2 expires

```
Timestamp qevercloud::EvernoteOAuthWebView::OAuthResult::expires
```

authenticationToken time of expiration.

### 7.48.2.3 noteStoreUrl

```
QString qevercloud::EvernoteOAuthWebView::OAuthResult::noteStoreUrl
```

note store url for the user; no need to question [UserStore::getNoteStoreUrl](#) for it.

### 7.48.2.4 shardId

```
QString qevercloud::EvernoteOAuthWebView::OAuthResult::shardId
```

usually is not used

### 7.48.2.5 userId

```
UserID qevercloud::EvernoteOAuthWebView::OAuthResult::userId
```

same as [PublicUserInfo::userId](#)

### 7.48.2.6 webApiUrlPrefix

```
QString qevercloud::EvernoteOAuthWebView::OAuthResult::webApiUrlPrefix
```

see [PublicUserInfo::webApiUrlPrefix](#)

## 7.49 qevercloud::Optional< T > Class Template Reference

```
#include <Optional.h>
```

## Public Member Functions

- `Optional ()`
- `Optional (const Optional &o)`
- `template<typename X > Optional (const Optional< X > &o)`
- `Optional (const T &value)`
- `template<typename X > Optional (const X &value)`
- `Optional & operator= (const Optional &o)`
- `template<typename X > Optional & operator= (const Optional< X > &o)`
- `Optional & operator= (const T &value)`
- `template<typename X > Optional & operator= (const X &value)`
- `operator const T & () const`
- `operator T& ()`
- `const T & ref () const`
- `T & ref ()`
- `bool isSet () const`  
*Checks if value is set.*
- `void clear ()`
- `Optional & init ()`
- `T * operator-> ()`
- `const T * operator-> () const`
- `T value (T defaultValue=T()) const`
- `bool isEqual (const Optional< T > &other) const`

## Friends

- `template<typename X > class Optional`
- `void swap (Optional &first, Optional &second)`

### 7.49.1 Detailed Description

```
template<typename T>
class qevercloud::Optional< T >
```

Supports optional values.

Most of the fields in the Evernote API structs are optional. But C++ does not support this notion directly.

To implement the concept of optional values conventional Thrift C++ wrapper uses a special field of a struct type where each field is of type bool with the same name as a field in the struct. This bool flag indicated was the field with the same name in the outer struct assigned or not.

While this method have its advantages (obviousness and simplicity) I found it very inconvenient to work with. You have to check by hand that both values (value itself and its `__isset` flag) are in sync. There is no checks whatsoever against an error and such an error is too easy to make.

So for my library I created a special class that supports the optional value notion explicitly. Basically `Optional` class just holds a bool value that tracks the fact that a value was assigned. But this tracking is done automatically and attempts to use unassigned values throw exceptions. In this way errors are much harder to make and it's harder for them to slip through testing unnoticed too.

## 7.49.2 Constructor & Destructor Documentation

### 7.49.2.1 Optional() [1/5]

```
template<typename T >
qevercloud::Optional< T >::Optional ( ) [inline]
```

Default constructor. Default [Optional](#) is not set.

### 7.49.2.2 Optional() [2/5]

```
template<typename T >
qevercloud::Optional< T >::Optional (
    const Optional< T > & o ) [inline]
```

Copy constructor.

### 7.49.2.3 Optional() [3/5]

```
template<typename T >
template<typename X >
qevercloud::Optional< T >::Optional (
    const Optional< X > & o ) [inline]
```

Template copy constructor. Allows to be initialized with [Optional](#) of any compatible type.

### 7.49.2.4 Optional() [4/5]

```
template<typename T >
qevercloud::Optional< T >::Optional (
    const T & value ) [inline]
```

Initialization with a value of the type T. [Note](#): it's implicit.

### 7.49.2.5 Optional() [5/5]

```
template<typename T >
template<typename X >
qevercloud::Optional< T >::Optional (
    const X & value ) [inline]
```

Template initialization with a value of any compatible type.

## 7.49.3 Member Function Documentation

### 7.49.3.1 clear()

```
template<typename T >
void qevercloud::Optional< T >::clear ( ) [inline]
```

Clears an [Optional](#).

```
Optional<int> o(1);
o.clear();
cout << o; // exception
```

### 7.49.3.2 init()

```
template<typename T >
Optional& qevercloud::Optional< T >::init ( ) [inline]
```

Fast way to initialize an [Optional](#) with a default value.

It's very useful for structs.

```
struct S2 {int f;};
struct S {int f1; Optional<S2> f2};
Optional<S> o; // o.isSet() != true
// without init() it's cumbersome to access struct fields
// it's especially true for nested Optionals
o = S(); // now o is set
o->f2 = S2(); // and o.f2 is set
o->f2->f = 1; // so at last it can be used
// with init() it's simpler
o.init()->f2.init()->f = 1;
```

#### Returns

reference to itself

### 7.49.3.3 isEqual()

```
template<typename T >
bool qevercloud::Optional< T >::isEqual (
    const Optional< T > & other ) const [inline]
```

Two optionals are equal if they are both not set or have equal values.

I do not define operator== due to not easily resolvable conflicts with operator T&.

**Note** that optional == other\_optional may throw but optional.isEqual(other\_optional) will not.

### 7.49.3.4 isSet()

```
template<typename T >
bool qevercloud::Optional< T >::isSet ( ) const [inline]
```

Checks if value is set.

#### Returns

true if [Optional](#) have been assigned a value and false otherwise.

Access to an unassigned ("not set") [Optional](#) lead to an exception.



### 7.49.3.5 operator const T &()

```
template<typename T >
qevercloud::Optional< T >::operator const T & ( ) const [inline]
```

Implicit conversion of Optional<T> to T.

const version.

### 7.49.3.6 operator T&()

```
template<typename T >
qevercloud::Optional< T >::operator T& ( ) [inline]
```

Implicit conversion of Optional<T> to T.

**Note:** a reference is returned, not a copy.

### 7.49.3.7 operator->() [1/2]

```
template<typename T >
T* qevercloud::Optional< T >::operator-> ( ) [inline]
```

Two syntatic constructs come to mind to use for implementation of access to a struct's/class's field directly from [Optional](#).

One is the dereference operator. This is what boost::optional uses. While it's conceptually nice I found it to be not a very convenient way to refer to structs, especially nested ones. So I overloaded the operator-> and use smart pointer semantics.

```
struct S1 {int f1;};
struct S2 {Optional<S1> f2;};
Optional<S2> o;
*((*o).f2).f1; // boost way, not implemented
o->f2->f1;      // QEverCloud way
```

I admit, boost::optional is much more elegant overall. It uses pointer semantics quite clearly and in an instantly understandable way. It's universal (\* works for any type and not just structs). There is no need for implicit type conversions and so there is no subtleties because of it. And so on.

But then referring to struct fields is a chore. And this is the most common use case of Optionals in QEverCloud.

So I decided to use non-obvious-on-the-first-sight semantics for my [Optional](#). IMO it's much more convenient when gotten used to.

### 7.49.3.8 operator->() [2/2]

```
template<typename T >
const T* qevercloud::Optional< T >::operator-> ( ) const [inline]
```

const version.

**7.49.3.9 operator=()** [1/4]

```
template<typename T >
Optional& qevercloud::Optional< T >::operator= (
    const Optional< T > & o ) [inline]
```

Assignment.

**7.49.3.10 operator=()** [2/4]

```
template<typename T >
template<typename X >
Optional& qevercloud::Optional< T >::operator= (
    const Optional< X > & o ) [inline]
```

Template assignment with an [Optional](#) of any compatible value.

**7.49.3.11 operator=()** [3/4]

```
template<typename T >
Optional& qevercloud::Optional< T >::operator= (
    const T & value ) [inline]
```

Assignment with a value of the type T.

**7.49.3.12 operator=()** [4/4]

```
template<typename T >
template<typename X >
Optional& qevercloud::Optional< T >::operator= (
    const X & value ) [inline]
```

Template assignment with a value of any compatible type.

**7.49.3.13 ref()** [1/2]

```
template<typename T >
T& qevercloud::Optional< T >::ref ( ) [inline]
```

Returns reference to the holded value.

There are contexts in C++ where impicit type conversions can't help. For example:

```
Optional<QStringList> l;
for(auto s : l); // you will hear from your compiler
```

Explicit type conversion can be used...

```
Optional<QStringList> l;
for(auto s : static_cast<QStringList&>(l)); // ugh...
```

... but this is indeed ugly as hell.

So I implemented [ref\(\)](#) function that returns a reference to the holded value.

```
Optional<QStringList> l;
for(auto s : l.ref()); // not ideal but OK
```

**7.49.3.14 ref()** [2/2]

```
template<typename T >
const T& qevercloud::Optional< T >::ref ( ) const [inline]
```

Returns a reference to the holded value.

const version.

**7.49.3.15 value()**

```
template<typename T >
T qevercloud::Optional< T >::value (
    T defaultValue = T() ) const [inline]
```

The function is sometimes useful to simplify checking for the value being set.

**Parameters**

<i>defaultValue</i>	The value to return if <a href="#">Optional</a> is not set.
---------------------	---

**Returns**

[Optional](#) value if set and defaultValue otherwise.

**7.49.4 Friends And Related Function Documentation****7.49.4.1 Optional**

```
template<typename T >
template<typename X >
friend class Optional [friend]
```

**7.49.4.2 swap**

```
template<typename T >
void swap (
    Optional< T > & first,
    Optional< T > & second ) [friend]
```

**7.50 qevercloud::PremiumInfo Struct Reference**

```
#include <types.h>
```

## Public Member Functions

- bool `operator==` (const [PremiumInfo](#) &other) const
- bool `operator!=` (const [PremiumInfo](#) &other) const

## Public Attributes

- [Timestamp](#) `currentTime`
- bool `premium`
- bool `premiumRecurring`
- [Optional](#)< [Timestamp](#) > `premiumExpirationDate`
- bool `premiumExtendable`
- bool `premiumPending`
- bool `premiumCancellationPending`
- bool `canPurchaseUploadAllowance`
- [Optional](#)< [QString](#) > `sponsoredGroupName`
- [Optional](#)< [SponsoredGroupRole::type](#) > `sponsoredGroupRole`
- [Optional](#)< bool > `premiumUpgradable`

### 7.50.1 Detailed Description

This structure is used to provide information about a user's Premium account.

### 7.50.2 Member Function Documentation

#### 7.50.2.1 `operator"!=()`

```
bool qevercloud::PremiumInfo::operator!= (
    const PremiumInfo & other ) const [inline]
```

#### 7.50.2.2 `operator==(`

```
bool qevercloud::PremiumInfo::operator== (
    const PremiumInfo & other ) const [inline]
```

### 7.50.3 Member Data Documentation

### 7.50.3.1 canPurchaseUploadAllowance

```
bool qevercloud::PremiumInfo::canPurchaseUploadAllowance
```

True if the user is eligible for purchasing additional upload allowance.

### 7.50.3.2 currentTime

```
Timestamp qevercloud::PremiumInfo::currentTime
```

The server-side date and time when this data was generated.

### 7.50.3.3 premium

```
bool qevercloud::PremiumInfo::premium
```

True if the user's account is Premium.

### 7.50.3.4 premiumCancellationPending

```
bool qevercloud::PremiumInfo::premiumCancellationPending
```

True if the user has requested that no further charges to be made; the Premium account will remain active until it expires.

### 7.50.3.5 premiumExpirationDate

```
Optional< Timestamp > qevercloud::PremiumInfo::premiumExpirationDate
```

The date when the user's Premium account expires, or the date when the user's account is due for payment if it has a recurring payment method.

### 7.50.3.6 premiumExtendable

```
bool qevercloud::PremiumInfo::premiumExtendable
```

True if the user is eligible for purchasing Premium account extensions.

### 7.50.3.7 premiumPending

```
bool qevercloud::PremiumInfo::premiumPending
```

True if the user's Premium account is pending payment confirmation

### 7.50.3.8 premiumRecurring

```
bool qevercloud::PremiumInfo::premiumRecurring
```

True if the user's account is Premium and has a recurring payment method.

### 7.50.3.9 premiumUpgradable

```
Optional< bool > qevercloud::PremiumInfo::premiumUpgradable
```

True if the user is eligible for purchasing Premium account upgrade.

### 7.50.3.10 sponsoredGroupName

```
Optional< QString > qevercloud::PremiumInfo::sponsoredGroupName
```

The name of the sponsored group that the user is part of.

### 7.50.3.11 sponsoredGroupRole

```
Optional< SponsoredGroupRole::type > qevercloud::PremiumInfo::sponsoredGroupRole
```

DEPRECATED - will be removed in a future update.

## 7.51 qevercloud::PremiumOrderStatus Struct Reference

```
#include <types.h>
```

### Public Types

- enum `type` {  
`NONE` = 0, `PENDING` = 1, `ACTIVE` = 2, `FAILED` = 3,  
`CANCELLATION_PENDING` = 4, `CANCELED` = 5 }

#### 7.51.1 Detailed Description

This enumeration defines the possible states of a premium account

`NONE`: the user has never attempted to become a premium subscriber

`PENDING`: the user has requested a premium account but their charge has not been confirmed

`ACTIVE`: the user has been charged and their premium account is in good standing

`FAILED`: the system attempted to charge the was denied. Their premium privileges have been revoked. We will periodically attempt to re-validate their order.

`CANCELLATION_PENDING`: the user has requested that no further charges be made but the current account is still active.

`CANCELED`: the premium account was canceled either because of failure to pay or user cancelation. No more attempts will be made to activate the account.

## 7.51.2 Member Enumeration Documentation

### 7.51.2.1 type

enum `qevercloud::PremiumOrderStatus::type`

## Enumerator

NONE	
PENDING	
ACTIVE	
FAILED	
CANCELLATION_PENDING	
CANCELED	

## 7.52 qevercloud::PrivilegeLevel Struct Reference

```
#include <types.h>
```

### Public Types

- enum [type](#) {  
    [NORMAL](#) = 1, [PREMIUM](#) = 3, [VIP](#) = 5, [MANAGER](#) = 7,  
    [SUPPORT](#) = 8, [ADMIN](#) = 9 }

#### 7.52.1 Detailed Description

This enumeration defines the possible permission levels for a user. Free accounts will have a level of NORMAL and paid Premium accounts will have a level of PREMIUM.

#### 7.52.2 Member Enumeration Documentation

##### 7.52.2.1 type

```
enum qevercloud::PrivilegeLevel::type
```

## Enumerator

NORMAL	
PREMIUM	
VIP	
MANAGER	
SUPPORT	
ADMIN	



## 7.53 qevercloud::PublicUserInfo Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [PublicUserInfo](#) &other) const
- bool [operator!=](#) (const [PublicUserInfo](#) &other) const

### Public Attributes

- [UserID](#) [userId](#)
- [QString](#) [shardId](#)
- [Optional](#)< [PrivilegeLevel::type](#) > [privilege](#)
- [Optional](#)< [QString](#) > [username](#)
- [Optional](#)< [QString](#) > [noteStoreUrl](#)
- [Optional](#)< [QString](#) > [webApiUrlPrefix](#)

#### 7.53.1 Detailed Description

This structure is used to provide publicly-available user information about a particular account.

#### 7.53.2 Member Function Documentation

##### 7.53.2.1 [operator"!=\(\)](#)

```
bool qevercloud::PublicUserInfo::operator!= (
    const PublicUserInfo & other ) const    [inline]
```

##### 7.53.2.2 [operator==\(\(\)\)](#)

```
bool qevercloud::PublicUserInfo::operator== (
    const PublicUserInfo & other ) const    [inline]
```

#### 7.53.3 Member Data Documentation

### 7.53.3.1 noteStoreUrl

```
Optional< QString > qevercloud::PublicUserInfo::noteStoreUrl
```

This field will contain the full URL that clients should use to make [NoteStore](#) requests to the server shard that contains that user's data. I.e. this is the URL that should be used to create the Thrift HTTP client transport to send messages to the [NoteStore](#) service for the account.

### 7.53.3.2 privilege

```
Optional< PrivilegeLevel::type > qevercloud::PublicUserInfo::privilege
```

The privilege level of the account, to determine whether this is a Premium or Free account.

### 7.53.3.3 shardId

```
QString qevercloud::PublicUserInfo::shardId
```

DEPRECATED - Client applications should have no need to use this field.

### 7.53.3.4 userId

```
UserID qevercloud::PublicUserInfo::userId
```

The unique numeric user identifier for the user account.

### 7.53.3.5 username

```
Optional< QString > qevercloud::PublicUserInfo::username
```

NOT DOCUMENTED

### 7.53.3.6 webApiUrlPrefix

```
Optional< QString > qevercloud::PublicUserInfo::webApiUrlPrefix
```

This field will contain the initial part of the URLs that should be used to make requests to Evernote's thin client "web API", which provide optimized operations for clients that aren't capable of manipulating the full contents of accounts via the full Thrift data model. Clients should concatenate the relative path for the various servlets onto the end of this string to construct the full URL, as documented on our developer web site.

## 7.54 qevercloud::Publishing Struct Reference

```
#include <types.h>
```

## Public Member Functions

- bool `operator==` (const [Publishing](#) &other) const
- bool `operator!=` (const [Publishing](#) &other) const

## Public Attributes

- [Optional](#)< QString > `uri`
- [Optional](#)< [NoteSortOrder::type](#) > `order`
- [Optional](#)< bool > `ascending`
- [Optional](#)< QString > `publicDescription`

### 7.54.1 Detailed Description

If a [Notebook](#) has been opened to the public, the [Notebook](#) will have a reference to one of these structures, which gives the location and optional description of the externally-visible public [Notebook](#).

### 7.54.2 Member Function Documentation

#### 7.54.2.1 `operator"!=()`

```
bool qevercloud::Publishing::operator!= (
    const Publishing & other ) const    [inline]
```

#### 7.54.2.2 `operator==(`

```
bool qevercloud::Publishing::operator== (
    const Publishing & other ) const    [inline]
```

### 7.54.3 Member Data Documentation

#### 7.54.3.1 `ascending`

```
Optional< bool > qevercloud::Publishing::ascending
```

If this is set to true, then the public notes will be displayed in ascending order (e.g. from oldest to newest). Otherwise, the notes will be displayed in descending order (e.g. newest to oldest).

### 7.54.3.2 order

```
Optional< NoteSortOrder::type > qevercloud::Publishing::order
```

When the notes are publicly displayed, they will be sorted based on the requested criteria.

### 7.54.3.3 publicDescription

```
Optional< QString > qevercloud::Publishing::publicDescription
```

This field may be used to provide a short description of the notebook, which may be displayed when (e.g.) the notebook is shown in a public view. Can't begin or end with a space.

Length: EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MIN - EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MAX

Regex: EDAM\_PUBLISHING\_DESCRIPTION\_REGEX

### 7.54.3.4 uri

```
Optional< QString > qevercloud::Publishing::uri
```

If this field is present, then the notebook is published for mass consumption on the Internet under the provided URI, which is relative to a defined base publishing URI defined by the service. This field can only be modified via the web service GUI ... publishing cannot be modified via an offline client.

Length: EDAM\_PUBLISHING\_URI\_LEN\_MIN - EDAM\_PUBLISHING\_URI\_LEN\_MAX

Regex: EDAM\_PUBLISHING\_URI\_REGEX

## 7.55 qevercloud::QueryFormat Struct Reference

```
#include <types.h>
```

### Public Types

- enum `type` { `USER` = 1, `SEXP` = 2 }

### 7.55.1 Detailed Description

Every search query is specified as a sequence of characters. Currently, only the USER query format is supported.

### 7.55.2 Member Enumeration Documentation

#### 7.55.2.1 type

```
enum qevercloud::QueryFormat::type
```

## Enumerator

USER	
SEXP	

## 7.56 qevercloud::RelatedQuery Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [RelatedQuery](#) &other) const
- bool [operator!=](#) (const [RelatedQuery](#) &other) const

### Public Attributes

- [Optional](#)< [QString](#) > [noteGuid](#)
- [Optional](#)< [QString](#) > [plainText](#)
- [Optional](#)< [NoteFilter](#) > [filter](#)
- [Optional](#)< [QString](#) > [referenceUri](#)

#### 7.56.1 Detailed Description

A description of the thing for which we are searching for related entities.

You must specify either *noteGuid* or *plainText*, but not both. *filter* and *referenceUri* are optional.

#### 7.56.2 Member Function Documentation

##### 7.56.2.1 operator"!=()

```
bool qevercloud::RelatedQuery::operator!= (
    const RelatedQuery & other ) const [inline]
```

##### 7.56.2.2 operator==( )

```
bool qevercloud::RelatedQuery::operator== (
    const RelatedQuery & other ) const [inline]
```

### 7.56.3 Member Data Documentation

#### 7.56.3.1 filter

`Optional< NoteFilter > qevercloud::RelatedQuery::filter`

The list of criteria that will constrain the notes being considered related. Please note that some of the parameters may be ignored, such as *order* and *ascending*.

#### 7.56.3.2 noteGuid

`Optional< QString > qevercloud::RelatedQuery::noteGuid`

The GUID of an existing note in your account for which related entities will be found.

#### 7.56.3.3 plainText

`Optional< QString > qevercloud::RelatedQuery::plainText`

A string of plain text for which to find related entities. You should provide a text block with a number of characters between EDAM\_RELATED\_PLAINTEXT\_LEN\_MIN and EDAM\_RELATED\_PLAINTEXT\_LEN\_MAX.

#### 7.56.3.4 referenceUri

`Optional< QString > qevercloud::RelatedQuery::referenceUri`

A URI string specifying a reference entity, around which "relatedness" should be based. This can be an URL pointing to a web page, for example.

## 7.57 qevercloud::RelatedResult Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const `RelatedResult` &other) const
- bool `operator!=` (const `RelatedResult` &other) const

### Public Attributes

- `Optional< QList< Note > >` `notes`
- `Optional< QList< Notebook > >` `notebooks`
- `Optional< QList< Tag > >` `tags`
- `Optional< QList< NotebookDescriptor > >` `containingNotebooks`

### 7.57.1 Detailed Description

The result of calling `findRelated()`. The contents of the `notes`, `notebooks`, and `tags` fields will be in decreasing order of expected relevance. It is possible that fewer results than requested will be returned even if there are enough distinct entities in the account in cases where the relevance is estimated to be low.

### 7.57.2 Member Function Documentation

#### 7.57.2.1 `operator!=(())`

```
bool qevercloud::RelatedResult::operator!= (
    const RelatedResult & other ) const [inline]
```

#### 7.57.2.2 `operator==(())`

```
bool qevercloud::RelatedResult::operator== (
    const RelatedResult & other ) const [inline]
```

### 7.57.3 Member Data Documentation

#### 7.57.3.1 `containingNotebooks`

```
Optional< QList< NotebookDescriptor > > qevercloud::RelatedResult::containingNotebooks
```

If `includeContainingNotebooks` is set to `true` in the [RelatedResultSpec](#), return the list of notebooks to which the returned related notes belong. The notebooks in this list will occur once per notebook GUID and are represented as [NotebookDescriptor](#) objects.

#### 7.57.3.2 `notebooks`

```
Optional< QList< Notebook > > qevercloud::RelatedResult::notebooks
```

If notebooks have been requested to be included, this will be the list of notebooks.

#### 7.57.3.3 `notes`

```
Optional< QList< Note > > qevercloud::RelatedResult::notes
```

If notes have been requested to be included, this will be the list of notes.

### 7.57.3.4 tags

```
Optional< QList< Tag > > qevercloud::RelatedResult::tags
```

If tags have been requested to be included, this will be the list of tags.

## 7.58 qevercloud::RelatedResultSpec Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [RelatedResultSpec](#) &other) const
- bool `operator!=` (const [RelatedResultSpec](#) &other) const

### Public Attributes

- [Optional](#)< qint32 > `maxNotes`
- [Optional](#)< qint32 > `maxNotebooks`
- [Optional](#)< qint32 > `maxTags`
- [Optional](#)< bool > `writableNotebooksOnly`
- [Optional](#)< bool > `includeContainingNotebooks`

### 7.58.1 Detailed Description

A description of the thing for which the service will find related entities, via `findRelated()`, together with a description of what type of entities and how many you are seeking in the [RelatedResult](#).

### 7.58.2 Member Function Documentation

#### 7.58.2.1 `operator"!=()`

```
bool qevercloud::RelatedResultSpec::operator!= (
    const RelatedResultSpec & other ) const [inline]
```

#### 7.58.2.2 `operator==(`

```
bool qevercloud::RelatedResultSpec::operator== (
    const RelatedResultSpec & other ) const [inline]
```



### 7.58.3 Member Data Documentation

#### 7.58.3.1 includeContainingNotebooks

`Optional< bool > qevercloud::RelatedResultSpec::includeContainingNotebooks`

If set to `true`, return the `containingNotebooks` field in the [RelatedResult](#), which will contain the list of notebooks to which the returned related notes belong.

#### 7.58.3.2 maxNotebooks

`Optional< qint32 > qevercloud::RelatedResultSpec::maxNotebooks`

Return notebooks that are related to the query, but no more than this many. Any value greater than `EDAM_RELATED_MAX_NOTEBOOKS` will be silently capped. If you do not set this field, then no notebooks will be returned.

#### 7.58.3.3 maxNotes

`Optional< qint32 > qevercloud::RelatedResultSpec::maxNotes`

Return notes that are related to the query, but no more than this many. Any value greater than `EDAM_RELATED_MAX_NOTES` will be silently capped. If you do not set this field, then no notes will be returned.

#### 7.58.3.4 maxTags

`Optional< qint32 > qevercloud::RelatedResultSpec::maxTags`

Return tags that are related to the query, but no more than this many. Any value greater than `EDAM_RELATED_MAX_TAGS` will be silently capped. If you do not set this field, then no tags will be returned.

#### 7.58.3.5 writableNotebooksOnly

`Optional< bool > qevercloud::RelatedResultSpec::writableNotebooksOnly`

Require that all returned related notebooks are writable. The user will be able to create notes in all returned notebooks. However, individual notes returned may still belong to notebooks in which the user lacks the ability to create notes.

## 7.59 qevercloud::ReminderEmailConfig Struct Reference

```
#include <types.h>
```

## Public Types

- enum [type](#) { [DO\\_NOT\\_SEND](#) = 1, [SEND\\_DAILY\\_EMAIL](#) = 2 }

### 7.59.1 Detailed Description

An enumeration describing the configuration state related to receiving reminder e-mails from the service. Reminder e-mails summarize notes based on their `Note.attributes.reminderTime` values.

`DO_NOT_SEND`: The user has selected to not receive reminder e-mail.

`SEND_DAILY_EMAIL`: The user has selected to receive reminder e-mail for those days when there is a reminder.

### 7.59.2 Member Enumeration Documentation

#### 7.59.2.1 `type`

```
enum qevercloud::ReminderEmailConfig::type
```

Enumerator

<code>DO_NOT_SEND</code>	
<code>SEND_DAILY_EMAIL</code>	

## 7.60 `qevercloud::Resource` Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [Resource](#) &other) const
- bool [operator!=](#) (const [Resource](#) &other) const

### Public Attributes

- [Optional](#)< [Guid](#) > [guid](#)
- [Optional](#)< [Guid](#) > [noteGuid](#)
- [Optional](#)< [Data](#) > [data](#)
- [Optional](#)< [QString](#) > [mime](#)
- [Optional](#)< [qint16](#) > [width](#)
- [Optional](#)< [qint16](#) > [height](#)
- [Optional](#)< [qint16](#) > [duration](#)

- [Optional](#)< bool > [active](#)
- [Optional](#)< [Data](#) > [recognition](#)
- [Optional](#)< [ResourceAttributes](#) > [attributes](#)
- [Optional](#)< qint32 > [updateSequenceNum](#)
- [Optional](#)< [Data](#) > [alternateData](#)

### 7.60.1 Detailed Description

Every media file that is embedded or attached to a note is represented through a [Resource](#) entry.

### 7.60.2 Member Function Documentation

#### 7.60.2.1 operator"!="()

```
bool qevercloud::Resource::operator!= (
    const Resource & other ) const [inline]
```

#### 7.60.2.2 operator==()

```
bool qevercloud::Resource::operator== (
    const Resource & other ) const [inline]
```

### 7.60.3 Member Data Documentation

#### 7.60.3.1 active

[Optional](#)< bool > qevercloud::Resource::active

DEPRECATED: ignored.

#### 7.60.3.2 alternateData

[Optional](#)< [Data](#) > qevercloud::Resource::alternateData

Some Resources may be assigned an alternate data format by the service which may be more appropriate for indexing or rendering than the original data provided by the user. In these cases, the alternate data form will be available via this [Data](#) element. If a [Resource](#) has no alternate form, this field will be unset.

### 7.60.3.3 attributes

`Optional< ResourceAttributes > qevercloud::Resource::attributes`

A list of the attributes for this resource.

### 7.60.3.4 data

`Optional< Data > qevercloud::Resource::data`

The contents of the resource. Maximum length: The data.body is limited to EDAM\_RESOURCE\_SIZE\_MAX\_FREE for free accounts and EDAM\_RESOURCE\_SIZE\_MAX\_PREMIUM for premium accounts.

### 7.60.3.5 duration

`Optional< qint16 > qevercloud::Resource::duration`

DEPRECATED: ignored.

### 7.60.3.6 guid

`Optional< Guid > qevercloud::Resource::guid`

The unique identifier of this resource. Will be set whenever a resource is retrieved from the service, but may be null when a client is creating a resource.

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX

### 7.60.3.7 height

`Optional< qint16 > qevercloud::Resource::height`

If set, this contains the display height of this resource, in pixels.

### 7.60.3.8 mime

`Optional< QString > qevercloud::Resource::mime`

The MIME type for the embedded resource. E.g. "image/gif"

Length: EDAM\_MIME\_LEN\_MIN - EDAM\_MIME\_LEN\_MAX

Regex: EDAM\_MIME\_REGEX

### 7.60.3.9 noteGuid

`Optional< Guid > qevercloud::Resource::noteGuid`

The unique identifier of the [Note](#) that holds this [Resource](#). Will be set whenever the resource is retrieved from the service, but may be null when a client is creating a resource.

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX

### 7.60.3.10 recognition

`Optional< Data > qevercloud::Resource::recognition`

If set, this will hold the encoded data that provides information on search and recognition within this resource.

### 7.60.3.11 updateSequenceNum

`Optional< qint32 > qevercloud::Resource::updateSequenceNum`

A number identifying the last transaction to modify the state of this object. The USN values are sequential within an account, and can be used to compare the order of modifications within the service.

### 7.60.3.12 width

`Optional< qint16 > qevercloud::Resource::width`

If set, this contains the display width of this resource, in pixels.

## 7.61 qevercloud::ResourceAttributes Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [ResourceAttributes](#) &other) const
- bool `operator!=` (const [ResourceAttributes](#) &other) const

### Public Attributes

- `Optional< QString > sourceURL`
- `Optional< Timestamp > timestamp`
- `Optional< double > latitude`
- `Optional< double > longitude`
- `Optional< double > altitude`
- `Optional< QString > cameraMake`
- `Optional< QString > cameraModel`
- `Optional< bool > clientWillIndex`
- `Optional< QString > recoType`
- `Optional< QString > fileName`
- `Optional< bool > attachment`
- `Optional< LazyMap > applicationData`

### 7.61.1 Detailed Description

Structure holding the optional attributes of a [Resource](#)

## 7.61.2 Member Function Documentation

### 7.61.2.1 operator"!="()

```
bool qevercloud::ResourceAttributes::operator!= (
    const ResourceAttributes & other ) const [inline]
```

### 7.61.2.2 operator==()

```
bool qevercloud::ResourceAttributes::operator== (
    const ResourceAttributes & other ) const [inline]
```

## 7.61.3 Member Data Documentation

### 7.61.3.1 altitude

```
Optional< double > qevercloud::ResourceAttributes::altitude
```

the altitude where the resource was captured

### 7.61.3.2 applicationData

```
Optional< LazyMap > qevercloud::ResourceAttributes::applicationData
```

Provides a location for applications to store a relatively small (4kb) blob of data associated with a [Resource](#) that is not visible to the user and that is opaque to the Evernote service. A single application may use at most one entry in this map, using its API consumer key as the map key. See the documentation for [LazyMap](#) for a description of when the actual map values are returned by the service.

To safely add or modify your application's entry in the map, use [NoteStore.setResourceApplicationDataEntry](#). To safely remove your application's entry from the map, use [NoteStore.unsetResourceApplicationDataEntry](#).

Minimum length of a name (key): EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN  
Sum max size of key and value: EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX  
Syntax regex for name (key): EDAM\_APPLICATIONDATA\_NAME\_REGEX

### 7.61.3.3 attachment

```
Optional< bool > qevercloud::ResourceAttributes::attachment
```

this will be true if the resource should be displayed as an attachment, or false if the resource should be displayed inline (if possible).

#### 7.61.3.4 cameraMake

`Optional< QString > qevercloud::ResourceAttributes::cameraMake`

information about an image's camera, e.g. as embedded in the image's EXIF data  
Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.61.3.5 cameraModel

`Optional< QString > qevercloud::ResourceAttributes::cameraModel`

information about an image's camera, e.g. as embedded in the image's EXIF data  
Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.61.3.6 clientWillIndex

`Optional< bool > qevercloud::ResourceAttributes::clientWillIndex`

if true, then the original client that submitted the resource plans to submit the recognition index for this resource at a later time.

#### 7.61.3.7 fileName

`Optional< QString > qevercloud::ResourceAttributes::fileName`

if the resource came from a source that provided an explicit file name, the original name will be stored here. Many resources come from unnamed sources, so this will not always be set.

#### 7.61.3.8 latitude

`Optional< double > qevercloud::ResourceAttributes::latitude`

the latitude where the resource was captured

#### 7.61.3.9 longitude

`Optional< double > qevercloud::ResourceAttributes::longitude`

the longitude where the resource was captured

#### 7.61.3.10 recoType

`Optional< QString > qevercloud::ResourceAttributes::recoType`

DEPRECATED - this field is no longer set by the service, so should be ignored.

### 7.61.3.11 sourceURL

`Optional< QString > qevercloud::ResourceAttributes::sourceURL`

the original location where the resource was hosted

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

### 7.61.3.12 timestamp

`Optional< Timestamp > qevercloud::ResourceAttributes::timestamp`

the date and time that is associated with this resource (e.g. the time embedded in an image from a digital camera with a clock)

## 7.62 qevercloud::SavedSearch Struct Reference

```
#include <types.h>
```

### Public Member Functions

- `bool operator== (const SavedSearch &other) const`
- `bool operator!= (const SavedSearch &other) const`

### Public Attributes

- `Optional< Guid > guid`
- `Optional< QString > name`
- `Optional< QString > query`
- `Optional< QueryFormat::type > format`
- `Optional< qint32 > updateSequenceNum`
- `Optional< SavedSearchScope > scope`

### 7.62.1 Detailed Description

A named search associated with the account that can be quickly re-used.

### 7.62.2 Member Function Documentation

#### 7.62.2.1 operator"!=()

```
bool qevercloud::SavedSearch::operator!= (
    const SavedSearch & other ) const [inline]
```



### 7.62.2.2 operator==( )

```
bool qevercloud::SavedSearch::operator== (
    const SavedSearch & other ) const [inline]
```

## 7.62.3 Member Data Documentation

### 7.62.3.1 format

```
Optional< QueryFormat::type > qevercloud::SavedSearch::format
```

The format of the query string, to determine how to parse and process it.

### 7.62.3.2 guid

```
Optional< Guid > qevercloud::SavedSearch::guid
```

The unique identifier of this search. Will be set by the service, so may be omitted by the client when creating.

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX

### 7.62.3.3 name

```
Optional< QString > qevercloud::SavedSearch::name
```

The name of the saved search to display in the GUI. The account may only contain one search with a given name (case-insensitive compare). Can't begin or end with a space.

Length: EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MIN - EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MAX

Regex: EDAM\_SAVED\_SEARCH\_NAME\_REGEX

### 7.62.3.4 query

```
Optional< QString > qevercloud::SavedSearch::query
```

A string expressing the search to be performed.

Length: EDAM\_SAVED\_SEARCH\_QUERY\_LEN\_MIN - EDAM\_SAVED\_SEARCH\_QUERY\_LEN\_MAX

### 7.62.3.5 scope

```
Optional< SavedSearchScope > qevercloud::SavedSearch::scope
```

Specifies the set of notes that should be included in the search, if possible.

Clients are expected to search as much of the desired scope as possible, with the understanding that a given client may not be able to cover the full specified scope. For example, when executing a search that includes notes in both the owner's account and business notebooks, a mobile client may choose to only search within the user's account because it is not capable of searching both scopes simultaneously. When a search across multiple scopes is not possible, a client may choose which scope to search based on the current application context. If a client cannot search any of the desired scopes, it should refuse to execute the search.

### 7.62.3.6 updateSequenceNum

```
Optional< qint32 > qevercloud::SavedSearch::updateSequenceNum
```

A number identifying the last transaction to modify the state of this object. The USN values are sequential within an account, and can be used to compare the order of modifications within the service.

## 7.63 qevercloud::SavedSearchScope Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [SavedSearchScope](#) &other) const
- bool `operator!=` (const [SavedSearchScope](#) &other) const

### Public Attributes

- [Optional](#)< bool > `includeAccount`
- [Optional](#)< bool > `includePersonalLinkedNotebooks`
- [Optional](#)< bool > `includeBusinessLinkedNotebooks`

### 7.63.1 Detailed Description

A structure defining the scope of a [SavedSearch](#).

### 7.63.2 Member Function Documentation

#### 7.63.2.1 `operator!=()`

```
bool qevercloud::SavedSearchScope::operator!= (
    const SavedSearchScope & other ) const [inline]
```

#### 7.63.2.2 `operator==()`

```
bool qevercloud::SavedSearchScope::operator== (
    const SavedSearchScope & other ) const [inline]
```

### 7.63.3 Member Data Documentation

#### 7.63.3.1 includeAccount

`Optional< bool > qevercloud::SavedSearchScope::includeAccount`

The search should include notes from the account that contains the [SavedSearch](#).

#### 7.63.3.2 includeBusinessLinkedNotebooks

`Optional< bool > qevercloud::SavedSearchScope::includeBusinessLinkedNotebooks`

The search should include notes within those shared notebooks that the user has joined that are business notebooks in the business that the user is currently a member of.

#### 7.63.3.3 includePersonalLinkedNotebooks

`Optional< bool > qevercloud::SavedSearchScope::includePersonalLinkedNotebooks`

The search should include notes within those shared notebooks that the user has joined that are NOT business notebooks.

## 7.64 qevercloud::SharedNotebook Struct Reference

```
#include <types.h>
```

### Public Member Functions

- `bool operator==` (const [SharedNotebook](#) &other) const
- `bool operator!=` (const [SharedNotebook](#) &other) const

### Public Attributes

- `Optional< qint64 > id`
- `Optional< qint32 > userId`
- `Optional< QString > notebookGuid`
- `Optional< QString > email`
- `Optional< bool > notebookModifiable`
- `Optional< bool > requireLogin`
- `Optional< Timestamp > serviceCreated`
- `Optional< Timestamp > serviceUpdated`
- `Optional< QString > shareKey`
- `Optional< QString > username`
- `Optional< SharedNotebookPrivilegeLevel::type > privilege`
- `Optional< bool > allowPreview`
- `Optional< SharedNotebookRecipientSettings > recipientSettings`

### 7.64.1 Detailed Description

Shared notebooks represent a relationship between a notebook and a single share invitation recipient.

### 7.64.2 Member Function Documentation

#### 7.64.2.1 operator!=(())

```
bool qevercloud::SharedNotebook::operator!=(
    const SharedNotebook & other ) const [inline]
```

#### 7.64.2.2 operator==(())

```
bool qevercloud::SharedNotebook::operator==(
    const SharedNotebook & other ) const [inline]
```

### 7.64.3 Member Data Documentation

#### 7.64.3.1 allowPreview

```
Optional< bool > qevercloud::SharedNotebook::allowPreview
```

Whether or not to grant "READ\_NOTEBOOK" privilege without an authentication token, for authenticateToSharedNotebook(...). With the change to "requireLogin" always being true for new shared notebooks, this is the only way to access a shared notebook without an authorization token. This setting expires after the first use of authenticateToSharedNotebook(...) with a valid authentication token.

#### 7.64.3.2 email

```
Optional< QString > qevercloud::SharedNotebook::email
```

the email address of the recipient - used by the notebook owner to identify who they shared with.

#### 7.64.3.3 id

```
Optional< qint64 > qevercloud::SharedNotebook::id
```

the primary identifier of the share

#### 7.64.3.4 notebookGuid

Optional< QString > qevercloud::SharedNotebook::notebookGuid

the GUID of the associated notebook shared.

#### 7.64.3.5 notebookModifiable

Optional< bool > qevercloud::SharedNotebook::notebookModifiable

(DEPRECATED) a flag indicating the share is read/write -otherwise it's read only. This field is deprecated in favor of the new "privilege" field.

#### 7.64.3.6 privilege

Optional< SharedNotebookPrivilegeLevel::type > qevercloud::SharedNotebook::privilege

The privilege level granted to the notebook, activity stream, and invitations. See the corresponding enumeration for details.

#### 7.64.3.7 recipientSettings

Optional< SharedNotebookRecipientSettings > qevercloud::SharedNotebook::recipientSettings

Settings intended for use only by the recipient of this shared notebook. You should skip setting this value unless you want to change the value contained inside the structure, and only if you are the recipient.

#### 7.64.3.8 requireLogin

Optional< bool > qevercloud::SharedNotebook::requireLogin

(DEPRECATED) indicates that a user must login to access the share. This field is deprecated and will be "true" for all new shared notebooks. It is read-only and ignored when creating or modifying a shared notebook, except that a shared notebook can be modified to require login. See "allowPreview" for information on privileges and shared notebooks.

#### 7.64.3.9 serviceCreated

Optional< Timestamp > qevercloud::SharedNotebook::serviceCreated

the date the owner first created the share with the specific email address

#### 7.64.3.10 serviceUpdated

Optional< Timestamp > qevercloud::SharedNotebook::serviceUpdated

the date the shared notebook was last updated on the service. This will be updated when authenticateToSharedNotebook is called the first time with a shared notebook requiring login (i.e. when the username is bound to that shared notebook).

### 7.64.3.11 shareKey

`Optional< QString > qevercloud::SharedNotebook::shareKey`

NOT DOCUMENTED

### 7.64.3.12 userId

`Optional< qint32 > qevercloud::SharedNotebook::userId`

the user id of the owner of the notebook

### 7.64.3.13 username

`Optional< QString > qevercloud::SharedNotebook::username`

the username of the user who can access this share. Once it's assigned it cannot be changed.

## 7.65 qevercloud::SharedNotebookInstanceRestrictions Struct Reference

```
#include <types.h>
```

### Public Types

- enum `type` { `ONLY_JOINED_OR_PREVIEW` = 1, `NO_SHARED_NOTEBOOKS` = 2 }

### 7.65.1 Detailed Description

An enumeration describing restrictions on the domain of shared notebook instances that are valid for a given operation, as used, for example, in [NotebookRestrictions](#).

`ONLY_JOINED_OR_PREVIEW`: The domain consists of shared notebooks that "belong" to the recipient or still available for preview by any recipient. Shared notebooks that the recipient has joined (the username has already been assigned to our user) are in the domain. Additionally, shared notebooks that allow preview and have not yet been joined are in the domain.

`NO_SHARED_NOTEBOOKS`: No shared notebooks are applicable to the operation.

### 7.65.2 Member Enumeration Documentation

#### 7.65.2.1 type

enum `qevercloud::SharedNotebookInstanceRestrictions::type`

## Enumerator

ONLY_JOINED_OR_PREVIEW	
NO_SHARED_NOTEBOOKS	

## 7.66 qevercloud::SharedNotebookPrivilegeLevel Struct Reference

```
#include <types.h>
```

### Public Types

- enum `type` {  
`READ_NOTEBOOK` = 0, `MODIFY_NOTEBOOK_PLUS_ACTIVITY` = 1, `READ_NOTEBOOK_PLUS_ACTIVITY` = 2, `GROUP` = 3,  
`FULL_ACCESS` = 4, `BUSINESS_FULL_ACCESS` = 5 }

#### 7.66.1 Detailed Description

Privilege levels for accessing shared notebooks.

`READ_NOTEBOOK`: Recipient is able to read the contents of the shared notebook but does not have access to information about other recipients of the notebook or the activity stream information.

`MODIFY_NOTEBOOK_PLUS_ACTIVITY`: Recipient has rights to read and modify the contents of the shared notebook, including the right to move notes to the trash and to create notes in the notebook. The recipient can also access information about other recipients and the activity stream.

`READ_NOTEBOOK_PLUS_ACTIVITY`: Recipient has `READ_NOTEBOOK` rights and can also access information about other recipients and the activity stream.

`GROUP`: If the user belongs to a group, such as a Business, that has a defined privilege level, use the privilege level of the group as the privilege for the individual.

`FULL_ACCESS`: Recipient has full rights to the shared notebook and recipient lists, including privilege to revoke and create invitations and to change privilege levels on invitations for individuals. This privilege level is primarily intended for use by individual shares.

`BUSINESS_FULL_ACCESS`: Intended for use with Business Notebooks, a `BUSINESS_FULL_ACCESS` level is `FULL_ACCESS` with the additional rights to change how the notebook will appear in the business library, including the rights to publish and unpublish the notebook from the library.

#### 7.66.2 Member Enumeration Documentation

##### 7.66.2.1 type

```
enum qevercloud::SharedNotebookPrivilegeLevel::type
```

## Enumerator

READ_NOTEBOOK	
MODIFY_NOTEBOOK_PLUS_ACTIVITY	
READ_NOTEBOOK_PLUS_ACTIVITY	
GROUP	
FULL_ACCESS	
BUSINESS_FULL_ACCESS	

## 7.67 qevercloud::SharedNotebookRecipientSettings Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [SharedNotebookRecipientSettings](#) &other) const
- bool [operator!=](#) (const [SharedNotebookRecipientSettings](#) &other) const

### Public Attributes

- [Optional](#)< bool > [reminderNotifyEmail](#)
- [Optional](#)< bool > [reminderNotifyInApp](#)

#### 7.67.1 Detailed Description

Settings meant for the recipient of a shared notebook, such as for indicating which types of notifications the recipient wishes for reminders, etc.

The `reminderNotifyEmail` and `reminderNotifyInApp` fields have a 3-state read value but a 2-state write value. On read, it is possible to observe "unset", true, or false. The initial state is "unset". When you choose to set a value, you may set it to either true or false, but you cannot unset the value. Once one of these members has a true/false value, it will always have a true/false value.

#### 7.67.2 Member Function Documentation

##### 7.67.2.1 `operator!=()`

```
bool qevercloud::SharedNotebookRecipientSettings::operator!= (
    const SharedNotebookRecipientSettings & other ) const [inline]
```



### 7.67.2.2 operator==( )

```
bool qevercloud::SharedNotebookRecipientSettings::operator== (
    const SharedNotebookRecipientSettings & other ) const [inline]
```

## 7.67.3 Member Data Documentation

### 7.67.3.1 reminderNotifyEmail

```
Optional< bool > qevercloud::SharedNotebookRecipientSettings::reminderNotifyEmail
```

Indicates that the user wishes to receive daily e-mail notifications for reminders associated with the shared notebook. This may be true only for business notebooks that belong to the business of which the user is a member. You may only set this value on a notebook in your business.

### 7.67.3.2 reminderNotifyInApp

```
Optional< bool > qevercloud::SharedNotebookRecipientSettings::reminderNotifyInApp
```

Indicates that the user wishes to receive notifications for reminders by applications that support providing such notifications. The exact nature of the notification is defined by the individual applications.

## 7.68 qevercloud::SponsoredGroupRole Struct Reference

```
#include <types.h>
```

### Public Types

- enum type { GROUP\_MEMBER = 1, GROUP\_ADMIN = 2, GROUP\_OWNER = 3 }

### 7.68.1 Detailed Description

Enumeration of the roles that a [User](#) can have within a sponsored group.

GROUP\_MEMBER: The user is a member of the group with no special privileges.

GROUP\_ADMIN: The user is an administrator within the group.

GROUP\_OWNER: The user is the owner of the group.

### 7.68.2 Member Enumeration Documentation

#### 7.68.2.1 type

```
enum qevercloud::SponsoredGroupRole::type
```

## Enumerator

GROUP_MEMBER	
GROUP_ADMIN	
GROUP_OWNER	

## 7.69 qevercloud::SyncChunk Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool [operator==](#) (const [SyncChunk](#) &other) const
- bool [operator!=](#) (const [SyncChunk](#) &other) const

### Public Attributes

- [Timestamp](#) [currentTime](#)
- [Optional](#)< [qint32](#) > [chunkHighUSN](#)
- [qint32](#) [updateCount](#)
- [Optional](#)< [QList](#)< [Note](#) > > [notes](#)
- [Optional](#)< [QList](#)< [Notebook](#) > > [notebooks](#)
- [Optional](#)< [QList](#)< [Tag](#) > > [tags](#)
- [Optional](#)< [QList](#)< [SavedSearch](#) > > [searches](#)
- [Optional](#)< [QList](#)< [Resource](#) > > [resources](#)
- [Optional](#)< [QList](#)< [Guid](#) > > [expungedNotes](#)
- [Optional](#)< [QList](#)< [Guid](#) > > [expungedNotebooks](#)
- [Optional](#)< [QList](#)< [Guid](#) > > [expungedTags](#)
- [Optional](#)< [QList](#)< [Guid](#) > > [expungedSearches](#)
- [Optional](#)< [QList](#)< [LinkedNotebook](#) > > [linkedNotebooks](#)
- [Optional](#)< [QList](#)< [Guid](#) > > [expungedLinkedNotebooks](#)

### 7.69.1 Detailed Description

This structure is given out by the [NoteStore](#) when a client asks to receive the current state of an account. The client asks for the server's state one chunk at a time in order to allow clients to retrieve the state of a large account without needing to transfer the entire account in a single message.

The server always gives SyncChunks using an ascending series of Update Sequence Numbers (USNs).

### 7.69.2 Member Function Documentation

### 7.69.2.1 operator!=(())

```
bool qevercloud::SyncChunk::operator!= (
    const SyncChunk & other ) const [inline]
```

### 7.69.2.2 operator==(())

```
bool qevercloud::SyncChunk::operator== (
    const SyncChunk & other ) const [inline]
```

## 7.69.3 Member Data Documentation

### 7.69.3.1 chunkHighUSN

`Optional< qint32 > qevercloud::SyncChunk::chunkHighUSN`

The highest USN for any of the data objects represented in this sync chunk. If there are no objects in the chunk, this will not be set.

### 7.69.3.2 currentTime

`Timestamp qevercloud::SyncChunk::currentTime`

The server's current date and time.

### 7.69.3.3 expungedLinkedNotebooks

`Optional< QList< Guid > > qevercloud::SyncChunk::expungedLinkedNotebooks`

If present, the GUIDs of all of the LinkedNotebooks that were permanently expunged in this chunk.

### 7.69.3.4 expungedNotebooks

`Optional< QList< Guid > > qevercloud::SyncChunk::expungedNotebooks`

If present, the GUIDs of all of the notebooks that were permanently expunged in this chunk. When a notebook is expunged, this implies that all of its child notes (and their resources) were also expunged.

### 7.69.3.5 expungedNotes

`Optional< QList< Guid > > qevercloud::SyncChunk::expungedNotes`

If present, the GUIDs of all of the notes that were permanently expunged in this chunk.

### 7.69.3.6 expungedSearches

`Optional< QList< Guid > > qevercloud::SyncChunk::expungedSearches`

If present, the GUIDs of all of the saved searches that were permanently expunged in this chunk.

### 7.69.3.7 expungedTags

`Optional< QList< Guid > > qevercloud::SyncChunk::expungedTags`

If present, the GUIDs of all of the tags that were permanently expunged in this chunk.

### 7.69.3.8 linkedNotebooks

`Optional< QList< LinkedNotebook > > qevercloud::SyncChunk::linkedNotebooks`

If present, this is a list of non-expunged LinkedNotebooks that have a USN in this chunk.

### 7.69.3.9 notebooks

`Optional< QList< Notebook > > qevercloud::SyncChunk::notebooks`

If present, this is a list of non-expunged notebooks that have a USN in this chunk. This will include notebooks that are "deleted" but not expunged (i.e. in the trash).

### 7.69.3.10 notes

`Optional< QList< Note > > qevercloud::SyncChunk::notes`

If present, this is a list of non-expunged notes that have a USN in this chunk. This will include notes that are "deleted" but not expunged (i.e. in the trash). The notes will include their list of tags and resources, but the note content, resource content, resource recognition data and resource alternate data will not be supplied.

### 7.69.3.11 resources

`Optional< QList< Resource > > qevercloud::SyncChunk::resources`

If present, this is a list of the non-expunged resources that have a USN in this chunk. This will include the metadata for each resource, but not its binary contents or recognition data, which must be retrieved separately.

### 7.69.3.12 searches

`Optional< QList< SavedSearch > > qevercloud::SyncChunk::searches`

If present, this is a list of non-expunged searches that have a USN in this chunk.

### 7.69.3.13 tags

`Optional< QList< Tag > > qevercloud::SyncChunk::tags`

If present, this is a list of the non-expunged tags that have a USN in this chunk.

### 7.69.3.14 updateCount

`qint32 qevercloud::SyncChunk::updateCount`

The total number of updates that have been performed in the service for this account. This is equal to the highest USN within the account at the point that this [SyncChunk](#) was generated. If `updateCount` and `chunkHighUSN` are identical, that means that this is the last chunk in the account ... there is no more recent information.

## 7.70 qevercloud::SyncChunkFilter Struct Reference

```
#include <types.h>
```

### Public Member Functions

- `bool operator== (const SyncChunkFilter &other) const`
- `bool operator!= (const SyncChunkFilter &other) const`

### Public Attributes

- `Optional< bool > includeNotes`
- `Optional< bool > includeNoteResources`
- `Optional< bool > includeNoteAttributes`
- `Optional< bool > includeNotebooks`
- `Optional< bool > includeTags`
- `Optional< bool > includeSearches`
- `Optional< bool > includeResources`
- `Optional< bool > includeLinkedNotebooks`
- `Optional< bool > includeExpunged`
- `Optional< bool > includeNoteApplicationDataFullMap`
- `Optional< bool > includeResourceApplicationDataFullMap`
- `Optional< bool > includeNoteResourceApplicationDataFullMap`
- `Optional< QString > requireNoteContentClass`

### 7.70.1 Detailed Description

This structure is used with the 'getFilteredSyncChunk' call to provide fine-grained control over the data that's returned when a client needs to synchronize with the service. Each flag in this structure specifies whether to include one class of data in the results of that call.

## 7.70.2 Member Function Documentation

### 7.70.2.1 operator!=(())

```
bool qevercloud::SyncChunkFilter::operator!= (
    const SyncChunkFilter & other ) const [inline]
```

### 7.70.2.2 operator==(())

```
bool qevercloud::SyncChunkFilter::operator== (
    const SyncChunkFilter & other ) const [inline]
```

## 7.70.3 Member Data Documentation

### 7.70.3.1 includeExpunged

```
Optional< bool > qevercloud::SyncChunkFilter::includeExpunged
```

If true, then the server will include the 'expunged' data for any type of included data. For example, if 'includeTags' and 'includeExpunged' are both true, then the SyncChunks.expungedTags field will be set with the GUIDs of tags that have been expunged from the server.

### 7.70.3.2 includeLinkedNotebooks

```
Optional< bool > qevercloud::SyncChunkFilter::includeLinkedNotebooks
```

If true, then the server will include the SyncChunks.linkedNotebooks field.

### 7.70.3.3 includeNoteApplicationDataFullMap

```
Optional< bool > qevercloud::SyncChunkFilter::includeNoteApplicationDataFullMap
```

If true, then the values for the applicationData map will be filled in, assuming notes and note attributes are being returned. Otherwise, only the keysOnly field will be filled in.

### 7.70.3.4 includeNoteAttributes

```
Optional< bool > qevercloud::SyncChunkFilter::includeNoteAttributes
```

If true, then the server will include the 'attributes' field on all of the Notes that are in SyncChunks.notes. If 'include↵ Notes' is false, then this will have no effect.

### 7.70.3.5 includeNotebooks

`Optional< bool > qevercloud::SyncChunkFilter::includeNotebooks`

If true, then the server will include the SyncChunks.notebooks field

### 7.70.3.6 includeNoteResourceApplicationDataFullMap

`Optional< bool > qevercloud::SyncChunkFilter::includeNoteResourceApplicationDataFullMap`

If true, then the fullMap values for the applicationData map will be filled in for resources found inside of notes, assuming resources are being returned in notes (includeNoteResources is true). Otherwise, only the keysOnly field will be filled in.

### 7.70.3.7 includeNoteResources

`Optional< bool > qevercloud::SyncChunkFilter::includeNoteResources`

If true, then the server will include the 'resources' field on all of the Notes that are in [SyncChunk.notes](#). If 'includeNotes' is false, then this will have no effect.

### 7.70.3.8 includeNotes

`Optional< bool > qevercloud::SyncChunkFilter::includeNotes`

If true, then the server will include the SyncChunks.notes field

### 7.70.3.9 includeResourceApplicationDataFullMap

`Optional< bool > qevercloud::SyncChunkFilter::includeResourceApplicationDataFullMap`

If true, then the fullMap values for the applicationData map will be filled in, assuming resources and resource attributes are being returned (includeResources is true). Otherwise, only the keysOnly field will be filled in.

### 7.70.3.10 includeResources

`Optional< bool > qevercloud::SyncChunkFilter::includeResources`

If true, then the server will include the SyncChunks.resources field. Since the Resources are also provided with their [Note](#) (in the Notes.resources list), this is primarily useful for clients that want to watch for changes to individual Resources due to recognition data being added.

### 7.70.3.11 includeSearches

`Optional< bool > qevercloud::SyncChunkFilter::includeSearches`

If true, then the server will include the SyncChunks.searches field

### 7.70.3.12 includeTags

```
Optional< bool > qevercloud::SyncChunkFilter::includeTags
```

If true, then the server will include the SyncChunks.tags field

### 7.70.3.13 requireNoteContentClass

```
Optional< QString > qevercloud::SyncChunkFilter::requireNoteContentClass
```

If set, then only send notes whose content class matches this value. The value can be a literal match or, if the last character is an asterisk, a prefix match.

## 7.71 qevercloud::SyncState Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const [SyncState](#) &other) const
- bool `operator!=` (const [SyncState](#) &other) const

### Public Attributes

- [Timestamp](#) `currentTime`
- [Timestamp](#) `fullSyncBefore`
- quint32 `updateCount`
- [Optional](#)< quint64 > `uploaded`

### 7.71.1 Detailed Description

This structure encapsulates the information about the state of the user's account for the purpose of "state based" synchronization.

### 7.71.2 Member Function Documentation

#### 7.71.2.1 `operator"!=()`

```
bool qevercloud::SyncState::operator!= (
    const SyncState & other ) const [inline]
```



### 7.71.2.2 operator==( )

```
bool qevercloud::SyncState::operator== (
    const SyncState & other ) const [inline]
```

## 7.71.3 Member Data Documentation

### 7.71.3.1 currentTime

Timestamp qevercloud::SyncState::currentTime

The server's current date and time.

### 7.71.3.2 fullSyncBefore

Timestamp qevercloud::SyncState::fullSyncBefore

The cutoff date and time for client caches to be updated via incremental synchronization. Any clients that were last synched with the server before this date/time must do a full resync of all objects. This cutoff point will change over time as archival data is deleted or special circumstances on the service require resynchronization.

### 7.71.3.3 updateCount

qint32 qevercloud::SyncState::updateCount

Indicates the total number of transactions that have been committed within the account. This reflects (for example) the number of discrete additions or modifications that have been made to the data in this account (tags, notes, resources, etc.). This number is the "high water mark" for Update Sequence Numbers (USN) within the account.

### 7.71.3.4 uploaded

Optional< qint64 > qevercloud::SyncState::uploaded

The total number of bytes that have been uploaded to this account in the current monthly period. This can be compared against [Accounting.uploadLimit](#) (from the [UserStore](#)) to determine how close the user is to their monthly upload limit. This value may not be present if the [SyncState](#) has been retrieved by a caller that only has read access to the account.

## 7.72 qevercloud::Tag Struct Reference

```
#include <types.h>
```

## Public Member Functions

- bool `operator==` (const [Tag](#) &other) const
- bool `operator!=` (const [Tag](#) &other) const

## Public Attributes

- [Optional](#)< [Guid](#) > `guid`
- [Optional](#)< [QString](#) > `name`
- [Optional](#)< [Guid](#) > `parentGuid`
- [Optional](#)< [qint32](#) > `updateSequenceNum`

### 7.72.1 Detailed Description

A tag within a user's account is a unique name which may be organized a simple hierarchy.

### 7.72.2 Member Function Documentation

#### 7.72.2.1 `operator"!=()`

```
bool qevercloud::Tag::operator!= (
    const Tag & other ) const [inline]
```

#### 7.72.2.2 `operator==(`

```
bool qevercloud::Tag::operator== (
    const Tag & other ) const [inline]
```

### 7.72.3 Member Data Documentation

#### 7.72.3.1 `guid`

[Optional](#)< [Guid](#) > qevercloud::Tag::guid

The unique identifier of this tag. Will be set by the service, so may be omitted by the client when creating the [Tag](#).  
Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX  
Regex: EDAM\_GUID\_REGEX

### 7.72.3.2 name

`Optional< QString > qevercloud::Tag::name`

A sequence of characters representing the tag's identifier. Case is preserved, but is ignored for comparisons. This means that an account may only have one tag with a given name, via case-insensitive comparison, so an account may not have both "food" and "Food" tags. May not contain a comma (','), and may not begin or end with a space. Length: EDAM\_TAG\_NAME\_LEN\_MIN - EDAM\_TAG\_NAME\_LEN\_MAX  
Regex: EDAM\_TAG\_NAME\_REGEX

### 7.72.3.3 parentGuid

`Optional< Guid > qevercloud::Tag::parentGuid`

If this is set, then this is the GUID of the tag that holds this tag within the tag organizational hierarchy. If this is not set, then the tag has no parent and it is a "top level" tag. Cycles are not allowed (e.g. a->parent->parent == a) and will be rejected by the service. Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX  
Regex: EDAM\_GUID\_REGEX

### 7.72.3.4 updateSequenceNum

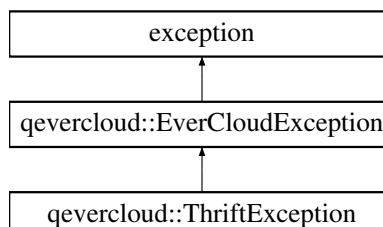
`Optional< qint32 > qevercloud::Tag::updateSequenceNum`

A number identifying the last transaction to modify the state of this object. The USN values are sequential within an account, and can be used to compare the order of modifications within the service.

## 7.73 qevercloud::ThriftException Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::ThriftException:



### Classes

- struct [Type](#)

## Public Member Functions

- [ThriftException](#) ()
- [ThriftException](#) ([Type::type](#) type)
- [ThriftException](#) ([Type::type](#) type, QString message)
- [Type::type](#) type () const
- const char \* [what](#) () const Q\_DECL\_OVERRIDE throw ()
- virtual QSharedPointer< [EverCloudExceptionData](#) > [exceptionData](#) () const Q\_DECL\_OVERRIDE

## Protected Attributes

- [Type::type](#) m\_type

### 7.73.1 Detailed Description

Errors of the Thrift protocol level. It could be wrongly formatted parameters or return values for example.

### 7.73.2 Constructor & Destructor Documentation

#### 7.73.2.1 [ThriftException\(\)](#) [1/3]

```
qevercloud::ThriftException::ThriftException ( )
```

#### 7.73.2.2 [ThriftException\(\)](#) [2/3]

```
qevercloud::ThriftException::ThriftException (
    Type::type type )
```

#### 7.73.2.3 [ThriftException\(\)](#) [3/3]

```
qevercloud::ThriftException::ThriftException (
    Type::type type,
    QString message )
```

### 7.73.3 Member Function Documentation

### 7.73.3.1 exceptionData()

```
QSharedPointer< EverCloudExceptionData > qevercloud::ThriftException::exceptionData ( ) const
[inline], [virtual]
```

Reimplemented from [qevercloud::EverCloudException](#).

### 7.73.3.2 type()

```
Type::type qevercloud::ThriftException::type ( ) const
```

### 7.73.3.3 what()

```
const char* qevercloud::ThriftException::what ( ) const throw ( )
```

## 7.73.4 Member Data Documentation

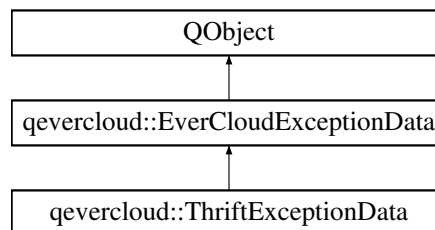
### 7.73.4.1 m\_type

```
Type::type qevercloud::ThriftException::m_type [protected]
```

## 7.74 qevercloud::ThriftExceptionData Class Reference

```
#include <exceptions.h>
```

Inheritance diagram for qevercloud::ThriftExceptionData:



## Public Member Functions

- [ThriftExceptionData](#) (QString error, [ThriftException::Type::type](#) type)
- virtual void [throwException](#) ( ) const Q\_DECL\_OVERRIDE

## Protected Attributes

- [ThriftException::Type::type m\\_type](#)

## Additional Inherited Members

### 7.74.1 Detailed Description

Asynchronous API conterpart of [ThriftException](#). See [EverCloudExceptionData](#) for more details.

### 7.74.2 Constructor & Destructor Documentation

#### 7.74.2.1 ThriftExceptionData()

```
qevercloud::ThriftExceptionData::ThriftExceptionData (
    QString error,
    ThriftException::Type::type type ) [explicit]
```

### 7.74.3 Member Function Documentation

#### 7.74.3.1 throwException()

```
virtual void qevercloud::ThriftExceptionData::throwException ( ) const [virtual]
```

If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant than call this function. Do not use `throw` statement, it's not polymorphic.

Reimplemented from [qevercloud::EverCloudExceptionData](#).

### 7.74.4 Member Data Documentation

#### 7.74.4.1 m\_type

```
ThriftException::Type::type qevercloud::ThriftExceptionData::m_type [protected]
```

## 7.75 qevercloud::Thumbnail Class Reference

The class is for downloading thumbnails for notes and resources from Evernote servers.

```
#include <thumbnail.h>
```

### Classes

- struct [ImageType](#)

### Public Member Functions

- [Thumbnail](#) ()  
*Default constructor.*
- [Thumbnail](#) (QString host, QString shardId, QString authenticationToken, int size=300, [ImageType::type](#) imageType=[ImageType::PNG](#))  
*Constructs [Thumbnail](#).*
- virtual [~Thumbnail](#) ()
- [Thumbnail](#) & [setHost](#) (QString host)
- [Thumbnail](#) & [setShardId](#) (QString shardId)
- [Thumbnail](#) & [setAuthenticationToken](#) (QString authenticationToken)
- [Thumbnail](#) & [setSize](#) (int size)
- [Thumbnail](#) & [setImageType](#) ([ImageType::type](#) imageType)
- QByteArray [download](#) ([Guid](#) guid, bool isPublic=false, bool isResourceGuid=false)  
*Downloads the thumbnail for a resource or a note.*
- [AsyncResult](#) \* [downloadAsync](#) ([Guid](#) guid, bool isPublic=false, bool isResourceGuid=false)
- QPair< QNetworkRequest, QByteArray > [createPostRequest](#) ([qevercloud::Guid](#) guid, bool isPublic=false, bool isResourceGuid=false)  
*Prepares a POST request for a thumbnail download.*

### 7.75.1 Detailed Description

The class is for downloading thumbnails for notes and resources from Evernote servers.

These thumbnails are not available with general EDAM Thrift interface as explained in the [documentation](#).

Usage:

```
Thumbnail thumb("www.evernote.com", sharId, authenticationToken);
QByteArray pngImage = thumb.download(noteGuid);
```

By default 300x300 PNG images are requested.

### 7.75.2 Constructor & Destructor Documentation

**7.75.2.1 Thumbnail()** [1/2]

```
qevercloud::Thumbnail::Thumbnail ( )
```

Default constructor.

host, shardId, authenticationToken have to be specified before calling [download](#) or [createPostRequest](#)

**7.75.2.2 Thumbnail()** [2/2]

```
qevercloud::Thumbnail::Thumbnail (
    QString host,
    QString shardId,
    QString authenticationToken,
    int size = 300,
    ImageType::type imageType = ImageType::PNG )
```

Constructs [Thumbnail](#).

**Parameters**

<i>host</i>	<a href="#">www.evernote.com</a> or <a href="#">sandbox.evernote.com</a>
<i>shardId</i>	You can get the value from <a href="#">UserStore</a> service or as a result of an authentication.
<i>authenticationToken</i>	For working private notes/resources you must supply a valid authentication token. For public resources the value specified is not used.
<i>size</i>	The size of the thumbnail. Evernote supports values from from 1 to 300. By default 300 is used.
<i>imageType</i>	<a href="#">Thumbnail</a> image type. See <a href="#">ImageType</a> . By default PNG is used.

**7.75.2.3 ~Thumbnail()**

```
virtual qevercloud::Thumbnail::~~Thumbnail ( ) [virtual]
```

**7.75.3 Member Function Documentation****7.75.3.1 createPostRequest()**

```
QPair<QNetworkRequest, QByteArray> qevercloud::Thumbnail::createPostRequest (
    qevercloud::Guid guid,
    bool isPublic = false,
    bool isResourceGuid = false )
```

Prepares a POST request for a thumbnail download.



## Parameters

<i>guid</i>	The note or resource guid
<i>isPublic</i>	Specify true for public notes/resources. In this case authentication token is not sent to with the request as it should be according to the docs.
<i>isResourceGuid</i>	true if guid denotes a resource and false if it denotes a note.

## Returns

a pair of QNetworkRequest for the POST request and data that must be posted with the request.

### 7.75.3.2 download()

```
QByteArray qevercloud::Thumbnail::download (
    Guid guid,
    bool isPublic = false,
    bool isResourceGuid = false )
```

Downloads the thumbnail for a resource or a note.

## Parameters

<i>guid</i>	The note or resource guid
<i>isPublic</i>	Specify true for public notes/resources. In this case authentication token is not sent to with the request as it should be according to the docs.
<i>isResourceGuid</i>	true if guid denotes a resource and false if it denotes a note.

## Returns

downloaded data.

### 7.75.3.3 downloadAsync()

```
AsyncResult* qevercloud::Thumbnail::downloadAsync (
    Guid guid,
    bool isPublic = false,
    bool isResourceGuid = false )
```

Asynchronous version of [download](#) function

### 7.75.3.4 setAuthenticationToken()

```
Thumbnail& qevercloud::Thumbnail::setAuthenticationToken (
    QString authenticationToken )
```

## Parameters

<i>authenticationToken</i>	For working private notes/resources you must supply a valid authentication token. For public resources the value specified is not used.
----------------------------	---

**7.75.3.5 setHost()**

```
Thumbnail& qevercloud::Thumbnail::setHost (
    QString host )
```

## Parameters

<i>host</i>	www.evernote.com or sandbox.evernote.com
-------------	--

**7.75.3.6 setImageType()**

```
Thumbnail& qevercloud::Thumbnail::setImageType (
    ImageType::type imageType )
```

## Parameters

<i>imageType</i>	<a href="#">Thumbnail</a> image type. See <a href="#">ImageType</a> . By default PNG is used.
------------------	---

**7.75.3.7 setShardId()**

```
Thumbnail& qevercloud::Thumbnail::setShardId (
    QString shardId )
```

## Parameters

<i>shardId</i>	You can get the value from <a href="#">UserStore</a> service or as a result of an authentication.
----------------	---

**7.75.3.8 setSize()**

```
Thumbnail& qevercloud::Thumbnail::setSize (
    int size )
```

## Parameters

size	The size of the thumbnail. Evernote supports values from from 1 to 300. By default 300 is used.
------	---

## 7.76 qevercloud::ThriftException::Type Struct Reference

```
#include <exceptions.h>
```

### Public Types

- enum `type` {  
`UNKNOWN` = 0, `UNKNOWN_METHOD` = 1, `INVALID_MESSAGE_TYPE` = 2, `WRONG_METHOD_NAME` = 3,  
`BAD_SEQUENCE_ID` = 4, `MISSING_RESULT` = 5, `INTERNAL_ERROR` = 6, `PROTOCOL_ERROR` = 7,  
`INVALID_DATA` = 8 }

### 7.76.1 Member Enumeration Documentation

#### 7.76.1.1 type

```
enum qevercloud::ThriftException::Type::type
```

#### Enumerator

UNKNOWN	
UNKNOWN_METHOD	
INVALID_MESSAGE_TYPE	
WRONG_METHOD_NAME	
BAD_SEQUENCE_ID	
MISSING_RESULT	
INTERNAL_ERROR	
PROTOCOL_ERROR	
INVALID_DATA	

## 7.77 qevercloud::User Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const `User` &other) const
- bool `operator!=` (const `User` &other) const

## Public Attributes

- [Optional< UserID > id](#)
- [Optional< QString > username](#)
- [Optional< QString > email](#)
- [Optional< QString > name](#)
- [Optional< QString > timezone](#)
- [Optional< PrivilegeLevel::type > privilege](#)
- [Optional< Timestamp > created](#)
- [Optional< Timestamp > updated](#)
- [Optional< Timestamp > deleted](#)
- [Optional< bool > active](#)
- [Optional< QString > shardId](#)
- [Optional< UserAttributes > attributes](#)
- [Optional< Accounting > accounting](#)
- [Optional< PremiumInfo > premiumInfo](#)
- [Optional< BusinessUserInfo > businessUserInfo](#)

### 7.77.1 Detailed Description

This represents the information about a single user account.

### 7.77.2 Member Function Documentation

#### 7.77.2.1 `operator!=(())`

```
bool qevercloud::User::operator!= (
    const User & other ) const [inline]
```

#### 7.77.2.2 `operator==(())`

```
bool qevercloud::User::operator== (
    const User & other ) const [inline]
```

### 7.77.3 Member Data Documentation

#### 7.77.3.1 `accounting`

[Optional< Accounting >](#) qevercloud::User::accounting

Bookkeeping information for the user's subscription.

### 7.77.3.2 active

`Optional< bool > qevercloud::User::active`

If the user account is available for login and synchronization, this flag will be set to true.

### 7.77.3.3 attributes

`Optional< UserAttributes > qevercloud::User::attributes`

If present, this will contain a list of the attributes for this user account.

### 7.77.3.4 businessUserInfo

`Optional< BusinessUserInfo > qevercloud::User::businessUserInfo`

If present, this will contain a set of business information relating to the user's business membership. If not present, the user is not currently part of a business.

### 7.77.3.5 created

`Optional< Timestamp > qevercloud::User::created`

The date and time when this user account was created in the service.

### 7.77.3.6 deleted

`Optional< Timestamp > qevercloud::User::deleted`

If the account has been deleted from the system (e.g. as the result of a legal request by the user), the date and time of the deletion will be represented here. If not, this value will not be set.

### 7.77.3.7 email

`Optional< QString > qevercloud::User::email`

The email address registered for the user. Must comply with RFC 2821 and RFC 2822.

Third party applications that authenticate using OAuth do not have access to this field. Length: EDAM\_EMAIL\_LENGTH\_MIN - EDAM\_EMAIL\_LENGTH\_MAX

Regex: EDAM\_EMAIL\_REGEX

### 7.77.3.8 id

`Optional< UserID > qevercloud::User::id`

The unique numeric identifier for the account, which will not change for the lifetime of the account.

### 7.77.3.9 name

`Optional< QString > qevercloud::User::name`

The printable name of the user, which may be a combination of given and family names. This is used instead of separate "first" and "last" names due to variations in international name format/order. May not start or end with a whitespace character. May contain any character but carriage return or newline (Unicode classes Zl and Zp).

Length: EDAM\_USER\_NAME\_LEN\_MIN - EDAM\_USER\_NAME\_LEN\_MAX

Regex: EDAM\_USER\_NAME\_REGEX

### 7.77.3.10 premiumInfo

`Optional< PremiumInfo > qevercloud::User::premiumInfo`

If present, this will contain a set of commerce information relating to the user's premium service level.

### 7.77.3.11 privilege

`Optional< PrivilegeLevel::type > qevercloud::User::privilege`

The level of access permitted for the user.

### 7.77.3.12 shardId

`Optional< QString > qevercloud::User::shardId`

DEPRECATED - Client applications should have no need to use this field.

### 7.77.3.13 timezone

`Optional< QString > qevercloud::User::timezone`

The zone ID for the user's default location. If present, this may be used to localize the display of any timestamp for which no other timezone is available. The format must be encoded as a standard zone ID such as "America/Los\_Angeles" or "GMT+08:00"

Length: EDAM\_TIMEZONE\_LEN\_MIN - EDAM\_TIMEZONE\_LEN\_MAX

Regex: EDAM\_TIMEZONE\_REGEX

### 7.77.3.14 updated

`Optional< Timestamp > qevercloud::User::updated`

The date and time when this user account was last modified in the service.

### 7.77.3.15 username

`Optional< QString > qevercloud::User::username`

The name that uniquely identifies a single user account. This name may be presented by the user, along with their password, to log into their account. May only contain a-z, 0-9, or '-', and may not start or end with the '-'

Length: EDAM\_USER\_USERNAME\_LEN\_MIN - EDAM\_USER\_USERNAME\_LEN\_MAX

Regex: EDAM\_USER\_USERNAME\_REGEX

## 7.78 qevercloud::UserAttributes Struct Reference

```
#include <types.h>
```

### Public Member Functions

- bool `operator==` (const `UserAttributes` &other) const
- bool `operator!=` (const `UserAttributes` &other) const

### Public Attributes

- `Optional< QString > defaultLocationName`
- `Optional< double > defaultLatitude`
- `Optional< double > defaultLongitude`
- `Optional< bool > preactivation`
- `Optional< QStringList > viewedPromotions`
- `Optional< QString > incomingEmailAddress`
- `Optional< QStringList > recentMailedAddresses`
- `Optional< QString > comments`
- `Optional< Timestamp > dateAgreedToTermsOfService`
- `Optional< qint32 > maxReferrals`
- `Optional< qint32 > referralCount`
- `Optional< QString > refererCode`
- `Optional< Timestamp > sentEmailDate`
- `Optional< qint32 > sentEmailCount`
- `Optional< qint32 > dailyEmailLimit`
- `Optional< Timestamp > emailOptOutDate`
- `Optional< Timestamp > partnerEmailOptInDate`
- `Optional< QString > preferredLanguage`
- `Optional< QString > preferredCountry`
- `Optional< bool > clipFullPage`
- `Optional< QString > twitterUserName`
- `Optional< QString > twitterId`
- `Optional< QString > groupName`
- `Optional< QString > recognitionLanguage`
- `Optional< QString > referralProof`
- `Optional< bool > educationalDiscount`
- `Optional< QString > businessAddress`
- `Optional< bool > hideSponsorBilling`
- `Optional< bool > taxExempt`
- `Optional< bool > useEmailAutoFiling`
- `Optional< ReminderEmailConfig::type > reminderEmailConfig`

### 7.78.1 Detailed Description

A structure holding the optional attributes that can be stored on a [User](#). These are generally less critical than the core [User](#) fields.

### 7.78.2 Member Function Documentation

#### 7.78.2.1 operator!=(())

```
bool qevercloud::UserAttributes::operator!= (
    const UserAttributes & other ) const [inline]
```

#### 7.78.2.2 operator==(())

```
bool qevercloud::UserAttributes::operator== (
    const UserAttributes & other ) const [inline]
```

### 7.78.3 Member Data Documentation

#### 7.78.3.1 businessAddress

```
Optional< QString > qevercloud::UserAttributes::businessAddress
```

A string recording the business address of a Sponsored Account user who has requested invoicing.

#### 7.78.3.2 clipFullPage

```
Optional< bool > qevercloud::UserAttributes::clipFullPage
```

Boolean flag set to true if the user wants to clip full pages by default when they use the web clipper without a selection.

#### 7.78.3.3 comments

```
Optional< QString > qevercloud::UserAttributes::comments
```

Free-form text field that may hold general support information, etc.  
Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX



#### 7.78.3.4 dailyEmailLimit

`Optional< qint32 > qevercloud::UserAttributes::dailyEmailLimit`

If set, this is the maximum number of emails that may be sent in a given day from this account. If unset, the server will use the configured default limit.

#### 7.78.3.5 dateAgreedToTermsOfService

`Optional< Timestamp > qevercloud::UserAttributes::dateAgreedToTermsOfService`

The date/time when the user agreed to the terms of service. This can be used as the effective "start date" for the account.

#### 7.78.3.6 defaultLatitude

`Optional< double > qevercloud::UserAttributes::defaultLatitude`

if set, this is the latitude that should be assigned to any notes that have no other latitude information.

#### 7.78.3.7 defaultLocationName

`Optional< QString > qevercloud::UserAttributes::defaultLocationName`

the location string that should be associated with the user in order to determine where notes are taken if not otherwise specified.

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.78.3.8 defaultLongitude

`Optional< double > qevercloud::UserAttributes::defaultLongitude`

if set, this is the longitude that should be assigned to any notes that have no other longitude information.

#### 7.78.3.9 educationalDiscount

`Optional< bool > qevercloud::UserAttributes::educationalDiscount`

NOT DOCUMENTED

#### 7.78.3.10 emailOptOutDate

`Optional< Timestamp > qevercloud::UserAttributes::emailOptOutDate`

If set, this is the date when the user asked to be excluded from offers and promotions sent by Evernote. If not set, then the user currently agrees to receive these messages.

### 7.78.3.11 groupName

`Optional< QString > qevercloud::UserAttributes::groupName`

A name identifier used to identify a particular set of branding and light customization.

### 7.78.3.12 hideSponsorBilling

`Optional< bool > qevercloud::UserAttributes::hideSponsorBilling`

A flag indicating whether to hide the billing information on a sponsored account owner's settings page

### 7.78.3.13 incomingEmailAddress

`Optional< QString > qevercloud::UserAttributes::incomingEmailAddress`

if set, this is the email address that the user may send email to in order to add an email note directly into the account via the SMTP email gateway. This is the part of the email address before the '@' symbol ... our domain is not included. If this is not set, the user may not add notes via the gateway.

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

### 7.78.3.14 maxReferrals

`Optional< qint32 > qevercloud::UserAttributes::maxReferrals`

The number of referrals that the user is permitted to make.

### 7.78.3.15 partnerEmailOptInDate

`Optional< Timestamp > qevercloud::UserAttributes::partnerEmailOptInDate`

If set, this is the date when the user asked to be included in offers and promotions sent by Evernote's partners. If not set, then the user currently does not agree to receive these emails.

### 7.78.3.16 preactivation

`Optional< bool > qevercloud::UserAttributes::preactivation`

if set, the user account is not yet confirmed for login. I.e. the account has been created, but we are still waiting for the user to complete the activation step.

### 7.78.3.17 preferredCountry

`Optional< QString > qevercloud::UserAttributes::preferredCountry`

Preferred country code based on ISO 3166-1-alpha-2 indicating the users preferred country

### 7.78.3.18 preferredLanguage

`Optional< QString > qevercloud::UserAttributes::preferredLanguage`

a 2 character language codes based on: <http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt> used for localization purposes to determine what language to use for the web interface and for other direct communication (e.g. emails).

### 7.78.3.19 recentMailedAddresses

`Optional< QStringList > qevercloud::UserAttributes::recentMailedAddresses`

if set, this will contain a list of email addresses that have recently been used as recipients of outbound emails by the user. This can be used to pre-populate a list of possible destinations when a user wishes to send a note via email. Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX each  
Max: EDAM\_USER\_RECENT\_MAILED\_ADDRESSES\_MAX entries

### 7.78.3.20 recognitionLanguage

`Optional< QString > qevercloud::UserAttributes::recognitionLanguage`

a 2 character language codes based on: <http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt> If set, this is used to determine the language that should be used when processing images and PDF files to find text. If not set, then the 'preferredLanguage' will be used.

### 7.78.3.21 refererCode

`Optional< QString > qevercloud::UserAttributes::refererCode`

A code indicating where the user was sent from. AKA promotion code

### 7.78.3.22 referralCount

`Optional< qint32 > qevercloud::UserAttributes::referralCount`

The number of referrals sent from this account.

### 7.78.3.23 referralProof

`Optional< QString > qevercloud::UserAttributes::referralProof`

NOT DOCUMENTED

### 7.78.3.24 reminderEmailConfig

`Optional< ReminderEmailConfig::type > qevercloud::UserAttributes::reminderEmailConfig`

Configuration state for whether or not the user wishes to receive reminder e-mail. This setting applies to both the reminder e-mail sent for personal reminder notes and for the reminder e-mail sent for reminder notes in the user's business notebooks that the user has configured for e-mail notifications.

### 7.78.3.25 **sentEmailCount**

```
Optional< qint32 > qevercloud::UserAttributes::sentEmailCount
```

The number of emails that were sent from the user via the service on sentEmailDate. Used to enforce a limit on the number of emails per user per day to prevent spamming.

### 7.78.3.26 **sentEmailDate**

```
Optional< Timestamp > qevercloud::UserAttributes::sentEmailDate
```

The most recent date when the user sent outbound emails from the service. Used with sentEmailCount to limit the number of emails that can be sent per day.

### 7.78.3.27 **taxExempt**

```
Optional< bool > qevercloud::UserAttributes::taxExempt
```

A flag indicating the user's sponsored group is exempt from sale tax

### 7.78.3.28 **twitterId**

```
Optional< QString > qevercloud::UserAttributes::twitterId
```

The unique identifier of the user's Twitter account if that user has chosen to enable Twittering into Evernote.

### 7.78.3.29 **twitterUserName**

```
Optional< QString > qevercloud::UserAttributes::twitterUserName
```

The username of the account of someone who has chosen to enable Twittering into Evernote. This value is subject to change, since users may change their Twitter user name.

### 7.78.3.30 **useEmailAutoFiling**

```
Optional< bool > qevercloud::UserAttributes::useEmailAutoFiling
```

A flag indicating whether the user chooses to allow Evernote to automatically file and tag emailed notes

### 7.78.3.31 **viewedPromotions**

```
Optional< QStringList > qevercloud::UserAttributes::viewedPromotions
```

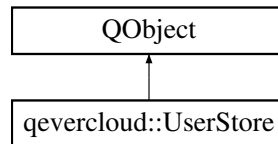
a list of promotions the user has seen. This list may occasionally be modified by the system when promotions are no longer available.

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

## 7.79 qevercloud::UserStore Class Reference

```
#include <services.h>
```

Inheritance diagram for qevercloud::UserStore:



### Public Member Functions

- [UserStore](#) (QString host, QString [authenticationToken](#)=QString(), QObject \*parent=0)
- void [setAuthenticationToken](#) (QString [authenticationToken](#))
- QString [authenticationToken](#) ()
- bool [checkVersion](#) (QString clientName, qint16 edamVersionMajor=[EDAM\\_VERSION\\_MAJOR](#), qint16 edamVersionMinor=[EDAM\\_VERSION\\_MINOR](#))
- [AsyncResult](#) \* [checkVersionAsync](#) (QString clientName, qint16 edamVersionMajor=[EDAM\\_VERSION\\_MAJOR](#), qint16 edamVersionMinor=[EDAM\\_VERSION\\_MINOR](#))
- [BootstrapInfo](#) [getBootstrapInfo](#) (QString locale)
- [AsyncResult](#) \* [getBootstrapInfoAsync](#) (QString locale)
- [AuthenticationResult](#) [authenticate](#) (QString username, QString password, QString consumerKey, QString consumerSecret, bool supportsTwoFactor)
- [AsyncResult](#) \* [authenticateAsync](#) (QString username, QString password, QString consumerKey, QString consumerSecret, bool supportsTwoFactor)
- [AuthenticationResult](#) [authenticateLongSession](#) (QString username, QString password, QString consumerKey, QString consumerSecret, QString deviceIdIdentifier, QString deviceDescription, bool supportsTwoFactor)
- [AsyncResult](#) \* [authenticateLongSessionAsync](#) (QString username, QString password, QString consumerKey, QString consumerSecret, QString deviceIdIdentifier, QString deviceDescription, bool supportsTwoFactor)
- [AuthenticationResult](#) [completeTwoFactorAuthentication](#) (QString oneTimeCode, QString deviceIdIdentifier, QString deviceDescription, QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [completeTwoFactorAuthenticationAsync](#) (QString oneTimeCode, QString deviceIdIdentifier, QString deviceDescription, QString [authenticationToken](#)=QString())
- void [revokeLongSession](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [revokeLongSessionAsync](#) (QString [authenticationToken](#)=QString())
- [AuthenticationResult](#) [authenticateToBusiness](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [authenticateToBusinessAsync](#) (QString [authenticationToken](#)=QString())
- [AuthenticationResult](#) [refreshAuthentication](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [refreshAuthenticationAsync](#) (QString [authenticationToken](#)=QString())
- [User](#) [getUser](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getUserAsync](#) (QString [authenticationToken](#)=QString())
- [PublicUserInfo](#) [getPublicUserInfo](#) (QString username)
- [AsyncResult](#) \* [getPublicUserInfoAsync](#) (QString username)
- [PremiumInfo](#) [getPremiumInfo](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getPremiumInfoAsync](#) (QString [authenticationToken](#)=QString())
- QString [getNoteStoreUrl](#) (QString [authenticationToken](#)=QString())
- [AsyncResult](#) \* [getNoteStoreUrlAsync](#) (QString [authenticationToken](#)=QString())

### 7.79.1 Detailed Description

Service: [UserStore](#)

The [UserStore](#) service is primarily used by EDAM clients to establish authentication via username and password over a trusted connection (e.g. SSL). A client's first call to this interface should be [checkVersion\(\)](#) to ensure that the client's software is up to date.

All calls which require an authenticationToken may throw an [EDAMUserException](#) for the following reasons:

- AUTH\_EXPIRED "authenticationToken" - token has expired
- BAD\_DATA\_FORMAT "authenticationToken" - token is malformed
- DATA\_REQUIRED "authenticationToken" - token is empty
- INVALID\_AUTH "authenticationToken" - token signature is invalid

### 7.79.2 Constructor & Destructor Documentation

#### 7.79.2.1 UserStore()

```
qevercloud::UserStore::UserStore (
    QString host,
    QString authenticationToken = QString(),
    QObject * parent = 0 ) [explicit]
```

### 7.79.3 Member Function Documentation

#### 7.79.3.1 authenticate()

```
AuthenticationResult qevercloud::UserStore::authenticate (
    QString username,
    QString password,
    QString consumerKey,
    QString consumerSecret,
    bool supportsTwoFactor )
```

This is used to check a username and password in order to create a short-lived authentication session that can be used for further actions.

This function is only available to Evernote's internal applications. Third party applications must authenticate using OAuth as described at [dev.evernote.com](http://dev.evernote.com).

## Parameters

<i>username</i>	The username (not numeric user ID) for the account to authenticate against. This function will also accept the user's registered email address in this parameter.
<i>password</i>	The plaintext password to check against the account. Since this is not protected by the EDAM protocol, this information must be provided over a protected transport (e.g. SSL).
<i>consumerKey</i>	The "consumer key" portion of the API key issued to the client application by Evernote.
<i>consumerSecret</i>	The "consumer secret" portion of the API key issued to the client application by Evernote.
<i>supportsTwoFactor</i>	Whether the calling application supports two-factor authentication. If this parameter is false, this method will fail with the error code INVALID_AUTH and the parameter "password" when called for a user who has enabled two-factor authentication.

## Returns

The result of the authentication. If the authentication was successful, the [AuthenticationResult.user](#) field will be set with the full information about the [User](#).

If the user has two-factor authentication enabled, [AuthenticationResult.secondFactorRequired](#) will be set and [AuthenticationResult.authenticationToken](#) will contain a short-lived token that may only be used to complete the two-factor authentication process by calling [UserStore.completeTwoFactorAuthentication](#).

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "username" - username is empty</li> <li>• DATA_REQUIRED "password" - password is empty</li> <li>• DATA_REQUIRED "consumerKey" - consumerKey is empty</li> <li>• INVALID_AUTH "username" - username not found</li> <li>• INVALID_AUTH "password" - password did not match</li> <li>• INVALID_AUTH "consumerKey" - consumerKey is not authorized</li> <li>• INVALID_AUTH "consumerSecret" - consumerSecret is incorrect</li> <li>• PERMISSION_DENIED "User.active" - user account is closed</li> <li>• PERMISSION_DENIED "User.tooManyFailuresTryAgainLater" - user has failed authentication too often</li> </ul>
-----------------------------------	---

## 7.79.3.2 authenticateAsync()

```
AsyncResult* qevercloud::UserStore::authenticateAsync (
    QString username,
    QString password,
    QString consumerKey,
```

```
QString consumerSecret,
bool supportsTwoFactor )
```

Asynchronous version of [authenticate](#)

### 7.79.3.3 authenticateLongSession()

```
AuthenticationResult qevercloud::UserStore::authenticateLongSession (
    QString username,
    QString password,
    QString consumerKey,
    QString consumerSecret,
    QString deviceIdIdentifier,
    QString deviceDescription,
    bool supportsTwoFactor )
```

This is used to check a username and password in order to create a long-lived authentication token that can be used for further actions.

This function is not available to most third party applications, which typically authenticate using OAuth as described at [dev.evernote.com](http://dev.evernote.com). If you believe that your application requires permission to authenticate using username and password instead of OAuth, please contact Evernote developer support by visiting [dev.evernote.com](http://dev.evernote.com).

#### Parameters

<i>username</i>	The username or registered email address of the account to authenticate against.
<i>password</i>	The plaintext password to check against the account. Since this is not protected by the EDAM protocol, this information must be provided over a protected transport (i.e. SSL).
<i>consumerKey</i>	The "consumer key" portion of the API key issued to the client application by Evernote.
<i>consumerSecret</i>	The "consumer secret" portion of the API key issued to the client application by Evernote.
<i>deviceIdIdentifier</i>	An optional string, no more than 32 characters in length, that uniquely identifies the device from which the authentication is being performed. This string allows the service to return the same authentication token when a given application requests authentication repeatedly from the same device. This may happen when the user logs out of an application and then logs back in, or when the application is uninstalled and later reinstalled. If no reliable device identifier can be created, this value should be omitted. If set, the device identifier must be between 1 and EDAM_DEVICE_ID_LEN_MAX characters long and must match the regular expression EDAM_DEVICE_ID_REGEX.
<i>deviceDescription</i>	A description of the device from which the authentication is being performed. This field is displayed to the user in a list of authorized applications to allow them to distinguish between multiple tokens issued to the same client application on different devices. For example, the Evernote iOS client on a user's iPhone and iPad might pass the iOS device names "Bob's iPhone" and "Bob's iPad". The device description must be between 1 and EDAM_DEVICE_DESCRIPTION_LEN_MAX characters long and must match the regular expression EDAM_DEVICE_DESCRIPTION_REGEX.
<i>supportsTwoFactor</i>	Whether the calling application supports two-factor authentication. If this parameter is false, this method will fail with the error code INVALID_AUTH and the parameter "password" when called for a user who has enabled two-factor authentication.

#### Returns

The result of the authentication. The level of detail provided in the returned AuthenticationResult.User structure depends on the access level granted by calling application's API key.



If the user has two-factor authentication enabled, [AuthenticationResult.secondFactorRequired](#) will be set and [AuthenticationResult.authenticationToken](#) will contain a short-lived token that may only be used to complete the two-factor authentication process by calling [UserStore.completeTwoFactorAuthentication](#).

#### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "username" - username is empty</li> <li>• DATA_REQUIRED "password" - password is empty</li> <li>• DATA_REQUIRED "consumerKey" - consumerKey is empty</li> <li>• DATA_REQUIRED "consumerSecret" - consumerSecret is empty</li> <li>• DATA_REQUIRED "deviceDescription" - deviceDescription is empty</li> <li>• BAD_DATA_FORMAT "deviceDescription" - deviceDescription is not valid.</li> <li>• BAD_DATA_FORMAT "deviceIdIdentifier" - deviceIdIdentifier is not valid.</li> <li>• INVALID_AUTH "username" - username not found</li> <li>• INVALID_AUTH "password" - password did not match</li> <li>• INVALID_AUTH "consumerKey" - consumerKey is not authorized</li> <li>• INVALID_AUTH "consumerSecret" - consumerSecret is incorrect</li> <li>• PERMISSION_DENIED "User.active" - user account is closed</li> <li>• PERMISSION_DENIED "User.tooManyFailuresTryAgainLater" - user has failed authentication too often</li> </ul>
-----------------------------------	---

#### 7.79.3.4 authenticateLongSessionAsync()

```
AsyncResult* qevercloud::UserStore::authenticateLongSessionAsync (
    QString username,
    QString password,
    QString consumerKey,
    QString consumerSecret,
    QString deviceIdIdentifier,
    QString deviceDescription,
    bool supportsTwoFactor )
```

Asynchronous version of [authenticateLongSession](#)

#### 7.79.3.5 authenticateToBusiness()

```
AuthenticationResult qevercloud::UserStore::authenticateToBusiness (
    QString authenticationToken = QString() )
```

This is used to take an existing authentication token that grants access to an individual user account (returned from 'authenticate', 'authenticateLongSession' or an OAuth authorization) and obtain an additional authentication token that may be used to access business notebooks if the user is a member of an Evernote Business account.

The resulting authentication token may be used to make [NoteStore](#) API calls against the business using the [NoteStore](#) URL returned in the result.

## Parameters

<i>authenticationToken</i>	The authentication token for the user. This may not be a shared authentication token (returned by <a href="#">NoteStore.authenticateToSharedNotebook</a> or <a href="#">NoteStore.authenticateToSharedNote</a> ) or a business authentication token.
----------------------------	--

## Returns

The result of the authentication, with the token granting access to the business in the result's 'authenticationToken' field. The URL that must be used to access the business account [NoteStore](#) will be returned in the result's 'noteStoreUrl' field. The 'User' field will not be set in the result.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "authenticationToken" - the provided authentication token is a shared or business authentication token.</li> <li>• PERMISSION_DENIED "Business" - the user identified by the provided authentication token is not currently a member of a business.</li> <li>• PERMISSION_DENIED "Business.status" - the business that the user is a member of is not currently in an active status.</li> </ul>
-----------------------------------	--

**7.79.3.6 authenticateToBusinessAsync()**

```
AsyncResult* qevercloud::UserStore::authenticateToBusinessAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [authenticateToBusiness](#)

**7.79.3.7 authenticationToken()**

```
QString qevercloud::UserStore::authenticationToken ( ) [inline]
```

**7.79.3.8 checkVersion()**

```
bool qevercloud::UserStore::checkVersion (
    QString clientName,
    qint16 edamVersionMajor = EDAM_VERSION_MAJOR,
    qint16 edamVersionMinor = EDAM_VERSION_MINOR )
```

This should be the first call made by a client to the EDAM service. It tells the service what protocol version is used by the client. The service will then return true if the client is capable of talking to the service, and false if the client's protocol version is incompatible with the service, so the client must upgrade. If a client receives a false value, it should report the incompatibility to the user and not continue with any more EDAM requests ([UserStore](#) or [NoteStore](#)).

## Parameters

<i>clientName</i>	This string provides some information about the client for tracking/logging on the service. It should provide information about the client's software and platform. The structure should be: application/version; platform/version; [ device/version ] E.g. "Evernote Windows/3.0.1; Windows/XP SP3".
<i>edamVersionMajor</i>	This should be the major protocol version that was compiled by the client. This should be the current value of the EDAM_VERSION_MAJOR constant for the client.
<i>edamVersionMinor</i>	This should be the major protocol version that was compiled by the client. This should be the current value of the EDAM_VERSION_MINOR constant for the client.

## 7.79.3.9 checkVersionAsync()

```
AsyncResult* qevercloud::UserStore::checkVersionAsync (
    QString clientName,
    qint16 edamVersionMajor = EDAM_VERSION_MAJOR,
    qint16 edamVersionMinor = EDAM_VERSION_MINOR )
```

Asynchronous version of [checkVersion](#)

## 7.79.3.10 completeTwoFactorAuthentication()

```
AuthenticationResult qevercloud::UserStore::completeTwoFactorAuthentication (
    QString oneTimeCode,
    QString deviceIdIdentifier,
    QString deviceDescription,
    QString authenticationToken = QString() )
```

Complete the authentication process when a second factor is required. This call is made after a successful call to [authenticate](#) or [authenticateLongSession](#) when the authenticating user has enabled two-factor authentication.

## Parameters

<i>authenticationToken</i>	An authentication token returned by a previous call to <a href="#">UserStore.authenticate</a> or <a href="#">UserStore.authenticateLongSession</a> that could not be completed in a single call because a second factor was required.
<i>oneTimeCode</i>	The one time code entered by the user. This value is delivered out-of-band, typically via SMS or an authenticator application.
<i>deviceIdIdentifier</i>	See the corresponding parameter in <a href="#">authenticateLongSession</a> .
<i>deviceDescription</i>	See the corresponding parameter in <a href="#">authenticateLongSession</a> .

## Returns

The result of the authentication. The level of detail provided in the returned [AuthenticationResult](#).User structure depends on the access level granted by the calling application's API key. If the initial authentication call was made to [authenticateLongSession](#), the [AuthenticationResult](#) will contain a long-lived authentication token.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "authenticationToken" - authenticationToken is empty</li> <li>• DATA_REQUIRED "oneTimeCode" - oneTimeCode is empty</li> <li>• BAD_DATA_FORMAT "authenticationToken" - authenticationToken is not well formed</li> <li>• INVALID_AUTH "oneTimeCode" - oneTimeCode did not match</li> <li>• AUTH_EXPIRED "authenticationToken" - authenticationToken has expired</li> <li>• PERMISSION_DENIED "authenticationToken" - authenticationToken is not valid</li> <li>• PERMISSION_DENIED "User.active" - user account is closed</li> <li>• PERMISSION_DENIED "User.tooManyFailuresTryAgainLater" - user has failed authentication too often</li> <li>• DATA_CONFLICT "User.twoFactorAuthentication" - The user has not enabled two-factor authentication.</li> </ul>
-----------------------------------	--

**7.79.3.11 completeTwoFactorAuthenticationAsync()**

```
AsyncResult* qevercloud::UserStore::completeTwoFactorAuthenticationAsync (
    QString oneTimeCode,
    QString deviceIdIdentifier,
    QString deviceDescription,
    QString authenticationToken = QString() )
```

Asynchronous version of [completeTwoFactorAuthentication](#)

**7.79.3.12 getBootstrapInfo()**

```
BootstrapInfo qevercloud::UserStore::getBootstrapInfo (
    QString locale )
```

This provides bootstrap information to the client. Various bootstrap profiles and settings may be used by the client to configure itself.

## Parameters

<i>locale</i>	The client's current locale, expressed in language[_country] format. E.g., "en_US". See ISO-639 and ISO-3166 for valid language and country codes.
---------------	--

## Returns

The bootstrap information suitable for this client.

### 7.79.3.13 getBootstrapInfoAsync()

```
AsyncResult* qevercloud::UserStore::getBootstrapInfoAsync (
    QString locale )
```

Asynchronous version of [getBootstrapInfo](#)

### 7.79.3.14 getNoteStoreUrl()

```
QString qevercloud::UserStore::getNoteStoreUrl (
    QString authenticationToken = QString() )
```

Returns the URL that should be used to talk to the [NoteStore](#) for the account represented by the provided authenticationToken. This method isn't needed by most clients, who can retrieve the correct [NoteStore](#) URL from the [AuthenticationResult](#) returned from the authenticate or refreshAuthentication calls. This method is typically only needed to look up the correct URL for a long-lived session token (e.g. for an OAuth web service).

### 7.79.3.15 getNoteStoreUrlAsync()

```
AsyncResult* qevercloud::UserStore::getNoteStoreUrlAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [getNoteStoreUrl](#)

### 7.79.3.16 getPremiumInfo()

```
PremiumInfo qevercloud::UserStore::getPremiumInfo (
    QString authenticationToken = QString() )
```

Returns information regarding a user's Premium account corresponding to the provided authentication token, or throws an exception if this token is not valid.

### 7.79.3.17 getPremiumInfoAsync()

```
AsyncResult* qevercloud::UserStore::getPremiumInfoAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [getPremiumInfo](#)

### 7.79.3.18 getPublicUserInfo()

```
PublicUserInfo qevercloud::UserStore::getPublicUserInfo (
    QString username )
```

Asks the [UserStore](#) about the publicly available location information for a particular username.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>DATA_REQUIRED "username" - username is empty</li> </ul>
-----------------------------------	--

**7.79.3.19 getPublicUserInfoAsync()**

```
AsyncResult* qevercloud::UserStore::getPublicUserInfoAsync (
    QString username )
```

Asynchronous version of [getPublicUserInfo](#)

**7.79.3.20 getUser()**

```
User qevercloud::UserStore::getUser (
    QString authenticationToken = QString() )
```

Returns the [User](#) corresponding to the provided authentication token, or throws an exception if this token is not valid. The level of detail provided in the returned [User](#) structure depends on the access level granted by the token, so a web service client may receive fewer fields than an integrated desktop client.

**7.79.3.21 getUserAsync()**

```
AsyncResult* qevercloud::UserStore::getUserAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [getUser](#)

**7.79.3.22 refreshAuthentication()**

```
AuthenticationResult qevercloud::UserStore::refreshAuthentication (
    QString authenticationToken = QString() )
```

This is used to take an existing authentication token (returned from 'authenticate') and exchange it for a newer token which will not expire as soon. This must be invoked before the previous token expires.

This function is only available to Evernote's internal applications.

## Parameters

<i>authenticationToken</i>	The previous authentication token from the <a href="#">authenticate()</a> result.
----------------------------	---

### Returns

The result of the authentication, with the new token in the result's 'authenticationToken' field. The 'User' field will not be set in the result.

#### 7.79.3.23 refreshAuthenticationAsync()

```
AsyncResult* qevercloud::UserStore::refreshAuthenticationAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [refreshAuthentication](#)

#### 7.79.3.24 revokeLongSession()

```
void qevercloud::UserStore::revokeLongSession (
    QString authenticationToken = QString() )
```

Revoke an existing long lived authentication token. This can be used to revoke OAuth tokens or tokens created by calling `authenticateLongSession`, and allows a user to effectively log out of Evernote from the perspective of the application that holds the token. The authentication token that is passed is immediately revoked and may not be used to call any authenticated EDAM function.

### Parameters

<i>authenticationToken</i>	the authentication token to revoke.
----------------------------	-------------------------------------

### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"><li>• DATA_REQUIRED "authenticationToken" - no authentication token provided</li><li>• BAD_DATA_FORMAT "authenticationToken" - the authentication token is not well formed</li><li>• INVALID_AUTH "authenticationToken" - the authentication token is invalid</li><li>• AUTH_EXPIRED "authenticationToken" - the authentication token is expired or is already revoked.</li></ul>
-----------------------------------	---

#### 7.79.3.25 revokeLongSessionAsync()

```
AsyncResult* qevercloud::UserStore::revokeLongSessionAsync (
    QString authenticationToken = QString() )
```

Asynchronous version of [revokeLongSession](#)

**7.79.3.26 setAuthenticationToken()**

```
void qevercloud::UserStore::setAuthenticationToken (
    QString authenticationToken ) [inline]
```



## Chapter 8

# File Documentation

### 8.1 AsyncResult.h File Reference

```
#include "qt4helpers.h"
#include "export.h"
#include "EverCloudException.h"
#include <QObject>
#include <QNetworkRequest>
```

#### Classes

- class [qevercloud::AsyncResult](#)  
*Returned by asynchronous versions of functions.*

#### Namespaces

- [qevercloud](#)

### 8.2 constants.h File Reference

```
#include "../Optional.h"
#include "../export.h"
#include <QMap>
#include <QList>
#include <QSet>
#include <QString>
#include <QStringList>
#include <QByteArray>
#include <QDateTime>
#include <QMetaType>
```

## Namespaces

- [qevercloud](#)

## Variables

- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_ATTRIBUTE\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_ATTRIBUTE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_ATTRIBUTE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_ATTRIBUTE\\_LIST\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_ATTRIBUTE\\_MAP\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_GUID\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_GUID\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_GUID\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_EMAIL\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_EMAIL\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_EMAIL\\_LOCAL\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_EMAIL\\_DOMAIN\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_EMAIL\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_VAT\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_TIMEZONE\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_TIMEZONE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_TIMEZONE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_MIME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_MIME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_GIF](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_JPEG](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_PNG](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_WAV](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_MP3](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_AMR](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_AAC](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_M4A](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_MP4\\_VIDEO](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_INK](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_PDF](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_MIME\\_TYPE\\_DEFAULT](#)
- const [QEVERCLOUD\\_EXPORT](#) QSet< QString > [qevercloud::EDAM\\_MIME\\_TYPES](#)
- const [QEVERCLOUD\\_EXPORT](#) QSet< QString > [qevercloud::EDAM\\_INDEXABLE\\_RESOURCE\\_MIME\\_TYPES](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_SEARCH\\_QUERY\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_SEARCH\\_QUERY\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_SEARCH\\_QUERY\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_HASH\\_LEN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_USERNAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_USERNAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_USER\\_USERNAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_USER\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_TAG\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_TAG\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_TAG\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_TITLE\\_LEN\\_MIN](#)

- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_TITLE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_NOTE\\_TITLE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_CONTENT\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_CONTENT\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_APPLICATIONDATA\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_APPLICATIONDATA\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_APPLICATIONDATA\\_VALUE\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_APPLICATIONDATA\\_VALUE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_APPLICATIONDATA\\_ENTRY\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_APPLICATIONDATA\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_APPLICATIONDATA\\_VALUE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTEBOOK\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTEBOOK\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_NOTEBOOK\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTEBOOK\\_STACK\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTEBOOK\\_STACK\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_NOTEBOOK\\_STACK\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PUBLISHING\\_URI\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PUBLISHING\\_URI\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_PUBLISHING\\_URI\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QSet< QString > [qevercloud::EDAM\\_PUBLISHING\\_URI\\_PROHIBITED](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PUBLISHING\\_DESCRIPTION\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PUBLISHING\\_DESCRIPTION\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_PUBLISHING\\_DESCRIPTION\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_SAVED\\_SEARCH\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_SAVED\\_SEARCH\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_SAVED\\_SEARCH\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_PASSWORD\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_PASSWORD\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_USER\\_PASSWORD\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_BUSINESS\\_URI\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_TAGS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_RESOURCES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_TAGS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_BUSINESS\\_TAGS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_SAVED\\_SEARCHES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_NOTES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_BUSINESS\\_NOTES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_NOTEBOOKS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_BUSINESS\\_NOTEBOOKS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_RECENT\\_MAILED\\_ADDRESSES\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_MAIL\\_LIMIT\\_DAILY\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_MAIL\\_LIMIT\\_DAILY\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint64 [qevercloud::EDAM\\_USER\\_UPLOAD\\_LIMIT\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint64 [qevercloud::EDAM\\_USER\\_UPLOAD\\_LIMIT\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint64 [qevercloud::EDAM\\_USER\\_UPLOAD\\_LIMIT\\_BUSINESS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_SIZE\\_MAX\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_SIZE\\_MAX\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_RESOURCE\\_SIZE\\_MAX\\_FREE](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_RESOURCE\\_SIZE\\_MAX\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_LINKED\\_NOTEBOOK\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_USER\\_LINKED\\_NOTEBOOK\\_MAX\\_PREMIUM](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTEBOOK\\_SHARED\\_NOTEBOOK\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_CONTENT\\_CLASS\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_NOTE\\_CONTENT\\_CLASS\\_LEN\\_MAX](#)

- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_NOTE\\_CONTENT\\_CLASS\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_HELLO\\_APP\\_CONTENT\\_CLASS\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_FOOD\\_APP\\_CONTENT\\_CLASS\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_HELLO\\_ENCOUNTER](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_HELLO\\_PROFILE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_FOOD\\_MEAL](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_SKITCH\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_SKITCH](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_SKITCH\\_PDF](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_PENULTIMATE\\_PREFIX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_CONTENT\\_CLASS\\_PENULTIMATE\\_NOTEBOOK](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_RELATED\\_PLAINTEXT\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_RELATED\\_PLAINTEXT\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_RELATED\\_MAX\\_NOTES](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_RELATED\\_MAX\\_NOTEBOOKS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_RELATED\\_MAX\\_TAGS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_BUSINESS\\_PHONE\\_NUMBER\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PREFERENCE\\_NAME\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PREFERENCE\\_NAME\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PREFERENCE\\_VALUE\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PREFERENCE\\_VALUE\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_MAX\\_PREFERENCES](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_MAX\\_VALUES\\_PER\\_PREFERENCE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_PREFERENCE\\_NAME\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_PREFERENCE\\_VALUE\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_PREFERENCE\\_SHORTCUTS](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_PREFERENCE\\_SHORTCUTS\\_MAX\\_VALUES](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_DEVICE\\_ID\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_DEVICE\\_ID\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_DEVICE\\_DESCRIPTION\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_DEVICE\\_DESCRIPTION\\_REGEX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_SEARCH\\_SUGGESTIONS\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_SEARCH\\_SUGGESTIONS\\_PREFIX\\_LEN\\_MAX](#)
- const [QEVERCLOUD\\_EXPORT](#) qint32 [qevercloud::EDAM\\_SEARCH\\_SUGGESTIONS\\_PREFIX\\_LEN\\_MIN](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::CLASSIFICATION\\_RECIPE\\_USER\\_NON\\_RECIPE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::CLASSIFICATION\\_RECIPE\\_USER\\_RECIPE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::CLASSIFICATION\\_RECIPE\\_SERVICE\\_RECIPE](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_NOTE\\_SOURCE\\_WEB\\_CLIP](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_NOTE\\_SOURCE\\_MAIL\\_CLIP](#)
- const [QEVERCLOUD\\_EXPORT](#) QString [qevercloud::EDAM\\_NOTE\\_SOURCE\\_MAIL\\_SMTP\\_GATEWAY](#)
- const [QEVERCLOUD\\_EXPORT](#) qint16 [qevercloud::EDAM\\_VERSION\\_MAJOR](#)
- const [QEVERCLOUD\\_EXPORT](#) qint16 [qevercloud::EDAM\\_VERSION\\_MINOR](#)

### 8.3 EDAMErrorCode.h File Reference

```
#include "../export.h"
```

## Classes

- struct [qevercloud::EDAMErrorCode](#)

## Namespaces

- [qevercloud](#)

## 8.4 EventLoopFinisher.h File Reference

```
#include "qt4helpers.h"
#include "export.h"
#include <QObject>
#include <QEventLoop>
```

## Classes

- class [qevercloud::EventLoopFinisher](#)

## Namespaces

- [qevercloud](#)

## 8.5 EverCloudException.h File Reference

```
#include "qt4helpers.h"
#include "export.h"
#include <QObject>
#include <QString>
#include <QSharedPointer>
#include <exception>
```

## Classes

- class [qevercloud::EverCloudException](#)
- class [qevercloud::EverCloudExceptionData](#)  
*EverCloudException counterpart for asynchronous API.*
- class [qevercloud::EvernoteException](#)
- class [qevercloud::EvernoteExceptionData](#)

## Namespaces

- [qevercloud](#)

## Variables

- class [QEVERCLOUD\\_EXPORT](#) [qevercloud::EverCloudExceptionData](#)

## 8.6 exceptions.h File Reference

```
#include "Optional.h"
#include "export.h"
#include "EverCloudException.h"
#include "generated/EDAMErrorCode.h"
#include "generated/types.h"
#include <QString>
#include <QObject>
#include <QSharedPointer>
```

## Classes

- class [qevercloud::ThriftException](#)
- struct [qevercloud::ThriftException::Type](#)
- class [qevercloud::ThriftExceptionData](#)
- class [qevercloud::EDAMUserExceptionData](#)
- class [qevercloud::EDAMSystemExceptionData](#)
- class [qevercloud::EDAMNotFoundExceptionData](#)
- class [qevercloud::EDAMSystemExceptionRateLimitReached](#)
- class [qevercloud::EDAMSystemExceptionRateLimitReachedData](#)
- class [qevercloud::EDAMSystemExceptionAuthExpired](#)
- class [qevercloud::EDAMSystemExceptionAuthExpiredData](#)

## Namespaces

- [qevercloud](#)

## 8.7 export.h File Reference

```
#include <QtCore/QtGlobal>
```

## Macros

- `#define` [QEVERCLOUD\\_EXPORT](#)

### 8.7.1 Macro Definition Documentation

### 8.7.1.1 QEVERCLOUD\_EXPORT

```
#define QEVERCLOUD_EXPORT
```

Original work: Copyright (c) 2014 Sergey Skoblikov Modified work: Copyright (c) 2015-2016 Dmitry Ivanov

This file is a part of QEverCloud project and is distributed under the terms of MIT license: <https://opensource.org/licenses/MIT>

## 8.8 globals.h File Reference

```
#include "export.h"  
#include <QNetworkAccessManager>
```

### Namespaces

- [qevercloud](#)

### Functions

- [QEVERCLOUD\\_EXPORT](#) [QNetworkAccessManager](#) \* [qevercloud::evernoteNetworkAccessManager](#) ()
- [QEVERCLOUD\\_EXPORT](#) int [qevercloud::libraryVersion](#) ()

## 8.9 oauth.h File Reference

```
#include "generated/types.h"  
#include "export.h"  
#include "qt4helpers.h"  
#include <QDialog>  
#include <QString>
```

### Classes

- class [qevercloud::EvernoteOAuthWebView](#)  
*The class is tailored specifically for OAuth authorization with Evernote.*
- struct [qevercloud::EvernoteOAuthWebView::OAuthResult](#)
- class [qevercloud::EvernoteOAuthDialog](#)  
*Authorizes your app with the Evernote service by means of OAuth authentication.*

### Namespaces

- [qevercloud](#)

## Functions

- void [qevercloud::setNonceGenerator](#) (quint64(\*nonceGenerator)())  
*Sets the function to use for nonce generation for OAuth authentication.*

## 8.10 Optional.h File Reference

```
#include "EverCloudException.h"  
#include <algorithm>
```

## Classes

- class [qevercloud::Optional](#)< T >

## Namespaces

- [qevercloud](#)

## 8.11 QEverCloud.h File Reference

```
#include "AsyncResult.h"  
#include "EventLoopFinisher.h"  
#include "EverCloudException.h"  
#include "exceptions.h"  
#include "export.h"  
#include "globals.h"  
#include "Optional.h"  
#include "qt4helpers.h"  
#include "thumbnail.h"  
#include "generated/EDAMErrorCode.h"  
#include "generated/constants.h"  
#include "generated/services.h"  
#include "generated/types.h"
```

## 8.12 QEverCloudOAuth.h File Reference

```
#include "QEverCloud.h"  
#include "oauth.h"
```

## 8.13 qt4helpers.h File Reference

```
#include <QtGlobal>
```



## Macros

- `#define QEC_SIGNAL(className, methodName, ...) &className::methodName`
- `#define QEC_SLOT(className, methodName, ...) &className::methodName`

### 8.13.1 Macro Definition Documentation

#### 8.13.1.1 QEC\_SIGNAL

```
#define QEC_SIGNAL(  
    className,  
    methodName,  
    ... ) &className::methodName
```

Original work: Copyright (c) 2014 Sergey Skoblikov Modified work: Copyright (c) 2015-2016 Dmitry Ivanov

This file is a part of QEverCloud project and is distributed under the terms of MIT license: <https://opensource.org/licenses/MIT>

This header provides the "backports" of several Qt5 macros into Qt4 so that one can use Qt5 candies with Qt4 as well

#### 8.13.1.2 QEC\_SLOT

```
#define QEC_SLOT(  
    className,  
    methodName,  
    ... ) &className::methodName
```

## 8.14 README.md File Reference

## 8.15 services.h File Reference

```
#include "../Optional.h"  
#include "../export.h"  
#include "../AsyncResult.h"  
#include "constants.h"  
#include "types.h"  
#include <QMap>  
#include <QList>  
#include <QSet>  
#include <QString>  
#include <QStringList>  
#include <QByteArray>  
#include <QDateTime>  
#include <QMetaType>  
#include <QObject>
```

## Classes

- class [qevercloud::NoteStore](#)
- class [qevercloud::UserStore](#)

## Namespaces

- [qevercloud](#)

## 8.16 thumbnail.h File Reference

```
#include "export.h"
#include "AsyncResult.h"
#include "generated/types.h"
#include <QByteArray>
#include <QString>
#include <QNetworkAccessManager>
```

## Classes

- class [qevercloud::Thumbnail](#)  
*The class is for downloading thumbnails for notes and resources from Evernote servers.*
- struct [qevercloud::Thumbnail::ImageType](#)

## Namespaces

- [qevercloud](#)

## 8.17 types.h File Reference

```
#include "../Optional.h"
#include "../export.h"
#include "EDAMErrorCode.h"
#include <QMap>
#include <QList>
#include <QSet>
#include <QString>
#include <QStringList>
#include <QByteArray>
#include <QDateTime>
#include <QMetaType>
#include <QSharedPointer>
```

## Classes

- struct [qevercloud::PrivilegeLevel](#)
- struct [qevercloud::QueryFormat](#)
- struct [qevercloud::NoteSortOrder](#)
- struct [qevercloud::PremiumOrderStatus](#)
- struct [qevercloud::SharedNotebookPrivilegeLevel](#)
- struct [qevercloud::SponsoredGroupRole](#)
- struct [qevercloud::BusinessUserRole](#)
- struct [qevercloud::SharedNotebookInstanceRestrictions](#)
- struct [qevercloud::ReminderEmailConfig](#)
- struct [qevercloud::SyncState](#)
- struct [qevercloud::SyncChunkFilter](#)
- struct [qevercloud::NoteFilter](#)
- struct [qevercloud::NotesMetadataResultSpec](#)
- struct [qevercloud::NoteCollectionCounts](#)
- struct [qevercloud::NoteVersionId](#)
- struct [qevercloud::ClientUsageMetrics](#)
- struct [qevercloud::RelatedQuery](#)
- struct [qevercloud::RelatedResultSpec](#)
- struct [qevercloud::Data](#)
- struct [qevercloud::UserAttributes](#)
- struct [qevercloud::Accounting](#)
- struct [qevercloud::BusinessUserInfo](#)
- struct [qevercloud::PremiumInfo](#)
- struct [qevercloud::User](#)
- struct [qevercloud::Tag](#)
- struct [qevercloud::LazyMap](#)
- struct [qevercloud::ResourceAttributes](#)
- struct [qevercloud::Resource](#)
- struct [qevercloud::NoteAttributes](#)
- struct [qevercloud::Note](#)
- struct [qevercloud::Publishing](#)
- struct [qevercloud::BusinessNotebook](#)
- struct [qevercloud::SavedSearchScope](#)
- struct [qevercloud::SavedSearch](#)
- struct [qevercloud::SharedNotebookRecipientSettings](#)
- struct [qevercloud::SharedNotebook](#)
- struct [qevercloud::NotebookRestrictions](#)
- struct [qevercloud::Notebook](#)
- struct [qevercloud::LinkedNotebook](#)
- struct [qevercloud::NotebookDescriptor](#)
- struct [qevercloud::PublicUserInfo](#)
- struct [qevercloud::AuthenticationResult](#)
- struct [qevercloud::BootstrapSettings](#)
- struct [qevercloud::BootstrapProfile](#)
- struct [qevercloud::BootstrapInfo](#)
- struct [qevercloud::SyncChunk](#)
- struct [qevercloud::NoteList](#)
- struct [qevercloud::NoteMetadata](#)
- struct [qevercloud::NotesMetadataList](#)
- struct [qevercloud::NoteEmailParameters](#)
- struct [qevercloud::RelatedResult](#)
- class [qevercloud::EDAMUserException](#)
- class [qevercloud::EDAMSystemException](#)
- class [qevercloud::EDAMNotFoundException](#)

## Namespaces

- [qevercloud](#)

## Typedefs

- typedef qint32 [qevercloud::UserID](#)
- typedef QString [qevercloud::Guid](#)
- typedef qint64 [qevercloud::Timestamp](#)

# Index

- ~AsyncResult
  - qevercloud::AsyncResult, [47](#)
- ~EDAMNotFoundException
  - qevercloud::EDAMNotFoundException, [64](#)
- ~EDAMSystemException
  - qevercloud::EDAMSystemException, [68](#)
- ~EDAMUserException
  - qevercloud::EDAMUserException, [76](#)
- ~EventLoopFinisher
  - qevercloud::EventLoopFinisher, [79](#)
- ~EverCloudException
  - qevercloud::EverCloudException, [80](#)
- ~EvernoteOAuthDialog
  - qevercloud::EvernoteOAuthDialog, [87](#)
- ~Thumbnail
  - qevercloud::Thumbnail, [246](#)
- accountEmailDomain
  - qevercloud::BootstrapSettings, [53](#)
- accounting
  - qevercloud::User, [250](#)
- ACTIVE
  - qevercloud::PremiumOrderStatus, [206](#)
- active
  - qevercloud::Note, [97](#)
  - qevercloud::Resource, [217](#)
  - qevercloud::User, [250](#)
- ADMIN
  - qevercloud::BusinessUserRole, [59](#)
  - qevercloud::PrivilegeLevel, [206](#)
- allowPreview
  - qevercloud::SharedNotebook, [226](#)
- alternateData
  - qevercloud::Resource, [217](#)
- altitude
  - qevercloud::NoteAttributes, [101](#)
  - qevercloud::ResourceAttributes, [220](#)
- applicationData
  - qevercloud::NoteAttributes, [101](#)
  - qevercloud::ResourceAttributes, [220](#)
- ascending
  - qevercloud::NoteFilter, [118](#)
  - qevercloud::Publishing, [209](#)
- AsyncResult
  - qevercloud::AsyncResult, [46](#), [47](#)
- AsyncResult.h, [271](#)
- attachment
  - qevercloud::ResourceAttributes, [220](#)
- attributes
  - qevercloud::Note, [97](#)
  - qevercloud::NoteMetadata, [122](#)
  - qevercloud::Resource, [217](#)
  - qevercloud::User, [251](#)
- AUTH\_EXPIRED
  - qevercloud::EDAMErrorCode, [63](#)
- authenticate
  - qevercloud::EvernoteOAuthWebView, [90](#)
  - qevercloud::UserStore, [260](#)
- authenticateAsync
  - qevercloud::UserStore, [261](#)
- authenticateLongSession
  - qevercloud::UserStore, [262](#)
- authenticateLongSessionAsync
  - qevercloud::UserStore, [263](#)
- authenticateToBusiness
  - qevercloud::UserStore, [263](#)
- authenticateToBusinessAsync
  - qevercloud::UserStore, [264](#)
- authenticateToSharedNote
  - qevercloud::NoteStore, [134](#)
- authenticateToSharedNoteAsync
  - qevercloud::NoteStore, [135](#)
- authenticateToSharedNotebook
  - qevercloud::NoteStore, [135](#)
- authenticateToSharedNotebookAsync
  - qevercloud::NoteStore, [136](#)
- authenticationFailed
  - qevercloud::EvernoteOAuthWebView, [90](#)
- authenticationFinished
  - qevercloud::EvernoteOAuthWebView, [90](#)
- authenticationSucceeded
  - qevercloud::EvernoteOAuthWebView, [90](#)
- authenticationToken
  - qevercloud::AuthenticationResult, [49](#)
  - qevercloud::EvernoteOAuthWebView::OAuthResult, [194](#)
  - qevercloud::NoteStore, [136](#)
  - qevercloud::UserStore, [264](#)
- author
  - qevercloud::NoteAttributes, [101](#)
- BAD\_DATA\_FORMAT
  - qevercloud::EDAMErrorCode, [63](#)
- BAD\_SEQUENCE\_ID
  - qevercloud::ThriftException::Type, [249](#)
- BMP
  - qevercloud::Thumbnail::ImageType, [92](#)
- body
  - qevercloud::Data, [61](#)
- bodyHash

- qevercloud::Data, 61
- BUSINESS\_FULL\_ACCESS
  - qevercloud::SharedNotebookPrivilegeLevel, 230
- businessAddress
  - qevercloud::UserAttributes, 254
- businessId
  - qevercloud::Accounting, 42
  - qevercloud::BusinessUserInfo, 57
  - qevercloud::LinkedNotebook, 94
- businessName
  - qevercloud::Accounting, 42
  - qevercloud::BusinessUserInfo, 58
- businessNotebook
  - qevercloud::Notebook, 106
- businessRole
  - qevercloud::Accounting, 42
- businessUserInfo
  - qevercloud::User, 251
- cameraMake
  - qevercloud::ResourceAttributes, 220
- cameraModel
  - qevercloud::ResourceAttributes, 221
- CANCELED
  - qevercloud::PremiumOrderStatus, 206
- CANCELLATION\_PENDING
  - qevercloud::PremiumOrderStatus, 206
- canPurchaseUploadAllowance
  - qevercloud::PremiumInfo, 202
- ccAddresses
  - qevercloud::NoteEmailParameters, 116
- checkVersion
  - qevercloud::UserStore, 264
- checkVersionAsync
  - qevercloud::UserStore, 265
- chunkHighUSN
  - qevercloud::SyncChunk, 233
- CLASSIFICATION\_RECIPE\_SERVICE\_RECIPE
  - qevercloud, 19
- CLASSIFICATION\_RECIPE\_USER\_NON\_RECIPE
  - qevercloud, 20
- CLASSIFICATION\_RECIPE\_USER\_RECIPE
  - qevercloud, 20
- classifications
  - qevercloud::NoteAttributes, 102
- clear
  - qevercloud::Optional< T >, 197
- clientWillIndex
  - qevercloud::ResourceAttributes, 221
- clipFullPage
  - qevercloud::UserAttributes, 254
- comments
  - qevercloud::UserAttributes, 254
- completeTwoFactorAuthentication
  - qevercloud::UserStore, 265
- completeTwoFactorAuthenticationAsync
  - qevercloud::UserStore, 266
- constants.h, 271
- contact
  - qevercloud::Notebook, 106
- contactName
  - qevercloud::NotebookDescriptor, 109
- containingNotebooks
  - qevercloud::RelatedResult, 213
- content
  - qevercloud::Note, 98
- contentClass
  - qevercloud::NoteAttributes, 102
- contentHash
  - qevercloud::Note, 98
- contentLength
  - qevercloud::Note, 98
  - qevercloud::NoteMetadata, 122
- copyNote
  - qevercloud::NoteStore, 136
- copyNoteAsync
  - qevercloud::NoteStore, 137
- CREATED
  - qevercloud::NoteSortOrder, 129
- created
  - qevercloud::Note, 98
  - qevercloud::NoteMetadata, 122
  - qevercloud::User, 251
- createLinkedNotebook
  - qevercloud::NoteStore, 137
- createLinkedNotebookAsync
  - qevercloud::NoteStore, 138
- createNote
  - qevercloud::NoteStore, 138
- createNoteAsync
  - qevercloud::NoteStore, 139
- createNotebook
  - qevercloud::NoteStore, 139
- createNotebookAsync
  - qevercloud::NoteStore, 141
- createPostRequest
  - qevercloud::Thumbnail, 246
- createSearch
  - qevercloud::NoteStore, 141
- createSearchAsync
  - qevercloud::NoteStore, 142
- createSharedNotebook
  - qevercloud::NoteStore, 142
- createSharedNotebookAsync
  - qevercloud::NoteStore, 143
- createTag
  - qevercloud::NoteStore, 143
- createTagAsync
  - qevercloud::NoteStore, 144
- creatorId
  - qevercloud::NoteAttributes, 102
- currency
  - qevercloud::Accounting, 42
- currentTime
  - qevercloud::AuthenticationResult, 49
  - qevercloud::PremiumInfo, 203
  - qevercloud::SyncChunk, 233

- qevercloud::SyncState, [239](#)
- dailyEmailLimit
  - qevercloud::UserAttributes, [254](#)
- data
  - qevercloud::Resource, [218](#)
- DATA\_CONFLICT
  - qevercloud::EDAMErrorCode, [63](#)
- DATA\_REQUIRED
  - qevercloud::EDAMErrorCode, [63](#)
- dateAgreedToTermsOfService
  - qevercloud::UserAttributes, [255](#)
- defaultLatitude
  - qevercloud::UserAttributes, [255](#)
- defaultLocationName
  - qevercloud::UserAttributes, [255](#)
- defaultLongitude
  - qevercloud::UserAttributes, [255](#)
- defaultNotebook
  - qevercloud::Notebook, [106](#)
- deleted
  - qevercloud::Note, [98](#)
  - qevercloud::NoteMetadata, [122](#)
  - qevercloud::User, [251](#)
- deleteNote
  - qevercloud::NoteStore, [144](#)
- deleteNoteAsync
  - qevercloud::NoteStore, [145](#)
- DO\_NOT\_SEND
  - qevercloud::ReminderEmailConfig, [216](#)
- download
  - qevercloud::Thumbnail, [247](#)
- downloadAsync
  - qevercloud::Thumbnail, [247](#)
- duration
  - qevercloud::Resource, [218](#)
- EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX
  - qevercloud, [20](#)
- EDAM\_APPLICATIONDATA\_NAME\_LEN\_MAX
  - qevercloud, [20](#)
- EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN
  - qevercloud, [20](#)
- EDAM\_APPLICATIONDATA\_NAME\_REGEX
  - qevercloud, [20](#)
- EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MAX
  - qevercloud, [20](#)
- EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MIN
  - qevercloud, [21](#)
- EDAM\_APPLICATIONDATA\_VALUE\_REGEX
  - qevercloud, [21](#)
- EDAM\_ATTRIBUTE\_LEN\_MAX
  - qevercloud, [21](#)
- EDAM\_ATTRIBUTE\_LEN\_MIN
  - qevercloud, [21](#)
- EDAM\_ATTRIBUTE\_LIST\_MAX
  - qevercloud, [21](#)
- EDAM\_ATTRIBUTE\_MAP\_MAX
  - qevercloud, [21](#)
- EDAM\_ATTRIBUTE\_REGEX
  - qevercloud, [21](#)
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MAX
  - qevercloud, [22](#)
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MIN
  - qevercloud, [22](#)
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_REGEX
  - qevercloud, [22](#)
- EDAM\_BUSINESS\_NOTEBOOKS\_MAX
  - qevercloud, [22](#)
- EDAM\_BUSINESS\_NOTES\_MAX
  - qevercloud, [22](#)
- EDAM\_BUSINESS\_PHONE\_NUMBER\_LEN\_MAX
  - qevercloud, [22](#)
- EDAM\_BUSINESS\_TAGS\_MAX
  - qevercloud, [22](#)
- EDAM\_BUSINESS\_URI\_LEN\_MAX
  - qevercloud, [23](#)
- EDAM\_CONTENT\_CLASS\_FOOD\_MEAL
  - qevercloud, [23](#)
- EDAM\_CONTENT\_CLASS\_HELLO\_ENCOUNTER
  - qevercloud, [23](#)
- EDAM\_CONTENT\_CLASS\_HELLO\_PROFILE
  - qevercloud, [23](#)
- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_NOTEBOOK
  - qevercloud, [23](#)
- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_PREFIX
  - qevercloud, [23](#)
- EDAM\_CONTENT\_CLASS\_SKITCH
  - qevercloud, [23](#)
- EDAM\_CONTENT\_CLASS\_SKITCH\_PDF
  - qevercloud, [24](#)
- EDAM\_CONTENT\_CLASS\_SKITCH\_PREFIX
  - qevercloud, [24](#)
- EDAM\_DEVICE\_DESCRIPTION\_LEN\_MAX
  - qevercloud, [24](#)
- EDAM\_DEVICE\_DESCRIPTION\_REGEX
  - qevercloud, [24](#)
- EDAM\_DEVICE\_ID\_LEN\_MAX
  - qevercloud, [24](#)
- EDAM\_DEVICE\_ID\_REGEX
  - qevercloud, [24](#)
- EDAM\_EMAIL\_DOMAIN\_REGEX
  - qevercloud, [24](#)
- EDAM\_EMAIL\_LEN\_MAX
  - qevercloud, [25](#)
- EDAM\_EMAIL\_LEN\_MIN
  - qevercloud, [25](#)
- EDAM\_EMAIL\_LOCAL\_REGEX
  - qevercloud, [25](#)
- EDAM\_EMAIL\_REGEX
  - qevercloud, [25](#)
- EDAM\_FOOD\_APP\_CONTENT\_CLASS\_PREFIX
  - qevercloud, [25](#)
- EDAM\_GUID\_LEN\_MAX
  - qevercloud, [25](#)
- EDAM\_GUID\_LEN\_MIN
  - qevercloud, [25](#)

EDAM\_GUID\_REGEX  
qevercloud, 26

EDAM\_HASH\_LEN  
qevercloud, 26

EDAM\_HELLO\_APP\_CONTENT\_CLASS\_PREFIX  
qevercloud, 26

EDAM\_INDEXABLE\_RESOURCE\_MIME\_TYPES  
qevercloud, 26

EDAM\_MAX\_PREFERENCES  
qevercloud, 26

EDAM\_MAX\_VALUES\_PER\_PREFERENCE  
qevercloud, 26

EDAM\_MIME\_LEN\_MAX  
qevercloud, 26

EDAM\_MIME\_LEN\_MIN  
qevercloud, 27

EDAM\_MIME\_REGEX  
qevercloud, 27

EDAM\_MIME\_TYPE\_AAC  
qevercloud, 27

EDAM\_MIME\_TYPE\_AMR  
qevercloud, 27

EDAM\_MIME\_TYPE\_DEFAULT  
qevercloud, 27

EDAM\_MIME\_TYPE\_GIF  
qevercloud, 27

EDAM\_MIME\_TYPE\_INK  
qevercloud, 27

EDAM\_MIME\_TYPE\_JPEG  
qevercloud, 27

EDAM\_MIME\_TYPE\_M4A  
qevercloud, 28

EDAM\_MIME\_TYPE\_MP3  
qevercloud, 28

EDAM\_MIME\_TYPE\_MP4\_VIDEO  
qevercloud, 28

EDAM\_MIME\_TYPE\_PDF  
qevercloud, 28

EDAM\_MIME\_TYPE\_PNG  
qevercloud, 28

EDAM\_MIME\_TYPE\_WAV  
qevercloud, 28

EDAM\_MIME\_TYPES  
qevercloud, 28

EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MAX  
qevercloud, 28

EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MIN  
qevercloud, 29

EDAM\_NOTE\_CONTENT\_CLASS\_REGEX  
qevercloud, 29

EDAM\_NOTE\_CONTENT\_LEN\_MAX  
qevercloud, 29

EDAM\_NOTE\_CONTENT\_LEN\_MIN  
qevercloud, 29

EDAM\_NOTE\_RESOURCES\_MAX  
qevercloud, 29

EDAM\_NOTE\_SIZE\_MAX\_FREE  
qevercloud, 29

EDAM\_NOTE\_SIZE\_MAX\_PREMIUM  
qevercloud, 29

EDAM\_NOTE\_SOURCE\_MAIL\_CLIP  
qevercloud, 30

EDAM\_NOTE\_SOURCE\_MAIL\_SMTP\_GATEWAY  
qevercloud, 30

EDAM\_NOTE\_SOURCE\_WEB\_CLIP  
qevercloud, 30

EDAM\_NOTE\_TAGS\_MAX  
qevercloud, 30

EDAM\_NOTE\_TITLE\_LEN\_MAX  
qevercloud, 30

EDAM\_NOTE\_TITLE\_LEN\_MIN  
qevercloud, 30

EDAM\_NOTE\_TITLE\_REGEX  
qevercloud, 30

EDAM\_NOTEBOOK\_NAME\_LEN\_MAX  
qevercloud, 31

EDAM\_NOTEBOOK\_NAME\_LEN\_MIN  
qevercloud, 31

EDAM\_NOTEBOOK\_NAME\_REGEX  
qevercloud, 31

EDAM\_NOTEBOOK\_SHARED\_NOTEBOOK\_MAX  
qevercloud, 31

EDAM\_NOTEBOOK\_STACK\_LEN\_MAX  
qevercloud, 31

EDAM\_NOTEBOOK\_STACK\_LEN\_MIN  
qevercloud, 31

EDAM\_NOTEBOOK\_STACK\_REGEX  
qevercloud, 31

EDAM\_PREFERENCE\_NAME\_LEN\_MAX  
qevercloud, 32

EDAM\_PREFERENCE\_NAME\_LEN\_MIN  
qevercloud, 32

EDAM\_PREFERENCE\_NAME\_REGEX  
qevercloud, 32

EDAM\_PREFERENCE\_SHORTCUTS  
qevercloud, 32

EDAM\_PREFERENCE\_SHORTCUTS\_MAX\_VALUES  
qevercloud, 32

EDAM\_PREFERENCE\_VALUE\_LEN\_MAX  
qevercloud, 32

EDAM\_PREFERENCE\_VALUE\_LEN\_MIN  
qevercloud, 32

EDAM\_PREFERENCE\_VALUE\_REGEX  
qevercloud, 33

EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MAX  
qevercloud, 33

EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MIN  
qevercloud, 33

EDAM\_PUBLISHING\_DESCRIPTION\_REGEX  
qevercloud, 33

EDAM\_PUBLISHING\_URI\_LEN\_MAX  
qevercloud, 33

EDAM\_PUBLISHING\_URI\_LEN\_MIN  
qevercloud, 33

EDAM\_PUBLISHING\_URI\_PROHIBITED  
qevercloud, 33



- EDAM\_PUBLISHING\_URI\_REGEX
  - qevercloud, [34](#)
- EDAM\_RELATED\_MAX\_NOTEBOOKS
  - qevercloud, [34](#)
- EDAM\_RELATED\_MAX\_NOTES
  - qevercloud, [34](#)
- EDAM\_RELATED\_MAX\_TAGS
  - qevercloud, [34](#)
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MAX
  - qevercloud, [34](#)
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MIN
  - qevercloud, [34](#)
- EDAM\_RESOURCE\_SIZE\_MAX\_FREE
  - qevercloud, [34](#)
- EDAM\_RESOURCE\_SIZE\_MAX\_PREMIUM
  - qevercloud, [34](#)
- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MAX
  - qevercloud, [35](#)
- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MIN
  - qevercloud, [35](#)
- EDAM\_SAVED\_SEARCH\_NAME\_REGEX
  - qevercloud, [35](#)
- EDAM\_SEARCH\_QUERY\_LEN\_MAX
  - qevercloud, [35](#)
- EDAM\_SEARCH\_QUERY\_LEN\_MIN
  - qevercloud, [35](#)
- EDAM\_SEARCH\_QUERY\_REGEX
  - qevercloud, [35](#)
- EDAM\_SEARCH\_SUGGESTIONS\_MAX
  - qevercloud, [35](#)
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MAX
  - qevercloud, [36](#)
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MIN
  - qevercloud, [36](#)
- EDAM\_TAG\_NAME\_LEN\_MAX
  - qevercloud, [36](#)
- EDAM\_TAG\_NAME\_LEN\_MIN
  - qevercloud, [36](#)
- EDAM\_TAG\_NAME\_REGEX
  - qevercloud, [36](#)
- EDAM\_TIMEZONE\_LEN\_MAX
  - qevercloud, [36](#)
- EDAM\_TIMEZONE\_LEN\_MIN
  - qevercloud, [36](#)
- EDAM\_TIMEZONE\_REGEX
  - qevercloud, [37](#)
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX
  - qevercloud, [37](#)
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX\_PREMIUM
  - qevercloud, [37](#)
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_FREE
  - qevercloud, [37](#)
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_PREMIUM
  - qevercloud, [37](#)
- EDAM\_USER\_NAME\_LEN\_MAX
  - qevercloud, [37](#)
- EDAM\_USER\_NAME\_LEN\_MIN
  - qevercloud, [38](#)
- EDAM\_USER\_NAME\_REGEX
  - qevercloud, [38](#)
- EDAM\_USER\_NOTEBOOKS\_MAX
  - qevercloud, [38](#)
- EDAM\_USER\_NOTES\_MAX
  - qevercloud, [38](#)
- EDAM\_USER\_PASSWORD\_LEN\_MAX
  - qevercloud, [38](#)
- EDAM\_USER\_PASSWORD\_LEN\_MIN
  - qevercloud, [38](#)
- EDAM\_USER\_PASSWORD\_REGEX
  - qevercloud, [38](#)
- EDAM\_USER\_RECENT\_MAILED\_ADDRESSES\_MAX
  - qevercloud, [38](#)
- EDAM\_USER\_SAVED\_SEARCHES\_MAX
  - qevercloud, [39](#)
- EDAM\_USER\_TAGS\_MAX
  - qevercloud, [39](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS
  - qevercloud, [39](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_FREE
  - qevercloud, [39](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_PREMIUM
  - qevercloud, [39](#)
- EDAM\_USER\_USERNAME\_LEN\_MAX
  - qevercloud, [39](#)
- EDAM\_USER\_USERNAME\_LEN\_MIN
  - qevercloud, [39](#)
- EDAM\_USER\_USERNAME\_REGEX
  - qevercloud, [40](#)
- EDAM\_VAT\_REGEX
  - qevercloud, [40](#)
- EDAM\_VERSION\_MAJOR
  - qevercloud, [40](#)
- EDAM\_VERSION\_MINOR
  - qevercloud, [40](#)
- EDAMErrorCode.h, [274](#)
- EDAMNotFoundException
  - qevercloud::EDAMNotFoundException, [64](#)
- EDAMNotFoundExceptionData
  - qevercloud::EDAMNotFoundExceptionData, [66](#)
- EDAMSystemException
  - qevercloud::EDAMSystemException, [67](#)
- EDAMSystemExceptionAuthExpiredData
  - qevercloud::EDAMSystemExceptionAuthExpiredData, [70](#)
- EDAMSystemExceptionData
  - qevercloud::EDAMSystemExceptionData, [71](#)
- EDAMSystemExceptionRateLimitReachedData
  - qevercloud::EDAMSystemExceptionRateLimitReachedData, [74](#)
- EDAMUserException
  - qevercloud::EDAMUserException, [76](#)
- EDAMUserExceptionData
  - qevercloud::EDAMUserExceptionData, [77](#)
- educationalDiscount
  - qevercloud::UserAttributes, [255](#)
- email

- qevercloud::BusinessUserInfo, 58
- qevercloud::SharedNotebook, 226
- qevercloud::User, 251
- emailNote
  - qevercloud::NoteStore, 145
- emailNoteAsync
  - qevercloud::NoteStore, 146
- emailOptOutDate
  - qevercloud::UserAttributes, 255
- emphasized
  - qevercloud::NoteFilter, 118
- enableFacebookSharing
  - qevercloud::BootstrapSettings, 53
- enableGiftSubscriptions
  - qevercloud::BootstrapSettings, 54
- enableLinkedInSharing
  - qevercloud::BootstrapSettings, 54
- enablePublicNotebooks
  - qevercloud::BootstrapSettings, 54
- enableSharedNotebooks
  - qevercloud::BootstrapSettings, 54
- enableSingleNoteSharing
  - qevercloud::BootstrapSettings, 54
- enableSponsoredAccounts
  - qevercloud::BootstrapSettings, 54
- enableSupportTickets
  - qevercloud::BootstrapSettings, 54
- enableTwitterSharing
  - qevercloud::BootstrapSettings, 54
- ENML\_VALIDATION
  - qevercloud::EDAMErrorCode, 63
- errorCode
  - qevercloud::EDAMSystemException, 68
  - qevercloud::EDAMUserException, 76
- errorMessage
  - qevercloud::EverCloudExceptionData, 83
- EventLoopFinisher
  - qevercloud::EventLoopFinisher, 79
- EventLoopFinisher.h, 275
- EverCloudException
  - qevercloud::EverCloudException, 80
- EverCloudException.h, 275
- EverCloudExceptionData
  - qevercloud, 40
  - qevercloud::EverCloudExceptionData, 82
- EvernoteException
  - qevercloud::EvernoteException, 84
- EvernoteExceptionData
  - qevercloud::EvernoteExceptionData, 85
- evernoteNetworkAccessManager
  - qevercloud, 19
- EvernoteOAuthDialog
  - qevercloud::EvernoteOAuthDialog, 87
- EvernoteOAuthWebView
  - qevercloud::EvernoteOAuthWebView, 89
- exceptionData
  - qevercloud::EDAMNotFoundException, 64
  - qevercloud::EDAMSystemException, 68
  - qevercloud::EDAMSystemExceptionAuthExpired, 69
  - qevercloud::EDAMSystemExceptionRateLimitReached, 73
  - qevercloud::EDAMUserException, 76
  - qevercloud::EverCloudException, 81
  - qevercloud::EvernoteException, 84
  - qevercloud::ThriftException, 242
- exceptions.h, 276
- exec
  - qevercloud::EvernoteOAuthDialog, 87
- expiration
  - qevercloud::AuthenticationResult, 49
- expires
  - qevercloud::EvernoteOAuthWebView::OAuthResult, 194
- export.h, 276
  - QEVERCLOUD\_EXPORT, 276
- expungedLinkedNotebooks
  - qevercloud::SyncChunk, 233
- expungedNotebooks
  - qevercloud::SyncChunk, 233
- expungedNotes
  - qevercloud::SyncChunk, 233
- expungedSearches
  - qevercloud::SyncChunk, 233
- expungedTags
  - qevercloud::SyncChunk, 234
- expungeInactiveNotes
  - qevercloud::NoteStore, 146
- expungeInactiveNotesAsync
  - qevercloud::NoteStore, 146
- expungeLinkedNotebook
  - qevercloud::NoteStore, 147
- expungeLinkedNotebookAsync
  - qevercloud::NoteStore, 147
- expungeNote
  - qevercloud::NoteStore, 147
- expungeNoteAsync
  - qevercloud::NoteStore, 148
- expungeNotebook
  - qevercloud::NoteStore, 148
- expungeNotebookAsync
  - qevercloud::NoteStore, 149
- expungeNotes
  - qevercloud::NoteStore, 149
- expungeNotesAsync
  - qevercloud::NoteStore, 150
- expungeSearch
  - qevercloud::NoteStore, 150
- expungeSearchAsync
  - qevercloud::NoteStore, 150
- expungeSharedNotebooks
  - qevercloud::NoteStore, 150
- expungeSharedNotebooksAsync
  - qevercloud::NoteStore, 151
- expungeTag
  - qevercloud::NoteStore, 151

- expungeTagAsync
  - qevercloud::NoteStore, [152](#)
- expungeWhichSharedNotebookRestrictions
  - qevercloud::NotebookRestrictions, [111](#)
- FAILED
  - qevercloud::PremiumOrderStatus, [206](#)
- fileName
  - qevercloud::ResourceAttributes, [221](#)
- filter
  - qevercloud::RelatedQuery, [212](#)
- findNoteCounts
  - qevercloud::NoteStore, [152](#)
- findNoteCountsAsync
  - qevercloud::NoteStore, [153](#)
- findNoteOffset
  - qevercloud::NoteStore, [153](#)
- findNoteOffsetAsync
  - qevercloud::NoteStore, [154](#)
- findNotes
  - qevercloud::NoteStore, [154](#)
- findNotesAsync
  - qevercloud::NoteStore, [154](#)
- findNotesMetadata
  - qevercloud::NoteStore, [154](#)
- findNotesMetadataAsync
  - qevercloud::NoteStore, [155](#)
- findRelated
  - qevercloud::NoteStore, [156](#)
- findRelatedAsync
  - qevercloud::NoteStore, [157](#)
- finished
  - qevercloud::AsyncResult, [47](#)
- format
  - qevercloud::SavedSearch, [223](#)
- FULL\_ACCESS
  - qevercloud::SharedNotebookPrivilegeLevel, [230](#)
- fullMap
  - qevercloud::LazyMap, [93](#)
- fullSyncBefore
  - qevercloud::SyncState, [239](#)
- getBootstrapInfo
  - qevercloud::UserStore, [266](#)
- getBootstrapInfoAsync
  - qevercloud::UserStore, [266](#)
- getDefaultNotebook
  - qevercloud::NoteStore, [157](#)
- getDefaultNotebookAsync
  - qevercloud::NoteStore, [157](#)
- getFilteredSyncChunk
  - qevercloud::NoteStore, [157](#)
- getFilteredSyncChunkAsync
  - qevercloud::NoteStore, [158](#)
- getLinkedNotebookSyncChunk
  - qevercloud::NoteStore, [158](#)
- getLinkedNotebookSyncChunkAsync
  - qevercloud::NoteStore, [159](#)
- getLinkedNotebookSyncState
  - qevercloud::NoteStore, [159](#)
- getLinkedNotebookSyncStateAsync
  - qevercloud::NoteStore, [159](#)
- getNote
  - qevercloud::NoteStore, [160](#)
- getNoteApplicationData
  - qevercloud::NoteStore, [160](#)
- getNoteApplicationDataAsync
  - qevercloud::NoteStore, [161](#)
- getNoteApplicationDataEntry
  - qevercloud::NoteStore, [161](#)
- getNoteApplicationDataEntryAsync
  - qevercloud::NoteStore, [161](#)
- getNoteAsync
  - qevercloud::NoteStore, [161](#)
- getNotebook
  - qevercloud::NoteStore, [161](#)
- getNotebookAsync
  - qevercloud::NoteStore, [162](#)
- getNoteContent
  - qevercloud::NoteStore, [162](#)
- getNoteContentAsync
  - qevercloud::NoteStore, [163](#)
- getNoteSearchText
  - qevercloud::NoteStore, [163](#)
- getNoteSearchTextAsync
  - qevercloud::NoteStore, [164](#)
- getNoteStoreUrl
  - qevercloud::UserStore, [267](#)
- getNoteStoreUrlAsync
  - qevercloud::UserStore, [267](#)
- getNoteTagNames
  - qevercloud::NoteStore, [164](#)
- getNoteTagNamesAsync
  - qevercloud::NoteStore, [164](#)
- getNoteVersion
  - qevercloud::NoteStore, [164](#)
- getNoteVersionAsync
  - qevercloud::NoteStore, [165](#)
- getPremiumInfo
  - qevercloud::UserStore, [267](#)
- getPremiumInfoAsync
  - qevercloud::UserStore, [267](#)
- getPublicNotebook
  - qevercloud::NoteStore, [165](#)
- getPublicNotebookAsync
  - qevercloud::NoteStore, [166](#)
- getPublicUserInfo
  - qevercloud::UserStore, [267](#)
- getPublicUserInfoAsync
  - qevercloud::UserStore, [268](#)
- getResource
  - qevercloud::NoteStore, [166](#)
- getResourceAlternateData
  - qevercloud::NoteStore, [167](#)
- getResourceAlternateDataAsync
  - qevercloud::NoteStore, [167](#)
- getResourceApplicationData

- qevercloud::NoteStore, 168
- getResourceApplicationDataAsync
  - qevercloud::NoteStore, 168
- getResourceApplicationDataEntry
  - qevercloud::NoteStore, 168
- getResourceApplicationDataEntryAsync
  - qevercloud::NoteStore, 168
- getResourceAsync
  - qevercloud::NoteStore, 169
- getResourceAttributes
  - qevercloud::NoteStore, 169
- getResourceAttributesAsync
  - qevercloud::NoteStore, 169
- getResourceByHash
  - qevercloud::NoteStore, 169
- getResourceByHashAsync
  - qevercloud::NoteStore, 170
- getResourceData
  - qevercloud::NoteStore, 170
- getResourceDataAsync
  - qevercloud::NoteStore, 171
- getResourceRecognition
  - qevercloud::NoteStore, 171
- getResourceRecognitionAsync
  - qevercloud::NoteStore, 172
- getResourceSearchText
  - qevercloud::NoteStore, 172
- getResourceSearchTextAsync
  - qevercloud::NoteStore, 173
- getSearch
  - qevercloud::NoteStore, 173
- getSearchAsync
  - qevercloud::NoteStore, 173
- getSharedNotebookByAuth
  - qevercloud::NoteStore, 173
- getSharedNotebookByAuthAsync
  - qevercloud::NoteStore, 174
- getSyncChunk
  - qevercloud::NoteStore, 174
- getSyncChunkAsync
  - qevercloud::NoteStore, 174
- getSyncState
  - qevercloud::NoteStore, 174
- getSyncStateAsync
  - qevercloud::NoteStore, 174
- getSyncStateWithMetrics
  - qevercloud::NoteStore, 175
- getSyncStateWithMetricsAsync
  - qevercloud::NoteStore, 175
- getTag
  - qevercloud::NoteStore, 175
- getTagAsync
  - qevercloud::NoteStore, 176
- getUser
  - qevercloud::UserStore, 268
- getUserAsync
  - qevercloud::UserStore, 268
- GIF
  - qevercloud::Thumbnail::ImageType, 92
- globals.h, 277
- GROUP
  - qevercloud::SharedNotebookPrivilegeLevel, 230
- GROUP\_ADMIN
  - qevercloud::SponsoredGroupRole, 232
- GROUP\_MEMBER
  - qevercloud::SponsoredGroupRole, 232
- GROUP\_OWNER
  - qevercloud::SponsoredGroupRole, 232
- groupName
  - qevercloud::UserAttributes, 255
- Guid
  - qevercloud, 18
- guid
  - qevercloud::LinkedNotebook, 95
  - qevercloud::Note, 98
  - qevercloud::Notebook, 106
  - qevercloud::NotebookDescriptor, 109
  - qevercloud::NoteEmailParameters, 116
  - qevercloud::NoteMetadata, 123
  - qevercloud::Resource, 218
  - qevercloud::SavedSearch, 223
  - qevercloud::Tag, 240
- hasSharedNotebook
  - qevercloud::NotebookDescriptor, 109
- height
  - qevercloud::Resource, 218
- hideSponsorBilling
  - qevercloud::UserAttributes, 256
- id
  - qevercloud::SharedNotebook, 226
  - qevercloud::User, 251
- identifier
  - qevercloud::EDAMNotFoundException, 65
- inactive
  - qevercloud::NoteFilter, 118
- includeAccount
  - qevercloud::SavedSearchScope, 225
- includeAttributes
  - qevercloud::NotesMetadataResultSpec, 127
- includeBusinessLinkedNotebooks
  - qevercloud::SavedSearchScope, 225
- includeContainingNotebooks
  - qevercloud::RelatedResultSpec, 215
- includeContentLength
  - qevercloud::NotesMetadataResultSpec, 127
- includeCreated
  - qevercloud::NotesMetadataResultSpec, 127
- includeDeleted
  - qevercloud::NotesMetadataResultSpec, 127
- includeExpunged
  - qevercloud::SyncChunkFilter, 236
- includeLargestResourceMime
  - qevercloud::NotesMetadataResultSpec, 127
- includeLargestResourceSize
  - qevercloud::NotesMetadataResultSpec, 127

- includeLinkedNotebooks
  - qevercloud::SyncChunkFilter, 236
- includeNoteApplicationDataFullMap
  - qevercloud::SyncChunkFilter, 236
- includeNoteAttributes
  - qevercloud::SyncChunkFilter, 236
- includeNotebookGuid
  - qevercloud::NotesMetadataResultSpec, 128
- includeNotebooks
  - qevercloud::SyncChunkFilter, 236
- includeNoteResourceApplicationDataFullMap
  - qevercloud::SyncChunkFilter, 237
- includeNoteResources
  - qevercloud::SyncChunkFilter, 237
- includeNotes
  - qevercloud::SyncChunkFilter, 237
- includePersonalLinkedNotebooks
  - qevercloud::SavedSearchScope, 225
- includeResourceApplicationDataFullMap
  - qevercloud::SyncChunkFilter, 237
- includeResources
  - qevercloud::SyncChunkFilter, 237
- includeSearches
  - qevercloud::SyncChunkFilter, 237
- includeTagGuids
  - qevercloud::NotesMetadataResultSpec, 128
- includeTags
  - qevercloud::SyncChunkFilter, 237
- includeTitle
  - qevercloud::NotesMetadataResultSpec, 128
- includeUpdated
  - qevercloud::NotesMetadataResultSpec, 128
- includeUpdateSequenceNum
  - qevercloud::NotesMetadataResultSpec, 128
- incomingEmailAddress
  - qevercloud::UserAttributes, 256
- init
  - qevercloud::Optional< T >, 198
- INTERNAL\_ERROR
  - qevercloud::EDAMErrorCode, 63
  - qevercloud::ThriftException::Type, 249
- INVALID\_AUTH
  - qevercloud::EDAMErrorCode, 63
- INVALID\_DATA
  - qevercloud::ThriftException::Type, 249
- INVALID\_MESSAGE\_TYPE
  - qevercloud::ThriftException::Type, 249
- isEqual
  - qevercloud::Optional< T >, 198
- isSet
  - qevercloud::Optional< T >, 198
- isSucceeded
  - qevercloud::EvernoteOAuthDialog, 87
  - qevercloud::EvernoteOAuthWebView, 91
- joinedUserCount
  - qevercloud::NotebookDescriptor, 109
- JPEG
  - qevercloud::Thumbnail::ImageType, 92
- key
  - qevercloud::EDAMNotFoundException, 65
- keysOnly
  - qevercloud::LazyMap, 93
- largestResourceMime
  - qevercloud::NoteMetadata, 123
- largestResourceSize
  - qevercloud::NoteMetadata, 123
- lastEditedBy
  - qevercloud::NoteAttributes, 102
- lastEditorId
  - qevercloud::NoteAttributes, 102
- lastFailedCharge
  - qevercloud::Accounting, 43
- lastFailedChargeReason
  - qevercloud::Accounting, 43
- lastRequestedCharge
  - qevercloud::Accounting, 43
- lastSuccessfulCharge
  - qevercloud::Accounting, 43
- latitude
  - qevercloud::NoteAttributes, 103
  - qevercloud::ResourceAttributes, 221
- LEN\_TOO\_LONG
  - qevercloud::EDAMErrorCode, 63
- LEN\_TOO\_SHORT
  - qevercloud::EDAMErrorCode, 63
- libraryVersion
  - qevercloud, 19
- LIMIT\_REACHED
  - qevercloud::EDAMErrorCode, 63
- linkedNotebooks
  - qevercloud::SyncChunk, 234
- listLinkedNotebooks
  - qevercloud::NoteStore, 176
- listLinkedNotebooksAsync
  - qevercloud::NoteStore, 176
- listNotebooks
  - qevercloud::NoteStore, 176
- listNotebooksAsync
  - qevercloud::NoteStore, 176
- listNoteVersions
  - qevercloud::NoteStore, 176
- listNoteVersionsAsync
  - qevercloud::NoteStore, 178
- listSearches
  - qevercloud::NoteStore, 178
- listSearchesAsync
  - qevercloud::NoteStore, 178
- listSharedNotebooks
  - qevercloud::NoteStore, 178
- listSharedNotebooksAsync
  - qevercloud::NoteStore, 178
- listTags
  - qevercloud::NoteStore, 179
- listTagsAsync
  - qevercloud::NoteStore, 179
- listTagsByNotebook

- qevercloud::NoteStore, 179
- listTagsByNotebookAsync
  - qevercloud::NoteStore, 179
- longitude
  - qevercloud::NoteAttributes, 103
  - qevercloud::ResourceAttributes, 221
- m\_error
  - qevercloud::EverCloudException, 81
- m\_errorCode
  - qevercloud::EDAMSystemExceptionData, 72
  - qevercloud::EDAMUserExceptionData, 78
- m\_identifier
  - qevercloud::EDAMNotFoundExceptionData, 66
- m\_key
  - qevercloud::EDAMNotFoundExceptionData, 66
- m\_message
  - qevercloud::EDAMSystemExceptionData, 72
- m\_parameter
  - qevercloud::EDAMUserExceptionData, 78
- m\_rateLimitDuration
  - qevercloud::EDAMSystemExceptionData, 72
- m\_type
  - qevercloud::ThriftException, 243
  - qevercloud::ThriftExceptionData, 244
- MANAGER
  - qevercloud::PrivilegeLevel, 206
- marketingUrl
  - qevercloud::BootstrapSettings, 55
- maxNotebooks
  - qevercloud::RelatedResultSpec, 215
- maxNotes
  - qevercloud::RelatedResultSpec, 215
- maxReferrals
  - qevercloud::UserAttributes, 256
- maxTags
  - qevercloud::RelatedResultSpec, 215
- message
  - qevercloud::EDAMSystemException, 68
  - qevercloud::NoteEmailParameters, 116
- mime
  - qevercloud::Resource, 218
- MISSING\_RESULT
  - qevercloud::ThriftException::Type, 249
- MODIFY\_NOTEBOOK\_PLUS\_ACTIVITY
  - qevercloud::SharedNotebookPrivilegeLevel, 230
- name
  - qevercloud::BootstrapProfile, 52
  - qevercloud::Notebook, 106
  - qevercloud::SavedSearch, 223
  - qevercloud::Tag, 240
  - qevercloud::User, 251
- nextChargeDate
  - qevercloud::Accounting, 43
- nextPaymentDue
  - qevercloud::Accounting, 43
- NO\_SHARED\_NOTEBOOKS
  - qevercloud::SharedNotebookInstanceRestrictions, 229
- noCreateNotes
  - qevercloud::NotebookRestrictions, 111
- noCreateSharedNotebooks
  - qevercloud::NotebookRestrictions, 111
- noCreateTags
  - qevercloud::NotebookRestrictions, 111
- noEmailNotes
  - qevercloud::NotebookRestrictions, 111
- noExpungeNotebook
  - qevercloud::NotebookRestrictions, 112
- noExpungeNotes
  - qevercloud::NotebookRestrictions, 112
- noExpungeTags
  - qevercloud::NotebookRestrictions, 112
- NONE
  - qevercloud::PremiumOrderStatus, 206
- noPublishToBusinessLibrary
  - qevercloud::NotebookRestrictions, 112
- noPublishToPublic
  - qevercloud::NotebookRestrictions, 112
- noReadNotes
  - qevercloud::NotebookRestrictions, 112
- NORMAL
  - qevercloud::BusinessUserRole, 59
  - qevercloud::PrivilegeLevel, 206
- noSendMessageToRecipients
  - qevercloud::NotebookRestrictions, 112
- noSetDefaultNotebook
  - qevercloud::NotebookRestrictions, 113
- noSetNotebookStack
  - qevercloud::NotebookRestrictions, 113
- noSetParentTag
  - qevercloud::NotebookRestrictions, 113
- noShareNotes
  - qevercloud::NotebookRestrictions, 113
- note
  - qevercloud::NoteEmailParameters, 116
- notebookCounts
  - qevercloud::NoteCollectionCounts, 115
- notebookDescription
  - qevercloud::BusinessNotebook, 56
- notebookDisplayName
  - qevercloud::NotebookDescriptor, 109
- notebookGuid
  - qevercloud::Note, 99
  - qevercloud::NoteFilter, 118
  - qevercloud::NoteMetadata, 123
  - qevercloud::SharedNotebook, 226
- notebookModifiable
  - qevercloud::SharedNotebook, 227
- notebooks
  - qevercloud::RelatedResult, 213
  - qevercloud::SyncChunk, 234
- noteGuid
  - qevercloud::RelatedQuery, 212
  - qevercloud::Resource, 218



- notes
  - qevercloud::NoteList, 120
  - qevercloud::NotesMetadataList, 125
  - qevercloud::RelatedResult, 213
  - qevercloud::SyncChunk, 234
- NoteStore
  - qevercloud::NoteStore, 134
- noteStoreUrl
  - qevercloud::AuthenticationResult, 49
  - qevercloud::EvernoteOAuthWebView::OAuthResult, 194
  - qevercloud::LinkedNotebook, 95
  - qevercloud::NoteStore, 179
  - qevercloud::PublicUserInfo, 207
- noUpdateNotebook
  - qevercloud::NotebookRestrictions, 113
- noUpdateNotes
  - qevercloud::NotebookRestrictions, 113
- noUpdateTags
  - qevercloud::NotebookRestrictions, 113
- oauth.h, 277
- oauthError
  - qevercloud::EvernoteOAuthDialog, 88
  - qevercloud::EvernoteOAuthWebView, 91
- OAuthResult
  - qevercloud::EvernoteOAuthDialog, 86
- oauthResult
  - qevercloud::EvernoteOAuthDialog, 88
  - qevercloud::EvernoteOAuthWebView, 91
- ONLY\_JOINED\_OR\_PREVIEW
  - qevercloud::SharedNotebookInstanceRestrictions, 229
- open
  - qevercloud::EvernoteOAuthDialog, 88
- operator const T &
  - qevercloud::Optional< T >, 198
- operator T&
  - qevercloud::Optional< T >, 199
- operator!=
  - qevercloud::Accounting, 42
  - qevercloud::AuthenticationResult, 48
  - qevercloud::BootstrapInfo, 51
  - qevercloud::BootstrapProfile, 52
  - qevercloud::BootstrapSettings, 53
  - qevercloud::BusinessNotebook, 56
  - qevercloud::BusinessUserInfo, 57
  - qevercloud::ClientUsageMetrics, 59
  - qevercloud::Data, 60
  - qevercloud::LazyMap, 93
  - qevercloud::LinkedNotebook, 94
  - qevercloud::Note, 97
  - qevercloud::NoteAttributes, 101
  - qevercloud::Notebook, 105
  - qevercloud::NotebookDescriptor, 109
  - qevercloud::NotebookRestrictions, 111
  - qevercloud::NoteCollectionCounts, 114
  - qevercloud::NoteEmailParameters, 116
  - qevercloud::NoteFilter, 118
  - qevercloud::NoteList, 120
  - qevercloud::NoteMetadata, 122
  - qevercloud::NotesMetadataList, 124
  - qevercloud::NotesMetadataResultSpec, 127
  - qevercloud::NoteVersionId, 193
  - qevercloud::PremiumInfo, 202
  - qevercloud::PublicUserInfo, 207
  - qevercloud::Publishing, 209
  - qevercloud::RelatedQuery, 211
  - qevercloud::RelatedResult, 213
  - qevercloud::NoteList, 120
  - qevercloud::NoteMetadata, 122
  - qevercloud::NotesMetadataList, 124
  - qevercloud::NotesMetadataResultSpec, 126
  - qevercloud::NoteVersionId, 193
  - qevercloud::PremiumInfo, 202
  - qevercloud::PublicUserInfo, 207
  - qevercloud::Publishing, 209
  - qevercloud::RelatedQuery, 211
  - qevercloud::RelatedResult, 213
  - qevercloud::RelatedResultSpec, 214
  - qevercloud::Resource, 217
  - qevercloud::ResourceAttributes, 220
  - qevercloud::SavedSearch, 222
  - qevercloud::SavedSearchScope, 224
  - qevercloud::SharedNotebook, 226
  - qevercloud::SharedNotebookRecipientSettings, 230
  - qevercloud::SyncChunk, 232
  - qevercloud::SyncChunkFilter, 236
  - qevercloud::SyncState, 238
  - qevercloud::Tag, 240
  - qevercloud::User, 250
  - qevercloud::UserAttributes, 254
- operator->
  - qevercloud::Optional< T >, 199
- operator=
  - qevercloud::Optional< T >, 199, 200
- operator==
  - qevercloud::Accounting, 42
  - qevercloud::AuthenticationResult, 48
  - qevercloud::BootstrapInfo, 51
  - qevercloud::BootstrapProfile, 52
  - qevercloud::BootstrapSettings, 53
  - qevercloud::BusinessNotebook, 56
  - qevercloud::BusinessUserInfo, 57
  - qevercloud::ClientUsageMetrics, 59
  - qevercloud::Data, 60
  - qevercloud::LazyMap, 93
  - qevercloud::LinkedNotebook, 94
  - qevercloud::Note, 97
  - qevercloud::NoteAttributes, 101
  - qevercloud::Notebook, 105
  - qevercloud::NotebookDescriptor, 109
  - qevercloud::NotebookRestrictions, 111
  - qevercloud::NoteCollectionCounts, 114
  - qevercloud::NoteEmailParameters, 116
  - qevercloud::NoteFilter, 118
  - qevercloud::NoteList, 120
  - qevercloud::NoteMetadata, 122
  - qevercloud::NotesMetadataList, 124
  - qevercloud::NotesMetadataResultSpec, 127
  - qevercloud::NoteVersionId, 193
  - qevercloud::PremiumInfo, 202
  - qevercloud::PublicUserInfo, 207
  - qevercloud::Publishing, 209
  - qevercloud::RelatedQuery, 211
  - qevercloud::RelatedResult, 213

- qevercloud::RelatedResultSpec, 214
- qevercloud::Resource, 217
- qevercloud::ResourceAttributes, 220
- qevercloud::SavedSearch, 222
- qevercloud::SavedSearchScope, 224
- qevercloud::SharedNotebook, 226
- qevercloud::SharedNotebookRecipientSettings, 230
- qevercloud::SyncChunk, 233
- qevercloud::SyncChunkFilter, 236
- qevercloud::SyncState, 238
- qevercloud::Tag, 240
- qevercloud::User, 250
- qevercloud::UserAttributes, 254
- Optional
  - qevercloud::Optional< T >, 197, 201
- Optional.h, 278
- order
  - qevercloud::NoteFilter, 118
  - qevercloud::Publishing, 209
- parameter
  - qevercloud::EDAMUserException, 76
- parentGuid
  - qevercloud::Tag, 241
- partnerEmailOptInDate
  - qevercloud::UserAttributes, 256
- PENDING
  - qevercloud::PremiumOrderStatus, 206
- PERMISSION\_DENIED
  - qevercloud::EDAMErrorCode, 63
- placeName
  - qevercloud::NoteAttributes, 103
- plainText
  - qevercloud::RelatedQuery, 212
- PNG
  - qevercloud::Thumbnail::ImageType, 92
- preactivation
  - qevercloud::UserAttributes, 256
- preferredCountry
  - qevercloud::UserAttributes, 256
- preferredLanguage
  - qevercloud::UserAttributes, 256
- PREMIUM
  - qevercloud::PrivilegeLevel, 206
- premium
  - qevercloud::PremiumInfo, 203
- premiumCancellationPending
  - qevercloud::PremiumInfo, 203
- premiumCommerceService
  - qevercloud::Accounting, 43
- premiumExpirationDate
  - qevercloud::PremiumInfo, 203
- premiumExtendable
  - qevercloud::PremiumInfo, 203
- premiumInfo
  - qevercloud::User, 252
- premiumLockUntil
  - qevercloud::Accounting, 43
- premiumOrderNumber
  - qevercloud::Accounting, 44
- premiumPending
  - qevercloud::PremiumInfo, 203
- premiumRecurring
  - qevercloud::PremiumInfo, 203
- premiumServiceSKU
  - qevercloud::Accounting, 44
- premiumServiceStart
  - qevercloud::Accounting, 44
- premiumServiceStatus
  - qevercloud::Accounting, 44
- premiumSubscriptionNumber
  - qevercloud::Accounting, 44
- premiumUpgradable
  - qevercloud::PremiumInfo, 204
- privilege
  - qevercloud::BusinessNotebook, 56
  - qevercloud::PublicUserInfo, 208
  - qevercloud::SharedNotebook, 227
  - qevercloud::User, 252
- profiles
  - qevercloud::BootstrapInfo, 51
- PROTOCOL\_ERROR
  - qevercloud::ThriftException::Type, 249
- publicDescription
  - qevercloud::Publishing, 210
- publicUserInfo
  - qevercloud::AuthenticationResult, 49
- published
  - qevercloud::Notebook, 106
- publishing
  - qevercloud::Notebook, 107
- QEC\_SIGNAL
  - qt4helpers.h, 279
- QEC\_SLOT
  - qt4helpers.h, 279
- qevercloud, 13
  - CLASSIFICATION\_RECIPE\_SERVICE\_RECIPE, 19
  - CLASSIFICATION\_RECIPE\_USER\_NON\_RECIPE, 20
  - CLASSIFICATION\_RECIPE\_USER\_RECIPE, 20
  - EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX, 20
  - EDAM\_APPLICATIONDATA\_NAME\_LEN\_MAX, 20
  - EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN, 20
  - EDAM\_APPLICATIONDATA\_NAME\_REGEX, 20
  - EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MAX, 20
  - EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MIN, 21
  - EDAM\_APPLICATIONDATA\_VALUE\_REGEX, 21
  - EDAM\_ATTRIBUTE\_LEN\_MAX, 21
  - EDAM\_ATTRIBUTE\_LEN\_MIN, 21
  - EDAM\_ATTRIBUTE\_LIST\_MAX, 21
  - EDAM\_ATTRIBUTE\_MAP\_MAX, 21



- EDAM\_ATTRIBUTE\_REGEX, [21](#)
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MAX, [22](#)
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_LEN\_MIN, [22](#)
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_REGEX, [22](#)
- EDAM\_BUSINESS\_NOTEBOOKS\_MAX, [22](#)
- EDAM\_BUSINESS\_NOTES\_MAX, [22](#)
- EDAM\_BUSINESS\_PHONE\_NUMBER\_LEN\_MAX, [22](#)
- EDAM\_BUSINESS\_TAGS\_MAX, [22](#)
- EDAM\_BUSINESS\_URI\_LEN\_MAX, [23](#)
- EDAM\_CONTENT\_CLASS\_FOOD\_MEAL, [23](#)
- EDAM\_CONTENT\_CLASS\_HELLO\_ENCOUNTER, [23](#)
- EDAM\_CONTENT\_CLASS\_HELLO\_PROFILE, [23](#)
- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_NOTEBOOK, [23](#)
- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_PREFIX, [23](#)
- EDAM\_CONTENT\_CLASS\_SKITCH, [23](#)
- EDAM\_CONTENT\_CLASS\_SKITCH\_PDF, [24](#)
- EDAM\_CONTENT\_CLASS\_SKITCH\_PREFIX, [24](#)
- EDAM\_DEVICE\_DESCRIPTION\_LEN\_MAX, [24](#)
- EDAM\_DEVICE\_DESCRIPTION\_REGEX, [24](#)
- EDAM\_DEVICE\_ID\_LEN\_MAX, [24](#)
- EDAM\_DEVICE\_ID\_REGEX, [24](#)
- EDAM\_EMAIL\_DOMAIN\_REGEX, [24](#)
- EDAM\_EMAIL\_LEN\_MAX, [25](#)
- EDAM\_EMAIL\_LEN\_MIN, [25](#)
- EDAM\_EMAIL\_LOCAL\_REGEX, [25](#)
- EDAM\_EMAIL\_REGEX, [25](#)
- EDAM\_FOOD\_APP\_CONTENT\_CLASS\_PREFIX, [25](#)
- EDAM\_GUID\_LEN\_MAX, [25](#)
- EDAM\_GUID\_LEN\_MIN, [25](#)
- EDAM\_GUID\_REGEX, [26](#)
- EDAM\_HASH\_LEN, [26](#)
- EDAM\_HELLO\_APP\_CONTENT\_CLASS\_PREFIX, [26](#)
- EDAM\_INDEXABLE\_RESOURCE\_MIME\_TYPES, [26](#)
- EDAM\_MAX\_PREFERENCES, [26](#)
- EDAM\_MAX\_VALUES\_PER\_PREFERENCE, [26](#)
- EDAM\_MIME\_LEN\_MAX, [26](#)
- EDAM\_MIME\_LEN\_MIN, [27](#)
- EDAM\_MIME\_REGEX, [27](#)
- EDAM\_MIME\_TYPE\_AAC, [27](#)
- EDAM\_MIME\_TYPE\_AMR, [27](#)
- EDAM\_MIME\_TYPE\_DEFAULT, [27](#)
- EDAM\_MIME\_TYPE\_GIF, [27](#)
- EDAM\_MIME\_TYPE\_INK, [27](#)
- EDAM\_MIME\_TYPE\_JPEG, [27](#)
- EDAM\_MIME\_TYPE\_M4A, [28](#)
- EDAM\_MIME\_TYPE\_MP3, [28](#)
- EDAM\_MIME\_TYPE\_MP4\_VIDEO, [28](#)
- EDAM\_MIME\_TYPE\_PDF, [28](#)
- EDAM\_MIME\_TYPE\_PNG, [28](#)
- EDAM\_MIME\_TYPE\_WAV, [28](#)
- EDAM\_MIME\_TYPES, [28](#)
- EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MAX, [28](#)
- EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MIN, [29](#)
- EDAM\_NOTE\_CONTENT\_CLASS\_REGEX, [29](#)
- EDAM\_NOTE\_CONTENT\_LEN\_MAX, [29](#)
- EDAM\_NOTE\_CONTENT\_LEN\_MIN, [29](#)
- EDAM\_NOTE\_RESOURCES\_MAX, [29](#)
- EDAM\_NOTE\_SIZE\_MAX\_FREE, [29](#)
- EDAM\_NOTE\_SIZE\_MAX\_PREMIUM, [29](#)
- EDAM\_NOTE\_SOURCE\_MAIL\_CLIP, [30](#)
- EDAM\_NOTE\_SOURCE\_MAIL\_SMTP\_GATEWAY, [30](#)
- EDAM\_NOTE\_SOURCE\_WEB\_CLIP, [30](#)
- EDAM\_NOTE\_TAGS\_MAX, [30](#)
- EDAM\_NOTE\_TITLE\_LEN\_MAX, [30](#)
- EDAM\_NOTE\_TITLE\_LEN\_MIN, [30](#)
- EDAM\_NOTE\_TITLE\_REGEX, [30](#)
- EDAM\_NOTEBOOK\_NAME\_LEN\_MAX, [31](#)
- EDAM\_NOTEBOOK\_NAME\_LEN\_MIN, [31](#)
- EDAM\_NOTEBOOK\_NAME\_REGEX, [31](#)
- EDAM\_NOTEBOOK\_SHARED\_NOTEBOOK\_MAX, [31](#)
- EDAM\_NOTEBOOK\_STACK\_LEN\_MAX, [31](#)
- EDAM\_NOTEBOOK\_STACK\_LEN\_MIN, [31](#)
- EDAM\_NOTEBOOK\_STACK\_REGEX, [31](#)
- EDAM\_PREFERENCE\_NAME\_LEN\_MAX, [32](#)
- EDAM\_PREFERENCE\_NAME\_LEN\_MIN, [32](#)
- EDAM\_PREFERENCE\_NAME\_REGEX, [32](#)
- EDAM\_PREFERENCE\_SHORTCUTS, [32](#)
- EDAM\_PREFERENCE\_SHORTCUTS\_MAX\_VALUES, [32](#)
- EDAM\_PREFERENCE\_VALUE\_LEN\_MAX, [32](#)
- EDAM\_PREFERENCE\_VALUE\_LEN\_MIN, [32](#)
- EDAM\_PREFERENCE\_VALUE\_REGEX, [33](#)
- EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MAX, [33](#)
- EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MIN, [33](#)
- EDAM\_PUBLISHING\_DESCRIPTION\_REGEX, [33](#)
- EDAM\_PUBLISHING\_URI\_LEN\_MAX, [33](#)
- EDAM\_PUBLISHING\_URI\_LEN\_MIN, [33](#)
- EDAM\_PUBLISHING\_URI\_PROHIBITED, [33](#)
- EDAM\_PUBLISHING\_URI\_REGEX, [34](#)
- EDAM\_RELATED\_MAX\_NOTEBOOKS, [34](#)
- EDAM\_RELATED\_MAX\_NOTES, [34](#)
- EDAM\_RELATED\_MAX\_TAGS, [34](#)
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MAX, [34](#)
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MIN, [34](#)
- EDAM\_RESOURCE\_SIZE\_MAX\_FREE, [34](#)
- EDAM\_RESOURCE\_SIZE\_MAX\_PREMIUM, [34](#)
- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MAX, [35](#)
- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MIN, [35](#)
- EDAM\_SAVED\_SEARCH\_NAME\_REGEX, [35](#)
- EDAM\_SEARCH\_QUERY\_LEN\_MAX, [35](#)
- EDAM\_SEARCH\_QUERY\_LEN\_MIN, [35](#)
- EDAM\_SEARCH\_QUERY\_REGEX, [35](#)

- EDAM\_SEARCH\_SUGGESTIONS\_MAX, 35
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MAX, 36
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MIN, 36
- EDAM\_TAG\_NAME\_LEN\_MAX, 36
- EDAM\_TAG\_NAME\_LEN\_MIN, 36
- EDAM\_TAG\_NAME\_REGEX, 36
- EDAM\_TIMEZONE\_LEN\_MAX, 36
- EDAM\_TIMEZONE\_LEN\_MIN, 36
- EDAM\_TIMEZONE\_REGEX, 37
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX, 37
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX\_PREMIUM, 37
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_FREE, 37
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_PREMIUM, 37
- EDAM\_USER\_NAME\_LEN\_MAX, 37
- EDAM\_USER\_NAME\_LEN\_MIN, 38
- EDAM\_USER\_NAME\_REGEX, 38
- EDAM\_USER\_NOTEBOOKS\_MAX, 38
- EDAM\_USER\_NOTES\_MAX, 38
- EDAM\_USER\_PASSWORD\_LEN\_MAX, 38
- EDAM\_USER\_PASSWORD\_LEN\_MIN, 38
- EDAM\_USER\_PASSWORD\_REGEX, 38
- EDAM\_USER\_RECENT\_MAILED\_ADDRESSES\_MAX, 38
- EDAM\_USER\_SAVED\_SEARCHES\_MAX, 39
- EDAM\_USER\_TAGS\_MAX, 39
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS, 39
- EDAM\_USER\_UPLOAD\_LIMIT\_FREE, 39
- EDAM\_USER\_UPLOAD\_LIMIT\_PREMIUM, 39
- EDAM\_USER\_USERNAME\_LEN\_MAX, 39
- EDAM\_USER\_USERNAME\_LEN\_MIN, 39
- EDAM\_USER\_USERNAME\_REGEX, 40
- EDAM\_VAT\_REGEX, 40
- EDAM\_VERSION\_MAJOR, 40
- EDAM\_VERSION\_MINOR, 40
- EverCloudExceptionData, 40
- evernoteNetworkAccessManager, 19
- Guid, 18
- libraryVersion, 19
- setNonceGenerator, 19
- Timestamp, 18
- UserID, 18
- QEverCloud.h, 278
- qevercloud::Accounting, 41
  - businessId, 42
  - businessName, 42
  - businessRole, 42
  - currency, 42
  - lastFailedCharge, 43
  - lastFailedChargeReason, 43
  - lastRequestedCharge, 43
  - lastSuccessfulCharge, 43
  - nextChargeDate, 43
  - nextPaymentDue, 43
  - operator!=, 42
  - operator==, 42
  - premiumCommerceService, 43
  - premiumLockUntil, 43
  - premiumOrderNumber, 44
  - premiumServiceSKU, 44
  - premiumServiceStart, 44
  - premiumServiceStatus, 44
  - premiumSubscriptionNumber, 44
  - unitDiscount, 44
  - unitPrice, 44
  - updated, 45
  - uploadLimit, 45
  - uploadLimitEnd, 45
  - uploadLimitNextMonth, 45
- qevercloud::AsyncResult, 45
  - ~AsyncResult, 47
  - AsyncResult, 46, 47
  - finished, 47
  - ReadFunctionType, 46
  - waitForFinished, 47
- qevercloud::AuthenticationResult, 48
  - authenticationToken, 49
  - currentTime, 49
  - expiration, 49
  - noteStoreUrl, 49
  - operator!=, 48
  - operator==, 48
  - publicUserInfo, 49
  - secondFactorDeliveryHint, 49
  - secondFactorRequired, 50
  - user, 50
  - webApiUrlPrefix, 50
- qevercloud::BootstrapInfo, 50
  - operator!=, 51
  - operator==, 51
  - profiles, 51
- qevercloud::BootstrapProfile, 51
  - name, 52
  - operator!=, 52
  - operator==, 52
  - settings, 52
- qevercloud::BootstrapSettings, 52
  - accountEmailDomain, 53
  - enableFacebookSharing, 53
  - enableGiftSubscriptions, 54
  - enableLinkedInSharing, 54
  - enablePublicNotebooks, 54
  - enableSharedNotebooks, 54
  - enableSingleNoteSharing, 54
  - enableSponsoredAccounts, 54
  - enableSupportTickets, 54
  - enableTwitterSharing, 54
  - marketingUrl, 55
  - operator!=, 53
  - operator==, 53
  - serviceHost, 55
  - supportUrl, 55
- qevercloud::BusinessNotebook, 55
  - notebookDescription, 56

- operator!=, 56
- operator==, 56
- privilege, 56
- recommended, 56
- qevercloud::BusinessUserInfo, 57
  - businessId, 57
  - businessName, 58
  - email, 58
  - operator!=, 57
  - operator==, 57
  - role, 58
- qevercloud::BusinessUserRole, 58
  - ADMIN, 59
  - NORMAL, 59
  - type, 58
- qevercloud::ClientUsageMetrics, 59
  - operator!=, 59
  - operator==, 59
  - sessions, 59
- qevercloud::Data, 60
  - body, 61
  - bodyHash, 61
  - operator!=, 60
  - operator==, 60
  - size, 61
- qevercloud::EDAMErrorCode, 61
  - AUTH\_EXPIRED, 63
  - BAD\_DATA\_FORMAT, 63
  - DATA\_CONFLICT, 63
  - DATA\_REQUIRED, 63
  - ENML\_VALIDATION, 63
  - INTERNAL\_ERROR, 63
  - INVALID\_AUTH, 63
  - LEN\_TOO\_LONG, 63
  - LEN\_TOO\_SHORT, 63
  - LIMIT\_REACHED, 63
  - PERMISSION\_DENIED, 63
  - QUOTA\_REACHED, 63
  - RATE\_LIMIT\_REACHED, 63
  - SHARD\_UNAVAILABLE, 63
  - TAKEN\_DOWN, 63
  - TOO\_FEW, 63
  - TOO\_MANY, 63
  - type, 62
  - UNKNOWN, 63
  - UNSUPPORTED\_OPERATION, 63
- qevercloud::EDAMNotFoundException, 63
  - ~EDAMNotFoundException, 64
  - EDAMNotFoundException, 64
  - exceptionData, 64
  - identifier, 65
  - key, 65
  - what, 64
- qevercloud::EDAMNotFoundExceptionData, 65
  - EDAMNotFoundExceptionData, 66
  - m\_identifier, 66
  - m\_key, 66
  - throwException, 66
- qevercloud::EDAMSystemException, 67
  - ~EDAMSystemException, 68
  - EDAMSystemException, 67
  - errorCode, 68
  - exceptionData, 68
  - message, 68
  - rateLimitDuration, 68
  - what, 68
- qevercloud::EDAMSystemExceptionAuthExpired, 69
  - exceptionData, 69
- qevercloud::EDAMSystemExceptionAuthExpiredData, 70
  - EDAMSystemExceptionAuthExpiredData, 70
  - throwException, 70
- qevercloud::EDAMSystemExceptionData, 71
  - EDAMSystemExceptionData, 71
  - m\_errorCode, 72
  - m\_message, 72
  - m\_rateLimitDuration, 72
  - throwException, 72
- qevercloud::EDAMSystemExceptionRateLimitReached, 73
  - exceptionData, 73
- qevercloud::EDAMSystemExceptionRateLimitReachedData, 74
  - EDAMSystemExceptionRateLimitReachedData, 74
  - throwException, 74
- qevercloud::EDAMUserException, 75
  - ~EDAMUserException, 76
  - EDAMUserException, 76
  - errorCode, 76
  - exceptionData, 76
  - parameter, 76
  - what, 76
- qevercloud::EDAMUserExceptionData, 77
  - EDAMUserExceptionData, 77
  - m\_errorCode, 78
  - m\_parameter, 78
  - throwException, 77
- qevercloud::EventLoopFinisher, 78
  - ~EventLoopFinisher, 79
  - EventLoopFinisher, 79
  - stopEventLoop, 79
- qevercloud::EverCloudException, 79
  - ~EverCloudException, 80
  - EverCloudException, 80
  - exceptionData, 81
  - m\_error, 81
  - what, 81
- qevercloud::EverCloudExceptionData, 81
  - errorMessage, 83
  - EverCloudExceptionData, 82
  - throwException, 82
- qevercloud::EvernoteException, 83
  - EvernoteException, 84
  - exceptionData, 84
- qevercloud::EvernoteExceptionData, 85

- EvernoteExceptionData, 85
- throwException, 85
- qevercloud::EvernoteOAuthDialog, 86
  - ~EvernoteOAuthDialog, 87
  - EvernoteOAuthDialog, 87
  - exec, 87
  - isSucceeded, 87
  - oauthError, 88
  - OAuthResult, 86
  - oauthResult, 88
  - open, 88
  - setWebViewSizeHint, 88
- qevercloud::EvernoteOAuthWebView, 89
  - authenticate, 90
  - authenticationFailed, 90
  - authenticationFinished, 90
  - authenticationSucceeded, 90
  - EvernoteOAuthWebView, 89
  - isSucceeded, 91
  - oauthError, 91
  - oauthResult, 91
  - setSizeHint, 91
  - sizeHint, 91
- qevercloud::EvernoteOAuthWebView::OAuthResult, 194
  - authenticationToken, 194
  - expires, 194
  - noteStoreUrl, 194
  - shardId, 195
  - userId, 195
  - webApiUrlPrefix, 195
- qevercloud::LazyMap, 92
  - fullMap, 93
  - keysOnly, 93
  - operator!=, 93
  - operator==, 93
- qevercloud::LinkedNotebook, 94
  - businessId, 94
  - guid, 95
  - noteStoreUrl, 95
  - operator!=, 94
  - operator==, 94
  - shardId, 95
  - shareKey, 95
  - shareName, 95
  - stack, 95
  - updateSequenceNum, 96
  - uri, 96
  - username, 96
  - webApiUrlPrefix, 96
- qevercloud::Note, 96
  - active, 97
  - attributes, 97
  - content, 98
  - contentHash, 98
  - contentLength, 98
  - created, 98
  - deleted, 98
  - guid, 98
  - notebookGuid, 99
  - operator!=, 97
  - operator==, 97
  - resources, 99
  - tagGuids, 99
  - tagNames, 99
  - title, 99
  - updated, 99
  - updateSequenceNum, 100
- qevercloud::NoteAttributes, 100
  - altitude, 101
  - applicationData, 101
  - author, 101
  - classifications, 102
  - contentClass, 102
  - creatorId, 102
  - lastEditedBy, 102
  - lastEditorId, 102
  - latitude, 103
  - longitude, 103
  - operator!=, 101
  - operator==, 101
  - placeName, 103
  - reminderDoneTime, 103
  - reminderOrder, 103
  - reminderTime, 103
  - shareDate, 104
  - source, 104
  - sourceApplication, 104
  - sourceURL, 104
  - subjectDate, 104
- qevercloud::Notebook, 105
  - businessNotebook, 106
  - contact, 106
  - defaultNotebook, 106
  - guid, 106
  - name, 106
  - operator!=, 105
  - operator==, 105
  - published, 106
  - publishing, 107
  - restrictions, 107
  - serviceCreated, 107
  - serviceUpdated, 107
  - sharedNotebookIds, 107
  - sharedNotebooks, 107
  - stack, 108
  - updateSequenceNum, 108
- qevercloud::NotebookDescriptor, 108
  - contactName, 109
  - guid, 109
  - hasSharedNotebook, 109
  - joinedUserCount, 109
  - notebookDisplayName, 109
  - operator!=, 109
  - operator==, 109
- qevercloud::NotebookRestrictions, 110
  - expungeWhichSharedNotebookRestrictions, 111

- noCreateNotes, 111
- noCreateSharedNotebooks, 111
- noCreateTags, 111
- noEmailNotes, 111
- noExpungeNotebook, 112
- noExpungeNotes, 112
- noExpungeTags, 112
- noPublishToBusinessLibrary, 112
- noPublishToPublic, 112
- noReadNotes, 112
- noSendMessageToRecipients, 112
- noSetDefaultNotebook, 113
- noSetNotebookStack, 113
- noSetParentTag, 113
- noShareNotes, 113
- noUpdateNotebook, 113
- noUpdateNotes, 113
- noUpdateTags, 113
- operator!=, 111
- operator==, 111
- updateWhichSharedNotebookRestrictions, 114
- qevercloud::NoteCollectionCounts, 114
  - notebookCounts, 115
  - operator!=, 114
  - operator==, 114
  - tagCounts, 115
  - trashCount, 115
- qevercloud::NoteEmailParameters, 115
  - ccAddresses, 116
  - guid, 116
  - message, 116
  - note, 116
  - operator!=, 116
  - operator==, 116
  - subject, 117
  - toAddresses, 117
- qevercloud::NoteFilter, 117
  - ascending, 118
  - emphasized, 118
  - inactive, 118
  - notebookGuid, 118
  - operator!=, 118
  - operator==, 118
  - order, 118
  - tagGuids, 119
  - timeZone, 119
  - words, 119
- qevercloud::NoteList, 119
  - notes, 120
  - operator!=, 120
  - operator==, 120
  - searchedWords, 120
  - startIndex, 120
  - stoppedWords, 120
  - totalNotes, 121
  - updateCount, 121
- qevercloud::NoteMetadata, 121
  - attributes, 122
  - contentLength, 122
  - created, 122
  - deleted, 122
  - guid, 123
  - largestResourceMime, 123
  - largestResourceSize, 123
  - notebookGuid, 123
  - operator!=, 122
  - operator==, 122
  - tagGuids, 123
  - title, 123
  - updated, 123
  - updateSequenceNum, 124
- qevercloud::NotesMetadataList, 124
  - notes, 125
  - operator!=, 124
  - operator==, 124
  - searchedWords, 125
  - startIndex, 125
  - stoppedWords, 125
  - totalNotes, 125
  - updateCount, 125
- qevercloud::NotesMetadataResultSpec, 126
  - includeAttributes, 127
  - includeContentLength, 127
  - includeCreated, 127
  - includeDeleted, 127
  - includeLargestResourceMime, 127
  - includeLargestResourceSize, 127
  - includeNotebookGuid, 128
  - includeTagGuids, 128
  - includeTitle, 128
  - includeUpdated, 128
  - includeUpdateSequenceNum, 128
  - operator!=, 126
  - operator==, 127
- qevercloud::NoteSortOrder, 128
  - CREATED, 129
  - RELEVANCE, 129
  - TITLE, 129
  - type, 129
  - UPDATE\_SEQUENCE\_NUMBER, 129
  - UPDATED, 129
- qevercloud::NoteStore, 129
  - authenticateToSharedNote, 134
  - authenticateToSharedNoteAsync, 135
  - authenticateToSharedNotebook, 135
  - authenticateToSharedNotebookAsync, 136
  - authenticationToken, 136
  - copyNote, 136
  - copyNoteAsync, 137
  - createLinkedNotebook, 137
  - createLinkedNotebookAsync, 138
  - createNote, 138
  - createNoteAsync, 139
  - createNotebook, 139
  - createNotebookAsync, 141
  - createSearch, 141

- createSearchAsync, 142
- createSharedNotebook, 142
- createSharedNotebookAsync, 143
- createTag, 143
- createTagAsync, 144
- deleteNote, 144
- deleteNoteAsync, 145
- emailNote, 145
- emailNoteAsync, 146
- expungeInactiveNotes, 146
- expungeInactiveNotesAsync, 146
- expungeLinkedNotebook, 147
- expungeLinkedNotebookAsync, 147
- expungeNote, 147
- expungeNoteAsync, 148
- expungeNotebook, 148
- expungeNotebookAsync, 149
- expungeNotes, 149
- expungeNotesAsync, 150
- expungeSearch, 150
- expungeSearchAsync, 150
- expungeSharedNotebooks, 150
- expungeSharedNotebooksAsync, 151
- expungeTag, 151
- expungeTagAsync, 152
- findNoteCounts, 152
- findNoteCountsAsync, 153
- findNoteOffset, 153
- findNoteOffsetAsync, 154
- findNotes, 154
- findNotesAsync, 154
- findNotesMetadata, 154
- findNotesMetadataAsync, 155
- findRelated, 156
- findRelatedAsync, 157
- getDefaultNotebook, 157
- getDefaultNotebookAsync, 157
- getFilteredSyncChunk, 157
- getFilteredSyncChunkAsync, 158
- getLinkedNotebookSyncChunk, 158
- getLinkedNotebookSyncChunkAsync, 159
- getLinkedNotebookSyncState, 159
- getLinkedNotebookSyncStateAsync, 159
- getNote, 160
- getNoteApplicationData, 160
- getNoteApplicationDataAsync, 161
- getNoteApplicationDataEntry, 161
- getNoteApplicationDataEntryAsync, 161
- getNoteAsync, 161
- getNotebook, 161
- getNotebookAsync, 162
- getNoteContent, 162
- getNoteContentAsync, 163
- getNoteSearchText, 163
- getNoteSearchTextAsync, 164
- getNoteTagNames, 164
- getNoteTagNamesAsync, 164
- getNoteVersion, 164
- getNoteVersionAsync, 165
- getPublicNotebook, 165
- getPublicNotebookAsync, 166
- getResource, 166
- getResourceAlternateData, 167
- getResourceAlternateDataAsync, 167
- getResourceApplicationData, 168
- getResourceApplicationDataAsync, 168
- getResourceApplicationDataEntry, 168
- getResourceApplicationDataEntryAsync, 168
- getResourceAsync, 169
- getResourceAttributes, 169
- getResourceAttributesAsync, 169
- getResourceByHash, 169
- getResourceByHashAsync, 170
- getResourceData, 170
- getResourceDataAsync, 171
- getResourceRecognition, 171
- getResourceRecognitionAsync, 172
- getResourceSearchText, 172
- getResourceSearchTextAsync, 173
- getSearch, 173
- getSearchAsync, 173
- getSharedNotebookByAuth, 173
- getSharedNotebookByAuthAsync, 174
- getSyncChunk, 174
- getSyncChunkAsync, 174
- getSyncState, 174
- getSyncStateAsync, 174
- getSyncStateWithMetrics, 175
- getSyncStateWithMetricsAsync, 175
- getTag, 175
- getTagAsync, 176
- listLinkedNotebooks, 176
- listLinkedNotebooksAsync, 176
- listNotebooks, 176
- listNotebooksAsync, 176
- listNoteVersions, 176
- listNoteVersionsAsync, 178
- listSearches, 178
- listSearchesAsync, 178
- listSharedNotebooks, 178
- listSharedNotebooksAsync, 178
- listTags, 179
- listTagsAsync, 179
- listTagsByNotebook, 179
- listTagsByNotebookAsync, 179
- NoteStore, 134
- noteStoreUrl, 179
- sendMessageToSharedNotebookMembers, 180
- sendMessageToSharedNotebookMembersAsync, 180
- setAuthenticationToken, 181
- setNoteApplicationDataEntry, 181
- setNoteApplicationDataEntryAsync, 181
- setNoteStoreUrl, 181
- setResourceApplicationDataEntry, 181
- setResourceApplicationDataEntryAsync, 181



- setSharedNotebookRecipientSettings, 182
- setSharedNotebookRecipientSettingsAsync, 182
- shareNote, 183
- shareNoteAsync, 183
- stopSharingNote, 183
- stopSharingNoteAsync, 184
- unsetNoteApplicationDataEntry, 184
- unsetNoteApplicationDataEntryAsync, 184
- unsetResourceApplicationDataEntry, 184
- unsetResourceApplicationDataEntryAsync, 184
- untagAll, 185
- untagAllAsync, 185
- updateLinkedNotebook, 185
- updateLinkedNotebookAsync, 186
- updateNote, 186
- updateNoteAsync, 187
- updateNotebook, 187
- updateNotebookAsync, 188
- updateResource, 188
- updateResourceAsync, 189
- updateSearch, 189
- updateSearchAsync, 190
- updateSharedNotebook, 190
- updateSharedNotebookAsync, 191
- updateTag, 191
- updateTagAsync, 192
- qevercloud::NoteVersionId, 192
  - operator!=, 193
  - operator==, 193
  - saved, 193
  - title, 193
  - updated, 193
  - updateSequenceNum, 193
- qevercloud::Optional< T >, 195
  - clear, 197
  - init, 198
  - isEqual, 198
  - isSet, 198
  - operator const T &, 198
  - operator T&, 199
  - operator->, 199
  - operator=, 199, 200
  - Optional, 197, 201
  - ref, 200
  - swap, 201
  - value, 201
- qevercloud::PremiumInfo, 201
  - canPurchaseUploadAllowance, 202
  - currentTime, 203
  - operator!=, 202
  - operator==, 202
  - premium, 203
  - premiumCancellationPending, 203
  - premiumExpirationDate, 203
  - premiumExtendable, 203
  - premiumPending, 203
  - premiumRecurring, 203
  - premiumUpgradable, 204
  - sponsoredGroupName, 204
  - sponsoredGroupRole, 204
- qevercloud::PremiumOrderStatus, 204
  - ACTIVE, 206
  - CANCELED, 206
  - CANCELLATION\_PENDING, 206
  - FAILED, 206
  - NONE, 206
  - PENDING, 206
  - type, 205
- qevercloud::PrivilegeLevel, 206
  - ADMIN, 206
  - MANAGER, 206
  - NORMAL, 206
  - PREMIUM, 206
  - SUPPORT, 206
  - type, 206
  - VIP, 206
- qevercloud::PublicUserInfo, 207
  - noteStoreUrl, 207
  - operator!=, 207
  - operator==, 207
  - privilege, 208
  - shardId, 208
  - userId, 208
  - username, 208
  - webApiUrlPrefix, 208
- qevercloud::Publishing, 208
  - ascending, 209
  - operator!=, 209
  - operator==, 209
  - order, 209
  - publicDescription, 210
  - uri, 210
- qevercloud::QueryFormat, 210
  - SEXP, 211
  - type, 210
  - USER, 211
- qevercloud::RelatedQuery, 211
  - filter, 212
  - noteGuid, 212
  - operator!=, 211
  - operator==, 211
  - plainText, 212
  - referenceUri, 212
- qevercloud::RelatedResult, 212
  - containingNotebooks, 213
  - notebooks, 213
  - notes, 213
  - operator!=, 213
  - operator==, 213
  - tags, 213
- qevercloud::RelatedResultSpec, 214
  - includeContainingNotebooks, 215
  - maxNotebooks, 215
  - maxNotes, 215
  - maxTags, 215
  - operator!=, 214

- operator==, 214
- writableNotebooksOnly, 215
- qevercloud::ReminderEmailConfig, 215
  - DO\_NOT\_SEND, 216
  - SEND\_DAILY\_EMAIL, 216
  - type, 216
- qevercloud::Resource, 216
  - active, 217
  - alternateData, 217
  - attributes, 217
  - data, 218
  - duration, 218
  - guid, 218
  - height, 218
  - mime, 218
  - noteGuid, 218
  - operator!=, 217
  - operator==, 217
  - recognition, 218
  - updateSequenceNum, 219
  - width, 219
- qevercloud::ResourceAttributes, 219
  - altitude, 220
  - applicationData, 220
  - attachment, 220
  - cameraMake, 220
  - cameraModel, 221
  - clientWillIndex, 221
  - fileName, 221
  - latitude, 221
  - longitude, 221
  - operator!=, 220
  - operator==, 220
  - recoType, 221
  - sourceURL, 221
  - timestamp, 222
- qevercloud::SavedSearch, 222
  - format, 223
  - guid, 223
  - name, 223
  - operator!=, 222
  - operator==, 222
  - query, 223
  - scope, 223
  - updateSequenceNum, 223
- qevercloud::SavedSearchScope, 224
  - includeAccount, 225
  - includeBusinessLinkedNotebooks, 225
  - includePersonalLinkedNotebooks, 225
  - operator!=, 224
  - operator==, 224
- qevercloud::SharedNotebook, 225
  - allowPreview, 226
  - email, 226
  - id, 226
  - notebookGuid, 226
  - notebookModifiable, 227
  - operator!=, 226
  - operator==, 226
  - privilege, 227
  - recipientSettings, 227
  - requireLogin, 227
  - serviceCreated, 227
  - serviceUpdated, 227
  - shareKey, 227
  - userId, 228
  - username, 228
- qevercloud::SharedNotebookInstanceRestrictions, 228
  - NO\_SHARED\_NOTEBOOKS, 229
  - ONLY\_JOINED\_OR\_PREVIEW, 229
  - type, 228
- qevercloud::SharedNotebookPrivilegeLevel, 229
  - BUSINESS\_FULL\_ACCESS, 230
  - FULL\_ACCESS, 230
  - GROUP, 230
  - MODIFY\_NOTEBOOK\_PLUS\_ACTIVITY, 230
  - READ\_NOTEBOOK, 230
  - READ\_NOTEBOOK\_PLUS\_ACTIVITY, 230
  - type, 229
- qevercloud::SharedNotebookRecipientSettings, 230
  - operator!=, 230
  - operator==, 230
  - reminderNotifyEmail, 231
  - reminderNotifyInApp, 231
- qevercloud::SponsoredGroupRole, 231
  - GROUP\_ADMIN, 232
  - GROUP\_MEMBER, 232
  - GROUP\_OWNER, 232
  - type, 231
- qevercloud::SyncChunk, 232
  - chunkHighUSN, 233
  - currentTime, 233
  - expungedLinkedNotebooks, 233
  - expungedNotebooks, 233
  - expungedNotes, 233
  - expungedSearches, 233
  - expungedTags, 234
  - linkedNotebooks, 234
  - notebooks, 234
  - notes, 234
  - operator!=, 232
  - operator==, 233
  - resources, 234
  - searches, 234
  - tags, 234
  - updateCount, 235
- qevercloud::SyncChunkFilter, 235
  - includeExpunged, 236
  - includeLinkedNotebooks, 236
  - includeNoteApplicationDataFullMap, 236
  - includeNoteAttributes, 236
  - includeNotebooks, 236
  - includeNoteResourceApplicationDataFullMap, 237
  - includeNoteResources, 237
  - includeNotes, 237
  - includeResourceApplicationDataFullMap, 237



- includeResources, 237
- includeSearches, 237
- includeTags, 237
- operator!=, 236
- operator==, 236
- requireNoteContentClass, 238
- qevercloud::SyncState, 238
  - currentTime, 239
  - fullSyncBefore, 239
  - operator!=, 238
  - operator==, 238
  - updateCount, 239
  - uploaded, 239
- qevercloud::Tag, 239
  - guid, 240
  - name, 240
  - operator!=, 240
  - operator==, 240
  - parentGuid, 241
  - updateSequenceNum, 241
- qevercloud::ThriftException, 241
  - exceptionData, 242
  - m\_type, 243
  - ThriftException, 242
  - type, 243
  - what, 243
- qevercloud::ThriftException::Type, 249
  - BAD\_SEQUENCE\_ID, 249
  - INTERNAL\_ERROR, 249
  - INVALID\_DATA, 249
  - INVALID\_MESSAGE\_TYPE, 249
  - MISSING\_RESULT, 249
  - PROTOCOL\_ERROR, 249
  - type, 249
  - UNKNOWN, 249
  - UNKNOWN\_METHOD, 249
  - WRONG\_METHOD\_NAME, 249
- qevercloud::ThriftExceptionData, 243
  - m\_type, 244
  - ThriftExceptionData, 244
  - throwException, 244
- qevercloud::Thumbnail, 245
  - ~Thumbnail, 246
  - createPostRequest, 246
  - download, 247
  - downloadAsync, 247
  - setAuthenticationToken, 247
  - setHost, 248
  - setImageType, 248
  - setShardId, 248
  - setSize, 248
  - Thumbnail, 245, 246
- qevercloud::Thumbnail::ImageType, 92
  - BMP, 92
  - GIF, 92
  - JPEG, 92
  - PNG, 92
  - type, 92
- qevercloud::User, 249
  - accounting, 250
  - active, 250
  - attributes, 251
  - businessUserInfo, 251
  - created, 251
  - deleted, 251
  - email, 251
  - id, 251
  - name, 251
  - operator!=, 250
  - operator==, 250
  - premiumInfo, 252
  - privilege, 252
  - shardId, 252
  - timezone, 252
  - updated, 252
  - username, 252
- qevercloud::UserAttributes, 253
  - businessAddress, 254
  - clipFullPage, 254
  - comments, 254
  - dailyEmailLimit, 254
  - dateAgreedToTermsOfService, 255
  - defaultLatitude, 255
  - defaultLocationName, 255
  - defaultLongitude, 255
  - educationalDiscount, 255
  - emailOptOutDate, 255
  - groupName, 255
  - hideSponsorBilling, 256
  - incomingEmailAddress, 256
  - maxReferrals, 256
  - operator!=, 254
  - operator==, 254
  - partnerEmailOptInDate, 256
  - preactivation, 256
  - preferredCountry, 256
  - preferredLanguage, 256
  - recentMailedAddresses, 257
  - recognitionLanguage, 257
  - referrerCode, 257
  - referralCount, 257
  - referralProof, 257
  - reminderEmailConfig, 257
  - sentEmailCount, 257
  - sentEmailDate, 258
  - taxExempt, 258
  - twitterId, 258
  - twitterUserName, 258
  - useEmailAutoFiling, 258
  - viewedPromotions, 258
- qevercloud::UserStore, 259
  - authenticate, 260
  - authenticateAsync, 261
  - authenticateLongSession, 262
  - authenticateLongSessionAsync, 263
  - authenticateToBusiness, 263

- authenticateToBusinessAsync, 264
- authenticationToken, 264
- checkVersion, 264
- checkVersionAsync, 265
- completeTwoFactorAuthentication, 265
- completeTwoFactorAuthenticationAsync, 266
- getBootstrapInfo, 266
- getBootstrapInfoAsync, 266
- getNoteStoreUrl, 267
- getNoteStoreUrlAsync, 267
- getPremiumInfo, 267
- getPremiumInfoAsync, 267
- getPublicUserInfo, 267
- getPublicUserInfoAsync, 268
- getUser, 268
- getUserAsync, 268
- refreshAuthentication, 268
- refreshAuthenticationAsync, 269
- revokeLongSession, 269
- revokeLongSessionAsync, 269
- setAuthenticationToken, 269
- UserStore, 260
- QEVERCLOUD\_EXPORT
  - export.h, 276
- QEverCloudOAuth.h, 278
- qt4helpers.h, 278
  - QEC\_SIGNAL, 279
  - QEC\_SLOT, 279
- query
  - qevercloud::SavedSearch, 223
- QUOTA\_REACHED
  - qevercloud::EDAMErrorCode, 63
- RATE\_LIMIT\_REACHED
  - qevercloud::EDAMErrorCode, 63
- rateLimitDuration
  - qevercloud::EDAMSystemException, 68
- READ\_NOTEBOOK
  - qevercloud::SharedNotebookPrivilegeLevel, 230
- READ\_NOTEBOOK\_PLUS\_ACTIVITY
  - qevercloud::SharedNotebookPrivilegeLevel, 230
- ReadFunctionType
  - qevercloud::AsyncResult, 46
- README.md, 279
- recentMailedAddresses
  - qevercloud::UserAttributes, 257
- recipientSettings
  - qevercloud::SharedNotebook, 227
- recognition
  - qevercloud::Resource, 218
- recognitionLanguage
  - qevercloud::UserAttributes, 257
- recommended
  - qevercloud::BusinessNotebook, 56
- recoType
  - qevercloud::ResourceAttributes, 221
- ref
  - qevercloud::Optional< T >, 200
- referenceUri
  - qevercloud::RelatedQuery, 212
- refererCode
  - qevercloud::UserAttributes, 257
- referralCount
  - qevercloud::UserAttributes, 257
- referralProof
  - qevercloud::UserAttributes, 257
- refreshAuthentication
  - qevercloud::UserStore, 268
- refreshAuthenticationAsync
  - qevercloud::UserStore, 269
- RELEVANCE
  - qevercloud::NoteSortOrder, 129
- reminderDoneTime
  - qevercloud::NoteAttributes, 103
- reminderEmailConfig
  - qevercloud::UserAttributes, 257
- reminderNotifyEmail
  - qevercloud::SharedNotebookRecipientSettings, 231
- reminderNotifyInApp
  - qevercloud::SharedNotebookRecipientSettings, 231
- reminderOrder
  - qevercloud::NoteAttributes, 103
- reminderTime
  - qevercloud::NoteAttributes, 103
- requireLogin
  - qevercloud::SharedNotebook, 227
- requireNoteContentClass
  - qevercloud::SyncChunkFilter, 238
- resources
  - qevercloud::Note, 99
  - qevercloud::SyncChunk, 234
- restrictions
  - qevercloud::Notebook, 107
- revokeLongSession
  - qevercloud::UserStore, 269
- revokeLongSessionAsync
  - qevercloud::UserStore, 269
- role
  - qevercloud::BusinessUserInfo, 58
- saved
  - qevercloud::NoteVersionId, 193
- scope
  - qevercloud::SavedSearch, 223
- searchedWords
  - qevercloud::NoteList, 120
  - qevercloud::NotesMetadataList, 125
- searches
  - qevercloud::SyncChunk, 234
- secondFactorDeliveryHint
  - qevercloud::AuthenticationResult, 49
- secondFactorRequired
  - qevercloud::AuthenticationResult, 50
- SEND\_DAILY\_EMAIL
  - qevercloud::ReminderEmailConfig, 216
- sendMessageToSharedNotebookMembers

- qevercloud::NoteStore, 180
- sendMessageToSharedNotebookMembersAsync
  - qevercloud::NoteStore, 180
- sentEmailCount
  - qevercloud::UserAttributes, 257
- sentEmailDate
  - qevercloud::UserAttributes, 258
- serviceCreated
  - qevercloud::Notebook, 107
  - qevercloud::SharedNotebook, 227
- serviceHost
  - qevercloud::BootstrapSettings, 55
- services.h, 279
- serviceUpdated
  - qevercloud::Notebook, 107
  - qevercloud::SharedNotebook, 227
- sessions
  - qevercloud::ClientUsageMetrics, 59
- setAuthenticationToken
  - qevercloud::NoteStore, 181
  - qevercloud::Thumbnail, 247
  - qevercloud::UserStore, 269
- setHost
  - qevercloud::Thumbnail, 248
- setImageType
  - qevercloud::Thumbnail, 248
- setNonceGenerator
  - qevercloud, 19
- setNoteApplicationDataEntry
  - qevercloud::NoteStore, 181
- setNoteApplicationDataEntryAsync
  - qevercloud::NoteStore, 181
- setNoteStoreUrl
  - qevercloud::NoteStore, 181
- setResourceApplicationDataEntry
  - qevercloud::NoteStore, 181
- setResourceApplicationDataEntryAsync
  - qevercloud::NoteStore, 181
- setShardId
  - qevercloud::Thumbnail, 248
- setSharedNotebookRecipientSettings
  - qevercloud::NoteStore, 182
- setSharedNotebookRecipientSettingsAsync
  - qevercloud::NoteStore, 182
- setSize
  - qevercloud::Thumbnail, 248
- setSizeHint
  - qevercloud::EvernoteOAuthWebView, 91
- settings
  - qevercloud::BootstrapProfile, 52
- setWebViewSizeHint
  - qevercloud::EvernoteOAuthDialog, 88
- SEXP
  - qevercloud::QueryFormat, 211
- SHARD\_UNAVAILABLE
  - qevercloud::EDAMErrorCode, 63
- shardId
  - qevercloud::EvernoteOAuthWebView::OAuthResult, 195
  - qevercloud::LinkedNotebook, 95
  - qevercloud::PublicUserInfo, 208
  - qevercloud::User, 252
- shareDate
  - qevercloud::NoteAttributes, 104
- sharedNotebookIds
  - qevercloud::Notebook, 107
- sharedNotebooks
  - qevercloud::Notebook, 107
- shareKey
  - qevercloud::LinkedNotebook, 95
  - qevercloud::SharedNotebook, 227
- shareName
  - qevercloud::LinkedNotebook, 95
- shareNote
  - qevercloud::NoteStore, 183
- shareNoteAsync
  - qevercloud::NoteStore, 183
- size
  - qevercloud::Data, 61
- sizeHint
  - qevercloud::EvernoteOAuthWebView, 91
- source
  - qevercloud::NoteAttributes, 104
- sourceApplication
  - qevercloud::NoteAttributes, 104
- sourceURL
  - qevercloud::NoteAttributes, 104
  - qevercloud::ResourceAttributes, 221
- sponsoredGroupName
  - qevercloud::PremiumInfo, 204
- sponsoredGroupRole
  - qevercloud::PremiumInfo, 204
- stack
  - qevercloud::LinkedNotebook, 95
  - qevercloud::Notebook, 108
- startIndex
  - qevercloud::NoteList, 120
  - qevercloud::NotesMetadataList, 125
- stopEventLoop
  - qevercloud::EventLoopFinisher, 79
- stoppedWords
  - qevercloud::NoteList, 120
  - qevercloud::NotesMetadataList, 125
- stopSharingNote
  - qevercloud::NoteStore, 183
- stopSharingNoteAsync
  - qevercloud::NoteStore, 184
- subject
  - qevercloud::NoteEmailParameters, 117
- subjectDate
  - qevercloud::NoteAttributes, 104
- SUPPORT
  - qevercloud::PrivilegeLevel, 206
- supportUrl
  - qevercloud::BootstrapSettings, 55

- swap
  - qevercloud::Optional< T >, 201
- tagCounts
  - qevercloud::NoteCollectionCounts, 115
- tagGuids
  - qevercloud::Note, 99
  - qevercloud::NoteFilter, 119
  - qevercloud::NoteMetadata, 123
- tagNames
  - qevercloud::Note, 99
- tags
  - qevercloud::RelatedResult, 213
  - qevercloud::SyncChunk, 234
- TAKEN\_DOWN
  - qevercloud::EDAMErrorCode, 63
- taxExempt
  - qevercloud::UserAttributes, 258
- ThriftException
  - qevercloud::ThriftException, 242
- ThriftExceptionData
  - qevercloud::ThriftExceptionData, 244
- throwException
  - qevercloud::EDAMNotFoundExceptionData, 66
  - qevercloud::EDAMSystemExceptionAuthExpiredData, 70
  - qevercloud::EDAMSystemExceptionData, 72
  - qevercloud::EDAMSystemExceptionRateLimitReachedData, 74
  - qevercloud::EDAMUserExceptionData, 77
  - qevercloud::EverCloudExceptionData, 82
  - qevercloud::EvernoteExceptionData, 85
  - qevercloud::ThriftExceptionData, 244
- Thumbnail
  - qevercloud::Thumbnail, 245, 246
- thumbnail.h, 280
- Timestamp
  - qevercloud, 18
- timestamp
  - qevercloud::ResourceAttributes, 222
- timeZone
  - qevercloud::NoteFilter, 119
- timezone
  - qevercloud::User, 252
- TITLE
  - qevercloud::NoteSortOrder, 129
- title
  - qevercloud::Note, 99
  - qevercloud::NoteMetadata, 123
  - qevercloud::NoteVersionId, 193
- toAddresses
  - qevercloud::NoteEmailParameters, 117
- TOO\_FEW
  - qevercloud::EDAMErrorCode, 63
- TOO\_MANY
  - qevercloud::EDAMErrorCode, 63
- totalNotes
  - qevercloud::NoteList, 121
  - qevercloud::NotesMetadataList, 125
- trashCount
  - qevercloud::NoteCollectionCounts, 115
- twitterId
  - qevercloud::UserAttributes, 258
- twitterUserName
  - qevercloud::UserAttributes, 258
- type
  - qevercloud::BusinessUserRole, 58
  - qevercloud::EDAMErrorCode, 62
  - qevercloud::NoteSortOrder, 129
  - qevercloud::PremiumOrderStatus, 205
  - qevercloud::PrivilegeLevel, 206
  - qevercloud::QueryFormat, 210
  - qevercloud::ReminderEmailConfig, 216
  - qevercloud::SharedNotebookInstanceRestrictions, 228
  - qevercloud::SharedNotebookPrivilegeLevel, 229
  - qevercloud::SponsoredGroupRole, 231
  - qevercloud::ThriftException, 243
  - qevercloud::ThriftException::Type, 249
  - qevercloud::Thumbnail::ImageType, 92
- types.h, 280
- unitDiscount
  - qevercloud::Accounting, 44
- unitPrice
  - qevercloud::Accounting, 44
- UNKNOWN
  - qevercloud::EDAMErrorCode, 63
  - qevercloud::ThriftException::Type, 249
- UNKNOWN\_METHOD
  - qevercloud::ThriftException::Type, 249
- unsetNoteApplicationDataEntry
  - qevercloud::NoteStore, 184
- unsetNoteApplicationDataEntryAsync
  - qevercloud::NoteStore, 184
- unsetResourceApplicationDataEntry
  - qevercloud::NoteStore, 184
- unsetResourceApplicationDataEntryAsync
  - qevercloud::NoteStore, 184
- UNSUPPORTED\_OPERATION
  - qevercloud::EDAMErrorCode, 63
- untagAll
  - qevercloud::NoteStore, 185
- untagAllAsync
  - qevercloud::NoteStore, 185
- UPDATE\_SEQUENCE\_NUMBER
  - qevercloud::NoteSortOrder, 129
- updateCount
  - qevercloud::NoteList, 121
  - qevercloud::NotesMetadataList, 125
  - qevercloud::SyncChunk, 235
  - qevercloud::SyncState, 239
- UPDATED
  - qevercloud::NoteSortOrder, 129
- updated
  - qevercloud::Accounting, 45
  - qevercloud::Note, 99
  - qevercloud::NoteMetadata, 123

- qevercloud::NoteVersionId, 193
- qevercloud::User, 252
- updateLinkedNotebook
  - qevercloud::NoteStore, 185
- updateLinkedNotebookAsync
  - qevercloud::NoteStore, 186
- updateNote
  - qevercloud::NoteStore, 186
- updateNoteAsync
  - qevercloud::NoteStore, 187
- updateNotebook
  - qevercloud::NoteStore, 187
- updateNotebookAsync
  - qevercloud::NoteStore, 188
- updateResource
  - qevercloud::NoteStore, 188
- updateResourceAsync
  - qevercloud::NoteStore, 189
- updateSearch
  - qevercloud::NoteStore, 189
- updateSearchAsync
  - qevercloud::NoteStore, 190
- updateSequenceNum
  - qevercloud::LinkedNotebook, 96
  - qevercloud::Note, 100
  - qevercloud::Notebook, 108
  - qevercloud::NoteMetadata, 124
  - qevercloud::NoteVersionId, 193
  - qevercloud::Resource, 219
  - qevercloud::SavedSearch, 223
  - qevercloud::Tag, 241
- updateSharedNotebook
  - qevercloud::NoteStore, 190
- updateSharedNotebookAsync
  - qevercloud::NoteStore, 191
- updateTag
  - qevercloud::NoteStore, 191
- updateTagAsync
  - qevercloud::NoteStore, 192
- updateWhichSharedNotebookRestrictions
  - qevercloud::NotebookRestrictions, 114
- uploaded
  - qevercloud::SyncState, 239
- uploadLimit
  - qevercloud::Accounting, 45
- uploadLimitEnd
  - qevercloud::Accounting, 45
- uploadLimitNextMonth
  - qevercloud::Accounting, 45
- uri
  - qevercloud::LinkedNotebook, 96
  - qevercloud::Publishing, 210
- useEmailAutoFiling
  - qevercloud::UserAttributes, 258
- USER
  - qevercloud::QueryFormat, 211
- user
  - qevercloud::AuthenticationResult, 50
- UserID
  - qevercloud, 18
- userId
  - qevercloud::EvernoteOAuthWebView::OAuthResult, 195
  - qevercloud::PublicUserInfo, 208
  - qevercloud::SharedNotebook, 228
- username
  - qevercloud::LinkedNotebook, 96
  - qevercloud::PublicUserInfo, 208
  - qevercloud::SharedNotebook, 228
  - qevercloud::User, 252
- UserStore
  - qevercloud::UserStore, 260
- value
  - qevercloud::Optional< T >, 201
- viewedPromotions
  - qevercloud::UserAttributes, 258
- VIP
  - qevercloud::PrivilegeLevel, 206
- waitFinished
  - qevercloud::AsyncResult, 47
- webApiUrlPrefix
  - qevercloud::AuthenticationResult, 50
  - qevercloud::EvernoteOAuthWebView::OAuthResult, 195
  - qevercloud::LinkedNotebook, 96
  - qevercloud::PublicUserInfo, 208
- what
  - qevercloud::EDAMNotFoundException, 64
  - qevercloud::EDAMSystemException, 68
  - qevercloud::EDAMUserException, 76
  - qevercloud::EverCloudException, 81
  - qevercloud::ThriftException, 243
- width
  - qevercloud::Resource, 219
- words
  - qevercloud::NoteFilter, 119
- writableNotebooksOnly
  - qevercloud::RelatedResultSpec, 215
- WRONG\_METHOD\_NAME
  - qevercloud::ThriftException::Type, 249