

QEverCloud

6.2.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>QEverCloud</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>5</b>
2.1	Namespace List . . . . .	5
<b>3</b>	<b>Hierarchical Index</b>	<b>7</b>
3.1	Class Hierarchy . . . . .	7
<b>4</b>	<b>Class Index</b>	<b>13</b>
4.1	Class List . . . . .	13
<b>5</b>	<b>File Index</b>	<b>17</b>
5.1	File List . . . . .	17
<b>6</b>	<b>Namespace Documentation</b>	<b>19</b>
6.1	qevercloud Namespace Reference . . . . .	19
6.1.1	Detailed Description . . . . .	29
6.1.2	Typedef Documentation . . . . .	30
6.1.2.1	EverCloudExceptionDataPtr . . . . .	30
6.1.2.2	Guid . . . . .	30
6.1.2.3	IdentityID . . . . .	30
6.1.2.4	IDurableServicePtr . . . . .	30
6.1.2.5	ILoggerPtr . . . . .	30
6.1.2.6	INoteStorePtr . . . . .	30
6.1.2.7	InvalidationSequenceNumber . . . . .	31
6.1.2.8	IRequestContextPtr . . . . .	31

6.1.2.9	<a href="#">IRetryPolicyPtr</a>	31
6.1.2.10	<a href="#">IUserStorePtr</a>	31
6.1.2.11	<a href="#">MessageEventID</a>	31
6.1.2.12	<a href="#">MessageThreadID</a>	31
6.1.2.13	<a href="#">Timestamp</a>	31
6.1.2.14	<a href="#">UserID</a>	32
6.1.3	<a href="#">Enumeration Type Documentation</a>	32
6.1.3.1	<a href="#">BusinessInvitationStatus</a>	32
6.1.3.2	<a href="#">BusinessUserRole</a>	32
6.1.3.3	<a href="#">BusinessUserStatus</a>	33
6.1.3.4	<a href="#">CanMoveToContainerStatus</a>	33
6.1.3.5	<a href="#">ContactType</a>	34
6.1.3.6	<a href="#">EDAMErrorCode</a>	34
6.1.3.7	<a href="#">EDAMInvalidContactReason</a>	36
6.1.3.8	<a href="#">EntityType</a>	36
6.1.3.9	<a href="#">LogLevel</a>	37
6.1.3.10	<a href="#">NoteSortOrder</a>	37
6.1.3.11	<a href="#">PremiumOrderStatus</a>	37
6.1.3.12	<a href="#">PrivilegeLevel</a>	38
6.1.3.13	<a href="#">QueryFormat</a>	38
6.1.3.14	<a href="#">RecipientStatus</a>	39
6.1.3.15	<a href="#">RelatedContentAccess</a>	39
6.1.3.16	<a href="#">RelatedContentType</a>	40
6.1.3.17	<a href="#">ReminderEmailConfig</a>	40
6.1.3.18	<a href="#">ServiceLevel</a>	40
6.1.3.19	<a href="#">SharedNotebookInstanceRestrictions</a>	41
6.1.3.20	<a href="#">SharedNotebookPrivilegeLevel</a>	41
6.1.3.21	<a href="#">SharedNotePrivilegeLevel</a>	42
6.1.3.22	<a href="#">ShareRelationshipPrivilegeLevel</a>	42
6.1.3.23	<a href="#">SponsoredGroupRole</a>	43

6.1.3.24	UserIdentityType . . . . .	43
6.1.4	Function Documentation . . . . .	43
6.1.4.1	evernoteNetworkProxy() . . . . .	43
6.1.4.2	initializeQEverCloud() . . . . .	44
6.1.4.3	libraryVersion() . . . . .	44
6.1.4.4	logger() . . . . .	44
6.1.4.5	newDurableService() . . . . .	44
6.1.4.6	newNoteStore() . . . . .	44
6.1.4.7	newRequestContext() . . . . .	45
6.1.4.8	newRetryPolicy() . . . . .	45
6.1.4.9	newStdErrLogger() . . . . .	45
6.1.4.10	newUserStore() . . . . .	45
6.1.4.11	nullLogger() . . . . .	45
6.1.4.12	nullRetryPolicy() . . . . .	45
6.1.4.13	operator<<() [1/48] . . . . .	46
6.1.4.14	operator<<() [2/48] . . . . .	46
6.1.4.15	operator<<() [3/48] . . . . .	46
6.1.4.16	operator<<() [4/48] . . . . .	46
6.1.4.17	operator<<() [5/48] . . . . .	46
6.1.4.18	operator<<() [6/48] . . . . .	46
6.1.4.19	operator<<() [7/48] . . . . .	47
6.1.4.20	operator<<() [8/48] . . . . .	47
6.1.4.21	operator<<() [9/48] . . . . .	47
6.1.4.22	operator<<() [10/48] . . . . .	47
6.1.4.23	operator<<() [11/48] . . . . .	47
6.1.4.24	operator<<() [12/48] . . . . .	47
6.1.4.25	operator<<() [13/48] . . . . .	48
6.1.4.26	operator<<() [14/48] . . . . .	48
6.1.4.27	operator<<() [15/48] . . . . .	48
6.1.4.28	operator<<() [16/48] . . . . .	48

6.1.4.29	operator<<()	[ 17 / 48 ]	48
6.1.4.30	operator<<()	[ 18 / 48 ]	48
6.1.4.31	operator<<()	[ 19 / 48 ]	49
6.1.4.32	operator<<()	[ 20 / 48 ]	49
6.1.4.33	operator<<()	[ 21 / 48 ]	49
6.1.4.34	operator<<()	[ 22 / 48 ]	49
6.1.4.35	operator<<()	[ 23 / 48 ]	49
6.1.4.36	operator<<()	[ 24 / 48 ]	49
6.1.4.37	operator<<()	[ 25 / 48 ]	50
6.1.4.38	operator<<()	[ 26 / 48 ]	50
6.1.4.39	operator<<()	[ 27 / 48 ]	50
6.1.4.40	operator<<()	[ 28 / 48 ]	50
6.1.4.41	operator<<()	[ 29 / 48 ]	50
6.1.4.42	operator<<()	[ 30 / 48 ]	50
6.1.4.43	operator<<()	[ 31 / 48 ]	51
6.1.4.44	operator<<()	[ 32 / 48 ]	51
6.1.4.45	operator<<()	[ 33 / 48 ]	51
6.1.4.46	operator<<()	[ 34 / 48 ]	51
6.1.4.47	operator<<()	[ 35 / 48 ]	51
6.1.4.48	operator<<()	[ 36 / 48 ]	51
6.1.4.49	operator<<()	[ 37 / 48 ]	52
6.1.4.50	operator<<()	[ 38 / 48 ]	52
6.1.4.51	operator<<()	[ 39 / 48 ]	52
6.1.4.52	operator<<()	[ 40 / 48 ]	52
6.1.4.53	operator<<()	[ 41 / 48 ]	52
6.1.4.54	operator<<()	[ 42 / 48 ]	52
6.1.4.55	operator<<()	[ 43 / 48 ]	53
6.1.4.56	operator<<()	[ 44 / 48 ]	53
6.1.4.57	operator<<()	[ 45 / 48 ]	53
6.1.4.58	operator<<()	[ 46 / 48 ]	53

6.1.4.59	<a href="#">operator&lt;&lt;()</a> [47/48]	53
6.1.4.60	<a href="#">operator&lt;&lt;()</a> [48/48]	53
6.1.4.61	<a href="#">qHash()</a> [1/23]	54
6.1.4.62	<a href="#">qHash()</a> [2/23]	54
6.1.4.63	<a href="#">qHash()</a> [3/23]	54
6.1.4.64	<a href="#">qHash()</a> [4/23]	54
6.1.4.65	<a href="#">qHash()</a> [5/23]	54
6.1.4.66	<a href="#">qHash()</a> [6/23]	54
6.1.4.67	<a href="#">qHash()</a> [7/23]	54
6.1.4.68	<a href="#">qHash()</a> [8/23]	55
6.1.4.69	<a href="#">qHash()</a> [9/23]	55
6.1.4.70	<a href="#">qHash()</a> [10/23]	55
6.1.4.71	<a href="#">qHash()</a> [11/23]	55
6.1.4.72	<a href="#">qHash()</a> [12/23]	55
6.1.4.73	<a href="#">qHash()</a> [13/23]	55
6.1.4.74	<a href="#">qHash()</a> [14/23]	55
6.1.4.75	<a href="#">qHash()</a> [15/23]	56
6.1.4.76	<a href="#">qHash()</a> [16/23]	56
6.1.4.77	<a href="#">qHash()</a> [17/23]	56
6.1.4.78	<a href="#">qHash()</a> [18/23]	56
6.1.4.79	<a href="#">qHash()</a> [19/23]	56
6.1.4.80	<a href="#">qHash()</a> [20/23]	56
6.1.4.81	<a href="#">qHash()</a> [21/23]	56
6.1.4.82	<a href="#">qHash()</a> [22/23]	57
6.1.4.83	<a href="#">qHash()</a> [23/23]	57
6.1.4.84	<a href="#">resetEvernoteNetworkProxy()</a>	57
6.1.4.85	<a href="#">setEvernoteNetworkProxy()</a>	57
6.1.4.86	<a href="#">setLogger()</a>	57
6.1.4.87	<a href="#">setNonceGenerator()</a>	58
6.1.4.88	<a href="#">toRange()</a> [1/2]	58

6.1.4.89	<code>toRange()</code> [2/2]	58
6.1.5	Variable Documentation	58
6.1.5.1	<code>CLASSIFICATION_RECIPE_SERVICE_RECIPE</code>	58
6.1.5.2	<code>CLASSIFICATION_RECIPE_USER_NON_RECIPE</code>	58
6.1.5.3	<code>CLASSIFICATION_RECIPE_USER_RECIPE</code>	59
6.1.5.4	<code>EDAM_APP_RATING_MAX</code>	59
6.1.5.5	<code>EDAM_APP_RATING_MIN</code>	59
6.1.5.6	<code>EDAM_APPLICATIONDATA_ENTRY_LEN_MAX</code>	59
6.1.5.7	<code>EDAM_APPLICATIONDATA_NAME_LEN_MAX</code>	59
6.1.5.8	<code>EDAM_APPLICATIONDATA_NAME_LEN_MIN</code>	59
6.1.5.9	<code>EDAM_APPLICATIONDATA_NAME_REGEX</code>	59
6.1.5.10	<code>EDAM_APPLICATIONDATA_VALUE_LEN_MAX</code>	60
6.1.5.11	<code>EDAM_APPLICATIONDATA_VALUE_LEN_MIN</code>	60
6.1.5.12	<code>EDAM_APPLICATIONDATA_VALUE_REGEX</code>	60
6.1.5.13	<code>EDAM_ATTRIBUTE_LEN_MAX</code>	60
6.1.5.14	<code>EDAM_ATTRIBUTE_LEN_MIN</code>	60
6.1.5.15	<code>EDAM_ATTRIBUTE_LIST_MAX</code>	60
6.1.5.16	<code>EDAM_ATTRIBUTE_MAP_MAX</code>	60
6.1.5.17	<code>EDAM_ATTRIBUTE_REGEX</code>	61
6.1.5.18	<code>EDAM_BUSINESS_MARKETING_CODE_REGEX_PATTERN</code>	61
6.1.5.19	<code>EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LEN_MAX</code>	61
6.1.5.20	<code>EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LEN_MIN</code>	61
6.1.5.21	<code>EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_REGEX</code>	61
6.1.5.22	<code>EDAM_BUSINESS_NOTEBOOKS_MAX</code>	61
6.1.5.23	<code>EDAM_BUSINESS_NOTES_MAX</code>	61
6.1.5.24	<code>EDAM_BUSINESS_PHONE_NUMBER_LEN_MAX</code>	62
6.1.5.25	<code>EDAM_BUSINESS_TAGS_MAX</code>	62
6.1.5.26	<code>EDAM_BUSINESS_URI_LEN_MAX</code>	62
6.1.5.27	<code>EDAM_BUSINESS_WORKSPACES_MAX</code>	62
6.1.5.28	<code>EDAM_CONNECTED_IDENTITY_REQUEST_MAX</code>	62



6.1.5.29	EDAM_CONTENT_CLASS_FOOD_MEAL . . . . .	62
6.1.5.30	EDAM_CONTENT_CLASS_HELLO_ENCOUNTER . . . . .	62
6.1.5.31	EDAM_CONTENT_CLASS_HELLO_PROFILE . . . . .	63
6.1.5.32	EDAM_CONTENT_CLASS_PENULTIMATE_NOTEBOOK . . . . .	63
6.1.5.33	EDAM_CONTENT_CLASS_PENULTIMATE_PREFIX . . . . .	63
6.1.5.34	EDAM_CONTENT_CLASS_SKITCH . . . . .	63
6.1.5.35	EDAM_CONTENT_CLASS_SKITCH_PDF . . . . .	63
6.1.5.36	EDAM_CONTENT_CLASS_SKITCH_PREFIX . . . . .	63
6.1.5.37	EDAM_DEVICE_DESCRIPTION_LEN_MAX . . . . .	63
6.1.5.38	EDAM_DEVICE_DESCRIPTION_REGEX . . . . .	64
6.1.5.39	EDAM_DEVICE_ID_LEN_MAX . . . . .	64
6.1.5.40	EDAM_DEVICE_ID_REGEX . . . . .	64
6.1.5.41	EDAM_EMAIL_DOMAIN_REGEX . . . . .	64
6.1.5.42	EDAM_EMAIL_LEN_MAX . . . . .	64
6.1.5.43	EDAM_EMAIL_LEN_MIN . . . . .	64
6.1.5.44	EDAM_EMAIL_LOCAL_REGEX . . . . .	64
6.1.5.45	EDAM_EMAIL_REGEX . . . . .	64
6.1.5.46	EDAM_FIND_CONTACT_DEFAULT_MAX_RESULTS . . . . .	65
6.1.5.47	EDAM_FIND_CONTACT_MAX_RESULTS . . . . .	65
6.1.5.48	EDAM_FOOD_APP_CONTENT_CLASS_PREFIX . . . . .	65
6.1.5.49	EDAM_GET_ORDERS_MAX_RESULTS . . . . .	65
6.1.5.50	EDAM_GUID_LEN_MAX . . . . .	65
6.1.5.51	EDAM_GUID_LEN_MIN . . . . .	65
6.1.5.52	EDAM_GUID_REGEX . . . . .	65
6.1.5.53	EDAM_HASH_LEN . . . . .	66
6.1.5.54	EDAM_HELLO_APP_CONTENT_CLASS_PREFIX . . . . .	66
6.1.5.55	EDAM_INDEXABLE_PLAINTEXT_MIME_TYPES . . . . .	66
6.1.5.56	EDAM_INDEXABLE_RESOURCE_MIME_TYPES . . . . .	66
6.1.5.57	EDAM_MAX_PREFERENCES . . . . .	66
6.1.5.58	EDAM_MAX_VALUES_PER_PREFERENCE . . . . .	66

6.1.5.59	EDAM_MESSAGE_ATTACHMENT_SNIPPET_LEN_MAX . . . . .	66
6.1.5.60	EDAM_MESSAGE_ATTACHMENT_SNIPPET_REGEX . . . . .	67
6.1.5.61	EDAM_MESSAGE_ATTACHMENT_TITLE_LEN_MAX . . . . .	67
6.1.5.62	EDAM_MESSAGE_ATTACHMENT_TITLE_REGEX . . . . .	67
6.1.5.63	EDAM_MESSAGE_ATTACHMENTS_MAX . . . . .	67
6.1.5.64	EDAM_MESSAGE_BODY_LEN_MAX . . . . .	67
6.1.5.65	EDAM_MESSAGE_BODY_REGEX . . . . .	67
6.1.5.66	EDAM_MESSAGE_RECIPIENTS_MAX . . . . .	67
6.1.5.67	EDAM_MIME_LEN_MAX . . . . .	67
6.1.5.68	EDAM_MIME_LEN_MIN . . . . .	68
6.1.5.69	EDAM_MIME_REGEX . . . . .	68
6.1.5.70	EDAM_MIME_TYPE_AAC . . . . .	68
6.1.5.71	EDAM_MIME_TYPE_AMR . . . . .	68
6.1.5.72	EDAM_MIME_TYPE_BMP . . . . .	68
6.1.5.73	EDAM_MIME_TYPE_DEFAULT . . . . .	68
6.1.5.74	EDAM_MIME_TYPE_GIF . . . . .	68
6.1.5.75	EDAM_MIME_TYPE_INK . . . . .	68
6.1.5.76	EDAM_MIME_TYPE_JPEG . . . . .	69
6.1.5.77	EDAM_MIME_TYPE_M4A . . . . .	69
6.1.5.78	EDAM_MIME_TYPE_MP3 . . . . .	69
6.1.5.79	EDAM_MIME_TYPE_MP4_VIDEO . . . . .	69
6.1.5.80	EDAM_MIME_TYPE_PDF . . . . .	69
6.1.5.81	EDAM_MIME_TYPE_PNG . . . . .	69
6.1.5.82	EDAM_MIME_TYPE_TIFF . . . . .	69
6.1.5.83	EDAM_MIME_TYPE_WAV . . . . .	69
6.1.5.84	EDAM_MIME_TYPES . . . . .	70
6.1.5.85	EDAM_NOTE_BUSINESS_SHARED_NOTE_MAX . . . . .	70
6.1.5.86	EDAM_NOTE_CONTENT_CLASS_LEN_MAX . . . . .	70
6.1.5.87	EDAM_NOTE_CONTENT_CLASS_LEN_MIN . . . . .	70
6.1.5.88	EDAM_NOTE_CONTENT_CLASS_REGEX . . . . .	70

6.1.5.89	EDAM_NOTE_CONTENT_LEN_MAX	70
6.1.5.90	EDAM_NOTE_CONTENT_LEN_MIN	70
6.1.5.91	EDAM_NOTE_LOCK_VIEWERS_NOTES_MAX	71
6.1.5.92	EDAM_NOTE_PERSONAL_SHARED_NOTE_MAX	71
6.1.5.93	EDAM_NOTE_RESOURCES_MAX	71
6.1.5.94	EDAM_NOTE_SIZE_MAX_FREE	71
6.1.5.95	EDAM_NOTE_SIZE_MAX_PREMIUM	71
6.1.5.96	EDAM_NOTE_SOURCE_MAIL_CLIP	71
6.1.5.97	EDAM_NOTE_SOURCE_MAIL_SMTP_GATEWAY	71
6.1.5.98	EDAM_NOTE_SOURCE_WEB_CLIP	72
6.1.5.99	EDAM_NOTE_SOURCE_WEB_CLIP_SIMPLIFIED	72
6.1.5.100	EDAM_NOTE_TAGS_MAX	72
6.1.5.101	EDAM_NOTE_TITLE_LEN_MAX	72
6.1.5.102	EDAM_NOTE_TITLE_LEN_MIN	72
6.1.5.103	EDAM_NOTE_TITLE_QUALITY_HIGH	72
6.1.5.104	EDAM_NOTE_TITLE_QUALITY_LOW	72
6.1.5.105	EDAM_NOTE_TITLE_QUALITY_MEDIUM	73
6.1.5.106	EDAM_NOTE_TITLE_QUALITY_UNTITLED	73
6.1.5.107	EDAM_NOTE_TITLE_REGEX	73
6.1.5.108	EDAM_NOTEBOOK_BUSINESS_SHARED_NOTEBOOK_MAX	73
6.1.5.109	EDAM_NOTEBOOK_NAME_LEN_MAX	73
6.1.5.110	EDAM_NOTEBOOK_NAME_LEN_MIN	73
6.1.5.111	EDAM_NOTEBOOK_NAME_REGEX	73
6.1.5.112	EDAM_NOTEBOOK_PERSONAL_SHARED_NOTEBOOK_MAX	74
6.1.5.113	EDAM_NOTEBOOK_STACK_LEN_MAX	74
6.1.5.114	EDAM_NOTEBOOK_STACK_LEN_MIN	74
6.1.5.115	EDAM_NOTEBOOK_STACK_REGEX	74
6.1.5.116	EDAM_OPEN_ID_ACCESS_TOKEN_MAX	74
6.1.5.117	EDAM_PREFERENCE_BUSINESS_DEFAULT_NOTEBOOK	74
6.1.5.118	EDAM_PREFERENCE_BUSINESS_QUICKNOTE	75

6.1.5.119 EDAM_PREFERENCE_NAME_LEN_MAX . . . . .	75
6.1.5.120 EDAM_PREFERENCE_NAME_LEN_MIN . . . . .	75
6.1.5.121 EDAM_PREFERENCE_NAME_REGEX . . . . .	75
6.1.5.122 EDAM_PREFERENCE_ONLY_ONE_VALUE_LEN_MAX . . . . .	75
6.1.5.123 EDAM_PREFERENCE_ONLY_ONE_VALUE_REGEX . . . . .	75
6.1.5.124 EDAM_PREFERENCE_SHORTCUTS . . . . .	76
6.1.5.125 EDAM_PREFERENCE_SHORTCUTS_MAX_VALUES . . . . .	76
6.1.5.126 EDAM_PREFERENCE_VALUE_LEN_MAX . . . . .	76
6.1.5.127 EDAM_PREFERENCE_VALUE_LEN_MIN . . . . .	76
6.1.5.128 EDAM_PREFERENCE_VALUE_REGEX . . . . .	76
6.1.5.129 EDAM_PROMOTION_ID_LEN_MAX . . . . .	76
6.1.5.130 EDAM_PROMOTION_ID_REGEX . . . . .	76
6.1.5.131 EDAM_PUBLISHING_DESCRIPTION_LEN_MAX . . . . .	76
6.1.5.132 EDAM_PUBLISHING_DESCRIPTION_LEN_MIN . . . . .	77
6.1.5.133 EDAM_PUBLISHING_DESCRIPTION_REGEX . . . . .	77
6.1.5.134 EDAM_PUBLISHING_URI_LEN_MAX . . . . .	77
6.1.5.135 EDAM_PUBLISHING_URI_LEN_MIN . . . . .	77
6.1.5.136 EDAM_PUBLISHING_URI_PROHIBITED . . . . .	77
6.1.5.137 EDAM_PUBLISHING_URI_REGEX . . . . .	77
6.1.5.138 EDAM_RELATED_MAX_EXPERTS . . . . .	77
6.1.5.139 EDAM_RELATED_MAX_NOTEBOOKS . . . . .	78
6.1.5.140 EDAM_RELATED_MAX_NOTES . . . . .	78
6.1.5.141 EDAM_RELATED_MAX_RELATED_CONTENT . . . . .	78
6.1.5.142 EDAM_RELATED_MAX_TAGS . . . . .	78
6.1.5.143 EDAM_RELATED_PLAINTEXT_LEN_MAX . . . . .	78
6.1.5.144 EDAM_RELATED_PLAINTEXT_LEN_MIN . . . . .	78
6.1.5.145 EDAM_RESOURCE_SIZE_MAX_FREE . . . . .	78
6.1.5.146 EDAM_RESOURCE_SIZE_MAX_PREMIUM . . . . .	78
6.1.5.147 EDAM_SAVED_SEARCH_NAME_LEN_MAX . . . . .	79
6.1.5.148 EDAM_SAVED_SEARCH_NAME_LEN_MIN . . . . .	79

6.1.5.149 EDAM_SAVED_SEARCH_NAME_REGEX . . . . .	79
6.1.5.150 EDAM_SEARCH_QUERY_LEN_MAX . . . . .	79
6.1.5.151 EDAM_SEARCH_QUERY_LEN_MIN . . . . .	79
6.1.5.152 EDAM_SEARCH_QUERY_REGEX . . . . .	79
6.1.5.153 EDAM_SEARCH_SUGGESTIONS_MAX . . . . .	79
6.1.5.154 EDAM_SEARCH_SUGGESTIONS_PREFIX_LEN_MAX . . . . .	80
6.1.5.155 EDAM_SEARCH_SUGGESTIONS_PREFIX_LEN_MIN . . . . .	80
6.1.5.156 EDAM_SNIPPETS_NOTES_MAX . . . . .	80
6.1.5.157 EDAM_SOURCE_APPLICATION_ANDROID_SHARE_EXTENSION . . . . .	80
6.1.5.158 EDAM_SOURCE_APPLICATION_EN_SCANSNAP . . . . .	80
6.1.5.159 EDAM_SOURCE_APPLICATION_EWC . . . . .	80
6.1.5.160 EDAM_SOURCE_APPLICATION_IOS_SHARE_EXTENSION . . . . .	80
6.1.5.161 EDAM_SOURCE_APPLICATION_MOLESKINE . . . . .	80
6.1.5.162 EDAM_SOURCE_APPLICATION_POSTIT . . . . .	81
6.1.5.163 EDAM_SOURCE_APPLICATION_WEB_CLIPPER . . . . .	81
6.1.5.164 EDAM_SOURCE_OUTLOOK_CLIPPER . . . . .	81
6.1.5.165 EDAM_TAG_NAME_LEN_MAX . . . . .	81
6.1.5.166 EDAM_TAG_NAME_LEN_MIN . . . . .	81
6.1.5.167 EDAM_TAG_NAME_REGEX . . . . .	81
6.1.5.168 EDAM_TIMEZONE_LEN_MAX . . . . .	81
6.1.5.169 EDAM_TIMEZONE_LEN_MIN . . . . .	82
6.1.5.170 EDAM_TIMEZONE_REGEX . . . . .	82
6.1.5.171 EDAM_USER_LINKED_NOTEBOOK_MAX . . . . .	82
6.1.5.172 EDAM_USER_LINKED_NOTEBOOK_MAX_PREMIUM . . . . .	82
6.1.5.173 EDAM_USER_MAIL_LIMIT_DAILY_FREE . . . . .	82
6.1.5.174 EDAM_USER_MAIL_LIMIT_DAILY_PREMIUM . . . . .	82
6.1.5.175 EDAM_USER_NAME_LEN_MAX . . . . .	83
6.1.5.176 EDAM_USER_NAME_LEN_MIN . . . . .	83
6.1.5.177 EDAM_USER_NAME_REGEX . . . . .	83
6.1.5.178 EDAM_USER_NOTEBOOKS_MAX . . . . .	83

6.1.5.179 EDAM_USER_NOTES_MAX . . . . .	83
6.1.5.180 EDAM_USER_PASSWORD_LEN_MAX . . . . .	83
6.1.5.181 EDAM_USER_PASSWORD_LEN_MIN . . . . .	83
6.1.5.182 EDAM_USER_PASSWORD_REGEX . . . . .	83
6.1.5.183 EDAM_USER_PROFILE_PHOTO_MAX_BYTES . . . . .	84
6.1.5.184 EDAM_USER_RECENT_MAILED_ADDRESSES_MAX . . . . .	84
6.1.5.185 EDAM_USER_SAVED_SEARCHES_MAX . . . . .	84
6.1.5.186 EDAM_USER_TAGS_MAX . . . . .	84
6.1.5.187 EDAM_USER_UPLOAD_LIMIT_BUSINESS . . . . .	84
6.1.5.188 EDAM_USER_UPLOAD_LIMIT_BUSINESS_FIRST_MONTH . . . . .	84
6.1.5.189 EDAM_USER_UPLOAD_LIMIT_BUSINESS_NEXT_MONTH . . . . .	84
6.1.5.190 EDAM_USER_UPLOAD_LIMIT_BUSINESS_PER_USER . . . . .	85
6.1.5.191 EDAM_USER_UPLOAD_LIMIT_FREE . . . . .	85
6.1.5.192 EDAM_USER_UPLOAD_LIMIT_PLUS . . . . .	85
6.1.5.193 EDAM_USER_UPLOAD_LIMIT_PREMIUM . . . . .	85
6.1.5.194 EDAM_USER_UPLOAD_SURVEY_THRESHOLD . . . . .	85
6.1.5.195 EDAM_USER_USERNAME_LEN_MAX . . . . .	85
6.1.5.196 EDAM_USER_USERNAME_LEN_MIN . . . . .	85
6.1.5.197 EDAM_USER_USERNAME_REGEX . . . . .	86
6.1.5.198 EDAM_USER_WORKSPACES_MAX . . . . .	86
6.1.5.199 EDAM_VAT_REGEX . . . . .	86
6.1.5.200 EDAM_VERSION_MAJOR . . . . .	86
6.1.5.201 EDAM_VERSION_MINOR . . . . .	86
6.1.5.202 EDAM_WORKSPACE_DESCRIPTION_LEN_MAX . . . . .	86
6.1.5.203 EDAM_WORKSPACE_NAME_LEN_MAX . . . . .	86
6.1.5.204 EDAM_WORKSPACE_NAME_LEN_MIN . . . . .	87
6.1.5.205 EDAM_WORKSPACE_NAME_REGEX . . . . .	87
6.1.5.206 EverCloudExceptionData . . . . .	87

<b>7</b>	<b>Class Documentation</b>	<b>89</b>
7.1	qevercloud::Accounting Struct Reference	89
7.1.1	Detailed Description	90
7.1.2	Member Function Documentation	90
7.1.2.1	operator!=()	90
7.1.2.2	operator==()	90
7.1.2.3	print()	90
7.1.3	Member Data Documentation	90
7.1.3.1	availablePoints	91
7.1.3.2	businessId	91
7.1.3.3	businessName	91
7.1.3.4	businessRole	91
7.1.3.5	currency	91
7.1.3.6	lastFailedCharge	91
7.1.3.7	lastFailedChargeReason	91
7.1.3.8	lastRequestedCharge	91
7.1.3.9	lastSuccessfulCharge	92
7.1.3.10	localData	92
7.1.3.11	nextChargeDate	92
7.1.3.12	nextPaymentDue	92
7.1.3.13	premiumCommerceService	92
7.1.3.14	premiumLockUntil	92
7.1.3.15	premiumOrderNumber	92
7.1.3.16	premiumServiceSKU	92
7.1.3.17	premiumServiceStart	93
7.1.3.18	premiumServiceStatus	93
7.1.3.19	premiumSubscriptionNumber	93
7.1.3.20	unitDiscount	93
7.1.3.21	unitPrice	93
7.1.3.22	updated	93

7.1.3.23	uploadLimitEnd	93
7.1.3.24	uploadLimitNextMonth	94
7.2	qevercloud::AccountLimits Struct Reference	94
7.2.1	Detailed Description	94
7.2.2	Member Function Documentation	94
7.2.2.1	operator"!=( )	95
7.2.2.2	operator==( )	95
7.2.2.3	print()	95
7.2.3	Member Data Documentation	95
7.2.3.1	localData	95
7.2.3.2	noteResourceCountMax	95
7.2.3.3	noteSizeMax	95
7.2.3.4	noteTagCountMax	96
7.2.3.5	resourceSizeMax	96
7.2.3.6	uploadLimit	96
7.2.3.7	userLinkedNotebookMax	96
7.2.3.8	userMailLimitDaily	96
7.2.3.9	userNotebookCountMax	96
7.2.3.10	userNoteCountMax	96
7.2.3.11	userSavedSearchesMax	97
7.2.3.12	userTagCountMax	97
7.3	qevercloud::IDurableService::AsyncRequest Struct Reference	97
7.3.1	Constructor & Destructor Documentation	97
7.3.1.1	AsyncRequest()	97
7.3.2	Member Data Documentation	97
7.3.2.1	m_call	98
7.3.2.2	m_description	98
7.3.2.3	m_name	98
7.4	qevercloud::AsyncResult Class Reference	98
7.4.1	Detailed Description	99



7.4.2	Member Typedef Documentation . . . . .	99
7.4.2.1	ReadFunctionType . . . . .	99
7.4.3	Constructor & Destructor Documentation . . . . .	99
7.4.3.1	AsyncResult() [1/3] . . . . .	99
7.4.3.2	AsyncResult() [2/3] . . . . .	100
7.4.3.3	AsyncResult() [3/3] . . . . .	100
7.4.3.4	~AsyncResult() . . . . .	100
7.4.4	Member Function Documentation . . . . .	100
7.4.4.1	asIs() . . . . .	100
7.4.4.2	finished . . . . .	100
7.4.4.3	waitForFinished() . . . . .	101
7.4.5	Friends And Related Function Documentation . . . . .	101
7.4.5.1	DurableService . . . . .	101
7.5	qevercloud::AuthenticationResult Struct Reference . . . . .	101
7.5.1	Detailed Description . . . . .	102
7.5.2	Member Function Documentation . . . . .	102
7.5.2.1	operator!=(()) . . . . .	102
7.5.2.2	operator==(()) . . . . .	102
7.5.2.3	print() . . . . .	102
7.5.3	Member Data Documentation . . . . .	103
7.5.3.1	authenticationToken . . . . .	103
7.5.3.2	currentTime . . . . .	103
7.5.3.3	expiration . . . . .	103
7.5.3.4	localData . . . . .	103
7.5.3.5	noteStoreUrl . . . . .	103
7.5.3.6	publicUserInfo . . . . .	103
7.5.3.7	secondFactorDeliveryHint . . . . .	104
7.5.3.8	secondFactorRequired . . . . .	104
7.5.3.9	urls . . . . .	104
7.5.3.10	user . . . . .	104

7.5.3.11	webApiUrlPrefix	104
7.6	qevercloud::BootstrapInfo Struct Reference	104
7.6.1	Detailed Description	105
7.6.2	Member Function Documentation	105
7.6.2.1	operator!=(())	105
7.6.2.2	operator==(())	105
7.6.2.3	print()	105
7.6.3	Member Data Documentation	105
7.6.3.1	localData	106
7.6.3.2	profiles	106
7.7	qevercloud::BootstrapProfile Struct Reference	106
7.7.1	Detailed Description	106
7.7.2	Member Function Documentation	106
7.7.2.1	operator!=(())	107
7.7.2.2	operator==(())	107
7.7.2.3	print()	107
7.7.3	Member Data Documentation	107
7.7.3.1	localData	107
7.7.3.2	name	107
7.7.3.3	settings	107
7.8	qevercloud::BootstrapSettings Struct Reference	108
7.8.1	Detailed Description	108
7.8.2	Member Function Documentation	108
7.8.2.1	operator!=(())	108
7.8.2.2	operator==(())	109
7.8.2.3	print()	109
7.8.3	Member Data Documentation	109
7.8.3.1	accountEmailDomain	109
7.8.3.2	enableFacebookSharing	109
7.8.3.3	enableGiftSubscriptions	109

7.8.3.4	<a href="#">enableGoogle</a>	109
7.8.3.5	<a href="#">enableLinkedInSharing</a>	110
7.8.3.6	<a href="#">enablePublicNotebooks</a>	110
7.8.3.7	<a href="#">enableSharedNotebooks</a>	110
7.8.3.8	<a href="#">enableSingleNoteSharing</a>	110
7.8.3.9	<a href="#">enableSponsoredAccounts</a>	110
7.8.3.10	<a href="#">enableSupportTickets</a>	110
7.8.3.11	<a href="#">enableTwitterSharing</a>	110
7.8.3.12	<a href="#">localData</a>	110
7.8.3.13	<a href="#">marketingUrl</a>	111
7.8.3.14	<a href="#">serviceHost</a>	111
7.8.3.15	<a href="#">supportUrl</a>	111
7.9	<a href="#">qevercloud::BusinessInvitation Struct Reference</a>	111
7.9.1	<a href="#">Detailed Description</a>	112
7.9.2	<a href="#">Member Function Documentation</a>	112
7.9.2.1	<a href="#">operator"!=( )</a>	112
7.9.2.2	<a href="#">operator==( )</a>	112
7.9.2.3	<a href="#">print( )</a>	112
7.9.3	<a href="#">Member Data Documentation</a>	112
7.9.3.1	<a href="#">businessId</a>	112
7.9.3.2	<a href="#">created</a>	112
7.9.3.3	<a href="#">email</a>	113
7.9.3.4	<a href="#">fromWorkChat</a>	113
7.9.3.5	<a href="#">localData</a>	113
7.9.3.6	<a href="#">mostRecentReminder</a>	113
7.9.3.7	<a href="#">requesterId</a>	113
7.9.3.8	<a href="#">role</a>	113
7.9.3.9	<a href="#">status</a>	113
7.10	<a href="#">qevercloud::BusinessNotebook Struct Reference</a>	114
7.10.1	<a href="#">Detailed Description</a>	114

7.10.2	Member Function Documentation	114
7.10.2.1	operator"!=(())	114
7.10.2.2	operator==(())	114
7.10.2.3	print()	115
7.10.3	Member Data Documentation	115
7.10.3.1	localData	115
7.10.3.2	notebookDescription	115
7.10.3.3	privilege	115
7.10.3.4	recommended	115
7.11	qevercloud::BusinessUserAttributes Struct Reference	115
7.11.1	Detailed Description	116
7.11.2	Member Function Documentation	116
7.11.2.1	operator"!=(())	116
7.11.2.2	operator==(())	116
7.11.2.3	print()	116
7.11.3	Member Data Documentation	117
7.11.3.1	companyStartDate	117
7.11.3.2	department	117
7.11.3.3	linkedinProfileUrl	117
7.11.3.4	localData	117
7.11.3.5	location	117
7.11.3.6	mobilePhone	117
7.11.3.7	title	117
7.11.3.8	workPhone	118
7.12	qevercloud::BusinessUserInfo Struct Reference	118
7.12.1	Detailed Description	118
7.12.2	Member Function Documentation	118
7.12.2.1	operator"!=(())	118
7.12.2.2	operator==(())	119
7.12.2.3	print()	119

7.12.3	Member Data Documentation . . . . .	119
7.12.3.1	businessId . . . . .	119
7.12.3.2	businessName . . . . .	119
7.12.3.3	email . . . . .	119
7.12.3.4	localData . . . . .	119
7.12.3.5	role . . . . .	120
7.12.3.6	updated . . . . .	120
7.13	qevercloud::CanMoveToContainerRestrictions Struct Reference . . . . .	120
7.13.1	Detailed Description . . . . .	120
7.13.2	Member Function Documentation . . . . .	120
7.13.2.1	operator"!=( ) . . . . .	121
7.13.2.2	operator==( ) . . . . .	121
7.13.2.3	print() . . . . .	121
7.13.3	Member Data Documentation . . . . .	121
7.13.3.1	canMoveToContainer . . . . .	121
7.13.3.2	localData . . . . .	121
7.14	qevercloud::Contact Struct Reference . . . . .	121
7.14.1	Detailed Description . . . . .	122
7.14.2	Member Function Documentation . . . . .	122
7.14.2.1	operator"!=( ) . . . . .	122
7.14.2.2	operator==( ) . . . . .	122
7.14.2.3	print() . . . . .	122
7.14.3	Member Data Documentation . . . . .	123
7.14.3.1	id . . . . .	123
7.14.3.2	localData . . . . .	123
7.14.3.3	messagingPermit . . . . .	123
7.14.3.4	messagingPermitExpires . . . . .	123
7.14.3.5	name . . . . .	123
7.14.3.6	photoLastUpdated . . . . .	123
7.14.3.7	photoUrl . . . . .	124

7.14.3.8	type	124
7.15	qevercloud::CreateOrUpdateNotebookSharesResult Struct Reference	124
7.15.1	Detailed Description	124
7.15.2	Member Function Documentation	125
7.15.2.1	operator!=(())	125
7.15.2.2	operator==(())	125
7.15.2.3	print()	125
7.15.3	Member Data Documentation	125
7.15.3.1	localData	125
7.15.3.2	matchingShares	125
7.15.3.3	updateSequenceNum	126
7.15.4	Property Documentation	126
7.15.4.1	QList	126
7.16	qevercloud::Data Struct Reference	126
7.16.1	Detailed Description	126
7.16.2	Member Function Documentation	127
7.16.2.1	operator!=(())	127
7.16.2.2	operator==(())	127
7.16.2.3	print()	127
7.16.3	Member Data Documentation	127
7.16.3.1	body	127
7.16.3.2	bodyHash	127
7.16.3.3	localData	128
7.16.3.4	size	128
7.17	qevercloud::EDAMInvalidContactsException Class Reference	128
7.17.1	Detailed Description	129
7.17.2	Constructor & Destructor Documentation	129
7.17.2.1	EDAMInvalidContactsException() [1/2]	129
7.17.2.2	~EDAMInvalidContactsException()	129
7.17.2.3	EDAMInvalidContactsException() [2/2]	129

7.17.3 Member Function Documentation . . . . .	129
7.17.3.1 exceptionData() . . . . .	130
7.17.3.2 operator!=(()) . . . . .	130
7.17.3.3 operator==(()) . . . . .	130
7.17.3.4 print() . . . . .	130
7.17.3.5 what() . . . . .	130
7.17.4 Member Data Documentation . . . . .	130
7.17.4.1 contacts . . . . .	130
7.17.4.2 parameter . . . . .	131
7.17.4.3 reasons . . . . .	131
7.17.5 Property Documentation . . . . .	131
7.17.5.1 QList . . . . .	131
7.18 qevercloud::EDAMInvalidContactsExceptionData Class Reference . . . . .	131
7.18.1 Detailed Description . . . . .	132
7.18.2 Constructor & Destructor Documentation . . . . .	132
7.18.2.1 EDAMInvalidContactsExceptionData() . . . . .	132
7.18.3 Member Function Documentation . . . . .	132
7.18.3.1 throwException() . . . . .	132
7.18.4 Member Data Documentation . . . . .	132
7.18.4.1 m_contacts . . . . .	132
7.18.4.2 m_parameter . . . . .	132
7.18.4.3 m_reasons . . . . .	133
7.19 qevercloud::EDAMNotFoundException Class Reference . . . . .	133
7.19.1 Detailed Description . . . . .	133
7.19.2 Constructor & Destructor Documentation . . . . .	134
7.19.2.1 EDAMNotFoundException() [1/2] . . . . .	134
7.19.2.2 ~EDAMNotFoundException() . . . . .	134
7.19.2.3 EDAMNotFoundException() [2/2] . . . . .	134
7.19.3 Member Function Documentation . . . . .	134
7.19.3.1 exceptionData() . . . . .	134

7.19.3.2	operator"!=()	134
7.19.3.3	operator==(())	135
7.19.3.4	print()	135
7.19.3.5	what()	135
7.19.4	Member Data Documentation	135
7.19.4.1	identifier	135
7.19.4.2	key	135
7.20	qevercloud::EDAMNotFoundExceptionData Class Reference	135
7.20.1	Detailed Description	136
7.20.2	Constructor & Destructor Documentation	136
7.20.2.1	EDAMNotFoundExceptionData()	136
7.20.3	Member Function Documentation	136
7.20.3.1	throwException()	136
7.20.4	Member Data Documentation	136
7.20.4.1	m_identifier	137
7.20.4.2	m_key	137
7.21	qevercloud::EDAMSystemException Class Reference	137
7.21.1	Detailed Description	138
7.21.2	Constructor & Destructor Documentation	138
7.21.2.1	EDAMSystemException() [1/2]	138
7.21.2.2	~EDAMSystemException()	138
7.21.2.3	EDAMSystemException() [2/2]	138
7.21.3	Member Function Documentation	138
7.21.3.1	exceptionData()	139
7.21.3.2	operator"!=()	139
7.21.3.3	operator==(())	139
7.21.3.4	print()	139
7.21.3.5	what()	139
7.21.4	Member Data Documentation	139
7.21.4.1	errorCode	140



7.21.4.2	message	140
7.21.4.3	rateLimitDuration	140
7.22	qevercloud::EDAMSystemExceptionAuthExpired Class Reference	140
7.22.1	Detailed Description	140
7.22.2	Member Function Documentation	141
7.22.2.1	exceptionData()	141
7.23	qevercloud::EDAMSystemExceptionAuthExpiredData Class Reference	141
7.23.1	Detailed Description	141
7.23.2	Constructor & Destructor Documentation	141
7.23.2.1	EDAMSystemExceptionAuthExpiredData()	142
7.23.3	Member Function Documentation	142
7.23.3.1	throwException()	142
7.24	qevercloud::EDAMSystemExceptionData Class Reference	142
7.24.1	Detailed Description	143
7.24.2	Constructor & Destructor Documentation	143
7.24.2.1	EDAMSystemExceptionData()	143
7.24.3	Member Function Documentation	143
7.24.3.1	throwException()	143
7.24.4	Member Data Documentation	143
7.24.4.1	m_errorCode	143
7.24.4.2	m_message	144
7.24.4.3	m_rateLimitDuration	144
7.25	qevercloud::EDAMSystemExceptionRateLimitReached Class Reference	144
7.25.1	Detailed Description	144
7.25.2	Member Function Documentation	144
7.25.2.1	exceptionData()	144
7.26	qevercloud::EDAMSystemExceptionRateLimitReachedData Class Reference	145
7.26.1	Detailed Description	145
7.26.2	Constructor & Destructor Documentation	145
7.26.2.1	EDAMSystemExceptionRateLimitReachedData()	145

7.26.3	Member Function Documentation	145
7.26.3.1	throwException()	146
7.27	qevercloud::EDAMUserException Class Reference	146
7.27.1	Detailed Description	147
7.27.2	Constructor & Destructor Documentation	147
7.27.2.1	EDAMUserException() [1/2]	147
7.27.2.2	~EDAMUserException()	147
7.27.2.3	EDAMUserException() [2/2]	147
7.27.3	Member Function Documentation	147
7.27.3.1	exceptionData()	148
7.27.3.2	operator!=(())	148
7.27.3.3	operator==(())	148
7.27.3.4	print()	148
7.27.3.5	what()	148
7.27.4	Member Data Documentation	148
7.27.4.1	errorCode	148
7.27.4.2	parameter	149
7.28	qevercloud::EDAMUserExceptionData Class Reference	149
7.28.1	Detailed Description	149
7.28.2	Constructor & Destructor Documentation	149
7.28.2.1	EDAMUserExceptionData()	149
7.28.3	Member Function Documentation	150
7.28.3.1	throwException()	150
7.28.4	Member Data Documentation	150
7.28.4.1	m_errorCode	150
7.28.4.2	m_parameter	150
7.29	qevercloud::EventLoopFinisher Class Reference	150
7.29.1	Constructor & Destructor Documentation	151
7.29.1.1	EventLoopFinisher()	151
7.29.1.2	~EventLoopFinisher()	151

7.29.2	Member Function Documentation	151
7.29.2.1	stopEventLoop	151
7.30	qevercloud::EverCloudException Class Reference	151
7.30.1	Detailed Description	152
7.30.2	Constructor & Destructor Documentation	152
7.30.2.1	EverCloudException() [1/4]	152
7.30.2.2	EverCloudException() [2/4]	152
7.30.2.3	EverCloudException() [3/4]	152
7.30.2.4	EverCloudException() [4/4]	152
7.30.2.5	~EverCloudException()	153
7.30.3	Member Function Documentation	153
7.30.3.1	exceptionData()	153
7.30.3.2	what()	153
7.30.4	Member Data Documentation	153
7.30.4.1	m_error	153
7.31	qevercloud::EverCloudExceptionData Class Reference	153
7.31.1	Detailed Description	154
7.31.2	Constructor & Destructor Documentation	155
7.31.2.1	EverCloudExceptionData()	155
7.31.3	Member Function Documentation	155
7.31.3.1	throwException()	155
7.31.4	Member Data Documentation	155
7.31.4.1	errorMessage	155
7.32	qevercloud::EverCloudLocalData Class Reference	156
7.32.1	Detailed Description	157
7.32.2	Member Typedef Documentation	157
7.32.2.1	Dict	157
7.32.3	Constructor & Destructor Documentation	157
7.32.3.1	EverCloudLocalData()	157
7.32.3.2	~EverCloudLocalData()	157

7.32.4	Member Function Documentation	157
7.32.4.1	operator!=()	157
7.32.4.2	operator==(())	158
7.32.4.3	print()	158
7.32.5	Member Data Documentation	158
7.32.5.1	dict	158
7.32.5.2	dirty	158
7.32.5.3	favorited	158
7.32.5.4	id	159
7.32.5.5	local	159
7.32.6	Property Documentation	159
7.32.6.1	dict	159
7.33	qevercloud::EvernoteException Class Reference	159
7.33.1	Detailed Description	160
7.33.2	Constructor & Destructor Documentation	160
7.33.2.1	EvernoteException() [1/4]	160
7.33.2.2	EvernoteException() [2/4]	160
7.33.2.3	EvernoteException() [3/4]	160
7.33.2.4	EvernoteException() [4/4]	160
7.33.3	Member Function Documentation	160
7.33.3.1	exceptionData()	161
7.34	qevercloud::EvernoteExceptionData Class Reference	161
7.34.1	Detailed Description	161
7.34.2	Constructor & Destructor Documentation	161
7.34.2.1	EvernoteExceptionData()	161
7.34.3	Member Function Documentation	161
7.34.3.1	throwException()	162
7.35	qevercloud::EvernoteOAuthDialog Class Reference	162
7.35.1	Detailed Description	163
7.35.2	Member Typedef Documentation	163

7.35.2.1	OAuthResult	163
7.35.3	Constructor & Destructor Documentation	163
7.35.3.1	EvernoteOAuthDialog()	163
7.35.3.2	~EvernoteOAuthDialog()	164
7.35.4	Member Function Documentation	164
7.35.4.1	exec()	164
7.35.4.2	isSucceeded()	164
7.35.4.3	oauthError()	164
7.35.4.4	oauthResult()	165
7.35.4.5	open()	165
7.35.4.6	setWebViewSizeHint()	165
7.36	qevercloud::EvernoteOAuthWebView Class Reference	165
7.36.1	Detailed Description	166
7.36.2	Constructor & Destructor Documentation	166
7.36.2.1	EvernoteOAuthWebView()	166
7.36.3	Member Function Documentation	166
7.36.3.1	authenticate()	166
7.36.3.2	authenticationFailed	167
7.36.3.3	authenticationFinished	167
7.36.3.4	authenticationSucceeded	167
7.36.3.5	isSucceeded()	167
7.36.3.6	oauthError()	167
7.36.3.7	oauthResult()	168
7.36.3.8	setSizeHint()	168
7.36.3.9	sizeHint()	168
7.37	qevercloud::Identity Struct Reference	168
7.37.1	Detailed Description	169
7.37.2	Member Function Documentation	169
7.37.2.1	operator!=()	169
7.37.2.2	operator==()	169

7.37.2.3	<a href="#">print()</a>	169
7.37.3	<a href="#">Member Data Documentation</a>	169
7.37.3.1	<a href="#">blocked</a>	169
7.37.3.2	<a href="#">contact</a>	169
7.37.3.3	<a href="#">deactivated</a>	170
7.37.3.4	<a href="#">eventId</a>	170
7.37.3.5	<a href="#">id</a>	170
7.37.3.6	<a href="#">localData</a>	170
7.37.3.7	<a href="#">sameBusiness</a>	170
7.37.3.8	<a href="#">userConnected</a>	170
7.37.3.9	<a href="#">userId</a>	171
7.38	<a href="#">qevercloud::IDurableService Class Reference</a>	171
7.38.1	<a href="#">Member Typedef Documentation</a>	171
7.38.1.1	<a href="#">AsyncServiceCall</a>	171
7.38.1.2	<a href="#">SyncResult</a>	171
7.38.1.3	<a href="#">SyncServiceCall</a>	172
7.38.2	<a href="#">Member Function Documentation</a>	172
7.38.2.1	<a href="#">executeAsyncRequest()</a>	172
7.38.2.2	<a href="#">executeSyncRequest()</a>	172
7.39	<a href="#">qevercloud::ILogger Class Reference</a>	172
7.39.1	<a href="#">Member Function Documentation</a>	172
7.39.1.1	<a href="#">level()</a>	172
7.39.1.2	<a href="#">log()</a>	173
7.39.1.3	<a href="#">setLevel()</a>	173
7.39.1.4	<a href="#">shouldLog()</a>	173
7.40	<a href="#">qevercloud::InkNotelImageDownloader Class Reference</a>	173
7.40.1	<a href="#">Detailed Description</a>	174
7.40.2	<a href="#">Constructor &amp; Destructor Documentation</a>	174
7.40.2.1	<a href="#">InkNotelImageDownloader() [1/2]</a>	174
7.40.2.2	<a href="#">InkNotelImageDownloader() [2/2]</a>	174

7.40.2.3	<a href="#">~InkNoteImageDownloader()</a>	175
7.40.3	Member Function Documentation	175
7.40.3.1	<a href="#">download()</a>	175
7.40.3.2	<a href="#">setAuthenticationToken()</a>	175
7.40.3.3	<a href="#">setHeight()</a>	176
7.40.3.4	<a href="#">setHost()</a>	176
7.40.3.5	<a href="#">setShardId()</a>	176
7.40.3.6	<a href="#">setWidth()</a>	176
7.41	<a href="#">qevercloud::INoteStore Class Reference</a>	177
7.41.1	Detailed Description	181
7.41.2	Constructor & Destructor Documentation	181
7.41.2.1	<a href="#">INoteStore()</a>	181
7.41.3	Member Function Documentation	181
7.41.3.1	<a href="#">authenticateToSharedNote()</a>	181
7.41.3.2	<a href="#">authenticateToSharedNoteAsync()</a>	182
7.41.3.3	<a href="#">authenticateToSharedNotebook()</a>	182
7.41.3.4	<a href="#">authenticateToSharedNotebookAsync()</a>	183
7.41.3.5	<a href="#">copyNote()</a>	183
7.41.3.6	<a href="#">copyNoteAsync()</a>	184
7.41.3.7	<a href="#">createLinkedNotebook()</a>	184
7.41.3.8	<a href="#">createLinkedNotebookAsync()</a>	185
7.41.3.9	<a href="#">createNote()</a>	185
7.41.3.10	<a href="#">createNoteAsync()</a>	187
7.41.3.11	<a href="#">createNotebook()</a>	187
7.41.3.12	<a href="#">createNotebookAsync()</a>	188
7.41.3.13	<a href="#">createOrUpdateNotebookShares()</a>	188
7.41.3.14	<a href="#">createOrUpdateNotebookSharesAsync()</a>	189
7.41.3.15	<a href="#">createSearch()</a>	189
7.41.3.16	<a href="#">createSearchAsync()</a>	190
7.41.3.17	<a href="#">createTag()</a>	190

7.41.3.18 createTagAsync()	191
7.41.3.19 deleteNote()	191
7.41.3.20 deleteNoteAsync()	192
7.41.3.21 emailNote()	192
7.41.3.22 emailNoteAsync()	193
7.41.3.23 expungeLinkedNotebook()	193
7.41.3.24 expungeLinkedNotebookAsync()	194
7.41.3.25 expungeNote()	194
7.41.3.26 expungeNoteAsync()	194
7.41.3.27 expungeNotebook()	195
7.41.3.28 expungeNotebookAsync()	195
7.41.3.29 expungeSearch()	195
7.41.3.30 expungeSearchAsync()	196
7.41.3.31 expungeTag()	196
7.41.3.32 expungeTagAsync()	197
7.41.3.33 findNoteCounts()	197
7.41.3.34 findNoteCountsAsync()	198
7.41.3.35 findNoteOffset()	198
7.41.3.36 findNoteOffsetAsync()	199
7.41.3.37 findNotesMetadata()	199
7.41.3.38 findNotesMetadataAsync()	200
7.41.3.39 findRelated()	200
7.41.3.40 findRelatedAsync()	201
7.41.3.41 getDefaultNotebook()	202
7.41.3.42 getDefaultNotebookAsync()	202
7.41.3.43 getFilteredSyncChunk()	202
7.41.3.44 getFilteredSyncChunkAsync()	203
7.41.3.45 getLinkedNotebookSyncChunk()	203
7.41.3.46 getLinkedNotebookSyncChunkAsync()	204
7.41.3.47 getLinkedNotebookSyncState()	204



7.41.3.48 getLinkedNotebookSyncStateAsync()	205
7.41.3.49 getNote()	205
7.41.3.50 getNoteApplicationData()	205
7.41.3.51 getNoteApplicationDataAsync()	206
7.41.3.52 getNoteApplicationDataEntry()	206
7.41.3.53 getNoteApplicationDataEntryAsync()	206
7.41.3.54 getNoteAsync()	206
7.41.3.55 getNotebook()	207
7.41.3.56 getNotebookAsync()	208
7.41.3.57 getNotebookShares()	208
7.41.3.58 getNotebookSharesAsync()	208
7.41.3.59 getNoteContent()	208
7.41.3.60 getNoteContentAsync()	209
7.41.3.61 getNoteSearchText()	209
7.41.3.62 getNoteSearchTextAsync()	210
7.41.3.63 getNoteTagNames()	210
7.41.3.64 getNoteTagNamesAsync()	210
7.41.3.65 getNoteVersion()	211
7.41.3.66 getNoteVersionAsync()	211
7.41.3.67 getNoteWithResultSpec()	212
7.41.3.68 getNoteWithResultSpecAsync()	212
7.41.3.69 getPublicNotebook()	212
7.41.3.70 getPublicNotebookAsync()	213
7.41.3.71 getResource()	213
7.41.3.72 getResourceAlternateData()	214
7.41.3.73 getResourceAlternateDataAsync()	215
7.41.3.74 getResourceApplicationData()	215
7.41.3.75 getResourceApplicationDataAsync()	215
7.41.3.76 getResourceApplicationDataEntry()	215
7.41.3.77 getResourceApplicationDataEntryAsync()	215

7.41.3.78 getResourceAsync()	216
7.41.3.79 getResourceAttributes()	216
7.41.3.80 getResourceAttributesAsync()	216
7.41.3.81 getResourceByHash()	217
7.41.3.82 getResourceByHashAsync()	217
7.41.3.83 getResourceData()	218
7.41.3.84 getResourceDataAsync()	218
7.41.3.85 getResourceRecognition()	218
7.41.3.86 getResourceRecognitionAsync()	219
7.41.3.87 getResourceSearchText()	219
7.41.3.88 getResourceSearchTextAsync()	220
7.41.3.89 getSearch()	220
7.41.3.90 getSearchAsync()	220
7.41.3.91 getSharedNotebookByAuth()	220
7.41.3.92 getSharedNotebookByAuthAsync()	221
7.41.3.93 getSyncState()	221
7.41.3.94 getSyncStateAsync()	221
7.41.3.95 getTag()	221
7.41.3.96 getTagAsync()	222
7.41.3.97 listAccessibleBusinessNotebooks()	222
7.41.3.98 listAccessibleBusinessNotebooksAsync()	223
7.41.3.99 listLinkedNotebooks()	223
7.41.3.100 listLinkedNotebooksAsync()	223
7.41.3.101 listNotebooks()	223
7.41.3.102 listNotebooksAsync()	223
7.41.3.103 listNoteVersions()	223
7.41.3.104 listNoteVersionsAsync()	224
7.41.3.105 listSearches()	224
7.41.3.106 listSearchesAsync()	224
7.41.3.107 listSharedNotebooks()	224

7.41.3.108	listSharedNotebooksAsync()	225
7.41.3.109	listTags()	225
7.41.3.110	listTagsAsync()	225
7.41.3.111	listTagsByNotebook()	225
7.41.3.112	listTagsByNotebookAsync()	225
7.41.3.113	manageNotebookShares()	226
7.41.3.114	manageNotebookSharesAsync()	226
7.41.3.115	noteStoreUrl()	226
7.41.3.116	setNoteApplicationDataEntry()	226
7.41.3.117	setNoteApplicationDataEntryAsync()	227
7.41.3.118	setNotebookRecipientSettings()	227
7.41.3.119	setNotebookRecipientSettingsAsync()	228
7.41.3.120	setNoteStoreUrl()	228
7.41.3.121	setResourceApplicationDataEntry()	228
7.41.3.122	setResourceApplicationDataEntryAsync()	228
7.41.3.123	shareNote()	229
7.41.3.124	shareNoteAsync()	229
7.41.3.125	shareNotebook()	229
7.41.3.126	shareNotebookAsync()	231
7.41.3.127	stopSharingNote()	231
7.41.3.128	stopSharingNoteAsync()	232
7.41.3.129	unsetNoteApplicationDataEntry()	232
7.41.3.130	unsetNoteApplicationDataEntryAsync()	232
7.41.3.131	unsetResourceApplicationDataEntry()	232
7.41.3.132	unsetResourceApplicationDataEntryAsync()	233
7.41.3.133	untagAll()	233
7.41.3.134	untagAllAsync()	233
7.41.3.135	updateLinkedNotebook()	233
7.41.3.136	updateLinkedNotebookAsync()	234
7.41.3.137	updateNote()	234

7.41.3.138	<a href="#">updateNoteAsync()</a>	235
7.41.3.139	<a href="#">updateNotebook()</a>	236
7.41.3.140	<a href="#">updateNotebookAsync()</a>	236
7.41.3.141	<a href="#">updateNotelfUsnMatches()</a>	237
7.41.3.142	<a href="#">updateNotelfUsnMatchesAsync()</a>	237
7.41.3.143	<a href="#">updateResource()</a>	237
7.41.3.144	<a href="#">updateResourceAsync()</a>	238
7.41.3.145	<a href="#">updateSearch()</a>	238
7.41.3.146	<a href="#">updateSearchAsync()</a>	239
7.41.3.147	<a href="#">updateSharedNotebook()</a>	239
7.41.3.148	<a href="#">updateSharedNotebookAsync()</a>	239
7.41.3.149	<a href="#">updateTag()</a>	240
7.41.3.150	<a href="#">updateTagAsync()</a>	241
7.42	<a href="#">qevercloud::InvitationShareRelationship Struct Reference</a>	241
7.42.1	<a href="#">Detailed Description</a>	242
7.42.2	<a href="#">Member Function Documentation</a>	242
7.42.2.1	<a href="#">operator"!=( )</a>	242
7.42.2.2	<a href="#">operator==( )</a>	242
7.42.2.3	<a href="#">print()</a>	242
7.42.3	<a href="#">Member Data Documentation</a>	242
7.42.3.1	<a href="#">displayName</a>	243
7.42.3.2	<a href="#">localData</a>	243
7.42.3.3	<a href="#">privilege</a>	243
7.42.3.4	<a href="#">recipientUserIdentity</a>	243
7.42.3.5	<a href="#">sharerUserId</a>	243
7.43	<a href="#">qevercloud::IRequestContext Class Reference</a>	243
7.43.1	<a href="#">Detailed Description</a>	244
7.43.2	<a href="#">Constructor &amp; Destructor Documentation</a>	244
7.43.2.1	<a href="#">~IRequestContext()</a>	244
7.43.3	<a href="#">Member Function Documentation</a>	244

7.43.3.1	<a href="#">authenticationToken()</a>	244
7.43.3.2	<a href="#">clone()</a>	244
7.43.3.3	<a href="#">cookies()</a>	244
7.43.3.4	<a href="#">increaseRequestTimeoutExponentially()</a>	244
7.43.3.5	<a href="#">maxRequestRetryCount()</a>	245
7.43.3.6	<a href="#">maxRequestTimeout()</a>	245
7.43.3.7	<a href="#">requestId()</a>	245
7.43.3.8	<a href="#">requestTimeout()</a>	245
7.43.4	<a href="#">Friends And Related Function Documentation</a>	245
7.43.4.1	<a href="#">operator&lt;&lt; [1/2]</a>	245
7.43.4.2	<a href="#">operator&lt;&lt; [2/2]</a>	245
7.44	<a href="#">qevercloud::IRetryPolicy Struct Reference</a>	245
7.44.1	<a href="#">Member Function Documentation</a>	246
7.44.1.1	<a href="#">shouldRetry()</a>	246
7.45	<a href="#">qevercloud::QAssociativeContainerReferenceWrapper&lt; Container &gt;::iterator Struct Reference</a>	246
7.45.1	<a href="#">Constructor &amp; Destructor Documentation</a>	246
7.45.1.1	<a href="#">iterator()</a>	246
7.45.2	<a href="#">Member Function Documentation</a>	246
7.45.2.1	<a href="#">operator!=(())</a>	247
7.45.2.2	<a href="#">operator*()</a>	247
7.45.2.3	<a href="#">operator++()</a>	247
7.45.3	<a href="#">Member Data Documentation</a>	247
7.45.3.1	<a href="#">m_iterator</a>	247
7.46	<a href="#">qevercloud::QAssociativeContainerConstReferenceWrapper&lt; Container &gt;::iterator Struct Reference</a>	247
7.46.1	<a href="#">Constructor &amp; Destructor Documentation</a>	248
7.46.1.1	<a href="#">iterator()</a>	248
7.46.2	<a href="#">Member Function Documentation</a>	248
7.46.2.1	<a href="#">operator!=(())</a>	248
7.46.2.2	<a href="#">operator*()</a>	248
7.46.2.3	<a href="#">operator++()</a>	248

7.46.3	Member Data Documentation . . . . .	248
7.46.3.1	m_iterator . . . . .	249
7.47	qevercloud::IUserStore Class Reference . . . . .	249
7.47.1	Detailed Description . . . . .	250
7.47.2	Constructor & Destructor Documentation . . . . .	250
7.47.2.1	IUserStore() . . . . .	250
7.47.3	Member Function Documentation . . . . .	250
7.47.3.1	authenticateLongSession() . . . . .	251
7.47.3.2	authenticateLongSessionAsync() . . . . .	252
7.47.3.3	authenticateToBusiness() . . . . .	252
7.47.3.4	authenticateToBusinessAsync() . . . . .	253
7.47.3.5	checkVersion() . . . . .	253
7.47.3.6	checkVersionAsync() . . . . .	254
7.47.3.7	completeTwoFactorAuthentication() . . . . .	254
7.47.3.8	completeTwoFactorAuthenticationAsync() . . . . .	255
7.47.3.9	getAccountLimits() . . . . .	255
7.47.3.10	getAccountLimitsAsync() . . . . .	256
7.47.3.11	getBootstrapInfo() . . . . .	256
7.47.3.12	getBootstrapInfoAsync() . . . . .	256
7.47.3.13	getPublicUserInfo() . . . . .	256
7.47.3.14	getPublicUserInfoAsync() . . . . .	257
7.47.3.15	getUser() . . . . .	257
7.47.3.16	getUserAsync() . . . . .	257
7.47.3.17	getUserUrls() . . . . .	257
7.47.3.18	getUserUrlsAsync() . . . . .	257
7.47.3.19	inviteToBusiness() . . . . .	258
7.47.3.20	inviteToBusinessAsync() . . . . .	258
7.47.3.21	listBusinessInvitations() . . . . .	259
7.47.3.22	listBusinessInvitationsAsync() . . . . .	259
7.47.3.23	listBusinessUsers() . . . . .	259

7.47.3.24	<a href="#">listBusinessUsersAsync()</a>	260
7.47.3.25	<a href="#">removeFromBusiness()</a>	260
7.47.3.26	<a href="#">removeFromBusinessAsync()</a>	260
7.47.3.27	<a href="#">revokeLongSession()</a>	261
7.47.3.28	<a href="#">revokeLongSessionAsync()</a>	261
7.47.3.29	<a href="#">setUserStoreUrl()</a>	261
7.47.3.30	<a href="#">updateBusinessUserIdentifier()</a>	261
7.47.3.31	<a href="#">updateBusinessUserIdentifierAsync()</a>	262
7.47.3.32	<a href="#">userStoreUrl()</a>	263
7.48	<a href="#">qevercloud::LazyMap Struct Reference</a>	263
7.48.1	<a href="#">Detailed Description</a>	263
7.48.2	<a href="#">Member Typedef Documentation</a>	264
7.48.2.1	<a href="#">FullMap</a>	264
7.48.3	<a href="#">Member Function Documentation</a>	264
7.48.3.1	<a href="#">operator"!=(())</a>	264
7.48.3.2	<a href="#">operator==(())</a>	264
7.48.3.3	<a href="#">print()</a>	264
7.48.4	<a href="#">Member Data Documentation</a>	264
7.48.4.1	<a href="#">fullMap</a>	264
7.48.4.2	<a href="#">keysOnly</a>	265
7.48.4.3	<a href="#">localData</a>	265
7.48.5	<a href="#">Property Documentation</a>	265
7.48.5.1	<a href="#">fullMap</a>	265
7.48.5.2	<a href="#">QSet</a>	265
7.49	<a href="#">qevercloud::LinkedNotebook Struct Reference</a>	265
7.49.1	<a href="#">Detailed Description</a>	266
7.49.2	<a href="#">Member Function Documentation</a>	266
7.49.2.1	<a href="#">operator"!=(())</a>	266
7.49.2.2	<a href="#">operator==(())</a>	266
7.49.2.3	<a href="#">print()</a>	266

7.49.3	Member Data Documentation . . . . .	266
7.49.3.1	businessId . . . . .	267
7.49.3.2	guid . . . . .	267
7.49.3.3	localData . . . . .	267
7.49.3.4	noteStoreUrl . . . . .	267
7.49.3.5	shardId . . . . .	267
7.49.3.6	sharedNotebookGlobalId . . . . .	267
7.49.3.7	shareName . . . . .	268
7.49.3.8	stack . . . . .	268
7.49.3.9	updateSequenceNum . . . . .	268
7.49.3.10	uri . . . . .	268
7.49.3.11	username . . . . .	268
7.49.3.12	webApiUrlPrefix . . . . .	268
7.50	qevercloud::ManageNotebookSharesError Struct Reference . . . . .	269
7.50.1	Detailed Description . . . . .	269
7.50.2	Member Function Documentation . . . . .	269
7.50.2.1	operator"!=( ) . . . . .	269
7.50.2.2	operator==( ) . . . . .	270
7.50.2.3	print() . . . . .	270
7.50.3	Member Data Documentation . . . . .	270
7.50.3.1	localData . . . . .	270
7.50.3.2	notFoundException . . . . .	270
7.50.3.3	userException . . . . .	270
7.50.3.4	userIdentity . . . . .	270
7.51	qevercloud::ManageNotebookSharesParameters Struct Reference . . . . .	271
7.51.1	Detailed Description . . . . .	271
7.51.2	Member Function Documentation . . . . .	271
7.51.2.1	operator"!=( ) . . . . .	271
7.51.2.2	operator==( ) . . . . .	272
7.51.2.3	print() . . . . .	272



7.51.3	Member Data Documentation	272
7.51.3.1	invitationsToCreateOrUpdate	272
7.51.3.2	inviteMessage	272
7.51.3.3	localData	272
7.51.3.4	membershipsToUpdate	272
7.51.3.5	notebookGuid	273
7.51.3.6	unshares	273
7.51.4	Property Documentation	273
7.51.4.1	QList	273
7.52	qevercloud::ManageNotebookSharesResult Struct Reference	273
7.52.1	Detailed Description	274
7.52.2	Member Function Documentation	274
7.52.2.1	operator"!=( )	274
7.52.2.2	operator==( )	274
7.52.2.3	print()	274
7.52.3	Member Data Documentation	274
7.52.3.1	errors	274
7.52.3.2	localData	275
7.52.4	Property Documentation	275
7.52.4.1	QList	275
7.53	qevercloud::ManageNoteSharesError Struct Reference	275
7.53.1	Detailed Description	275
7.53.2	Member Function Documentation	276
7.53.2.1	operator"!=( )	276
7.53.2.2	operator==( )	276
7.53.2.3	print()	276
7.53.3	Member Data Documentation	276
7.53.3.1	identityID	276
7.53.3.2	localData	276
7.53.3.3	notFoundException	276

7.53.3.4	<a href="#">userException</a>	277
7.53.3.5	<a href="#">userID</a>	277
7.54	<a href="#">qevercloud::ManageNoteSharesParameters Struct Reference</a>	277
7.54.1	<a href="#">Detailed Description</a>	278
7.54.2	<a href="#">Member Function Documentation</a>	278
7.54.2.1	<a href="#">operator"!=( )</a>	278
7.54.2.2	<a href="#">operator==( )</a>	278
7.54.2.3	<a href="#">print()</a>	278
7.54.3	<a href="#">Member Data Documentation</a>	278
7.54.3.1	<a href="#">invitationsToUnshare</a>	278
7.54.3.2	<a href="#">invitationsToUpdate</a>	279
7.54.3.3	<a href="#">localData</a>	279
7.54.3.4	<a href="#">membershipsToUnshare</a>	279
7.54.3.5	<a href="#">membershipsToUpdate</a>	279
7.54.3.6	<a href="#">noteGuid</a>	279
7.54.4	<a href="#">Property Documentation</a>	279
7.54.4.1	<a href="#">QList</a>	279
7.55	<a href="#">qevercloud::ManageNoteSharesResult Struct Reference</a>	280
7.55.1	<a href="#">Detailed Description</a>	280
7.55.2	<a href="#">Member Function Documentation</a>	280
7.55.2.1	<a href="#">operator"!=( )</a>	280
7.55.2.2	<a href="#">operator==( )</a>	281
7.55.2.3	<a href="#">print()</a>	281
7.55.3	<a href="#">Member Data Documentation</a>	281
7.55.3.1	<a href="#">errors</a>	281
7.55.3.2	<a href="#">localData</a>	281
7.55.4	<a href="#">Property Documentation</a>	281
7.55.4.1	<a href="#">QList</a>	281
7.56	<a href="#">qevercloud::MemberShareRelationship Struct Reference</a>	282
7.56.1	<a href="#">Detailed Description</a>	282

7.56.2	Member Function Documentation	282
7.56.2.1	operator"!=( )	282
7.56.2.2	operator==( )	282
7.56.2.3	print()	283
7.56.3	Member Data Documentation	283
7.56.3.1	bestPrivilege	283
7.56.3.2	displayName	283
7.56.3.3	individualPrivilege	283
7.56.3.4	localData	283
7.56.3.5	recipientUserId	283
7.56.3.6	restrictions	284
7.56.3.7	sharerUserId	284
7.57	qevercloud::NetworkException Class Reference	284
7.57.1	Detailed Description	285
7.57.2	Constructor & Destructor Documentation	285
7.57.2.1	NetworkException() [1/3]	285
7.57.2.2	NetworkException() [2/3]	285
7.57.2.3	NetworkException() [3/3]	285
7.57.2.4	~NetworkException()	285
7.57.3	Member Function Documentation	285
7.57.3.1	exceptionData()	285
7.57.3.2	operator"!=( )	286
7.57.3.3	operator==( )	286
7.57.3.4	type()	286
7.57.3.5	what()	286
7.57.4	Member Data Documentation	286
7.57.4.1	m_type	286
7.58	qevercloud::NetworkExceptionData Class Reference	286
7.58.1	Detailed Description	287
7.58.2	Constructor & Destructor Documentation	287

7.58.2.1	<a href="#">NetworkExceptionData()</a>	287
7.58.3	<a href="#">Member Function Documentation</a>	287
7.58.3.1	<a href="#">throwException()</a>	287
7.58.4	<a href="#">Member Data Documentation</a>	287
7.58.4.1	<a href="#">m_type</a>	288
7.59	<a href="#">qevercloud::Note Struct Reference</a>	288
7.59.1	<a href="#">Detailed Description</a>	289
7.59.2	<a href="#">Member Function Documentation</a>	289
7.59.2.1	<a href="#">operator!=(())</a>	289
7.59.2.2	<a href="#">operator==(())</a>	289
7.59.2.3	<a href="#">print()</a>	289
7.59.3	<a href="#">Member Data Documentation</a>	289
7.59.3.1	<a href="#">active</a>	289
7.59.3.2	<a href="#">attributes</a>	289
7.59.3.3	<a href="#">content</a>	290
7.59.3.4	<a href="#">contentHash</a>	290
7.59.3.5	<a href="#">contentLength</a>	290
7.59.3.6	<a href="#">created</a>	290
7.59.3.7	<a href="#">deleted</a>	290
7.59.3.8	<a href="#">guid</a>	290
7.59.3.9	<a href="#">limits</a>	291
7.59.3.10	<a href="#">localData</a>	291
7.59.3.11	<a href="#">notebookGuid</a>	291
7.59.3.12	<a href="#">resources</a>	291
7.59.3.13	<a href="#">restrictions</a>	291
7.59.3.14	<a href="#">sharedNotes</a>	291
7.59.3.15	<a href="#">tagGuids</a>	292
7.59.3.16	<a href="#">tagNames</a>	292
7.59.3.17	<a href="#">title</a>	292
7.59.3.18	<a href="#">updated</a>	292

7.59.3.19 updateSequenceNum . . . . .	292
7.59.4 Property Documentation . . . . .	292
7.59.4.1 QList . . . . .	293
7.60 qevercloud::NoteAttributes Struct Reference . . . . .	293
7.60.1 Detailed Description . . . . .	294
7.60.2 Member Typedef Documentation . . . . .	294
7.60.2.1 Classifications . . . . .	294
7.60.3 Member Function Documentation . . . . .	294
7.60.3.1 operator"!=( ) . . . . .	294
7.60.3.2 operator==( ) . . . . .	294
7.60.3.3 print( ) . . . . .	294
7.60.4 Member Data Documentation . . . . .	294
7.60.4.1 altitude . . . . .	295
7.60.4.2 applicationData . . . . .	295
7.60.4.3 author . . . . .	295
7.60.4.4 classifications . . . . .	295
7.60.4.5 conflictSourceNoteGuid . . . . .	295
7.60.4.6 contentClass . . . . .	296
7.60.4.7 creatorId . . . . .	296
7.60.4.8 lastEditedBy . . . . .	296
7.60.4.9 lastEditorId . . . . .	296
7.60.4.10 latitude . . . . .	296
7.60.4.11 localData . . . . .	297
7.60.4.12 longitude . . . . .	297
7.60.4.13 noteTitleQuality . . . . .	297
7.60.4.14 placeName . . . . .	297
7.60.4.15 reminderDoneTime . . . . .	297
7.60.4.16 reminderOrder . . . . .	298
7.60.4.17 reminderTime . . . . .	298
7.60.4.18 shareDate . . . . .	298

7.60.4.19	sharedWithBusiness	298
7.60.4.20	source	298
7.60.4.21	sourceApplication	299
7.60.4.22	sourceURL	299
7.60.4.23	subjectDate	299
7.60.5	Property Documentation	299
7.60.5.1	classifications	299
7.61	qevercloud::Notebook Struct Reference	299
7.61.1	Detailed Description	300
7.61.2	Member Function Documentation	300
7.61.2.1	operator"!=(())	300
7.61.2.2	operator==(())	300
7.61.2.3	print()	301
7.61.3	Member Data Documentation	301
7.61.3.1	businessNotebook	301
7.61.3.2	contact	301
7.61.3.3	defaultNotebook	301
7.61.3.4	guid	301
7.61.3.5	localData	302
7.61.3.6	name	302
7.61.3.7	published	302
7.61.3.8	publishing	302
7.61.3.9	recipientSettings	302
7.61.3.10	restrictions	302
7.61.3.11	serviceCreated	303
7.61.3.12	serviceUpdated	303
7.61.3.13	sharedNotebookIds	303
7.61.3.14	sharedNotebooks	303
7.61.3.15	stack	303
7.61.3.16	updateSequenceNum	303

7.61.4	Property Documentation	304
7.61.4.1	QList	304
7.62	qevercloud::NotebookDescriptor Struct Reference	304
7.62.1	Detailed Description	304
7.62.2	Member Function Documentation	304
7.62.2.1	operator"!=( )	305
7.62.2.2	operator==( )	305
7.62.2.3	print()	305
7.62.3	Member Data Documentation	305
7.62.3.1	contactName	305
7.62.3.2	guid	305
7.62.3.3	hasSharedNotebook	305
7.62.3.4	joinedUserCount	306
7.62.3.5	localData	306
7.62.3.6	notebookDisplayName	306
7.63	qevercloud::NotebookRecipientSettings Struct Reference	306
7.63.1	Detailed Description	307
7.63.2	Member Function Documentation	307
7.63.2.1	operator"!=( )	307
7.63.2.2	operator==( )	307
7.63.2.3	print()	307
7.63.3	Member Data Documentation	307
7.63.3.1	inMyList	307
7.63.3.2	localData	308
7.63.3.3	recipientStatus	308
7.63.3.4	reminderNotifyEmail	308
7.63.3.5	reminderNotifyInApp	308
7.63.3.6	stack	308
7.64	qevercloud::NotebookRestrictions Struct Reference	308
7.64.1	Detailed Description	309

7.64.2	Member Function Documentation	310
7.64.2.1	operator"!=( )	310
7.64.2.2	operator==( )	310
7.64.2.3	print()	310
7.64.3	Member Data Documentation	310
7.64.3.1	canMoveToContainerRestrictions	310
7.64.3.2	expungeWhichSharedNotebookRestrictions	310
7.64.3.3	localData	311
7.64.3.4	noCanMoveNote	311
7.64.3.5	noChangeContact	311
7.64.3.6	noCreateNotes	311
7.64.3.7	noCreateSharedNotebooks	311
7.64.3.8	noCreateTags	311
7.64.3.9	noEmailNotes	311
7.64.3.10	noExpungeNotebook	312
7.64.3.11	noExpungeNotes	312
7.64.3.12	noExpungeTags	312
7.64.3.13	noPublishToBusinessLibrary	312
7.64.3.14	noPublishToPublic	312
7.64.3.15	noReadNotes	312
7.64.3.16	noRenameNotebook	312
7.64.3.17	noSendMessageToRecipients	313
7.64.3.18	noSetDefaultNotebook	313
7.64.3.19	noSetInMyList	313
7.64.3.20	noSetNotebookStack	313
7.64.3.21	noSetParentTag	313
7.64.3.22	noSetRecipientSettingsStack	313
7.64.3.23	noSetReminderNotifyEmail	313
7.64.3.24	noSetReminderNotifyInApp	314
7.64.3.25	noShareNotes	314



7.64.3.26 noShareNotesWithBusiness . . . . .	314
7.64.3.27 noUpdateNotebook . . . . .	314
7.64.3.28 noUpdateNotes . . . . .	314
7.64.3.29 noUpdateTags . . . . .	314
7.64.3.30 updateWhichSharedNotebookRestrictions . . . . .	314
7.65 qevercloud::NotebookShareTemplate Struct Reference . . . . .	315
7.65.1 Detailed Description . . . . .	315
7.65.2 Member Function Documentation . . . . .	315
7.65.2.1 operator"!=( ) . . . . .	315
7.65.2.2 operator==( ) . . . . .	316
7.65.2.3 print() . . . . .	316
7.65.3 Member Data Documentation . . . . .	316
7.65.3.1 localData . . . . .	316
7.65.3.2 notebookGuid . . . . .	316
7.65.3.3 privilege . . . . .	316
7.65.3.4 recipientContacts . . . . .	316
7.65.3.5 recipientThreadId . . . . .	317
7.65.4 Property Documentation . . . . .	317
7.65.4.1 QList . . . . .	317
7.66 qevercloud::NoteCollectionCounts Struct Reference . . . . .	317
7.66.1 Detailed Description . . . . .	318
7.66.2 Member Typedef Documentation . . . . .	318
7.66.2.1 TagCounts . . . . .	318
7.66.3 Member Function Documentation . . . . .	318
7.66.3.1 operator"!=( ) . . . . .	318
7.66.3.2 operator==( ) . . . . .	318
7.66.3.3 print() . . . . .	318
7.66.4 Member Data Documentation . . . . .	319
7.66.4.1 localData . . . . .	319
7.66.4.2 notebookCounts . . . . .	319

7.66.4.3	tagCounts	319
7.66.4.4	trashCount	319
7.66.5	Property Documentation	319
7.66.5.1	notebookCounts	319
7.66.5.2	tagCounts	319
7.67	qevercloud::NoteEmailParameters Struct Reference	320
7.67.1	Detailed Description	320
7.67.2	Member Function Documentation	320
7.67.2.1	operator"!="()	320
7.67.2.2	operator=="()	321
7.67.2.3	print()	321
7.67.3	Member Data Documentation	321
7.67.3.1	ccAddresses	321
7.67.3.2	guid	321
7.67.3.3	localData	321
7.67.3.4	message	321
7.67.3.5	note	322
7.67.3.6	subject	322
7.67.3.7	toAddresses	322
7.68	qevercloud::NoteFilter Struct Reference	322
7.68.1	Detailed Description	323
7.68.2	Member Function Documentation	323
7.68.2.1	operator"!="()	323
7.68.2.2	operator=="()	323
7.68.2.3	print()	324
7.68.3	Member Data Documentation	324
7.68.3.1	ascending	324
7.68.3.2	context	324
7.68.3.3	emphasized	324
7.68.3.4	inactive	324

7.68.3.5	<a href="#">includeAllReadableNotebooks</a>	324
7.68.3.6	<a href="#">includeAllReadableWorkspaces</a>	325
7.68.3.7	<a href="#">localData</a>	325
7.68.3.8	<a href="#">notebookGuid</a>	325
7.68.3.9	<a href="#">order</a>	325
7.68.3.10	<a href="#">rawWords</a>	325
7.68.3.11	<a href="#">searchContextBytes</a>	325
7.68.3.12	<a href="#">tagGuids</a>	325
7.68.3.13	<a href="#">timeZone</a>	326
7.68.3.14	<a href="#">words</a>	326
7.68.4	<a href="#">Property Documentation</a>	326
7.68.4.1	<a href="#">QList</a>	326
7.69	<a href="#">qevercloud::NoteInvitationShareRelationship Struct Reference</a>	326
7.69.1	<a href="#">Detailed Description</a>	327
7.69.2	<a href="#">Member Function Documentation</a>	327
7.69.2.1	<a href="#">operator"!=( )</a>	327
7.69.2.2	<a href="#">operator==( )</a>	327
7.69.2.3	<a href="#">print()</a>	327
7.69.3	<a href="#">Member Data Documentation</a>	327
7.69.3.1	<a href="#">displayName</a>	327
7.69.3.2	<a href="#">localData</a>	327
7.69.3.3	<a href="#">privilege</a>	328
7.69.3.4	<a href="#">recipientIdentityId</a>	328
7.69.3.5	<a href="#">sharerUserId</a>	328
7.70	<a href="#">qevercloud::NoteLimits Struct Reference</a>	328
7.70.1	<a href="#">Detailed Description</a>	329
7.70.2	<a href="#">Member Function Documentation</a>	329
7.70.2.1	<a href="#">operator"!=( )</a>	329
7.70.2.2	<a href="#">operator==( )</a>	329
7.70.2.3	<a href="#">print()</a>	329

7.70.3	Member Data Documentation . . . . .	329
7.70.3.1	localData . . . . .	329
7.70.3.2	noteResourceCountMax . . . . .	330
7.70.3.3	noteSizeMax . . . . .	330
7.70.3.4	resourceSizeMax . . . . .	330
7.70.3.5	uploaded . . . . .	330
7.70.3.6	uploadLimit . . . . .	330
7.71	qevercloud::NoteList Struct Reference . . . . .	330
7.71.1	Detailed Description . . . . .	331
7.71.2	Member Function Documentation . . . . .	331
7.71.2.1	operator"!=( ) . . . . .	331
7.71.2.2	operator==( ) . . . . .	331
7.71.2.3	print() . . . . .	331
7.71.3	Member Data Documentation . . . . .	331
7.71.3.1	debugInfo . . . . .	332
7.71.3.2	localData . . . . .	332
7.71.3.3	notes . . . . .	332
7.71.3.4	searchContextBytes . . . . .	332
7.71.3.5	searchedWords . . . . .	332
7.71.3.6	startIndex . . . . .	332
7.71.3.7	stoppedWords . . . . .	332
7.71.3.8	totalNotes . . . . .	333
7.71.3.9	updateCount . . . . .	333
7.72	qevercloud::NoteMemberShareRelationship Struct Reference . . . . .	333
7.72.1	Detailed Description . . . . .	333
7.72.2	Member Function Documentation . . . . .	334
7.72.2.1	operator"!=( ) . . . . .	334
7.72.2.2	operator==( ) . . . . .	334
7.72.2.3	print() . . . . .	334
7.72.3	Member Data Documentation . . . . .	334

7.72.3.1	displayName	334
7.72.3.2	localData	334
7.72.3.3	privilege	334
7.72.3.4	recipientUserId	335
7.72.3.5	restrictions	335
7.72.3.6	sharerUserId	335
7.73	qevercloud::NoteMetadata Struct Reference	335
7.73.1	Detailed Description	336
7.73.2	Member Function Documentation	336
7.73.2.1	operator"!="()	336
7.73.2.2	operator=="()	336
7.73.2.3	print()	336
7.73.3	Member Data Documentation	336
7.73.3.1	attributes	336
7.73.3.2	contentLength	337
7.73.3.3	created	337
7.73.3.4	deleted	337
7.73.3.5	guid	337
7.73.3.6	largestResourceMime	337
7.73.3.7	largestResourceSize	337
7.73.3.8	localData	337
7.73.3.9	notebookGuid	338
7.73.3.10	tagGuids	338
7.73.3.11	title	338
7.73.3.12	updated	338
7.73.3.13	updateSequenceNum	338
7.73.4	Property Documentation	338
7.73.4.1	QList	338
7.74	qevercloud::NoteRestrictions Struct Reference	338
7.74.1	Detailed Description	339

7.74.2	Member Function Documentation	339
7.74.2.1	operator"!=()	340
7.74.2.2	operator==(())	340
7.74.2.3	print()	340
7.74.3	Member Data Documentation	340
7.74.3.1	localData	340
7.74.3.2	noEmail	340
7.74.3.3	noShare	340
7.74.3.4	noSharePublicly	341
7.74.3.5	noUpdateContent	341
7.74.3.6	noUpdateTitle	341
7.75	qevercloud::NoteResultSpec Struct Reference	341
7.75.1	Detailed Description	342
7.75.2	Member Function Documentation	342
7.75.2.1	operator"!=()	342
7.75.2.2	operator==(())	342
7.75.2.3	print()	342
7.75.3	Member Data Documentation	342
7.75.3.1	includeAccountLimits	342
7.75.3.2	includeContent	343
7.75.3.3	includeNoteAppDataValues	343
7.75.3.4	includeResourceAppDataValues	343
7.75.3.5	includeResourcesAlternateData	343
7.75.3.6	includeResourcesData	343
7.75.3.7	includeResourcesRecognition	343
7.75.3.8	includeSharedNotes	343
7.75.3.9	localData	344
7.76	qevercloud::NoteShareRelationshipRestrictions Struct Reference	344
7.76.1	Detailed Description	344
7.76.2	Member Function Documentation	344

7.76.2.1	operator"!=()	344
7.76.2.2	operator==(())	345
7.76.2.3	print()	345
7.76.3	Member Data Documentation	345
7.76.3.1	localData	345
7.76.3.2	noSetFullAccess	345
7.76.3.3	noSetModifyNote	345
7.76.3.4	noSetReadNote	345
7.77	qevercloud::NoteShareRelationships Struct Reference	346
7.77.1	Detailed Description	346
7.77.2	Member Function Documentation	346
7.77.2.1	operator"!=()	346
7.77.2.2	operator==(())	347
7.77.2.3	print()	347
7.77.3	Member Data Documentation	347
7.77.3.1	invitationRestrictions	347
7.77.3.2	invitations	347
7.77.3.3	localData	347
7.77.3.4	memberships	347
7.77.4	Property Documentation	348
7.77.4.1	QList	348
7.78	qevercloud::NotesMetadataList Struct Reference	348
7.78.1	Detailed Description	348
7.78.2	Member Function Documentation	348
7.78.2.1	operator"!=()	349
7.78.2.2	operator==(())	349
7.78.2.3	print()	349
7.78.3	Member Data Documentation	349
7.78.3.1	debugInfo	349
7.78.3.2	localData	349

7.78.3.3	notes	349
7.78.3.4	searchContextBytes	350
7.78.3.5	searchedWords	350
7.78.3.6	startIndex	350
7.78.3.7	stoppedWords	350
7.78.3.8	totalNotes	350
7.78.3.9	updateCount	350
7.79	qevercloud::NotesMetadataResultSpec Struct Reference	351
7.79.1	Detailed Description	351
7.79.2	Member Function Documentation	351
7.79.2.1	operator"!=(())	352
7.79.2.2	operator==(())	352
7.79.2.3	print()	352
7.79.3	Member Data Documentation	352
7.79.3.1	includeAttributes	352
7.79.3.2	includeContentLength	352
7.79.3.3	includeCreated	352
7.79.3.4	includeDeleted	353
7.79.3.5	includeLargestResourceMime	353
7.79.3.6	includeLargestResourceSize	353
7.79.3.7	includeNotebookGuid	353
7.79.3.8	includeTagGuids	353
7.79.3.9	includeTitle	353
7.79.3.10	includeUpdated	353
7.79.3.11	includeUpdateSequenceNum	353
7.79.3.12	localData	354
7.80	qevercloud::NoteStoreServer Class Reference	354
7.80.1	Detailed Description	359
7.80.2	Constructor & Destructor Documentation	359
7.80.2.1	NoteStoreServer()	359



7.80.3 Member Function Documentation . . . . .	359
7.80.3.1 authenticateToSharedNotebookRequest . . . . .	360
7.80.3.2 authenticateToSharedNotebookRequestReady . . . . .	360
7.80.3.3 authenticateToSharedNoteRequest . . . . .	360
7.80.3.4 authenticateToSharedNoteRequestReady . . . . .	360
7.80.3.5 copyNoteRequest . . . . .	360
7.80.3.6 copyNoteRequestReady . . . . .	360
7.80.3.7 createLinkedNotebookRequest . . . . .	361
7.80.3.8 createLinkedNotebookRequestReady . . . . .	361
7.80.3.9 createNotebookRequest . . . . .	361
7.80.3.10 createNotebookRequestReady . . . . .	361
7.80.3.11 createNoteRequest . . . . .	361
7.80.3.12 createNoteRequestReady . . . . .	361
7.80.3.13 createOrUpdateNotebookSharesRequest . . . . .	362
7.80.3.14 createOrUpdateNotebookSharesRequestReady . . . . .	362
7.80.3.15 createSearchRequest . . . . .	362
7.80.3.16 createSearchRequestReady . . . . .	362
7.80.3.17 createTagRequest . . . . .	362
7.80.3.18 createTagRequestReady . . . . .	362
7.80.3.19 deleteNoteRequest . . . . .	363
7.80.3.20 deleteNoteRequestReady . . . . .	363
7.80.3.21 emailNoteRequest . . . . .	363
7.80.3.22 emailNoteRequestReady . . . . .	363
7.80.3.23 expungeLinkedNotebookRequest . . . . .	363
7.80.3.24 expungeLinkedNotebookRequestReady . . . . .	363
7.80.3.25 expungeNotebookRequest . . . . .	364
7.80.3.26 expungeNotebookRequestReady . . . . .	364
7.80.3.27 expungeNoteRequest . . . . .	364
7.80.3.28 expungeNoteRequestReady . . . . .	364
7.80.3.29 expungeSearchRequest . . . . .	364

7.80.3.30 expungeSearchRequestReady . . . . .	364
7.80.3.31 expungeTagRequest . . . . .	365
7.80.3.32 expungeTagRequestReady . . . . .	365
7.80.3.33 findNoteCountsRequest . . . . .	365
7.80.3.34 findNoteCountsRequestReady . . . . .	365
7.80.3.35 findNoteOffsetRequest . . . . .	365
7.80.3.36 findNoteOffsetRequestReady . . . . .	365
7.80.3.37 findNotesMetadataRequest . . . . .	366
7.80.3.38 findNotesMetadataRequestReady . . . . .	366
7.80.3.39 findRelatedRequest . . . . .	366
7.80.3.40 findRelatedRequestReady . . . . .	366
7.80.3.41 getDefaultNotebookRequest . . . . .	366
7.80.3.42 getDefaultNotebookRequestReady . . . . .	366
7.80.3.43 getFilteredSyncChunkRequest . . . . .	367
7.80.3.44 getFilteredSyncChunkRequestReady . . . . .	367
7.80.3.45 getLinkedNotebookSyncChunkRequest . . . . .	367
7.80.3.46 getLinkedNotebookSyncChunkRequestReady . . . . .	367
7.80.3.47 getLinkedNotebookSyncStateRequest . . . . .	367
7.80.3.48 getLinkedNotebookSyncStateRequestReady . . . . .	367
7.80.3.49 getNoteApplicationDataEntryRequest . . . . .	368
7.80.3.50 getNoteApplicationDataEntryRequestReady . . . . .	368
7.80.3.51 getNoteApplicationDataRequest . . . . .	368
7.80.3.52 getNoteApplicationDataRequestReady . . . . .	368
7.80.3.53 getNotebookRequest . . . . .	368
7.80.3.54 getNotebookRequestReady . . . . .	368
7.80.3.55 getNotebookSharesRequest . . . . .	369
7.80.3.56 getNotebookSharesRequestReady . . . . .	369
7.80.3.57 getNoteContentRequest . . . . .	369
7.80.3.58 getNoteContentRequestReady . . . . .	369
7.80.3.59 getNoteRequest . . . . .	369

7.80.3.60	getNoteRequestReady	369
7.80.3.61	getNoteSearchTextRequest	370
7.80.3.62	getNoteSearchTextRequestReady	370
7.80.3.63	getNoteTagNamesRequest	370
7.80.3.64	getNoteTagNamesRequestReady	370
7.80.3.65	getNoteVersionRequest	370
7.80.3.66	getNoteVersionRequestReady	370
7.80.3.67	getNoteWithResultSpecRequest	371
7.80.3.68	getNoteWithResultSpecRequestReady	371
7.80.3.69	getPublicNotebookRequest	371
7.80.3.70	getPublicNotebookRequestReady	371
7.80.3.71	getResourceAlternateDataRequest	371
7.80.3.72	getResourceAlternateDataRequestReady	371
7.80.3.73	getResourceApplicationDataEntryRequest	372
7.80.3.74	getResourceApplicationDataEntryRequestReady	372
7.80.3.75	getResourceApplicationDataRequest	372
7.80.3.76	getResourceApplicationDataRequestReady	372
7.80.3.77	getResourceAttributesRequest	372
7.80.3.78	getResourceAttributesRequestReady	372
7.80.3.79	getResourceByHashRequest	373
7.80.3.80	getResourceByHashRequestReady	373
7.80.3.81	getResourceDataRequest	373
7.80.3.82	getResourceDataRequestReady	373
7.80.3.83	getResourceRecognitionRequest	373
7.80.3.84	getResourceRecognitionRequestReady	373
7.80.3.85	getResourceRequest	374
7.80.3.86	getResourceRequestReady	374
7.80.3.87	getResourceSearchTextRequest	374
7.80.3.88	getResourceSearchTextRequestReady	374
7.80.3.89	getSearchRequest	374

7.80.3.90	getSearchRequestReady	374
7.80.3.91	getSharedNotebookByAuthRequest	375
7.80.3.92	getSharedNotebookByAuthRequestReady	375
7.80.3.93	getSyncStateRequest	375
7.80.3.94	getSyncStateRequestReady	375
7.80.3.95	getTagRequest	375
7.80.3.96	getTagRequestReady	375
7.80.3.97	listAccessibleBusinessNotebooksRequest	375
7.80.3.98	listAccessibleBusinessNotebooksRequestReady	376
7.80.3.99	listLinkedNotebooksRequest	376
7.80.3.100	listLinkedNotebooksRequestReady	376
7.80.3.101	listNotebooksRequest	376
7.80.3.102	listNotebooksRequestReady	376
7.80.3.103	listNoteVersionsRequest	376
7.80.3.104	listNoteVersionsRequestReady	376
7.80.3.105	listSearchesRequest	377
7.80.3.106	listSearchesRequestReady	377
7.80.3.107	listSharedNotebooksRequest	377
7.80.3.108	listSharedNotebooksRequestReady	377
7.80.3.109	listTagsByNotebookRequest	377
7.80.3.110	listTagsByNotebookRequestReady	377
7.80.3.111	listTagsRequest	377
7.80.3.112	listTagsRequestReady	378
7.80.3.113	manageNotebookSharesRequest	378
7.80.3.114	manageNotebookSharesRequestReady	378
7.80.3.115	onAuthenticateToSharedNotebookRequestReady	378
7.80.3.116	onAuthenticateToSharedNoteRequestReady	378
7.80.3.117	onCopyNoteRequestReady	378
7.80.3.118	onCreateLinkedNotebookRequestReady	379
7.80.3.119	onCreateNotebookRequestReady	379

7.80.3.120onCreateNoteRequestReady . . . . .	379
7.80.3.121onCreateOrUpdateNotebookSharesRequestReady . . . . .	379
7.80.3.122onCreateSearchRequestReady . . . . .	379
7.80.3.123onCreateTagRequestReady . . . . .	379
7.80.3.124onDeleteNoteRequestReady . . . . .	380
7.80.3.125onEmailNoteRequestReady . . . . .	380
7.80.3.126onExpungeLinkedNotebookRequestReady . . . . .	380
7.80.3.127onExpungeNotebookRequestReady . . . . .	380
7.80.3.128onExpungeNoteRequestReady . . . . .	380
7.80.3.129onExpungeSearchRequestReady . . . . .	380
7.80.3.130onExpungeTagRequestReady . . . . .	381
7.80.3.131onFindNoteCountsRequestReady . . . . .	381
7.80.3.132onFindNoteOffsetRequestReady . . . . .	381
7.80.3.133onFindNotesMetadataRequestReady . . . . .	381
7.80.3.134onFindRelatedRequestReady . . . . .	381
7.80.3.135onGetDefaultNotebookRequestReady . . . . .	381
7.80.3.136onGetFilteredSyncChunkRequestReady . . . . .	382
7.80.3.137onGetLinkedNotebookSyncChunkRequestReady . . . . .	382
7.80.3.138onGetLinkedNotebookSyncStateRequestReady . . . . .	382
7.80.3.139onGetNoteApplicationDataEntryRequestReady . . . . .	382
7.80.3.140onGetNoteApplicationDataRequestReady . . . . .	382
7.80.3.141onGetNotebookRequestReady . . . . .	382
7.80.3.142onGetNotebookSharesRequestReady . . . . .	383
7.80.3.143onGetNoteContentRequestReady . . . . .	383
7.80.3.144onGetNoteRequestReady . . . . .	383
7.80.3.145onGetNoteSearchTextRequestReady . . . . .	383
7.80.3.146onGetNoteTagNamesRequestReady . . . . .	383
7.80.3.147onGetNoteVersionRequestReady . . . . .	383
7.80.3.148onGetNoteWithResultSpecRequestReady . . . . .	384
7.80.3.149onGetPublicNotebookRequestReady . . . . .	384

7.80.3.150nGetResourceAlternateDataRequestReady . . . . .	384
7.80.3.151nGetResourceApplicationDataEntryRequestReady . . . . .	384
7.80.3.152nGetResourceApplicationDataRequestReady . . . . .	384
7.80.3.153nGetResourceAttributesRequestReady . . . . .	384
7.80.3.154nGetResourceByHashRequestReady . . . . .	385
7.80.3.155nGetResourceDataRequestReady . . . . .	385
7.80.3.156nGetResourceRecognitionRequestReady . . . . .	385
7.80.3.157nGetResourceRequestReady . . . . .	385
7.80.3.158nGetResourceSearchTextRequestReady . . . . .	385
7.80.3.159nGetSearchRequestReady . . . . .	385
7.80.3.160nGetSharedNotebookByAuthRequestReady . . . . .	386
7.80.3.161nGetSyncStateRequestReady . . . . .	386
7.80.3.162nGetTagRequestReady . . . . .	386
7.80.3.163nListAccessibleBusinessNotebooksRequestReady . . . . .	386
7.80.3.164nListLinkedNotebooksRequestReady . . . . .	386
7.80.3.165nListNotebooksRequestReady . . . . .	386
7.80.3.166nListNoteVersionsRequestReady . . . . .	387
7.80.3.167nListSearchesRequestReady . . . . .	387
7.80.3.168nListSharedNotebooksRequestReady . . . . .	387
7.80.3.169nListTagsByNotebookRequestReady . . . . .	387
7.80.3.170nListTagsRequestReady . . . . .	387
7.80.3.171nManageNotebookSharesRequestReady . . . . .	387
7.80.3.172nRequest . . . . .	388
7.80.3.173nSetNoteApplicationDataEntryRequestReady . . . . .	388
7.80.3.174nSetNotebookRecipientSettingsRequestReady . . . . .	388
7.80.3.175nSetResourceApplicationDataEntryRequestReady . . . . .	388
7.80.3.176nShareNotebookRequestReady . . . . .	388
7.80.3.177nShareNoteRequestReady . . . . .	388
7.80.3.178nStopSharingNoteRequestReady . . . . .	389
7.80.3.179nUnsetNoteApplicationDataEntryRequestReady . . . . .	389

7.80.3.180onUnsetResourceApplicationDataEntryRequestReady . . . . .	389
7.80.3.181onUntagAllRequestReady . . . . .	389
7.80.3.182onUpdateLinkedNotebookRequestReady . . . . .	389
7.80.3.183onUpdateNotebookRequestReady . . . . .	389
7.80.3.184onUpdateNoteIfUsnMatchesRequestReady . . . . .	390
7.80.3.185onUpdateNoteRequestReady . . . . .	390
7.80.3.186onUpdateResourceRequestReady . . . . .	390
7.80.3.187onUpdateSearchRequestReady . . . . .	390
7.80.3.188onUpdateSharedNotebookRequestReady . . . . .	390
7.80.3.189onUpdateTagRequestReady . . . . .	390
7.80.3.190setNoteApplicationDataEntryRequest . . . . .	391
7.80.3.191setNoteApplicationDataEntryRequestReady . . . . .	391
7.80.3.192setNotebookRecipientSettingsRequest . . . . .	391
7.80.3.193setNotebookRecipientSettingsRequestReady . . . . .	391
7.80.3.194setResourceApplicationDataEntryRequest . . . . .	391
7.80.3.195setResourceApplicationDataEntryRequestReady . . . . .	391
7.80.3.196shareNotebookRequest . . . . .	392
7.80.3.197shareNotebookRequestReady . . . . .	392
7.80.3.198shareNoteRequest . . . . .	392
7.80.3.199shareNoteRequestReady . . . . .	392
7.80.3.200stopSharingNoteRequest . . . . .	392
7.80.3.201stopSharingNoteRequestReady . . . . .	392
7.80.3.202unsetNoteApplicationDataEntryRequest . . . . .	393
7.80.3.203unsetNoteApplicationDataEntryRequestReady . . . . .	393
7.80.3.204unsetResourceApplicationDataEntryRequest . . . . .	393
7.80.3.205unsetResourceApplicationDataEntryRequestReady . . . . .	393
7.80.3.206untagAllRequest . . . . .	393
7.80.3.207untagAllRequestReady . . . . .	393
7.80.3.208updateLinkedNotebookRequest . . . . .	394
7.80.3.209updateLinkedNotebookRequestReady . . . . .	394

7.80.3.210	updateNotebookRequest	394
7.80.3.211	updateNotebookRequestReady	394
7.80.3.212	updateNoteIfUsnMatchesRequest	394
7.80.3.213	updateNoteIfUsnMatchesRequestReady	394
7.80.3.214	updateNoteRequest	395
7.80.3.215	updateNoteRequestReady	395
7.80.3.216	updateResourceRequest	395
7.80.3.217	updateResourceRequestReady	395
7.80.3.218	updateSearchRequest	395
7.80.3.219	updateSearchRequestReady	395
7.80.3.220	updateSharedNotebookRequest	396
7.80.3.221	updateSharedNotebookRequestReady	396
7.80.3.222	updateTagRequest	396
7.80.3.223	updateTagRequestReady	396
7.81	qevercloud::NoteVersionId Struct Reference	396
7.81.1	Detailed Description	397
7.81.2	Member Function Documentation	397
7.81.2.1	operator!=(())	397
7.81.2.2	operator==(())	397
7.81.2.3	print()	397
7.81.3	Member Data Documentation	397
7.81.3.1	lastEditorId	398
7.81.3.2	localData	398
7.81.3.3	saved	398
7.81.3.4	title	398
7.81.3.5	updated	398
7.81.3.6	updateSequenceNum	398
7.82	qevercloud::EvernoteOAuthWebView::OAuthResult Struct Reference	399
7.82.1	Detailed Description	399
7.82.2	Member Function Documentation	399



7.82.2.1	<code>print()</code>	399
7.82.3	Member Data Documentation	400
7.82.3.1	<code>authenticationToken</code>	400
7.82.3.2	<code>cookies</code>	400
7.82.3.3	<code>expires</code>	400
7.82.3.4	<code>noteStoreUrl</code>	400
7.82.3.5	<code>shardId</code>	400
7.82.3.6	<code>userId</code>	401
7.82.3.7	<code>webApiUrlPrefix</code>	401
7.83	<code>qevercloud::Optional&lt; T &gt;</code> Class Template Reference	401
7.83.1	Detailed Description	402
7.83.2	Constructor & Destructor Documentation	402
7.83.2.1	<code>Optional()</code> [1/7]	402
7.83.2.2	<code>Optional()</code> [2/7]	402
7.83.2.3	<code>Optional()</code> [3/7]	403
7.83.2.4	<code>Optional()</code> [4/7]	403
7.83.2.5	<code>Optional()</code> [5/7]	403
7.83.2.6	<code>Optional()</code> [6/7]	403
7.83.2.7	<code>Optional()</code> [7/7]	403
7.83.3	Member Function Documentation	403
7.83.3.1	<code>clear()</code>	404
7.83.3.2	<code>init()</code>	404
7.83.3.3	<code>isEqual()</code>	404
7.83.3.4	<code>isSet()</code>	405
7.83.3.5	<code>operator const T &amp;()</code>	405
7.83.3.6	<code>operator T &amp;()</code>	405
7.83.3.7	<code>operator"!=()</code> [1/2]	405
7.83.3.8	<code>operator"!=()</code> [2/2]	405
7.83.3.9	<code>operator-&gt;()</code> [1/2]	406
7.83.3.10	<code>operator-&gt;()</code> [2/2]	406

7.83.3.11 operator=()	[ 1/6]	406
7.83.3.12 operator=()	[ 2/6]	406
7.83.3.13 operator=()	[ 3/6]	407
7.83.3.14 operator=()	[ 4/6]	407
7.83.3.15 operator=()	[ 5/6]	407
7.83.3.16 operator=()	[ 6/6]	407
7.83.3.17 operator==()	[ 1/2]	407
7.83.3.18 operator==()	[ 2/2]	407
7.83.3.19 ref()	[ 1/2]	408
7.83.3.20 ref()	[ 2/2]	408
7.83.3.21 value()		408
7.83.4 Friends And Related Function Documentation		409
7.83.4.1 Optional		409
7.83.4.2 swap		409
7.84 qevercloud::Printable Class Reference		409
7.84.1 Constructor & Destructor Documentation		410
7.84.1.1 Printable()		410
7.84.1.2 ~Printable()		411
7.84.2 Member Function Documentation		411
7.84.2.1 print()		411
7.84.2.2 toString()		411
7.84.3 Friends And Related Function Documentation		411
7.84.3.1 operator<<	[ 1/2]	412
7.84.3.2 operator<<	[ 2/2]	412
7.85 qevercloud::PublicUserInfo Struct Reference		412
7.85.1 Detailed Description		412
7.85.2 Member Function Documentation		413
7.85.2.1 operator!=(())		413
7.85.2.2 operator==(())		413
7.85.2.3 print()		413

7.85.3	Member Data Documentation . . . . .	413
7.85.3.1	localData . . . . .	413
7.85.3.2	noteStoreUrl . . . . .	413
7.85.3.3	serviceLevel . . . . .	414
7.85.3.4	userId . . . . .	414
7.85.3.5	username . . . . .	414
7.85.3.6	webApiUrlPrefix . . . . .	414
7.86	qevercloud::Publishing Struct Reference . . . . .	414
7.86.1	Detailed Description . . . . .	415
7.86.2	Member Function Documentation . . . . .	415
7.86.2.1	operator"!="() . . . . .	415
7.86.2.2	operator=="() . . . . .	415
7.86.2.3	print() . . . . .	415
7.86.3	Member Data Documentation . . . . .	415
7.86.3.1	ascending . . . . .	415
7.86.3.2	localData . . . . .	416
7.86.3.3	order . . . . .	416
7.86.3.4	publicDescription . . . . .	416
7.86.3.5	uri . . . . .	416
7.87	qevercloud::QAssociativeContainerConstReferenceWrapper< Container > Class Template Reference . . . . .	416
7.87.1	Constructor & Destructor Documentation . . . . .	417
7.87.1.1	QAssociativeContainerConstReferenceWrapper() . . . . .	417
7.87.2	Member Function Documentation . . . . .	417
7.87.2.1	begin() . . . . .	417
7.87.2.2	end() . . . . .	417
7.88	qevercloud::QAssociativeContainerReferenceWrapper< Container > Class Template Reference . . . . .	417
7.88.1	Constructor & Destructor Documentation . . . . .	418
7.88.1.1	QAssociativeContainerReferenceWrapper() . . . . .	418
7.88.2	Member Function Documentation . . . . .	418
7.88.2.1	begin() . . . . .	418

7.88.2.2	<code>end()</code>	418
7.89	<code>qevercloud::RelatedContent</code> Struct Reference	418
7.89.1	Detailed Description	419
7.89.2	Member Function Documentation	419
7.89.2.1	<code>operator!=()</code>	419
7.89.2.2	<code>operator==()</code>	419
7.89.2.3	<code>print()</code>	420
7.89.3	Member Data Documentation	420
7.89.3.1	<code>accessType</code>	420
7.89.3.2	<code>authors</code>	420
7.89.3.3	<code>clipUrl</code>	420
7.89.3.4	<code>contact</code>	420
7.89.3.5	<code>contentId</code>	420
7.89.3.6	<code>contentType</code>	420
7.89.3.7	<code>date</code>	421
7.89.3.8	<code>localData</code>	421
7.89.3.9	<code>sourceFaviconUrl</code>	421
7.89.3.10	<code>sourceId</code>	421
7.89.3.11	<code>sourceName</code>	421
7.89.3.12	<code>sourceUrl</code>	421
7.89.3.13	<code>teaser</code>	421
7.89.3.14	<code>thumbnails</code>	421
7.89.3.15	<code>title</code>	422
7.89.3.16	<code>url</code>	422
7.89.3.17	<code>visibleUrl</code>	422
7.89.4	Property Documentation	422
7.89.4.1	<code>QList</code>	422
7.90	<code>qevercloud::RelatedContentImage</code> Struct Reference	422
7.90.1	Detailed Description	423
7.90.2	Member Function Documentation	423

7.90.2.1	operator"!=()	423
7.90.2.2	operator==(	423
7.90.2.3	print()	423
7.90.3	Member Data Documentation	423
7.90.3.1	fileSize	424
7.90.3.2	height	424
7.90.3.3	localData	424
7.90.3.4	pixelRatio	424
7.90.3.5	url	424
7.90.3.6	width	424
7.91	qevercloud::RelatedQuery Struct Reference	424
7.91.1	Detailed Description	425
7.91.2	Member Function Documentation	425
7.91.2.1	operator"!=()	425
7.91.2.2	operator==(	425
7.91.2.3	print()	425
7.91.3	Member Data Documentation	426
7.91.3.1	cacheKey	426
7.91.3.2	context	426
7.91.3.3	filter	426
7.91.3.4	localData	426
7.91.3.5	noteGuid	426
7.91.3.6	plainText	426
7.91.3.7	referenceUri	427
7.92	qevercloud::RelatedResult Struct Reference	427
7.92.1	Detailed Description	427
7.92.2	Member Function Documentation	428
7.92.2.1	operator"!=()	428
7.92.2.2	operator==(	428
7.92.2.3	print()	428

7.92.3	Member Data Documentation . . . . .	428
7.92.3.1	cacheExpires . . . . .	428
7.92.3.2	cacheKey . . . . .	429
7.92.3.3	containingNotebooks . . . . .	429
7.92.3.4	debugInfo . . . . .	429
7.92.3.5	experts . . . . .	429
7.92.3.6	localData . . . . .	430
7.92.3.7	notebooks . . . . .	430
7.92.3.8	notes . . . . .	430
7.92.3.9	relatedContent . . . . .	430
7.92.3.10	tags . . . . .	430
7.92.4	Property Documentation . . . . .	430
7.92.4.1	QList . . . . .	430
7.93	qevercloud::RelatedResultSpec Struct Reference . . . . .	430
7.93.1	Detailed Description . . . . .	431
7.93.2	Member Function Documentation . . . . .	431
7.93.2.1	operator"!=( ) . . . . .	431
7.93.2.2	operator==( ) . . . . .	431
7.93.2.3	print() . . . . .	432
7.93.3	Member Data Documentation . . . . .	432
7.93.3.1	includeContainingNotebooks . . . . .	432
7.93.3.2	includeDebugInfo . . . . .	432
7.93.3.3	localData . . . . .	432
7.93.3.4	maxExperts . . . . .	432
7.93.3.5	maxNotebooks . . . . .	432
7.93.3.6	maxNotes . . . . .	433
7.93.3.7	maxRelatedContent . . . . .	433
7.93.3.8	maxTags . . . . .	433
7.93.3.9	relatedContentTypes . . . . .	433
7.93.3.10	writableNotebooksOnly . . . . .	433

7.93.4	Property Documentation	433
7.93.4.1	QSet	433
7.94	qevercloud::Resource Struct Reference	434
7.94.1	Detailed Description	434
7.94.2	Member Function Documentation	434
7.94.2.1	operator"!=( )	434
7.94.2.2	operator==( )	435
7.94.2.3	print()	435
7.94.3	Member Data Documentation	435
7.94.3.1	active	435
7.94.3.2	alternateData	435
7.94.3.3	attributes	435
7.94.3.4	data	435
7.94.3.5	duration	436
7.94.3.6	guid	436
7.94.3.7	height	436
7.94.3.8	localData	436
7.94.3.9	mime	436
7.94.3.10	noteGuid	436
7.94.3.11	recognition	436
7.94.3.12	updateSequenceNum	437
7.94.3.13	width	437
7.95	qevercloud::ResourceAttributes Struct Reference	437
7.95.1	Detailed Description	438
7.95.2	Member Function Documentation	438
7.95.2.1	operator"!=( )	438
7.95.2.2	operator==( )	438
7.95.2.3	print()	438
7.95.3	Member Data Documentation	438
7.95.3.1	altitude	438

7.95.3.2	<a href="#">applicationData</a>	439
7.95.3.3	<a href="#">attachment</a>	439
7.95.3.4	<a href="#">cameraMake</a>	439
7.95.3.5	<a href="#">cameraModel</a>	439
7.95.3.6	<a href="#">clientWillIndex</a>	439
7.95.3.7	<a href="#">fileName</a>	439
7.95.3.8	<a href="#">latitude</a>	440
7.95.3.9	<a href="#">localData</a>	440
7.95.3.10	<a href="#">longitude</a>	440
7.95.3.11	<a href="#">recoType</a>	440
7.95.3.12	<a href="#">sourceURL</a>	440
7.95.3.13	<a href="#">timestamp</a>	440
7.96	<a href="#">qevercloud::SavedSearch Struct Reference</a>	440
7.96.1	<a href="#">Detailed Description</a>	441
7.96.2	<a href="#">Member Function Documentation</a>	441
7.96.2.1	<a href="#">operator"!=( )</a>	441
7.96.2.2	<a href="#">operator==( )</a>	441
7.96.2.3	<a href="#">print()</a>	441
7.96.3	<a href="#">Member Data Documentation</a>	442
7.96.3.1	<a href="#">format</a>	442
7.96.3.2	<a href="#">guid</a>	442
7.96.3.3	<a href="#">localData</a>	442
7.96.3.4	<a href="#">name</a>	442
7.96.3.5	<a href="#">query</a>	442
7.96.3.6	<a href="#">scope</a>	442
7.96.3.7	<a href="#">updateSequenceNum</a>	443
7.97	<a href="#">qevercloud::SavedSearchScope Struct Reference</a>	443
7.97.1	<a href="#">Detailed Description</a>	443
7.97.2	<a href="#">Member Function Documentation</a>	443
7.97.2.1	<a href="#">operator"!=( )</a>	443



7.97.2.2	operator==( )	444
7.97.2.3	print( )	444
7.97.3	Member Data Documentation	444
7.97.3.1	includeAccount	444
7.97.3.2	includeBusinessLinkedNotebooks	444
7.97.3.3	includePersonalLinkedNotebooks	444
7.97.3.4	localData	444
7.98	qevercloud::SharedNote Struct Reference	445
7.98.1	Detailed Description	445
7.98.2	Member Function Documentation	445
7.98.2.1	operator!=( )	445
7.98.2.2	operator==( )	446
7.98.2.3	print( )	446
7.98.3	Member Data Documentation	446
7.98.3.1	localData	446
7.98.3.2	privilege	446
7.98.3.3	recipientIdentity	446
7.98.3.4	serviceAssigned	446
7.98.3.5	serviceCreated	447
7.98.3.6	serviceUpdated	447
7.98.3.7	sharerUserID	447
7.99	qevercloud::SharedNotebook Struct Reference	447
7.99.1	Detailed Description	448
7.99.2	Member Function Documentation	448
7.99.2.1	operator!=( )	448
7.99.2.2	operator==( )	448
7.99.2.3	print( )	448
7.99.3	Member Data Documentation	448
7.99.3.1	email	448
7.99.3.2	globalId	448

7.99.3.3	id	449
7.99.3.4	localData	449
7.99.3.5	notebookGuid	449
7.99.3.6	notebookModifiable	449
7.99.3.7	privilege	449
7.99.3.8	recipientIdentityId	449
7.99.3.9	recipientSettings	449
7.99.3.10	recipientUserId	450
7.99.3.11	recipientUsername	450
7.99.3.12	serviceAssigned	450
7.99.3.13	serviceCreated	450
7.99.3.14	serviceUpdated	450
7.99.3.15	sharerUserId	450
7.99.3.16	userId	451
7.99.3.17	username	451
7.100	qevercloud::SharedNotebookRecipientSettings Struct Reference	451
7.100.1	Detailed Description	451
7.100.2	Member Function Documentation	452
7.100.2.1	operator"!=( )	452
7.100.2.2	operator==( )	452
7.100.2.3	print()	452
7.100.3	Member Data Documentation	452
7.100.3.1	localData	452
7.100.3.2	reminderNotifyEmail	452
7.100.3.3	reminderNotifyInApp	453
7.101	qevercloud::SharedNoteTemplate Struct Reference	453
7.101.1	Detailed Description	453
7.101.2	Member Function Documentation	453
7.101.2.1	operator"!=( )	454
7.101.2.2	operator==( )	454

7.101.2.3 print()	454
7.101.3 Member Data Documentation	454
7.101.3.1 localData	454
7.101.3.2 noteGuid	454
7.101.3.3 privilege	454
7.101.3.4 recipientContacts	455
7.101.3.5 recipientThreadId	455
7.101.4 Property Documentation	455
7.101.4.1 QList	455
7.102qevercloud::ShareRelationshipRestrictions Struct Reference	455
7.102.1 Detailed Description	456
7.102.2 Member Function Documentation	456
7.102.2.1 operator"!=(	456
7.102.2.2 operator==(	456
7.102.2.3 print()	456
7.102.3 Member Data Documentation	456
7.102.3.1 localData	456
7.102.3.2 noSetFullAccess	456
7.102.3.3 noSetModify	457
7.102.3.4 noSetReadOnly	457
7.102.3.5 noSetReadPlusActivity	457
7.103qevercloud::ShareRelationships Struct Reference	457
7.103.1 Detailed Description	458
7.103.2 Member Function Documentation	458
7.103.2.1 operator"!=(	458
7.103.2.2 operator==(	458
7.103.2.3 print()	458
7.103.3 Member Data Documentation	458
7.103.3.1 invitationRestrictions	458
7.103.3.2 invitations	459

7.103.3.3 localData . . . . .	459
7.103.3.4 memberships . . . . .	459
7.103.4 Property Documentation . . . . .	459
7.103.4.1 QList . . . . .	459
7.104qevercloud::SyncChunk Struct Reference . . . . .	459
7.104.1 Detailed Description . . . . .	460
7.104.2 Member Function Documentation . . . . .	460
7.104.2.1 operator!=(()) . . . . .	460
7.104.2.2 operator==(()) . . . . .	460
7.104.2.3 print() . . . . .	461
7.104.3 Member Data Documentation . . . . .	461
7.104.3.1 chunkHighUSN . . . . .	461
7.104.3.2 currentTime . . . . .	461
7.104.3.3 expungedLinkedNotebooks . . . . .	461
7.104.3.4 expungedNotebooks . . . . .	461
7.104.3.5 expungedNotes . . . . .	461
7.104.3.6 expungedSearches . . . . .	462
7.104.3.7 expungedTags . . . . .	462
7.104.3.8 linkedNotebooks . . . . .	462
7.104.3.9 localData . . . . .	462
7.104.3.10notebooks . . . . .	462
7.104.3.11notes . . . . .	462
7.104.3.12resources . . . . .	462
7.104.3.13searches . . . . .	463
7.104.3.14tags . . . . .	463
7.104.3.15updateCount . . . . .	463
7.104.4 Property Documentation . . . . .	463
7.104.4.1 QList . . . . .	463
7.105qevercloud::SyncChunkFilter Struct Reference . . . . .	463
7.105.1 Detailed Description . . . . .	464

7.105.2 Member Function Documentation . . . . .	464
7.105.2.1 operator"!=( ) . . . . .	464
7.105.2.2 operator==( ) . . . . .	464
7.105.2.3 print( ) . . . . .	465
7.105.3 Member Data Documentation . . . . .	465
7.105.3.1 includeExpunged . . . . .	465
7.105.3.2 includeLinkedNotebooks . . . . .	465
7.105.3.3 includeNoteApplicationDataFullMap . . . . .	465
7.105.3.4 includeNoteAttributes . . . . .	465
7.105.3.5 includeNotebooks . . . . .	465
7.105.3.6 includeNoteResourceApplicationDataFullMap . . . . .	466
7.105.3.7 includeNoteResources . . . . .	466
7.105.3.8 includeNotes . . . . .	466
7.105.3.9 includeResourceApplicationDataFullMap . . . . .	466
7.105.3.10includeResources . . . . .	466
7.105.3.11includeSearches . . . . .	466
7.105.3.12includeSharedNotes . . . . .	466
7.105.3.13includeTags . . . . .	467
7.105.3.14localData . . . . .	467
7.105.3.15notebookGuids . . . . .	467
7.105.3.16omitSharedNotebooks . . . . .	467
7.105.3.17requireNoteContentClass . . . . .	467
7.105.4 Property Documentation . . . . .	467
7.105.4.1 QSet . . . . .	467
7.106qevercloud::IDurableService::SyncRequest Struct Reference . . . . .	468
7.106.1 Constructor & Destructor Documentation . . . . .	468
7.106.1.1 SyncRequest( ) . . . . .	468
7.106.2 Member Data Documentation . . . . .	468
7.106.2.1 m_call . . . . .	468
7.106.2.2 m_description . . . . .	468

7.106.2.3 m_name . . . . .	468
7.107qevercloud::SyncState Struct Reference . . . . .	469
7.107.1 Detailed Description . . . . .	469
7.107.2 Member Function Documentation . . . . .	469
7.107.2.1 operator!=(()) . . . . .	469
7.107.2.2 operator==(()) . . . . .	470
7.107.2.3 print() . . . . .	470
7.107.3 Member Data Documentation . . . . .	470
7.107.3.1 currentTime . . . . .	470
7.107.3.2 fullSyncBefore . . . . .	470
7.107.3.3 localData . . . . .	470
7.107.3.4 updateCount . . . . .	470
7.107.3.5 uploaded . . . . .	471
7.107.3.6 userLastUpdated . . . . .	471
7.107.3.7 userMaxMessageEventId . . . . .	471
7.108qevercloud::Tag Struct Reference . . . . .	471
7.108.1 Detailed Description . . . . .	472
7.108.2 Member Function Documentation . . . . .	472
7.108.2.1 operator!=(()) . . . . .	472
7.108.2.2 operator==(()) . . . . .	472
7.108.2.3 print() . . . . .	472
7.108.3 Member Data Documentation . . . . .	472
7.108.3.1 guid . . . . .	473
7.108.3.2 localData . . . . .	473
7.108.3.3 name . . . . .	473
7.108.3.4 parentGuid . . . . .	473
7.108.3.5 updateSequenceNum . . . . .	473
7.109qevercloud::ThriftException Class Reference . . . . .	473
7.109.1 Detailed Description . . . . .	474
7.109.2 Member Enumeration Documentation . . . . .	474

7.109.2.1 Type . . . . .	474
7.109.3 Constructor & Destructor Documentation . . . . .	475
7.109.3.1 ThriftException() [1/3] . . . . .	475
7.109.3.2 ThriftException() [2/3] . . . . .	475
7.109.3.3 ThriftException() [3/3] . . . . .	475
7.109.3.4 ~ThriftException() . . . . .	475
7.109.4 Member Function Documentation . . . . .	475
7.109.4.1 exceptionData() . . . . .	475
7.109.4.2 operator!=() . . . . .	476
7.109.4.3 operator==() . . . . .	476
7.109.4.4 type() . . . . .	476
7.109.4.5 what() . . . . .	476
7.109.5 Friends And Related Function Documentation . . . . .	476
7.109.5.1 operator<< . . . . .	476
7.109.6 Member Data Documentation . . . . .	476
7.109.6.1 m_type . . . . .	476
7.110qevercloud::ThriftExceptionData Class Reference . . . . .	477
7.110.1 Detailed Description . . . . .	477
7.110.2 Constructor & Destructor Documentation . . . . .	477
7.110.2.1 ThriftExceptionData() . . . . .	477
7.110.3 Member Function Documentation . . . . .	477
7.110.3.1 throwException() . . . . .	478
7.110.4 Member Data Documentation . . . . .	478
7.110.4.1 m_type . . . . .	478
7.111qevercloud::Thumbnail Class Reference . . . . .	478
7.111.1 Detailed Description . . . . .	479
7.111.2 Member Enumeration Documentation . . . . .	479
7.111.2.1 ImageType . . . . .	479
7.111.3 Constructor & Destructor Documentation . . . . .	479
7.111.3.1 Thumbnail() [1/2] . . . . .	479

7.111.3.2 Thumbnail() [2/2] . . . . .	480
7.111.3.3 ~Thumbnail() . . . . .	480
7.111.4 Member Function Documentation . . . . .	480
7.111.4.1 createPostRequest() . . . . .	480
7.111.4.2 download() . . . . .	481
7.111.4.3 downloadAsync() . . . . .	481
7.111.4.4 setAuthenticationToken() . . . . .	481
7.111.4.5 setHost() . . . . .	482
7.111.4.6 setImageType() . . . . .	482
7.111.4.7 setShardId() . . . . .	482
7.111.4.8 setSize() . . . . .	482
7.111.5 Friends And Related Function Documentation . . . . .	483
7.111.5.1 operator<< [1/2] . . . . .	483
7.111.5.2 operator<< [2/2] . . . . .	483
7.112qevercloud::UpdateNotelfUsnMatchesResult Struct Reference . . . . .	483
7.112.1 Detailed Description . . . . .	483
7.112.2 Member Function Documentation . . . . .	484
7.112.2.1 operator"!=( ) . . . . .	484
7.112.2.2 operator==( ) . . . . .	484
7.112.2.3 print() . . . . .	484
7.112.3 Member Data Documentation . . . . .	484
7.112.3.1 localData . . . . .	484
7.112.3.2 note . . . . .	484
7.112.3.3 updated . . . . .	485
7.113qevercloud::User Struct Reference . . . . .	485
7.113.1 Detailed Description . . . . .	485
7.113.2 Member Function Documentation . . . . .	486
7.113.2.1 operator"!=( ) . . . . .	486
7.113.2.2 operator==( ) . . . . .	486
7.113.2.3 print() . . . . .	486



7.113.3 Member Data Documentation . . . . .	486
7.113.3.1 accounting . . . . .	486
7.113.3.2 accountLimits . . . . .	486
7.113.3.3 active . . . . .	486
7.113.3.4 attributes . . . . .	487
7.113.3.5 businessUserInfo . . . . .	487
7.113.3.6 created . . . . .	487
7.113.3.7 deleted . . . . .	487
7.113.3.8 email . . . . .	487
7.113.3.9 id . . . . .	487
7.113.3.10 localData . . . . .	487
7.113.3.11 name . . . . .	488
7.113.3.12 photoLastUpdated . . . . .	488
7.113.3.13 photoUrl . . . . .	488
7.113.3.14 privilege . . . . .	488
7.113.3.15 serviceLevel . . . . .	488
7.113.3.16 shardId . . . . .	488
7.113.3.17 timezone . . . . .	489
7.113.3.18 updated . . . . .	489
7.113.3.19 username . . . . .	489
7.114 qevercloud::UserAttributes Struct Reference . . . . .	489
7.114.1 Detailed Description . . . . .	490
7.114.2 Member Function Documentation . . . . .	490
7.114.2.1 operator"!=( ) . . . . .	491
7.114.2.2 operator==( ) . . . . .	491
7.114.2.3 print( ) . . . . .	491
7.114.3 Member Data Documentation . . . . .	491
7.114.3.1 businessAddress . . . . .	491
7.114.3.2 clipFullPage . . . . .	491
7.114.3.3 comments . . . . .	491

7.114.3.4 dailyEmailLimit . . . . .	492
7.114.3.5 dateAgreedToTermsOfService . . . . .	492
7.114.3.6 defaultLatitude . . . . .	492
7.114.3.7 defaultLocationName . . . . .	492
7.114.3.8 defaultLongitude . . . . .	492
7.114.3.9 educationalDiscount . . . . .	492
7.114.3.10emailAddressLastConfirmed . . . . .	492
7.114.3.11emailOptOutDate . . . . .	493
7.114.3.12groupName . . . . .	493
7.114.3.13hideSponsorBilling . . . . .	493
7.114.3.14incomingEmailAddress . . . . .	493
7.114.3.15localData . . . . .	493
7.114.3.16maxReferrals . . . . .	493
7.114.3.17optOutMachineLearning . . . . .	493
7.114.3.18partnerEmailOptInDate . . . . .	494
7.114.3.19passwordUpdated . . . . .	494
7.114.3.20preactivation . . . . .	494
7.114.3.21preferredCountry . . . . .	494
7.114.3.22preferredLanguage . . . . .	494
7.114.3.23recentMailedAddresses . . . . .	494
7.114.3.24recognitionLanguage . . . . .	495
7.114.3.25referrerCode . . . . .	495
7.114.3.26referralCount . . . . .	495
7.114.3.27referralProof . . . . .	495
7.114.3.28reminderEmailConfig . . . . .	495
7.114.3.29salesforcePushEnabled . . . . .	495
7.114.3.30sentEmailCount . . . . .	495
7.114.3.31sentEmailDate . . . . .	496
7.114.3.32shouldLogClientEvent . . . . .	496
7.114.3.33twitterId . . . . .	496

7.114.3.34 <code>twitterUserName</code> . . . . .	496
7.114.3.35 <code>useEmailAutoFiling</code> . . . . .	496
7.114.3.36 <code>viewedPromotions</code> . . . . .	496
7.115 <code>qevercloud::UserIdentity</code> Struct Reference . . . . .	496
7.115.1 Detailed Description . . . . .	497
7.115.2 Member Function Documentation . . . . .	497
7.115.2.1 <code>operator!=()</code> . . . . .	497
7.115.2.2 <code>operator==()</code> . . . . .	497
7.115.2.3 <code>print()</code> . . . . .	498
7.115.3 Member Data Documentation . . . . .	498
7.115.3.1 <code>localData</code> . . . . .	498
7.115.3.2 <code>longIdentifier</code> . . . . .	498
7.115.3.3 <code>stringIdentifier</code> . . . . .	498
7.115.3.4 <code>type</code> . . . . .	498
7.116 <code>qevercloud::UserProfile</code> Struct Reference . . . . .	498
7.116.1 Detailed Description . . . . .	499
7.116.2 Member Function Documentation . . . . .	499
7.116.2.1 <code>operator!=()</code> . . . . .	499
7.116.2.2 <code>operator==()</code> . . . . .	499
7.116.2.3 <code>print()</code> . . . . .	499
7.116.3 Member Data Documentation . . . . .	500
7.116.3.1 <code>attributes</code> . . . . .	500
7.116.3.2 <code>email</code> . . . . .	500
7.116.3.3 <code>id</code> . . . . .	500
7.116.3.4 <code>joined</code> . . . . .	500
7.116.3.5 <code>localData</code> . . . . .	500
7.116.3.6 <code>name</code> . . . . .	500
7.116.3.7 <code>photoLastUpdated</code> . . . . .	500
7.116.3.8 <code>photoUrl</code> . . . . .	501
7.116.3.9 <code>role</code> . . . . .	501

7.116.3.10status . . . . .	501
7.116.3.11username . . . . .	501
7.117qevercloud::UserStoreServer Class Reference . . . . .	501
7.117.1 Detailed Description . . . . .	503
7.117.2 Constructor & Destructor Documentation . . . . .	503
7.117.2.1 UserStoreServer() . . . . .	503
7.117.3 Member Function Documentation . . . . .	503
7.117.3.1 authenticateLongSessionRequest . . . . .	503
7.117.3.2 authenticateLongSessionRequestReady . . . . .	504
7.117.3.3 authenticateToBusinessRequest . . . . .	504
7.117.3.4 authenticateToBusinessRequestReady . . . . .	504
7.117.3.5 checkVersionRequest . . . . .	504
7.117.3.6 checkVersionRequestReady . . . . .	504
7.117.3.7 completeTwoFactorAuthenticationRequest . . . . .	504
7.117.3.8 completeTwoFactorAuthenticationRequestReady . . . . .	505
7.117.3.9 getAccountLimitsRequest . . . . .	505
7.117.3.10getAccountLimitsRequestReady . . . . .	505
7.117.3.11getBootstrapInfoRequest . . . . .	505
7.117.3.12getBootstrapInfoRequestReady . . . . .	505
7.117.3.13getPublicUserInfoRequest . . . . .	505
7.117.3.14getPublicUserInfoRequestReady . . . . .	506
7.117.3.15getUserRequest . . . . .	506
7.117.3.16getUserRequestReady . . . . .	506
7.117.3.17getUserUrlsRequest . . . . .	506
7.117.3.18getUserUrlsRequestReady . . . . .	506
7.117.3.19inviteToBusinessRequest . . . . .	506
7.117.3.20inviteToBusinessRequestReady . . . . .	506
7.117.3.21listBusinessInvitationsRequest . . . . .	507
7.117.3.22listBusinessInvitationsRequestReady . . . . .	507
7.117.3.23listBusinessUsersRequest . . . . .	507

7.117.3.24	listBusinessUsersRequestReady	507
7.117.3.25	onAuthenticateLongSessionRequestReady	507
7.117.3.26	onAuthenticateToBusinessRequestReady	507
7.117.3.27	onCheckVersionRequestReady	508
7.117.3.28	onCompleteTwoFactorAuthenticationRequestReady	508
7.117.3.29	onGetAccountLimitsRequestReady	508
7.117.3.30	onGetBootstrapInfoRequestReady	508
7.117.3.31	onGetPublicUserInfoRequestReady	508
7.117.3.32	onGetUserRequestReady	508
7.117.3.33	onGetUserUrlsRequestReady	509
7.117.3.34	onInviteToBusinessRequestReady	509
7.117.3.35	onListBusinessInvitationsRequestReady	509
7.117.3.36	onListBusinessUsersRequestReady	509
7.117.3.37	onRemoveFromBusinessRequestReady	509
7.117.3.38	onRequest	509
7.117.3.39	onRevokeLongSessionRequestReady	510
7.117.3.40	onUpdateBusinessUserIdentifierRequestReady	510
7.117.3.41	removeFromBusinessRequest	510
7.117.3.42	removeFromBusinessRequestReady	510
7.117.3.43	revokeLongSessionRequest	510
7.117.3.44	revokeLongSessionRequestReady	510
7.117.3.45	updateBusinessUserIdentifierRequest	510
7.117.3.46	updateBusinessUserIdentifierRequestReady	511
7.118	qevercloud::UserUrls Struct Reference	511
7.118.1	Member Function Documentation	511
7.118.1.1	operator"!=( )	511
7.118.1.2	operator==( )	512
7.118.1.3	print( )	512
7.118.2	Member Data Documentation	512
7.118.2.1	localData	512
7.118.2.2	messageStoreUrl	512
7.118.2.3	noteStoreUrl	512
7.118.2.4	userStoreUrl	512
7.118.2.5	userWebSocketUrl	513
7.118.2.6	utilityUrl	513
7.118.2.7	webApiUrlPrefix	513

<b>8 File Documentation</b>	<b>515</b>
8.1 AsyncResult.h File Reference . . . . .	515
8.2 Constants.h File Reference . . . . .	515
8.3 DurableService.h File Reference . . . . .	520
8.4 EDAMErrorCode.h File Reference . . . . .	520
8.5 EventLoopFinisher.h File Reference . . . . .	524
8.6 EverCloudException.h File Reference . . . . .	525
8.7 Exceptions.h File Reference . . . . .	525
8.8 Export.h File Reference . . . . .	526
8.8.1 Macro Definition Documentation . . . . .	526
8.8.1.1 QEVERCLOUD_EXPORT . . . . .	526
8.9 Globals.h File Reference . . . . .	526
8.10 Helpers.h File Reference . . . . .	527
8.11 InkNotelImageDownloader.h File Reference . . . . .	527
8.12 Log.h File Reference . . . . .	528
8.12.1 Macro Definition Documentation . . . . .	529
8.12.1.1 __QEVERCLOUD_LOG_BASE . . . . .	529
8.12.1.2 QEC_DEBUG . . . . .	529
8.12.1.3 QEC_ERROR . . . . .	530
8.12.1.4 QEC_INFO . . . . .	530
8.12.1.5 QEC_TRACE . . . . .	530
8.12.1.6 QEC_WARNING . . . . .	530
8.13 OAuth.h File Reference . . . . .	530
8.14 Optional.h File Reference . . . . .	531
8.15 Printable.h File Reference . . . . .	531
8.16 QEverCloud.h File Reference . . . . .	532
8.17 QEverCloudOAuth.h File Reference . . . . .	532
8.18 README.md File Reference . . . . .	532
8.19 RequestContext.h File Reference . . . . .	532
8.20 Servers.h File Reference . . . . .	533
8.21 Services.h File Reference . . . . .	533
8.22 Thumbnail.h File Reference . . . . .	534
8.23 Types.h File Reference . . . . .	535
<b>Index</b>	<b>539</b>

# Chapter 1

## QEverCloud

### Unofficial Evernote Cloud API for Qt

#### 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 can also be generated in the form of a .qch file which you can register with your copy of Qt Creator to have context-sensitive help. See below for more details.

#### How to contribute

See contribution guide for detailed info.

#### Downloads

Prebuilt versions of the library can be downloaded from the following locations:

- Stable version:
  - Windows binaries:
    - \* [MSVC 2019 32 bit Qt 5.15.2](#)
    - \* [MSVC 2019 64 bit Qt 5.15.2](#)
  - [Mac binary](#) (built with Qt 5.15.2)
  - [Linux binary](#) built on Ubuntu 20.04 with Qt 5.12.8
- Unstable version:
  - Windows binaries:
    - \* [MSVC 2019 32 bit Qt 5.15.2](#)
    - \* [MSVC 2019 64 bit Qt 5.15.2](#)
  - [Mac binary](#) (built with Qt 5.15.2)
  - [Linux binary](#) built on Ubuntu 20.04 with Qt 5.12.8

## How to build

QEverCloud uses CMake build system which can be used as simply as follows (on Unix platforms):

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=<...> ../
make
make install
```

The library can be built and shipped either as a static library or a shared library. DLL export/import symbols necessary for Windows platform are supported.

QEverCloud uses C++14 standard. CMake automatically checks whether the compiler is capable enough of building QEverCloud so if the pre-build configuration step was successful, the build step should be successful as well. Known capable compilers are g++ 9 or later and Visual Studio 2019 or later.

The recommended version of Qt5 for building the library is the latest Qt 5.15. However, it is also known to build and work with Qt 5.12.

QEverCloud depends on the following Qt components:

- Qt5Core
- Qt5Widgets
- Qt5Network
- (Optional) Qt5WebKit and Qt5WebKitWidgets
- (Optional) Qt5WebEngine and Qt5WebEngineWidgets

The dependencies on Qt5WebKit or Qt5WebEngine are only actual if the library is built with OAuth support. But even then there is an option to build the library with OAuth support but without the dependency on either of these components. More on this below.

By default the library is built with OAuth support and uses Qt5WebEngine for it. The following cmake parameters are available to alter this behaviour:

- `-DBUILD_WITH_OAUTH_SUPPORT=NO` would disable building with OAuth support entirely.
- `-DUSE_QT5_WEBKIT=ON` would build the library with OAuth using Qt5WebKit for web page rendering.
- `-DQEVERCLOUD_USE_SYSTEM_BROWSER=ON` would build the library with OAuth not using either Qt5WebKit or Qt5WebEngine but instead delegating some portion of OAuth procedure to the system browser.

If Qt5's Qt5Test module is found during the pre-build configuration step, the unit tests are enabled and can be run with `make test` and more verbose `make check` commands.

Other available CMake configurations options:

**BUILD\_DOCUMENTATION** - when *ON*, attempts to find Doxygen and in case of success adds *doc* target so the documentation can be built using `make doc` command after the pre-build configuration step. By default this option is on.

**BUILD\_QCH\_DOCUMENTATION** - when *ON*, passes instructions on to Doxygen to build the documentation in *qch* format. This option only has any meaning if **BUILD\_DOCUMENTATION** option is on. By default this option is off.



**BUILD\_SHARED** - when *ON*, CMake configures the build for the shared library. By default this option is on.

**BUILD\_WITH\_Q\_NAMESPACE** - when *ON*, `Q_NAMESPACE` and `Q_ENUM_NS` macros are used to add introspection capabilities to enumerations within `qevercloud` namespace. Qt  $\geq$  5.8 is required to enable this option. By default this option is enabled.

**BUILD\_TRANSLATIONS** - when *ON*, builds and installs translation files for translatable strings from QEverCloud.

If **BUILD\_SHARED** is *ON*, `make install` installs CMake module necessary for applications using CMake's `find_package` command to find the installation of QEverCloud.

It is possible to build the library with enabled sanitizers using additional CMake options:

- `-DSANITIZE_ADDRESS=ON` to enable address sanitizer
- `-DSANITIZE_MEMORY=ON` to enable memory sanitizer
- `-DSANITIZE_THREAD=ON` to enable thread sanitizer
- `-DSANITIZE_UNDEFINED=ON` to enable undefined behaviour sanitizer

## 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 can also be used if needed.

## Seeding random numbers generator for Qt < 5.10

QEverCloud requires random numbers generator for OAuth procedure. When QEverCloud is built against Qt  $\geq$  5.10, it uses `QRandomGenerator` which is cryptographically secure on supported platforms and is seeded by Qt internals. With Qt < 5.10 QEverCloud uses `qrand`. It requires the client application to call `qsrand` with seed value before using OAuth calls of QEverCloud. So if you are using QEverCloud built with Qt < 5.10, make sure to call `qsrand` before using QEverCloud's OAuth.

## Related projects

- [QEverCloudGenerator](#) repository hosts code generating parser of [Evernote Thrift IDL files](#). This parser is used to autogenerate a portion of QEverCloud's headers and sources.
- [libquentier](#) is a library for creating feature rich full sync Evernote clients built on top of QEverCloud
- [Quentier](#) is an open source desktop note taking app capable of working as Evernote client built on top of [libquentier](#) and QEverCloud



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">qevercloud</a> . . . . .	19
--------------------------------------	----



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

qevercloud::IDurableService::AsyncRequest . . . . .	97
exception	
qevercloud::EverCloudException . . . . .	151
qevercloud::EvernoteException . . . . .	159
qevercloud::EDAMInvalidContactsException . . . . .	128
qevercloud::EDAMNotFoundException . . . . .	133
qevercloud::EDAMSystemException . . . . .	137
qevercloud::EDAMSystemExceptionAuthExpired . . . . .	140
qevercloud::EDAMSystemExceptionRateLimitReached . . . . .	144
qevercloud::EDAMUserException . . . . .	146
qevercloud::NetworkException . . . . .	284
qevercloud::ThriftException . . . . .	473
qevercloud::IDurableService . . . . .	171
qevercloud::ILogger . . . . .	172
qevercloud::InkNoteImageDownloader . . . . .	173
qevercloud::IRequestContext . . . . .	243
qevercloud::IRetryPolicy . . . . .	245
qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator . . . . .	246
qevercloud::QAssociativeContainerConstReferenceWrapper< Container >::iterator . . . . .	247
qevercloud::Optional< T > . . . . .	401
qevercloud::Optional< bool > . . . . .	401
qevercloud::Optional< BusinessInvitationStatus > . . . . .	401
qevercloud::Optional< BusinessUserRole > . . . . .	401
qevercloud::Optional< BusinessUserStatus > . . . . .	401
qevercloud::Optional< CanMoveToContainerStatus > . . . . .	401
qevercloud::Optional< ContactType > . . . . .	401
qevercloud::Optional< double > . . . . .	401
qevercloud::Optional< Guid > . . . . .	401
qevercloud::Optional< IdentityID > . . . . .	401
qevercloud::Optional< MessageEventID > . . . . .	401
qevercloud::Optional< MessageThreadID > . . . . .	401
qevercloud::Optional< NoteSortOrder > . . . . .	401
qevercloud::Optional< PremiumOrderStatus > . . . . .	401
qevercloud::Optional< PrivilegeLevel > . . . . .	401
qevercloud::Optional< QByteArray > . . . . .	401

qevercloud::Optional< qevercloud::Accounting > . . . . .	401
qevercloud::Optional< qevercloud::AccountLimits > . . . . .	401
qevercloud::Optional< qevercloud::BusinessNotebook > . . . . .	401
qevercloud::Optional< qevercloud::BusinessUserAttributes > . . . . .	401
qevercloud::Optional< qevercloud::BusinessUserInfo > . . . . .	401
qevercloud::Optional< qevercloud::CanMoveToContainerRestrictions > . . . . .	401
qevercloud::Optional< qevercloud::Contact > . . . . .	401
qevercloud::Optional< qevercloud::Data > . . . . .	401
qevercloud::Optional< qevercloud::EDAMNotFoundException > . . . . .	401
qevercloud::Optional< qevercloud::EDAMUserException > . . . . .	401
qevercloud::Optional< qevercloud::Identity > . . . . .	401
qevercloud::Optional< qevercloud::LazyMap > . . . . .	401
qevercloud::Optional< qevercloud::Note > . . . . .	401
qevercloud::Optional< qevercloud::NoteAttributes > . . . . .	401
qevercloud::Optional< qevercloud::NotebookRecipientSettings > . . . . .	401
qevercloud::Optional< qevercloud::NotebookRestrictions > . . . . .	401
qevercloud::Optional< qevercloud::NoteFilter > . . . . .	401
qevercloud::Optional< qevercloud::NoteLimits > . . . . .	401
qevercloud::Optional< qevercloud::NoteRestrictions > . . . . .	401
qevercloud::Optional< qevercloud::NoteShareRelationshipRestrictions > . . . . .	401
qevercloud::Optional< qevercloud::PublicUserInfo > . . . . .	401
qevercloud::Optional< qevercloud::Publishing > . . . . .	401
qevercloud::Optional< qevercloud::ResourceAttributes > . . . . .	401
qevercloud::Optional< qevercloud::SavedSearchScope > . . . . .	401
qevercloud::Optional< qevercloud::SharedNotebookRecipientSettings > . . . . .	401
qevercloud::Optional< qevercloud::ShareRelationshipRestrictions > . . . . .	401
qevercloud::Optional< qevercloud::User > . . . . .	401
qevercloud::Optional< qevercloud::UserAttributes > . . . . .	401
qevercloud::Optional< qevercloud::UserIdentity > . . . . .	401
qevercloud::Optional< qevercloud::UserUrls > . . . . .	401
qevercloud::Optional< qint16 > . . . . .	401
qevercloud::Optional< qint32 > . . . . .	401
qevercloud::Optional< qint64 > . . . . .	401
qevercloud::Optional< QList< EDAMInvalidContactReason > > . . . . .	401
qevercloud::Optional< QList< Guid > > . . . . .	401
qevercloud::Optional< QList< IdentityID > > . . . . .	401
qevercloud::Optional< QList< qevercloud::Contact > > . . . . .	401
qevercloud::Optional< QList< qevercloud::InvitationShareRelationship > > . . . . .	401
qevercloud::Optional< QList< qevercloud::LinkedNotebook > > . . . . .	401
qevercloud::Optional< QList< qevercloud::ManageNotebookSharesError > > . . . . .	401
qevercloud::Optional< QList< qevercloud::ManageNoteSharesError > > . . . . .	401
qevercloud::Optional< QList< qevercloud::MemberShareRelationship > > . . . . .	401
qevercloud::Optional< QList< qevercloud::Note > > . . . . .	401
qevercloud::Optional< QList< qevercloud::Notebook > > . . . . .	401
qevercloud::Optional< QList< qevercloud::NotebookDescriptor > > . . . . .	401
qevercloud::Optional< QList< qevercloud::NoteInvitationShareRelationship > > . . . . .	401
qevercloud::Optional< QList< qevercloud::NoteMemberShareRelationship > > . . . . .	401
qevercloud::Optional< QList< qevercloud::RelatedContent > > . . . . .	401
qevercloud::Optional< QList< qevercloud::RelatedContentImage > > . . . . .	401
qevercloud::Optional< QList< qevercloud::Resource > > . . . . .	401
qevercloud::Optional< QList< qevercloud::SavedSearch > > . . . . .	401
qevercloud::Optional< QList< qevercloud::SharedNote > > . . . . .	401
qevercloud::Optional< QList< qevercloud::SharedNotebook > > . . . . .	401
qevercloud::Optional< QList< qevercloud::Tag > > . . . . .	401
qevercloud::Optional< QList< qevercloud::UserIdentity > > . . . . .	401
qevercloud::Optional< QList< qevercloud::UserProfile > > . . . . .	401
qevercloud::Optional< QList< qint64 > > . . . . .	401
qevercloud::Optional< QList< UserID > > . . . . .	401

qevercloud::Optional< QMap< Guid, qint32 > > . . . . .	401
qevercloud::Optional< QMap< QString, QString > > . . . . .	401
qevercloud::Optional< QSet< QString > > . . . . .	401
qevercloud::Optional< QSet< RelatedContentType > > . . . . .	401
qevercloud::Optional< QString > . . . . .	401
qevercloud::Optional< QStringList > . . . . .	401
qevercloud::Optional< QueryFormat > . . . . .	401
qevercloud::Optional< RecipientStatus > . . . . .	401
qevercloud::Optional< RelatedContentAccess > . . . . .	401
qevercloud::Optional< RelatedContentType > . . . . .	401
qevercloud::Optional< ReminderEmailConfig > . . . . .	401
qevercloud::Optional< ServiceLevel > . . . . .	401
qevercloud::Optional< SharedNotebookInstanceRestrictions > . . . . .	401
qevercloud::Optional< SharedNotebookPrivilegeLevel > . . . . .	401
qevercloud::Optional< SharedNotePrivilegeLevel > . . . . .	401
qevercloud::Optional< ShareRelationshipPrivilegeLevel > . . . . .	401
qevercloud::Optional< Timestamp > . . . . .	401
qevercloud::Optional< UserID > . . . . .	401
qevercloud::Optional< UserIdentityType > . . . . .	401
qevercloud::Printable . . . . .	409
qevercloud::Accounting . . . . .	89
qevercloud::AccountLimits . . . . .	94
qevercloud::AuthenticationResult . . . . .	101
qevercloud::BootstrapInfo . . . . .	104
qevercloud::BootstrapProfile . . . . .	106
qevercloud::BootstrapSettings . . . . .	108
qevercloud::BusinessInvitation . . . . .	111
qevercloud::BusinessNotebook . . . . .	114
qevercloud::BusinessUserAttributes . . . . .	115
qevercloud::BusinessUserInfo . . . . .	118
qevercloud::CanMoveToContainerRestrictions . . . . .	120
qevercloud::Contact . . . . .	121
qevercloud::CreateOrUpdateNotebookSharesResult . . . . .	124
qevercloud::Data . . . . .	126
qevercloud::EDAMInvalidContactsException . . . . .	128
qevercloud::EDAMNotFoundException . . . . .	133
qevercloud::EDAMSystemException . . . . .	137
qevercloud::EDAMUserException . . . . .	146
qevercloud::EverCloudLocalData . . . . .	156
qevercloud::EvernoteOAuthWebView::OAuthResult . . . . .	399
qevercloud::Identity . . . . .	168
qevercloud::InvitationShareRelationship . . . . .	241
qevercloud::LazyMap . . . . .	263
qevercloud::LinkedNotebook . . . . .	265
qevercloud::ManageNotebookSharesError . . . . .	269
qevercloud::ManageNotebookSharesParameters . . . . .	271
qevercloud::ManageNotebookSharesResult . . . . .	273
qevercloud::ManageNoteSharesError . . . . .	275
qevercloud::ManageNoteSharesParameters . . . . .	277
qevercloud::ManageNoteSharesResult . . . . .	280
qevercloud::MemberShareRelationship . . . . .	282
qevercloud::Note . . . . .	288
qevercloud::NoteAttributes . . . . .	293
qevercloud::Notebook . . . . .	299
qevercloud::NotebookDescriptor . . . . .	304
qevercloud::NotebookRecipientSettings . . . . .	306
qevercloud::NotebookRestrictions . . . . .	308
qevercloud::NotebookShareTemplate . . . . .	315

qevercloud::NoteCollectionCounts	317
qevercloud::NoteEmailParameters	320
qevercloud::NoteFilter	322
qevercloud::NoteInvitationShareRelationship	326
qevercloud::NoteLimits	328
qevercloud::NoteList	330
qevercloud::NoteMemberShareRelationship	333
qevercloud::NoteMetadata	335
qevercloud::NoteRestrictions	338
qevercloud::NoteResultSpec	341
qevercloud::NoteShareRelationshipRestrictions	344
qevercloud::NoteShareRelationships	346
qevercloud::NotesMetadataList	348
qevercloud::NotesMetadataResultSpec	351
qevercloud::NoteVersionId	396
qevercloud::PublicUserInfo	412
qevercloud::Publishing	414
qevercloud::RelatedContent	418
qevercloud::RelatedContentImage	422
qevercloud::RelatedQuery	424
qevercloud::RelatedResult	427
qevercloud::RelatedResultSpec	430
qevercloud::Resource	434
qevercloud::ResourceAttributes	437
qevercloud::SavedSearch	440
qevercloud::SavedSearchScope	443
qevercloud::SharedNote	445
qevercloud::SharedNotebook	447
qevercloud::SharedNotebookRecipientSettings	451
qevercloud::SharedNoteTemplate	453
qevercloud::ShareRelationshipRestrictions	455
qevercloud::ShareRelationships	457
qevercloud::SyncChunk	459
qevercloud::SyncChunkFilter	463
qevercloud::SyncState	469
qevercloud::Tag	471
qevercloud::UpdateNoteIfUsnMatchesResult	483
qevercloud::User	485
qevercloud::UserAttributes	489
qevercloud::UserIdentity	496
qevercloud::UserProfile	498
qevercloud::UserUrls	511
qevercloud::QAssociativeContainerConstReferenceWrapper< Container >	416
qevercloud::QAssociativeContainerReferenceWrapper< Container >	417
QDialog	
qevercloud::EvernoteOAuthDialog	162
QObject	
qevercloud::AsyncResult	98
qevercloud::EventLoopFinisher	150
qevercloud::EverCloudExceptionData	153
qevercloud::EvernoteExceptionData	161
qevercloud::EDAMInvalidContactsExceptionData	131
qevercloud::EDAMNotFoundExceptionData	135
qevercloud::EDAMSystemExceptionData	142
qevercloud::EDAMSystemExceptionAuthExpiredData	141
qevercloud::EDAMSystemExceptionRateLimitReachedData	145
qevercloud::EDAMUserExceptionData	149
qevercloud::NetworkExceptionData	286



qevercloud::ThriftExceptionData . . . . .	477
qevercloud::INoteStore . . . . .	177
qevercloud::IUserStore . . . . .	249
qevercloud::NoteStoreServer . . . . .	354
qevercloud::UserStoreServer . . . . .	501
QWidget	
qevercloud::EvernoteOAuthWebView . . . . .	165
qevercloud::IDurableService::SyncRequest . . . . .	468
qevercloud::Thumbnail . . . . .	478



## Chapter 4

# Class Index

### 4.1 Class List

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

qevercloud::Accounting . . . . .	89
qevercloud::AccountLimits . . . . .	94
qevercloud::IDurableService::AsyncRequest . . . . .	97
qevercloud::AsyncResult	
Returned by asynchronous versions of functions . . . . .	98
qevercloud::AuthenticationResult . . . . .	101
qevercloud::BootstrapInfo . . . . .	104
qevercloud::BootstrapProfile . . . . .	106
qevercloud::BootstrapSettings . . . . .	108
qevercloud::BusinessInvitation . . . . .	111
qevercloud::BusinessNotebook . . . . .	114
qevercloud::BusinessUserAttributes . . . . .	115
qevercloud::BusinessUserInfo . . . . .	118
qevercloud::CanMoveToContainerRestrictions . . . . .	120
qevercloud::Contact . . . . .	121
qevercloud::CreateOrUpdateNotebookSharesResult . . . . .	124
qevercloud::Data . . . . .	126
qevercloud::EDAMInvalidContactsException . . . . .	128
qevercloud::EDAMInvalidContactsExceptionData . . . . .	131
qevercloud::EDAMNotFoundException . . . . .	133
qevercloud::EDAMNotFoundExceptionData . . . . .	135
qevercloud::EDAMSystemException . . . . .	137
qevercloud::EDAMSystemExceptionAuthExpired . . . . .	140
qevercloud::EDAMSystemExceptionAuthExpiredData . . . . .	141
qevercloud::EDAMSystemExceptionData . . . . .	142
qevercloud::EDAMSystemExceptionRateLimitReached . . . . .	144
qevercloud::EDAMSystemExceptionRateLimitReachedData . . . . .	145
qevercloud::EDAMUserException . . . . .	146
qevercloud::EDAMUserExceptionData . . . . .	149
qevercloud::EventLoopFinisher . . . . .	150
qevercloud::EverCloudException . . . . .	151
qevercloud::EverCloudExceptionData	
EverCloudException counterpart for asynchronous API . . . . .	153

<a href="#">qevercloud::EverCloudLocalData</a>	
Several data elements which are not synchronized with Evernote service but which are nevertheless useful in applications using QEverCloud to implement feature rich full sync Evernote clients. Values of this class' types are contained within QEverCloud types corresponding to actual Evernote API types	156
<a href="#">qevercloud::EvernoteException</a>	159
<a href="#">qevercloud::EvernoteExceptionData</a>	161
<a href="#">qevercloud::EvernoteOAuthDialog</a>	
Authorizes your app with the Evernote service by means of OAuth authentication	162
<a href="#">qevercloud::EvernoteOAuthWebView</a>	
The class is tailored specifically for OAuth authorization with Evernote	165
<a href="#">qevercloud::Identity</a>	168
<a href="#">qevercloud::IDurableService</a>	171
<a href="#">qevercloud::ILogger</a>	172
<a href="#">qevercloud::InkNoteImageDownloader</a>	
InkNoteImageDownloader class is for downloading the images of ink notes which can be created with the official Evernote client on Windows (only with it, at least at the time of this writing)	173
<a href="#">qevercloud::INoteStore</a>	177
<a href="#">qevercloud::InvitationShareRelationship</a>	241
<a href="#">qevercloud::IRequestContext</a>	243
<a href="#">qevercloud::IRetryPolicy</a>	245
<a href="#">qevercloud::QAssociativeContainerReferenceWrapper&lt; Container &gt;::iterator</a>	246
<a href="#">qevercloud::QAssociativeContainerConstReferenceWrapper&lt; Container &gt;::iterator</a>	247
<a href="#">qevercloud::IUserStore</a>	249
<a href="#">qevercloud::LazyMap</a>	263
<a href="#">qevercloud::LinkedNotebook</a>	265
<a href="#">qevercloud::ManageNotebookSharesError</a>	269
<a href="#">qevercloud::ManageNotebookSharesParameters</a>	271
<a href="#">qevercloud::ManageNotebookSharesResult</a>	273
<a href="#">qevercloud::ManageNoteSharesError</a>	275
<a href="#">qevercloud::ManageNoteSharesParameters</a>	277
<a href="#">qevercloud::ManageNoteSharesResult</a>	280
<a href="#">qevercloud::MemberShareRelationship</a>	282
<a href="#">qevercloud::NetworkException</a>	
QNetworkReply level errors	284
<a href="#">qevercloud::NetworkExceptionData</a>	286
<a href="#">qevercloud::Note</a>	288
<a href="#">qevercloud::NoteAttributes</a>	293
<a href="#">qevercloud::Notebook</a>	299
<a href="#">qevercloud::NotebookDescriptor</a>	304
<a href="#">qevercloud::NotebookRecipientSettings</a>	306
<a href="#">qevercloud::NotebookRestrictions</a>	308
<a href="#">qevercloud::NotebookShareTemplate</a>	315
<a href="#">qevercloud::NoteCollectionCounts</a>	317
<a href="#">qevercloud::NoteEmailParameters</a>	320
<a href="#">qevercloud::NoteFilter</a>	322
<a href="#">qevercloud::NoteInvitationShareRelationship</a>	326
<a href="#">qevercloud::NoteLimits</a>	328
<a href="#">qevercloud::NoteList</a>	330
<a href="#">qevercloud::NoteMemberShareRelationship</a>	333
<a href="#">qevercloud::NoteMetadata</a>	335
<a href="#">qevercloud::NoteRestrictions</a>	338
<a href="#">qevercloud::NoteResultSpec</a>	341
<a href="#">qevercloud::NoteShareRelationshipRestrictions</a>	344
<a href="#">qevercloud::NoteShareRelationships</a>	346
<a href="#">qevercloud::NotesMetadataList</a>	348
<a href="#">qevercloud::NotesMetadataResultSpec</a>	351

qevercloud::NoteStoreServer	
Represents customizable server for NoteStore requests. It is primarily used for testing of Q←	
EverCloud	354
qevercloud::NoteVersionId	396
qevercloud::EvernoteOAuthWebView::OAuthResult	399
qevercloud::Optional< T >	401
qevercloud::Printable	409
qevercloud::PublicUserInfo	412
qevercloud::Publishing	414
qevercloud::QAssociativeContainerConstReferenceWrapper< Container >	416
qevercloud::QAssociativeContainerReferenceWrapper< Container >	417
qevercloud::RelatedContent	418
qevercloud::RelatedContentImage	422
qevercloud::RelatedQuery	424
qevercloud::RelatedResult	427
qevercloud::RelatedResultSpec	430
qevercloud::Resource	434
qevercloud::ResourceAttributes	437
qevercloud::SavedSearch	440
qevercloud::SavedSearchScope	443
qevercloud::SharedNote	445
qevercloud::SharedNotebook	447
qevercloud::SharedNotebookRecipientSettings	451
qevercloud::SharedNoteTemplate	453
qevercloud::ShareRelationshipRestrictions	455
qevercloud::ShareRelationships	457
qevercloud::SyncChunk	459
qevercloud::SyncChunkFilter	463
qevercloud::IDurableService::SyncRequest	468
qevercloud::SyncState	469
qevercloud::Tag	471
qevercloud::ThriftException	473
qevercloud::ThriftExceptionData	477
qevercloud::Thumbnail	
The class is for downloading thumbnails for notes and resources from Evernote servers	478
qevercloud::UpdateNoteIfUsnMatchesResult	483
qevercloud::User	485
qevercloud::UserAttributes	489
qevercloud::UserIdentity	496
qevercloud::UserProfile	498
qevercloud::UserStoreServer	
Represents customizable server for UserStore requests. It is primarily used for testing of Q←	
EverCloud	501
qevercloud::UserUrls	511



## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

AsyncResult.h	515
Constants.h	515
DurableService.h	520
EDAMErrorCode.h	520
EventLoopFinisher.h	524
EverCloudException.h	525
Exceptions.h	525
Export.h	526
Globals.h	526
Helpers.h	527
InkNoteImageDownloader.h	527
Log.h	528
OAuth.h	530
Optional.h	531
Printable.h	531
QEverCloud.h	532
QEverCloudOAuth.h	532
RequestContext.h	532
Servers.h	533
Services.h	533
Thumbnail.h	534
Types.h	535





## Chapter 6

# Namespace Documentation

### 6.1 qevercloud Namespace Reference

#### Classes

- struct [Accounting](#)
- struct [AccountLimits](#)
- class [AsyncResult](#)

*Returned by asynchronous versions of functions.*

- struct [AuthenticationResult](#)
  - struct [BootstrapInfo](#)
  - struct [BootstrapProfile](#)
  - struct [BootstrapSettings](#)
  - struct [BusinessInvitation](#)
  - struct [BusinessNotebook](#)
  - struct [BusinessUserAttributes](#)
  - struct [BusinessUserInfo](#)
  - struct [CanMoveToContainerRestrictions](#)
  - struct [Contact](#)
  - struct [CreateOrUpdateNotebookSharesResult](#)
  - struct [Data](#)
  - class [EDAMInvalidContactsException](#)
  - class [EDAMInvalidContactsExceptionData](#)
  - 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 [EverCloudLocalData](#)

The [EverCloudLocalData](#) class contains several data elements which are not synchronized with Evernote service but which are nevertheless useful in applications using QEverCloud to implement feature rich full sync Evernote clients. Values of this class' types are contained within QEverCloud types corresponding to actual Evernote API types.

- 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 [Identity](#)
- class [IDurableService](#)
- class [ILogger](#)
- class [InkNoteImageDownloader](#)

the [InkNoteImageDownloader](#) class is for downloading the images of ink notes which can be created with the official Evernote client on Windows (only with it, at least at the time of this writing).

- class [INoteStore](#)
- struct [InvitationShareRelationship](#)
- class [IRequestContext](#)
- struct [IRetryPolicy](#)
- class [IUserStore](#)
- struct [LazyMap](#)
- struct [LinkedNotebook](#)
- struct [ManageNotebookSharesError](#)
- struct [ManageNotebookSharesParameters](#)
- struct [ManageNotebookSharesResult](#)
- struct [ManageNoteSharesError](#)
- struct [ManageNoteSharesParameters](#)
- struct [ManageNoteSharesResult](#)
- struct [MemberShareRelationship](#)
- class [NetworkException](#)

The [NetworkException](#) class represents QNetworkReply level errors.

- class [NetworkExceptionData](#)
- struct [Note](#)
- struct [NoteAttributes](#)
- struct [Notebook](#)
- struct [NotebookDescriptor](#)
- struct [NotebookRecipientSettings](#)
- struct [NotebookRestrictions](#)
- struct [NotebookShareTemplate](#)
- struct [NoteCollectionCounts](#)
- struct [NoteEmailParameters](#)
- struct [NoteFilter](#)
- struct [NoteInvitationShareRelationship](#)
- struct [NoteLimits](#)
- struct [NoteList](#)
- struct [NoteMemberShareRelationship](#)
- struct [NoteMetadata](#)
- struct [NoteRestrictions](#)
- struct [NoteResultSpec](#)
- struct [NoteShareRelationshipRestrictions](#)
- struct [NoteShareRelationships](#)
- struct [NotesMetadataList](#)
- struct [NotesMetadataResultSpec](#)
- class [NoteStoreServer](#)

The [NoteStoreServer](#) class represents customizable server for NoteStore requests. It is primarily used for testing of QEverCloud.

- struct [NoteVersionId](#)
- class [Optional](#)
- class [Printable](#)
- struct [PublicUserInfo](#)
- struct [Publishing](#)
- class [QAssociativeContainerConstReferenceWrapper](#)
- class [QAssociativeContainerReferenceWrapper](#)
- struct [RelatedContent](#)
- struct [RelatedContentImage](#)
- struct [RelatedQuery](#)
- struct [RelatedResult](#)
- struct [RelatedResultSpec](#)
- struct [Resource](#)
- struct [ResourceAttributes](#)
- struct [SavedSearch](#)
- struct [SavedSearchScope](#)
- struct [SharedNote](#)
- struct [SharedNotebook](#)
- struct [SharedNotebookRecipientSettings](#)
- struct [SharedNoteTemplate](#)
- struct [ShareRelationshipRestrictions](#)
- struct [ShareRelationships](#)
- 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 [UpdateNoteIfUsnMatchesResult](#)
- struct [User](#)
- struct [UserAttributes](#)
- struct [UserIdentity](#)
- struct [UserProfile](#)
- class [UserStoreServer](#)

The [UserStoreServer](#) class represents customizable server for UserStore requests. It is primarily used for testing of QEverCloud.

- struct [UserUrls](#)

## Typedefs

- using [IRetryPolicyPtr](#) = std::shared\_ptr< [IRetryPolicy](#) >
- using [IDurableServicePtr](#) = std::shared\_ptr< [IDurableService](#) >
- using [EverCloudExceptionDataPtr](#) = std::shared\_ptr< [EverCloudExceptionData](#) >
- using [INoteStorePtr](#) = std::shared\_ptr< [INoteStore](#) >
- using [IUserStorePtr](#) = std::shared\_ptr< [IUserStore](#) >
- using [InvalidationSequenceNumber](#) = quint64
- using [IdentityID](#) = quint64
- using [UserID](#) = quint32
- using [Guid](#) = QString
- using [Timestamp](#) = quint64
- using [MessageEventID](#) = quint64
- using [MessageThreadID](#) = quint64
- using [ILoggerPtr](#) = std::shared\_ptr< [ILogger](#) >
- using [IRequestContextPtr](#) = std::shared\_ptr< [IRequestContext](#) >

## Enumerations

- enum `EDAMErrorCode` {  
`EDAMErrorCode::UNKNOWN` = 1, `EDAMErrorCode::BAD_DATA_FORMAT` = 2, `EDAMErrorCode::PERMISSION_DENIED` = 3, `EDAMErrorCode::INTERNAL_ERROR` = 4,  
`EDAMErrorCode::DATA_REQUIRED` = 5, `EDAMErrorCode::LIMIT_REACHED` = 6, `EDAMErrorCode::QUOTA_REACHED` = 7, `EDAMErrorCode::INVALID_AUTH` = 8,  
`EDAMErrorCode::AUTH_EXPIRED` = 9, `EDAMErrorCode::DATA_CONFLICT` = 10, `EDAMErrorCode::EMAIL_VALIDATION` = 11, `EDAMErrorCode::SHARD_UNAVAILABLE` = 12,  
`EDAMErrorCode::LEN_TOO_SHORT` = 13, `EDAMErrorCode::LEN_TOO_LONG` = 14, `EDAMErrorCode::TOO_FEW` = 15, `EDAMErrorCode::TOO_MANY` = 16,  
`EDAMErrorCode::UNSUPPORTED_OPERATION` = 17, `EDAMErrorCode::TAKEN_DOWN` = 18, `EDAMErrorCode::RATE_LIMIT_REACHED` = 19, `EDAMErrorCode::BUSINESS_SECURITY_LOGIN_REQUIRED` = 20,  
`EDAMErrorCode::DEVICE_LIMIT_REACHED` = 21, `EDAMErrorCode::OPENID_ALREADY_TAKEN` = 22, `EDAMErrorCode::INVALID_OPENID_TOKEN` = 23, `EDAMErrorCode::USER_NOT_ASSOCIATED` = 24,  
`EDAMErrorCode::USER_NOT_REGISTERED` = 25, `EDAMErrorCode::USER_ALREADY_ASSOCIATED` = 26, `EDAMErrorCode::ACCOUNT_CLEAR` = 27, `EDAMErrorCode::SSO_AUTHENTICATION_REQUIRED` = 28 }
- enum `EDAMInvalidContactReason` { `EDAMInvalidContactReason::BAD_ADDRESS`, `EDAMInvalidContactReason::DUPLICATE_CONTACT`, `EDAMInvalidContactReason::NO_CONNECTION` }
- enum `ShareRelationshipPrivilegeLevel` { `ShareRelationshipPrivilegeLevel::READ_NOTEBOOK` = 0, `ShareRelationshipPrivilegeLevel::READ_NOTEBOOK_PLUS_ACTIVITY` = 10, `ShareRelationshipPrivilegeLevel::MODIFY_NOTEBOOK_PLUS_ACTIVITY` = 20, `ShareRelationshipPrivilegeLevel::FULL_ACCESS` = 30 }
- enum `PrivilegeLevel` {  
`PrivilegeLevel::NORMAL` = 1, `PrivilegeLevel::PREMIUM` = 3, `PrivilegeLevel::VIP` = 5, `PrivilegeLevel::MANAGER` = 7,  
`PrivilegeLevel::SUPPORT` = 8, `PrivilegeLevel::ADMIN` = 9 }
- enum `ServiceLevel` { `ServiceLevel::BASIC` = 1, `ServiceLevel::PLUS` = 2, `ServiceLevel::PREMIUM` = 3, `ServiceLevel::BUSINESS` = 4 }
- enum `QueryFormat` { `QueryFormat::USER` = 1, `QueryFormat::SEXP` = 2 }
- enum `NoteSortOrder` {  
`NoteSortOrder::CREATED` = 1, `NoteSortOrder::UPDATED` = 2, `NoteSortOrder::RELEVANCE` = 3, `NoteSortOrder::UPDATE_SEQUENCE_NUMBER` = 4,  
`NoteSortOrder::TITLE` = 5 }
- enum `PremiumOrderStatus` {  
`PremiumOrderStatus::NONE` = 0, `PremiumOrderStatus::PENDING` = 1, `PremiumOrderStatus::ACTIVE` = 2,  
`PremiumOrderStatus::FAILED` = 3,  
`PremiumOrderStatus::CANCELLATION_PENDING` = 4, `PremiumOrderStatus::CANCELED` = 5 }
- enum `SharedNotebookPrivilegeLevel` {  
`SharedNotebookPrivilegeLevel::READ_NOTEBOOK` = 0, `SharedNotebookPrivilegeLevel::MODIFY_NOTEBOOK_PLUS_ACTIVITY` = 1, `SharedNotebookPrivilegeLevel::READ_NOTEBOOK_PLUS_ACTIVITY` = 2,  
`SharedNotebookPrivilegeLevel::GROUP` = 3,  
`SharedNotebookPrivilegeLevel::FULL_ACCESS` = 4, `SharedNotebookPrivilegeLevel::BUSINESS_FULL_ACCESS` = 5 }
- enum `SharedNotePrivilegeLevel` { `SharedNotePrivilegeLevel::READ_NOTE` = 0, `SharedNotePrivilegeLevel::MODIFY_NOTE` = 1, `SharedNotePrivilegeLevel::FULL_ACCESS` = 2 }
- enum `SponsoredGroupRole` { `SponsoredGroupRole::GROUP_MEMBER` = 1, `SponsoredGroupRole::GROUP_ADMIN` = 2, `SponsoredGroupRole::GROUP_OWNER` = 3 }
- enum `BusinessUserRole` { `BusinessUserRole::ADMIN` = 1, `BusinessUserRole::NORMAL` = 2 }
- enum `BusinessUserStatus` { `BusinessUserStatus::ACTIVE` = 1, `BusinessUserStatus::DEACTIVATED` = 2 }
- enum `SharedNotebookInstanceRestrictions` { `SharedNotebookInstanceRestrictions::ASSIGNED` = 1, `SharedNotebookInstanceRestrictions::NO_SHARED_NOTEBOOKS` = 2 }
- enum `ReminderEmailConfig` { `ReminderEmailConfig::DO_NOT_SEND` = 1, `ReminderEmailConfig::SEND_DAILY_EMAIL` = 2 }
- enum `BusinessInvitationStatus` { `BusinessInvitationStatus::APPROVED` = 0, `BusinessInvitationStatus::REQUESTED` = 1, `BusinessInvitationStatus::REDEEMED` = 2 }

- enum [ContactType](#) { [ContactType::EVERNOTE](#) = 1, [ContactType::SMS](#) = 2, [ContactType::FACEBOOK](#) = 3, [ContactType::EMAIL](#) = 4, [ContactType::TWITTER](#) = 5, [ContactType::LINKEDIN](#) = 6 }
- enum [EntityType](#) { [EntityType::NOTE](#) = 1, [EntityType::NOTEBOOK](#) = 2, [EntityType::WORKSPACE](#) = 3 }
- enum [RecipientStatus](#) { [RecipientStatus::NOT\\_IN\\_MY\\_LIST](#) = 1, [RecipientStatus::IN\\_MY\\_LIST](#) = 2, [RecipientStatus::IN\\_MY\\_LIST\\_AND\\_DEFAULT\\_NOTEBOOK](#) = 3 }
- enum [CanMoveToContainerStatus](#) { [CanMoveToContainerStatus::CAN\\_BE\\_MOVED](#) = 1, [CanMoveToContainerStatus::INSUFFICIENT\\_ENTITY\\_PRIVILEGE](#) = 2, [CanMoveToContainerStatus::INSUFFICIENT\\_CONTAINER\\_PRIVILEGE](#) = 3 }
- enum [RelatedContentType](#) { [RelatedContentType::NEWS\\_ARTICLE](#) = 1, [RelatedContentType::PROFILE\\_PERSON](#) = 2, [RelatedContentType::PROFILE\\_ORGANIZATION](#) = 3, [RelatedContentType::REFERENCE\\_MATERIAL](#) = 4 }
- enum [RelatedContentAccess](#) { [RelatedContentAccess::NOT\\_ACCESSIBLE](#) = 0, [RelatedContentAccess::DIRECT\\_LINK\\_ACCESS\\_OK](#) = 1, [RelatedContentAccess::DIRECT\\_LINK\\_LOGIN\\_REQUIRED](#) = 2, [RelatedContentAccess::DIRECT\\_LINK\\_EMBEDDED\\_VIEW](#) = 3 }
- enum [UserIdentityType](#) { [UserIdentityType::EVERNOTE\\_USERID](#) = 1, [UserIdentityType::EMAIL](#) = 2, [UserIdentityType::IDENTITYID](#) = 3 }
- enum [LogLevel](#) { [LogLevel::Trace](#) = 0, [LogLevel::Debug](#), [LogLevel::Info](#), [LogLevel::Warn](#), [LogLevel::Error](#) }

## Functions

- [QEVERCLOUD\\_EXPORT IRetryPolicyPtr newRetryPolicy \(\)](#)
- [QEVERCLOUD\\_EXPORT IRetryPolicyPtr nullRetryPolicy \(\)](#)
- [QEVERCLOUD\\_EXPORT IDurableServicePtr newDurableService \(IRetryPolicyPtr={}, IRequestContextPtr={}\)](#)
- [uint qHash \(EDAMErrorCode value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const EDAMErrorCode value\)](#)
- [QEVERCLOUD\\_EXPORT QDebug & operator<< \(QDebug &out, const EDAMErrorCode value\)](#)
- [uint qHash \(EDAMInvalidContactReason value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const EDAMInvalidContactReason value\)](#)
- [QEVERCLOUD\\_EXPORT QDebug & operator<< \(QDebug &out, const EDAMInvalidContactReason value\)](#)
- [uint qHash \(ShareRelationshipPrivilegeLevel value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const ShareRelationshipPrivilegeLevel value\)](#)
- [QEVERCLOUD\\_EXPORT QDebug & operator<< \(QDebug &out, const ShareRelationshipPrivilegeLevel value\)](#)
- [uint qHash \(PrivilegeLevel value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const PrivilegeLevel value\)](#)
- [QEVERCLOUD\\_EXPORT QDebug & operator<< \(QDebug &out, const PrivilegeLevel value\)](#)
- [uint qHash \(ServiceLevel value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const ServiceLevel value\)](#)
- [QEVERCLOUD\\_EXPORT QDebug & operator<< \(QDebug &out, const ServiceLevel value\)](#)
- [uint qHash \(QueryFormat value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const QueryFormat value\)](#)
- [QEVERCLOUD\\_EXPORT QDebug & operator<< \(QDebug &out, const QueryFormat value\)](#)
- [uint qHash \(NoteSortOrder value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const NoteSortOrder value\)](#)
- [QEVERCLOUD\\_EXPORT QDebug & operator<< \(QDebug &out, const NoteSortOrder value\)](#)
- [uint qHash \(PremiumOrderStatus value\)](#)
- [QEVERCLOUD\\_EXPORT QTextStream & operator<< \(QTextStream &out, const PremiumOrderStatus value\)](#)

- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [PremiumOrderStatus](#) value)
- [uint](#) [qHash](#) ([SharedNotebookPrivilegeLevel](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [SharedNotebookPrivilegeLevel](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [SharedNotebookPrivilegeLevel](#) value)
- [uint](#) [qHash](#) ([SharedNotePrivilegeLevel](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [SharedNotePrivilegeLevel](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [SharedNotePrivilegeLevel](#) value)
- [uint](#) [qHash](#) ([SponsoredGroupRole](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [SponsoredGroupRole](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [SponsoredGroupRole](#) value)
- [uint](#) [qHash](#) ([BusinessUserRole](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [BusinessUserRole](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [BusinessUserRole](#) value)
- [uint](#) [qHash](#) ([BusinessUserStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [BusinessUserStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [BusinessUserStatus](#) value)
- [uint](#) [qHash](#) ([SharedNotebookInstanceRestrictions](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [SharedNotebookInstanceRestrictions](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [SharedNotebookInstanceRestrictions](#) value)
- [uint](#) [qHash](#) ([ReminderEmailConfig](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [ReminderEmailConfig](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [ReminderEmailConfig](#) value)
- [uint](#) [qHash](#) ([BusinessInvitationStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [BusinessInvitationStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [BusinessInvitationStatus](#) value)
- [uint](#) [qHash](#) ([ContactType](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [ContactType](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [ContactType](#) value)
- [uint](#) [qHash](#) ([EntityType](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [EntityType](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [EntityType](#) value)
- [uint](#) [qHash](#) ([RecipientStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [RecipientStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [RecipientStatus](#) value)
- [uint](#) [qHash](#) ([CanMoveToContainerStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [CanMoveToContainerStatus](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [CanMoveToContainerStatus](#) value)
- [uint](#) [qHash](#) ([RelatedContentType](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [RelatedContentType](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [RelatedContentType](#) value)
- [uint](#) [qHash](#) ([RelatedContentAccess](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QTextStream](#) & [operator<<](#) ([QTextStream](#) &out, const [RelatedContentAccess](#) value)
- [QEVERCLOUD\\_EXPORT](#) [QDebug](#) & [operator<<](#) ([QDebug](#) &out, const [RelatedContentAccess](#) value)

- uint [qHash](#) ([UserIdentityType](#) value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [operator<<](#) (QTextStream &out, const [UserIdentityType](#) value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [operator<<](#) (QDebug &out, const [UserIdentityType](#) value)
- [QEVERCLOUD\\_EXPORT](#) INoteStore \* [newNoteStore](#) (QString noteStoreUrl={}, [IRequestContextPtr](#) ctx={}, QObject \*parent=nullptr, [IRetryPolicyPtr](#) retryPolicy={})
- [QEVERCLOUD\\_EXPORT](#) IUserStore \* [newUserStore](#) (QString userStoreUrl={}, [IRequestContextPtr](#) ctx={}, QObject \*parent=nullptr, [IRetryPolicyPtr](#) retryPolicy={})
- [QEVERCLOUD\\_EXPORT](#) QNetworkProxy [evernoteNetworkProxy](#) ()
- [QEVERCLOUD\\_EXPORT](#) void [setEvernoteNetworkProxy](#) (QNetworkProxy proxy)
- [QEVERCLOUD\\_EXPORT](#) void [resetEvernoteNetworkProxy](#) ()
- [QEVERCLOUD\\_EXPORT](#) int [libraryVersion](#) ()
- [QEVERCLOUD\\_EXPORT](#) void [initializeQEverCloud](#) ()
- template<class Container >  
  [QAssociativeContainerReferenceWrapper](#)< Container > [toRange](#) (Container &container)
- template<class Container >  
  [QAssociativeContainerConstReferenceWrapper](#)< Container > [toRange](#) (const Container &container)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [operator<<](#) (QTextStream &out, const [LogLevel](#) level)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [operator<<](#) (QDebug &out, const [LogLevel](#) level)
- [QEVERCLOUD\\_EXPORT](#) ILoggerPtr [logger](#) ()
- [QEVERCLOUD\\_EXPORT](#) void [setLogger](#) (ILoggerPtr logger)
- [QEVERCLOUD\\_EXPORT](#) ILoggerPtr [nullLogger](#) ()
- [QEVERCLOUD\\_EXPORT](#) ILoggerPtr [newStdErrLogger](#) ([LogLevel](#) level=[LogLevel::Warn](#))
- void [setNonceGenerator](#) (quint64(\*nonceGenerator)())  
  *Sets the function to use for nonce generation for OAuth authentication.*
- [QEVERCLOUD\\_EXPORT](#) [IRequestContextPtr](#) [newRequestContext](#) (QString authenticationToken={}, quint64 requestTimeout=DEFAULT\_REQUEST\_TIMEOUT\_MSEC, bool increaseRequestTimeout↵ Exponentially=DEFAULT\_REQUEST\_TIMEOUT\_EXPONENTIAL\_INCREASE, quint64 maxRequest↵ Timeout=DEFAULT\_MAX\_REQUEST\_TIMEOUT\_MSEC, quint32 maxRequestRetryCount=DEFAULT↵ \_MAX\_REQUEST\_RETRY\_COUNT, QList< QNetworkCookie > cookies={})

## Variables

- class [QEVERCLOUD\\_EXPORT](#) [EverCloudExceptionData](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_ATTRIBUTE\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_ATTRIBUTE\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_ATTRIBUTE\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_ATTRIBUTE\\_LIST\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_ATTRIBUTE\\_MAP\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_GUID\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_GUID\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_GUID\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_EMAIL\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_EMAIL\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_EMAIL\\_LOCAL\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_EMAIL\\_DOMAIN\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_EMAIL\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_VAT\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_TIMEZONE\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_TIMEZONE\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_TIMEZONE\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_MIME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const quint32 [EDAM\\_MIME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_GIF](#)



- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_JPEG](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_PNG](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_TIFF](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_BMP](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_WAV](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_MP3](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_AMR](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_AAC](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_M4A](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_MP4\\_VIDEO](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_INK](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_PDF](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MIME\\_TYPE\\_DEFAULT](#)
- [QEVERCLOUD\\_EXPORT](#) const QSet< QString > [EDAM\\_MIME\\_TYPES](#)
- [QEVERCLOUD\\_EXPORT](#) const QSet< QString > [EDAM\\_INDEXABLE\\_RESOURCE\\_MIME\\_TYPES](#)
- [QEVERCLOUD\\_EXPORT](#) const QSet< QString > [EDAM\\_INDEXABLE\\_PLAINTEXT\\_MIME\\_TYPES](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_SEARCH\\_QUERY\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_SEARCH\\_QUERY\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SEARCH\\_QUERY\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_HASH\\_LEN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_USER\\_USERNAME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_USER\\_USERNAME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_USER\\_USERNAME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_USER\\_NAME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_USER\\_NAME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_USER\\_NAME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_TAG\\_NAME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_TAG\\_NAME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_TAG\\_NAME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_TITLE\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_TITLE\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_NOTE\\_TITLE\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_CONTENT\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_CONTENT\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_APPLICATIONDATA\\_NAME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_APPLICATIONDATA\\_NAME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_APPLICATIONDATA\\_VALUE\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_APPLICATIONDATA\\_VALUE\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_APPLICATIONDATA\\_ENTRY\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_APPLICATIONDATA\\_NAME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_APPLICATIONDATA\\_VALUE\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTEBOOK\\_NAME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTEBOOK\\_NAME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_NOTEBOOK\\_NAME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTEBOOK\\_STACK\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTEBOOK\\_STACK\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_NOTEBOOK\\_STACK\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_WORKSPACE\\_NAME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_WORKSPACE\\_NAME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_WORKSPACE\\_DESCRIPTION\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_WORKSPACE\\_NAME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PUBLISHING\\_URI\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PUBLISHING\\_URI\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_PUBLISHING\\_URI\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QSet< QString > [EDAM\\_PUBLISHING\\_URI\\_PROHIBITED](#)



- `QEVERCLOUD_EXPORT` const qint32 `EDAM_PUBLISHING_DESCRIPTION_LEN_MIN`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_PUBLISHING_DESCRIPTION_LEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_PUBLISHING_DESCRIPTION_REGEX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_SAVED_SEARCH_NAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_SAVED_SEARCH_NAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_SAVED_SEARCH_NAME_REGEX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_PASSWORD_LEN_MIN`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_PASSWORD_LEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_USER_PASSWORD_REGEX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_BUSINESS_URI_LEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_BUSINESS_MARKETING_CODE_REGEX_PATTERN`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_TAGS_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_RESOURCES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_TAGS_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_BUSINESS_TAGS_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_SAVED_SEARCHES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_NOTES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_BUSINESS_NOTES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_NOTEBOOKS_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_WORKSPACES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_BUSINESS_NOTEBOOKS_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_BUSINESS_WORKSPACES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_RECENT_MAILED_ADDRESSES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_MAIL_LIMIT_DAILY_FREE`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_MAIL_LIMIT_DAILY_PREMIUM`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_LIMIT_FREE`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_LIMIT_PREMIUM`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_LIMIT_BUSINESS_FIRST_MONTH`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_LIMIT_BUSINESS_NEXT_MONTH`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_LIMIT_PLUS`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_SURVEY_THRESHOLD`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_LIMIT_BUSINESS`
- `QEVERCLOUD_EXPORT` const qint64 `EDAM_USER_UPLOAD_LIMIT_BUSINESS_PER_USER`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_SIZE_MAX_FREE`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_SIZE_MAX_PREMIUM`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_RESOURCE_SIZE_MAX_FREE`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_RESOURCE_SIZE_MAX_PREMIUM`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_LINKED_NOTEBOOK_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_LINKED_NOTEBOOK_MAX_PREMIUM`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTEBOOK_BUSINESS_SHARED_NOTEBOOK_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTEBOOK_PERSONAL_SHARED_NOTEBOOK_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_BUSINESS_SHARED_NOTE_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_PERSONAL_SHARED_NOTE_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_CONTENT_CLASS_LEN_MIN`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_NOTE_CONTENT_CLASS_LEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_NOTE_CONTENT_CLASS_REGEX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_HELLO_APP_CONTENT_CLASS_PREFIX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_FOOD_APP_CONTENT_CLASS_PREFIX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_CONTENT_CLASS_HELLO_ENCOUNTER`
- `QEVERCLOUD_EXPORT` const QString `EDAM_CONTENT_CLASS_HELLO_PROFILE`
- `QEVERCLOUD_EXPORT` const QString `EDAM_CONTENT_CLASS_FOOD_MEAL`
- `QEVERCLOUD_EXPORT` const QString `EDAM_CONTENT_CLASS_SKITCH_PREFIX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_CONTENT_CLASS_SKITCH`
- `QEVERCLOUD_EXPORT` const QString `EDAM_CONTENT_CLASS_SKITCH_PDF`
- `QEVERCLOUD_EXPORT` const QString `EDAM_CONTENT_CLASS_PENULTIMATE_PREFIX`

- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_CONTENT\\_CLASS\\_PENULTIMATE\\_NOTEBOOK](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_APPLICATION\\_POSTIT](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_APPLICATION\\_MOLESKINE](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_APPLICATION\\_EN\\_SCANSNAP](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_APPLICATION\\_EWC](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_APPLICATION\\_ANDROID\\_SHARE\\_EXTENSION](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_APPLICATION\\_IOS\\_SHARE\\_EXTENSION](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_APPLICATION\\_WEB\\_CLIPPER](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_SOURCE\\_OUTLOOK\\_CLIPPER](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_TITLE\\_QUALITY\\_UNTITLED](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_TITLE\\_QUALITY\\_LOW](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_TITLE\\_QUALITY\\_MEDIUM](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_TITLE\\_QUALITY\\_HIGH](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_RELATED\\_PLAINTEXT\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_RELATED\\_PLAINTEXT\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_RELATED\\_MAX\\_NOTES](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_RELATED\\_MAX\\_NOTEBOOKS](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_RELATED\\_MAX\\_TAGS](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_RELATED\\_MAX\\_EXPERTS](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_RELATED\\_MAX\\_RELATED\\_CONTENT](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_BUSINESS\\_NOTEBOOK\\_DESCRIPTION\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_BUSINESS\\_PHONE\\_NUMBER\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PREFERENCE\\_NAME\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PREFERENCE\\_NAME\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PREFERENCE\\_VALUE\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PREFERENCE\\_VALUE\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_MAX\\_PREFERENCES](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_MAX\\_VALUES\\_PER\\_PREFERENCE](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PREFERENCE\\_ONLY\\_ONE\\_VALUE\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_PREFERENCE\\_NAME\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_PREFERENCE\\_VALUE\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_PREFERENCE\\_ONLY\\_ONE\\_VALUE\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_PREFERENCE\\_SHORTCUTS](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_PREFERENCE\\_BUSINESS\\_DEFAULT\\_NOTEBOOK](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_PREFERENCE\\_BUSINESS\\_QUICKNOTE](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_PREFERENCE\\_SHORTCUTS\\_MAX\\_VALUES](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_DEVICE\\_ID\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_DEVICE\\_ID\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_DEVICE\\_DESCRIPTION\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_DEVICE\\_DESCRIPTION\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_SEARCH\\_SUGGESTIONS\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_SEARCH\\_SUGGESTIONS\\_PREFIX\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_SEARCH\\_SUGGESTIONS\\_PREFIX\\_LEN\\_MIN](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_FIND\\_CONTACT\\_DEFAULT\\_MAX\\_RESULTS](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_FIND\\_CONTACT\\_MAX\\_RESULTS](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_NOTE\\_LOCK\\_VIEWERS\\_NOTES\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_GET\\_ORDERS\\_MAX\\_RESULTS](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_MESSAGE\\_BODY\\_LEN\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const QString [EDAM\\_MESSAGE\\_BODY\\_REGEX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_MESSAGE\\_RECIPIENTS\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_MESSAGE\\_ATTACHMENTS\\_MAX](#)
- [QEVERCLOUD\\_EXPORT](#) const qint32 [EDAM\\_MESSAGE\\_ATTACHMENT\\_TITLE\\_LEN\\_MAX](#)

- `QEVERCLOUD_EXPORT` const QString `EDAM_MESSAGE_ATTACHMENT_TITLE_REGEX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_MESSAGE_ATTACHMENT_SNIPPET_LEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_MESSAGE_ATTACHMENT_SNIPPET_REGEX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_USER_PROFILE_PHOTO_MAX_BYTES`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_PROMOTION_ID_LEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `EDAM_PROMOTION_ID_REGEX`
- `QEVERCLOUD_EXPORT` const qint16 `EDAM_APP_RATING_MIN`
- `QEVERCLOUD_EXPORT` const qint16 `EDAM_APP_RATING_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_SNIPPETS_NOTES_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_CONNECTED_IDENTITY_REQUEST_MAX`
- `QEVERCLOUD_EXPORT` const qint32 `EDAM_OPEN_ID_ACCESS_TOKEN_MAX`
- `QEVERCLOUD_EXPORT` const QString `CLASSIFICATION_RECIPE_USER_NON_RECIPE`
- `QEVERCLOUD_EXPORT` const QString `CLASSIFICATION_RECIPE_USER_RECIPE`
- `QEVERCLOUD_EXPORT` const QString `CLASSIFICATION_RECIPE_SERVICE_RECIPE`
- `QEVERCLOUD_EXPORT` const QString `EDAM_NOTE_SOURCE_WEB_CLIP`
- `QEVERCLOUD_EXPORT` const QString `EDAM_NOTE_SOURCE_WEB_CLIP_SIMPLIFIED`
- `QEVERCLOUD_EXPORT` const QString `EDAM_NOTE_SOURCE_MAIL_CLIP`
- `QEVERCLOUD_EXPORT` const QString `EDAM_NOTE_SOURCE_MAIL_SMTP_GATEWAY`
- `QEVERCLOUD_EXPORT` const qint16 `EDAM_VERSION_MAJOR`
- `QEVERCLOUD_EXPORT` const qint16 `EDAM_VERSION_MINOR`

### 6.1.1 Detailed Description

Original work: Copyright (c) 2014 Sergey Skoblikov Modified work: Copyright (c) 2015-2019 Dmitry Ivanov

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

Copyright (c) 2019 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) 2015-2020 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) 2015-2020 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.

Copyright (c) 2019-2020 Dmitry Ivanov

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

Copyright (c) 2016-2020 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) 2015-2020 Dmitry Ivanov

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

## 6.1.2 Typedef Documentation

### 6.1.2.1 EverCloudExceptionDataPtr

```
using qevercloud::EverCloudExceptionDataPtr = typedef std::shared_ptr<EverCloudExceptionData>
```

### 6.1.2.2 Guid

```
using qevercloud::Guid = typedef QString
```

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.

### 6.1.2.3 IdentityID

```
using qevercloud::IdentityID = typedef quint64
```

A type alias for the primary identifiers for [Identity](#) objects.

### 6.1.2.4 IDurableServicePtr

```
using qevercloud::IDurableServicePtr = typedef std::shared_ptr<IDurableService>
```

### 6.1.2.5 ILoggerPtr

```
using qevercloud::ILoggerPtr = typedef std::shared_ptr<ILogger>
```

### 6.1.2.6 INoteStorePtr

```
using qevercloud::INoteStorePtr = typedef std::shared_ptr<INoteStore>
```

#### 6.1.2.7 InvalidationSequenceNumber

```
using qevercloud::InvalidationSequenceNumber = typedef quint64
```

A monotonically incrementing number on each shard that identifies a cross shard cache invalidation event.

#### 6.1.2.8 IRequestContextPtr

```
using qevercloud::IRequestContextPtr = typedef std::shared_ptr<IRequestContext>
```

#### 6.1.2.9 IRetryPolicyPtr

```
using qevercloud::IRetryPolicyPtr = typedef std::shared_ptr<IRetryPolicy>
```

#### 6.1.2.10 IUserStorePtr

```
using qevercloud::IUserStorePtr = typedef std::shared_ptr<IUserStore>
```

#### 6.1.2.11 MessageEventID

```
using qevercloud::MessageEventID = typedef quint64
```

A sequence number for the MessageStore subsystem.

#### 6.1.2.12 MessageThreadID

```
using qevercloud::MessageThreadID = typedef quint64
```

A type alias for the primary identifiers for MessageThread objects.

#### 6.1.2.13 Timestamp

```
using qevercloud::Timestamp = typedef quint64
```

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

#### 6.1.2.14 UserID

```
using qevercloud::UserID = typedef qint32
```

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 Enumeration Type Documentation

#### 6.1.3.1 BusinessInvitationStatus

```
enum qevercloud::BusinessInvitationStatus [strong]
```

An enumeration defining the possible states of a [BusinessInvitation](#).

**APPROVED:** The invitation was created or approved by a business admin and may be redeemed by the invited email.

**REQUESTED:** The invitation was requested by a non-admin member of the business and must be approved by an admin before it may be redeemed. Invitations in this state do not count against a business' seat limit.

**REDEEMED:** The invitation has already been redeemed. Invitations in this state do not count against a business' seat limit.

Enumerator

APPROVED	
REQUESTED	
REDEEMED	

#### 6.1.3.2 BusinessUserRole

```
enum qevercloud::BusinessUserRole [strong]
```

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.

Enumerator

ADMIN	
NORMAL	

### 6.1.3.3 BusinessUserStatus

```
enum qevercloud::BusinessUserStatus [strong]
```

The BusinessUserStatus indicates the status of the user in the business.

A BusinessUser will typically start as ACTIVE. Only ACTIVE users can authenticate to the Business.

#### ACTIVE

The business user can authenticate to and access the business.

#### DEACTIVATED

The business user has been deactivated and cannot access the business

#### Enumerator

ACTIVE	
DEACTIVATED	

### 6.1.3.4 CanMoveToContainerStatus

```
enum qevercloud::CanMoveToContainerStatus [strong]
```

This enumeration defines the possible types of canMoveToContainer outcomes.

An outdated client is expected to signal a "Cannot Move, Please Upgrade To Learn Why" like response to the user if an unknown enumeration value is received.

**CAN\_BE\_MOVED** Can move [Notebook](#) to Workspace.

**INSUFFICIENT\_ENTITY\_PRIVILEGE** Can not move [Notebook](#) to Workspace, because either: a) [Notebook](#) not in Workspace and insufficient privilege on [Notebook](#) or b) [Notebook](#) in Workspace and membership on Workspace with insufficient privilege for move

**INSUFFICIENT\_CONTAINER\_PRIVILEGE** [Notebook](#) in Workspace and no membership on Workspace.

#### Enumerator

CAN_BE_MOVED	
INSUFFICIENT_ENTITY_PRIVILEGE	
INSUFFICIENT_CONTAINER_PRIVILEGE	

### 6.1.3.5 ContactType

```
enum qevercloud::ContactType [strong]
```

What kinds of Contacts does the Evernote service know about?

Enumerator

EVERNOTE	
SMS	
FACEBOOK	
EMAIL	
TWITTER	
LINKEDIN	

### 6.1.3.6 EDAMErrorCode

```
enum qevercloud::EDAMErrorCode [strong]
```

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.

**BUSINESS\_SECURITY\_LOGIN\_REQUIRED** Access to a business account has been denied because the user must complete additional steps in order to comply with business security requirements.

**DEVICE\_LIMIT\_REACHED** Operation denied because the user has exceeded their maximum allowed number of devices.

**OPENID\_ALREADY\_TAKEN** Operation failed because the Open ID is already associated with another user.

**INVALID\_OPENID\_TOKEN** Operation denied because the Open ID token is invalid. Please re-issue a valid token.

**USER\_NOT\_REGISTERED** There is no Evernote user associated with this OpenID account, and no Evernote user with a matching email

**USER\_NOT\_ASSOCIATED** There is no Evernote user associated with this OpenID account, but Evernote user with matching email exists

**USER\_ALREADY\_ASSOCIATED** Evernote user is already associated with this provider using a different email address.

**ACCOUNT\_CLEAR** The user's account has been disabled. Clients should deal with this errorCode by logging the user out and purging all locally saved content, including local edits not yet pushed to the server.

**SSO\_AUTHENTICATION\_REQUIRED** SSO authentication is the only type of authentication allowed for the user's account. This error is thrown when the user attempts to authenticate by another method (password, OpenId, etc).

#### 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	
BUSINESS_SECURITY_LOGIN_REQUIRED	
DEVICE_LIMIT_REACHED	
OPENID_ALREADY_TAKEN	
INVALID_OPENID_TOKEN	
USER_NOT_ASSOCIATED	

## Enumerator

USER_NOT_REGISTERED	
USER_ALREADY_ASSOCIATED	
ACCOUNT_CLEAR	
SSO_AUTHENTICATION_REQUIRED	

## 6.1.3.7 EDAMInvalidContactReason

```
enum qevercloud::EDAMInvalidContactReason [strong]
```

An enumeration that provides a reason for why a given contact was invalid, for example, as thrown via an [EDAMInvalidContactsException](#).

**BAD\_ADDRESS** The contact information does not represent a valid address for a recipient. Clients should be validating and normalizing contacts, so receiving this error code commonly represents a client error.

**DUPLICATE\_CONTACT** If the method throwing this exception accepts a list of contacts, this error code indicates that the given contact is a duplicate of another contact in the list. [Note](#) that the server may clean up contacts, and that this cleanup occurs before checking for duplication. Receiving this error is commonly an indication of a client issue, since client should be normalizing contacts and removing duplicates. All instances that are duplicates are returned. For example, if a list of 5 contacts has the same e-mail address twice, the two conflicting e-mail address contacts will be returned.

**NO\_CONNECTION** Indicates that the given contact, an Evernote type contact, is not connected to the user for which the call is being made. It is possible that clients are out of sync with the server and should re-synchronize their identities and business user state. See [Identity.userConnected](#) for more information on user connections.

[Note](#) that if multiple reasons may apply, only one is returned. The precedence order is BAD\_ADDRESS, DUPLICATE\_CONTACT, NO\_CONNECTION, meaning that if a contact has a bad address and is also duplicated, it will be returned as a BAD\_ADDRESS.

## Enumerator

BAD_ADDRESS	
DUPLICATE_CONTACT	
NO_CONNECTION	

## 6.1.3.8 EntityType

```
enum qevercloud::EntityType [strong]
```

## Entity types

## Enumerator

NOTE	
NOTEBOOK	
WORKSPACE	

## 6.1.3.9 LogLevel

```
enum qevercloud::LogLevel [strong]
```

## Enumerator

Trace	
Debug	
Info	
Warn	
Error	

## 6.1.3.10 NoteSortOrder

```
enum qevercloud::NoteSortOrder [strong]
```

This enumeration defines the possible sort ordering for notes when they are returned from a search result.

## Enumerator

CREATED	
UPDATED	
RELEVANCE	
UPDATE_SEQUENCE_NUMBER	
TITLE	

## 6.1.3.11 PremiumOrderStatus

```
enum qevercloud::PremiumOrderStatus [strong]
```

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. 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.

#### Enumerator

NONE	
PENDING	
ACTIVE	
FAILED	
CANCELLATION_PENDING	
CANCELED	

#### 6.1.3.12 PrivilegeLevel

```
enum qevercloud::PrivilegeLevel [strong]
```

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.

#### Enumerator

NORMAL	
PREMIUM	
VIP	
MANAGER	
SUPPORT	
ADMIN	

#### 6.1.3.13 QueryFormat

```
enum qevercloud::QueryFormat [strong]
```

Every search query is specified as a sequence of characters. Currently, only the USER query format is supported.

#### Enumerator

USER	
SEXP	

#### 6.1.3.14 RecipientStatus

```
enum qevercloud::RecipientStatus [strong]
```

This enumeration defines the possible states that a notebook can be in for a recipient. It encompasses the "inMyList" boolean and default notebook status.

**NOT\_IN\_MY\_LIST** The notebook is not in the recipient's list (not "joined").

**IN\_MY\_LIST** The notebook is in the recipient's notebook list (formerly, we would say that the recipient has "joined" the notebook)

**IN\_MY\_LIST\_AND\_DEFAULT\_NOTEBOOK** The same as IN\_MY\_LIST and this notebook is the user's default notebook.

##### Enumerator

NOT_IN_MY_LIST	
IN_MY_LIST	
IN_MY_LIST_AND_DEFAULT_NOTEBOOK	

#### 6.1.3.15 RelatedContentAccess

```
enum qevercloud::RelatedContentAccess [strong]
```

This enumeration defines the possible ways to access related content.

**NOT\_ACCESSIBLE**: The content is not accessible given the user's privilege level, but still worth showing as a snippet. The content url may point to a webpage that explains why not, or explains how to access that content.

**DIRECT\_LINK\_ACCESS\_OK**: The content is accessible directly, and no additional login is required.

**DIRECT\_LINK\_LOGIN\_REQUIRED**: The content is accessible directly, but an additional login is required.

**DIRECT\_LINK\_EMBEDDED\_VIEW**: The content is accessible directly, and should be shown in an embedded web view. If the URL refers to a secured location under our control (for example, <https://www.evernote.com/> <smth>), the client may include user-specific authentication credentials with the request.

##### Enumerator

NOT_ACCESSIBLE	
DIRECT_LINK_ACCESS_OK	
DIRECT_LINK_LOGIN_REQUIRED	
DIRECT_LINK_EMBEDDED_VIEW	

### 6.1.3.16 RelatedContentType

```
enum qevercloud::RelatedContentType [strong]
```

This enumeration defines the possible types of related content.

NEWS\_ARTICLE: This related content is a news article  
 PROFILE\_PERSON: This match refers to the profile of an individual person  
 PROFILE\_ORGANIZATION: This match refers to the profile of an organization  
 REFERENCE\_MATERIAL: This related content is material from reference works

#### Enumerator

NEWS_ARTICLE	
PROFILE_PERSON	
PROFILE_ORGANIZATION	
REFERENCE_MATERIAL	

### 6.1.3.17 ReminderEmailConfig

```
enum qevercloud::ReminderEmailConfig [strong]
```

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.

#### Enumerator

DO_NOT_SEND	
SEND_DAILY_EMAIL	

### 6.1.3.18 ServiceLevel

```
enum qevercloud::ServiceLevel [strong]
```

This enumeration defines the possible tiers of service that a user may have. A ServiceLevel of BUSINESS signifies a business-only account, which can never be any other ServiceLevel.

#### Enumerator

BASIC	
PLUS	
PREMIUM	
BUSINESS	

## 6.1.3.19 SharedNotebookInstanceRestrictions

```
enum qevercloud::SharedNotebookInstanceRestrictions [strong]
```

An enumeration describing restrictions on the domain of shared notebook instances that are valid for a given operation, as used, for example, in [NotebookRestrictions](#).

ASSIGNED: The domain consists of shared notebooks that belong, or are assigned, to the recipient.

NO\_SHARED\_NOTEBOOKS: No shared notebooks are applicable to the operation.

## Enumerator

ASSIGNED	
NO_SHARED_NOTEBOOKS	

## 6.1.3.20 SharedNotebookPrivilegeLevel

```
enum qevercloud::SharedNotebookPrivilegeLevel [strong]
```

Privilege levels for accessing shared notebooks.

**Note** that as of 2014-04, FULL\_ACCESS is synonymous with BUSINESS\_FULL\_ACCESS. If a user is a member of a business and has FULL\_ACCESS privileges, then they will automatically be granted BUSINESS\_FULL\_ACCESS for notebooks in their business. This will happen implicitly when they attempt to access the corresponding notebooks of the business. BUSINESS\_FULL\_ACCESS is therefore deprecated.

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. For members of a business, FULL\_ACCESS privilege on business notebooks also grants the ability to change how the notebook will appear when shared with the business, including the rights to share and unshare the notebook with the business.

BUSINESS\_FULL\_ACCESS: Deprecated. See the note above about BUSINESS\_FULL\_ACCESS and FULL\_ACCESS being synonymous.

## Enumerator

READ_NOTEBOOK	
MODIFY_NOTEBOOK_PLUS_ACTIVITY	
READ_NOTEBOOK_PLUS_ACTIVITY	
GROUP	
FULL_ACCESS	
BUSINESS_FULL_ACCESS	

## 6.1.3.21 SharedNotePrivilegeLevel

```
enum qevercloud::SharedNotePrivilegeLevel [strong]
```

Privilege levels for accessing a shared note. All privilege levels convey "activity feed" access, which allows the recipient to access information about other recipients and the activity stream.

READ\_NOTE: Recipient has rights to read the shared note.

MODIFY\_NOTE: Recipient has all of the rights of READ\_NOTE, plus rights to modify the shared note's content, title and resources. Other fields, including the notebook, tags and metadata, may not be modified.

FULL\_ACCESS: Recipient has all of the rights of MODIFY\_NOTE, plus rights to share the note with other users via email, public note links, and note sharing. Recipient may also update and remove other recipient's note sharing rights.

## Enumerator

READ_NOTE	
MODIFY_NOTE	
FULL_ACCESS	

## 6.1.3.22 ShareRelationshipPrivilegeLevel

```
enum qevercloud::ShareRelationshipPrivilegeLevel [strong]
```

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.

READ\_NOTEBOOK\_PLUS\_ACTIVITY: Recipient has READ\_NOTEBOOK rights and can also access information about other recipients and the activity stream.

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.

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. If the user is a member of the same group, (e.g. the same business) as the shared notebook, they will additionally be granted permissions to update the publishing status of the notebook.



## Enumerator

READ_NOTEBOOK	
READ_NOTEBOOK_PLUS_ACTIVITY	
MODIFY_NOTEBOOK_PLUS_ACTIVITY	
FULL_ACCESS	

## 6.1.3.23 SponsoredGroupRole

```
enum qevercloud::SponsoredGroupRole [strong]
```

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.

## Enumerator

GROUP_MEMBER	
GROUP_ADMIN	
GROUP_OWNER	

## 6.1.3.24 UserIdentityType

```
enum qevercloud::UserIdentityType [strong]
```

## Enumerator

EVERNOTE_USERID	
EMAIL	
IDENTITYID	

## 6.1.4 Function Documentation

## 6.1.4.1 evernoteNetworkProxy()

```
QEVERCLOUD_EXPORT QNetworkProxy qevercloud::evernoteNetworkProxy ( )
```

Getter for network proxy settings used by QEverCloud. If none were set explicitly, returns the same result as QNetworkProxy::applicationProxy. Hence, QEverCloud uses the same proxy settings as the application which uses QEverCloud by default.

This function is thread-safe although internally it operates on a static object containing proxy settings so it's not recommended to read and write proxy settings concurrently to avoid contention for static object.

WARNING: when QEverCloud is built with QtWebEngine and some proxy settings different from QNetworkProxy::applicationProxy are set, the OAuth call which loads the web page would not use them; instead it would use proxy settings from QNetworkProxy::applicationProxy. This limitation is imposed by Qt: <https://doc.qt.io/qt-5/qtwebengine-overview.html#proxy-support>

#### 6.1.4.2 initializeQEverCloud()

```
QEVERCLOUD_EXPORT void qevercloud::initializeQEverCloud ( )
```

Initialization function for QEverCloud, needs to be called once before using the library. There is no harm if it is called multiple times

#### 6.1.4.3 libraryVersion()

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

QEverCloud library version.

#### 6.1.4.4 logger()

```
QEVERCLOUD_EXPORT ILoggerPtr qevercloud::logger ( )
```

#### 6.1.4.5 newDurableService()

```
QEVERCLOUD_EXPORT IDurableServicePtr qevercloud::newDurableService (
    IRetryPolicyPtr    = {},
    IRequestContextPtr = {} )
```

#### 6.1.4.6 newNoteStore()

```
QEVERCLOUD_EXPORT INoteStore* qevercloud::newNoteStore (
    QString noteStoreUrl = {},
    IRequestContextPtr ctx = {},
    QObject * parent = nullptr,
    IRetryPolicyPtr retryPolicy = {} )
```

#### 6.1.4.7 newRequestContext()

```
QEVERCLOUD_EXPORT IRequestContextPtr qevercloud::newRequestContext (
    QString authenticationToken = {},
    qint64 requestTimeout = DEFAULT_REQUEST_TIMEOUT_MSEC,
    bool increaseRequestTimeoutExponentially = DEFAULT_REQUEST_TIMEOUT_EXPONENTIAL_↵
    INCREASE,
    qint64 maxRequestTimeout = DEFAULT_MAX_REQUEST_TIMEOUT_MSEC,
    quint32 maxRequestRetryCount = DEFAULT_MAX_REQUEST_RETRY_COUNT,
    QList< QNetworkCookie > cookies = {} )
```

#### 6.1.4.8 newRetryPolicy()

```
QEVERCLOUD_EXPORT IRetryPolicyPtr qevercloud::newRetryPolicy ( )
```

#### 6.1.4.9 newStdErrLogger()

```
QEVERCLOUD_EXPORT ILoggerPtr qevercloud::newStdErrLogger (
    LogLevel level = LogLevel::Warn )
```

#### 6.1.4.10 newUserStore()

```
QEVERCLOUD_EXPORT IUserStore* qevercloud::newUserStore (
    QString userStoreUrl = {},
    IRequestContextPtr ctx = {},
    QObject * parent = nullptr,
    IRetryPolicyPtr retryPolicy = {} )
```

#### 6.1.4.11 nullLogger()

```
QEVERCLOUD_EXPORT ILoggerPtr qevercloud::nullLogger ( )
```

#### 6.1.4.12 nullRetryPolicy()

```
QEVERCLOUD_EXPORT IRetryPolicyPtr qevercloud::nullRetryPolicy ( )
```

**6.1.4.13 operator<<()** [1/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const LogLevel level )
```

**6.1.4.14 operator<<()** [2/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const LogLevel level )
```

**6.1.4.15 operator<<()** [3/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const EDAMErrorCode value )
```

**6.1.4.16 operator<<()** [4/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const EDAMErrorCode value )
```

**6.1.4.17 operator<<()** [5/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const EDAMInvalidContactReason value )
```

**6.1.4.18 operator<<()** [6/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const EDAMInvalidContactReason value )
```

**6.1.4.19 operator<<()** [7/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const ShareRelationshipPrivilegeLevel value )
```

**6.1.4.20 operator<<()** [8/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const ShareRelationshipPrivilegeLevel value )
```

**6.1.4.21 operator<<()** [9/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const PrivilegeLevel value )
```

**6.1.4.22 operator<<()** [10/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const PrivilegeLevel value )
```

**6.1.4.23 operator<<()** [11/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const ServiceLevel value )
```

**6.1.4.24 operator<<()** [12/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const ServiceLevel value )
```

**6.1.4.25 operator<<()** [13/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const QueryFormat value )
```

**6.1.4.26 operator<<()** [14/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const QueryFormat value )
```

**6.1.4.27 operator<<()** [15/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const NoteSortOrder value )
```

**6.1.4.28 operator<<()** [16/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const NoteSortOrder value )
```

**6.1.4.29 operator<<()** [17/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const PremiumOrderStatus value )
```

**6.1.4.30 operator<<()** [18/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const PremiumOrderStatus value )
```

**6.1.4.31 operator<<()** [19/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const SharedNotebookPrivilegeLevel value )
```

**6.1.4.32 operator<<()** [20/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const SharedNotebookPrivilegeLevel value )
```

**6.1.4.33 operator<<()** [21/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const SharedNotePrivilegeLevel value )
```

**6.1.4.34 operator<<()** [22/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const SharedNotePrivilegeLevel value )
```

**6.1.4.35 operator<<()** [23/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const SponsoredGroupRole value )
```

**6.1.4.36 operator<<()** [24/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const SponsoredGroupRole value )
```

**6.1.4.37 operator<<()** [25/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const BusinessUserRole value )
```

**6.1.4.38 operator<<()** [26/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const BusinessUserRole value )
```

**6.1.4.39 operator<<()** [27/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const BusinessUserStatus value )
```

**6.1.4.40 operator<<()** [28/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const BusinessUserStatus value )
```

**6.1.4.41 operator<<()** [29/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const SharedNotebookInstanceRestrictions value )
```

**6.1.4.42 operator<<()** [30/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const SharedNotebookInstanceRestrictions value )
```



**6.1.4.43 operator<<()** [31/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const ReminderEmailConfig value )
```

**6.1.4.44 operator<<()** [32/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const ReminderEmailConfig value )
```

**6.1.4.45 operator<<()** [33/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const BusinessInvitationStatus value )
```

**6.1.4.46 operator<<()** [34/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const BusinessInvitationStatus value )
```

**6.1.4.47 operator<<()** [35/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const ContactType value )
```

**6.1.4.48 operator<<()** [36/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const ContactType value )
```

**6.1.4.49 operator<<()** [37/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const EntityType value )
```

**6.1.4.50 operator<<()** [38/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const EntityType value )
```

**6.1.4.51 operator<<()** [39/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const RecipientStatus value )
```

**6.1.4.52 operator<<()** [40/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const RecipientStatus value )
```

**6.1.4.53 operator<<()** [41/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const CanMoveToContainerStatus value )
```

**6.1.4.54 operator<<()** [42/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const CanMoveToContainerStatus value )
```

**6.1.4.55 operator<<()** [43/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const RelatedContentType value )
```

**6.1.4.56 operator<<()** [44/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const RelatedContentType value )
```

**6.1.4.57 operator<<()** [45/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const RelatedContentAccess value )
```

**6.1.4.58 operator<<()** [46/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const RelatedContentAccess value )
```

**6.1.4.59 operator<<()** [47/48]

```
QEVERCLOUD_EXPORT QTextStream& qevercloud::operator<< (
    QTextStream & out,
    const UserIdentityType value )
```

**6.1.4.60 operator<<()** [48/48]

```
QEVERCLOUD_EXPORT QDebug& qevercloud::operator<< (
    QDebug & out,
    const UserIdentityType value )
```

**6.1.4.61 qHash()** [1/23]

```
uint qevercloud::qHash (  
    EDAMErrorCode value ) [inline]
```

**6.1.4.62 qHash()** [2/23]

```
uint qevercloud::qHash (  
    EDAMInvalidContactReason value ) [inline]
```

**6.1.4.63 qHash()** [3/23]

```
uint qevercloud::qHash (  
    ShareRelationshipPrivilegeLevel value ) [inline]
```

**6.1.4.64 qHash()** [4/23]

```
uint qevercloud::qHash (  
    PrivilegeLevel value ) [inline]
```

**6.1.4.65 qHash()** [5/23]

```
uint qevercloud::qHash (  
    ServiceLevel value ) [inline]
```

**6.1.4.66 qHash()** [6/23]

```
uint qevercloud::qHash (  
    QueryFormat value ) [inline]
```

**6.1.4.67 qHash()** [7/23]

```
uint qevercloud::qHash (  
    NoteSortOrder value ) [inline]
```

**6.1.4.68 qHash()** [8/23]

```
uint qevercloud::qHash (
    PremiumOrderStatus value ) [inline]
```

**6.1.4.69 qHash()** [9/23]

```
uint qevercloud::qHash (
    SharedNotebookPrivilegeLevel value ) [inline]
```

**6.1.4.70 qHash()** [10/23]

```
uint qevercloud::qHash (
    SharedNotePrivilegeLevel value ) [inline]
```

**6.1.4.71 qHash()** [11/23]

```
uint qevercloud::qHash (
    SponsoredGroupRole value ) [inline]
```

**6.1.4.72 qHash()** [12/23]

```
uint qevercloud::qHash (
    BusinessUserRole value ) [inline]
```

**6.1.4.73 qHash()** [13/23]

```
uint qevercloud::qHash (
    BusinessUserStatus value ) [inline]
```

**6.1.4.74 qHash()** [14/23]

```
uint qevercloud::qHash (
    SharedNotebookInstanceRestrictions value ) [inline]
```

**6.1.4.75 qHash()** [15/23]

```
uint qevercloud::qHash (  
    ReminderEmailConfig value ) [inline]
```

**6.1.4.76 qHash()** [16/23]

```
uint qevercloud::qHash (  
    BusinessInvitationStatus value ) [inline]
```

**6.1.4.77 qHash()** [17/23]

```
uint qevercloud::qHash (  
    ContactType value ) [inline]
```

**6.1.4.78 qHash()** [18/23]

```
uint qevercloud::qHash (  
    EntityType value ) [inline]
```

**6.1.4.79 qHash()** [19/23]

```
uint qevercloud::qHash (  
    RecipientStatus value ) [inline]
```

**6.1.4.80 qHash()** [20/23]

```
uint qevercloud::qHash (  
    CanMoveToContainerStatus value ) [inline]
```

**6.1.4.81 qHash()** [21/23]

```
uint qevercloud::qHash (  
    RelatedContentType value ) [inline]
```

**6.1.4.82 qHash()** [22/23]

```
uint qevercloud::qHash (
    RelatedContentAccess value ) [inline]
```

**6.1.4.83 qHash()** [23/23]

```
uint qevercloud::qHash (
    UserIdentityType value ) [inline]
```

**6.1.4.84 resetEvernoteNetworkProxy()**

```
QEVCLOUD_EXPORT void qevercloud::resetEvernoteNetworkProxy ( )
```

Reset network proxy settings used by QEverCloud to those returned from QNetworkProxy::applicationProxy static method.

This function is thread-safe although internally it operates on a static object containing proxy settings so it's not recommended to read and write proxy settings concurrently to avoid contention for static object.

**6.1.4.85 setEvernoteNetworkProxy()**

```
QEVCLOUD_EXPORT void qevercloud::setEvernoteNetworkProxy (
    QNetworkProxy proxy )
```

Setter for network proxy settings used by QEverCloud. If this function is never called, QEverCloud would use proxy settings returned from QNetworkProxy::applicationProxy static method.

This function is thread-safe although internally it operates on a static object containing proxy settings so it's not recommended to read and write proxy settings concurrently to avoid contention for static object.

WARNING: when QEverCloud is built with QtWebEngine and some proxy settings different from QNetworkProxy::applicationProxy are set, the OAuth call which loads the web page would not use them; instead it would use proxy settings from QNetworkProxy::applicationProxy. This limitation is imposed by Qt: <https://doc.qt.io/qt-5/qtwebengine-overview.html#proxy-support>

**6.1.4.86 setLogger()**

```
QEVCLOUD_EXPORT void qevercloud::setLogger (
    ILoggerPtr logger )
```

#### 6.1.4.87 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 you own algorithm.

`setNonceGenerator` is NOT thread safe.

#### 6.1.4.88 toRange() [1/2]

```
template<class Container >
QAssociativeContainerReferenceWrapper<Container> qevercloud::toRange (
    Container & container )
```

#### 6.1.4.89 toRange() [2/2]

```
template<class Container >
QAssociativeContainerConstReferenceWrapper<Container> qevercloud::toRange (
    const Container & container )
```

### 6.1.5 Variable Documentation

#### 6.1.5.1 CLASSIFICATION\_RECIPE\_SERVICE\_RECIPE

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.2 CLASSIFICATION\_RECIPE\_USER\_NON\_RECIPE

```
QEVERCLOUD_EXPORT const 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.



### 6.1.5.3 CLASSIFICATION\_RECIPE\_USER\_RECIPE

```
QEVERCLOUD_EXPORT const 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.

### 6.1.5.4 EDAM\_APP\_RATING\_MAX

```
QEVERCLOUD_EXPORT const qint16 qevercloud::EDAM_APP_RATING_MAX
```

### 6.1.5.5 EDAM\_APP\_RATING\_MIN

```
QEVERCLOUD_EXPORT const qint16 qevercloud::EDAM_APP_RATING_MIN
```

App Feedback Rating range

### 6.1.5.6 EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX

```
QEVERCLOUD_EXPORT const qint32 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.

### 6.1.5.7 EDAM\_APPLICATIONDATA\_NAME\_LEN\_MAX

```
QEVERCLOUD_EXPORT const qint32 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.

### 6.1.5.8 EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN

```
QEVERCLOUD_EXPORT const qint32 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.

### 6.1.5.9 EDAM\_APPLICATIONDATA\_NAME\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.10 EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 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.

#### 6.1.5.11 EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MIN

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_APPLICATIONDATA_VALUE_LEN_MIN
```

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

#### 6.1.5.12 EDAM\_APPLICATIONDATA\_VALUE\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.13 EDAM\_ATTRIBUTE\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_ATTRIBUTE_LEN_MAX
```

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

#### 6.1.5.14 EDAM\_ATTRIBUTE\_LEN\_MIN

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_ATTRIBUTE_LEN_MIN
```

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

#### 6.1.5.15 EDAM\_ATTRIBUTE\_LIST\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_ATTRIBUTE_LIST_MAX
```

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

#### 6.1.5.16 EDAM\_ATTRIBUTE\_MAP\_MAX

```
QEVERCLOUD_EXPORT const quint32 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.

#### 6.1.5.17 EDAM\_ATTRIBUTE\_REGEX

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

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

#### 6.1.5.18 EDAM\_BUSINESS\_MARKETING\_CODE\_REGEX\_PATTERN

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_BUSINESS_MARKETING_CODE_REGEX_PATTERN
```

Valid Evernote Business marketing code / affiliate code format.

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

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

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

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

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

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

#### 6.1.5.21 EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.22 EDAM\_BUSINESS\_NOTEBOOKS\_MAX

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

Maximum number of Notebooks in a business account

#### 6.1.5.23 EDAM\_BUSINESS\_NOTES\_MAX

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

Maximum number of Notes per business account

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

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

The maximum length of a business phone number.

#### 6.1.5.25 EDAM\_BUSINESS\_TAGS\_MAX

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

Maximum number of Tags per business account.

#### 6.1.5.26 EDAM\_BUSINESS\_URI\_LEN\_MAX

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

The maximum length of an Evernote Business URI

#### 6.1.5.27 EDAM\_BUSINESS\_WORKSPACES\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_BUSINESS_WORKSPACES_MAX
```

Maximum number of Workspaces in a business account

#### 6.1.5.28 EDAM\_CONNECTED\_IDENTITY\_REQUEST\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_CONNECTED_IDENTITY_REQUEST_MAX
```

The maximum number of connected identities a client can request.

#### 6.1.5.29 EDAM\_CONTENT\_CLASS\_FOOD\_MEAL

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.30 EDAM\_CONTENT\_CLASS\_HELLO\_ENCOUNTER

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.31 EDAM\_CONTENT\_CLASS\_HELLO\_PROFILE

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.32 EDAM\_CONTENT\_CLASS\_PENULTIMATE\_NOTEBOOK

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

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

#### 6.1.5.33 EDAM\_CONTENT\_CLASS\_PENULTIMATE\_PREFIX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.34 EDAM\_CONTENT\_CLASS\_SKITCH

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

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

#### 6.1.5.35 EDAM\_CONTENT\_CLASS\_SKITCH\_PDF

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

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

#### 6.1.5.36 EDAM\_CONTENT\_CLASS\_SKITCH\_PREFIX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.37 EDAM\_DEVICE\_DESCRIPTION\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_DEVICE_DESCRIPTION_LEN_MAX
```

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

#### 6.1.5.38 EDAM\_DEVICE\_DESCRIPTION\_REGEX

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

Regular expression for device description strings associated with long sessions.

#### 6.1.5.39 EDAM\_DEVICE\_ID\_LEN\_MAX

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

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

#### 6.1.5.40 EDAM\_DEVICE\_ID\_REGEX

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

Regular expression for device identifier strings associated with long sessions.

#### 6.1.5.41 EDAM\_EMAIL\_DOMAIN\_REGEX

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

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

#### 6.1.5.42 EDAM\_EMAIL\_LEN\_MAX

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

The maximum length of any email address

#### 6.1.5.43 EDAM\_EMAIL\_LEN\_MIN

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

The minimum length of any email address

#### 6.1.5.44 EDAM\_EMAIL\_LOCAL\_REGEX

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

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

#### 6.1.5.45 EDAM\_EMAIL\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.46 EDAM\_FIND\_CONTACT\_DEFAULT\_MAX\_RESULTS

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_FIND_CONTACT_DEFAULT_MAX_RESULTS
```

Default maximum number of results the service will return for findContact

#### 6.1.5.47 EDAM\_FIND\_CONTACT\_MAX\_RESULTS

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_FIND_CONTACT_MAX_RESULTS
```

Absolute maximum number of results the service will return for findContact

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

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.49 EDAM\_GET\_ORDERS\_MAX\_RESULTS

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_GET_ORDERS_MAX_RESULTS
```

Absolute maximum number of results the service will return for PersistentInternalMarket.getOrders()

#### 6.1.5.50 EDAM\_GUID\_LEN\_MAX

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

The maximum length of a GUID generated by the Evernote service

#### 6.1.5.51 EDAM\_GUID\_LEN\_MIN

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

The minimum length of a GUID generated by the Evernote service

#### 6.1.5.52 EDAM\_GUID\_REGEX

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

GUIDs generated by the Evernote service will match the provided pattern

#### 6.1.5.53 EDAM\_HASH\_LEN

```
QEVERCLOUD_EXPORT const quint32 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.

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

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.55 EDAM\_INDEXABLE\_PLAINTEXT\_MIME\_TYPES

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

The set of plain text MIME types that Evernote will parse and index for searching. The MIME types which start with "text/" will be handled separately by each client (i.e. hard-coded in each client).

#### 6.1.5.56 EDAM\_INDEXABLE\_RESOURCE\_MIME\_TYPES

```
QEVERCLOUD_EXPORT const 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, PDFs and plain text files, which are handled in a different way.

#### 6.1.5.57 EDAM\_MAX\_PREFERENCES

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MAX_PREFERENCES
```

Maximum number of name/value pairs allowed

#### 6.1.5.58 EDAM\_MAX\_VALUES\_PER\_PREFERENCE

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MAX_VALUES_PER_PREFERENCE
```

Maximum number of values per preference name when using values of size no greater than EDAM\_PREFERENCE\_VALUE\_LEN\_MAX.

#### 6.1.5.59 EDAM\_MESSAGE\_ATTACHMENT\_SNIPPET\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MESSAGE_ATTACHMENT_SNIPPET_LEN_MAX
```

The maximum length of a message attachment snippet in unicode characters.



#### 6.1.5.60 EDAM\_MESSAGE\_ATTACHMENT\_SNIPPET\_REGEX

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MESSAGE_ATTACHMENT_SNIPPET_REGEX
```

The regex to validate message attachment snippets against

#### 6.1.5.61 EDAM\_MESSAGE\_ATTACHMENT\_TITLE\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MESSAGE_ATTACHMENT_TITLE_LEN_MAX
```

The maximum length of a message attachment title in unicode characters.

#### 6.1.5.62 EDAM\_MESSAGE\_ATTACHMENT\_TITLE\_REGEX

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MESSAGE_ATTACHMENT_TITLE_REGEX
```

The regex to validate message attachment titles against

#### 6.1.5.63 EDAM\_MESSAGE\_ATTACHMENTS\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MESSAGE_ATTACHMENTS_MAX
```

The maximum number of attachments a Message can have.

#### 6.1.5.64 EDAM\_MESSAGE\_BODY\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MESSAGE_BODY_LEN_MAX
```

The maximum length of a message body in unicode characters.

#### 6.1.5.65 EDAM\_MESSAGE\_BODY\_REGEX

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MESSAGE_BODY_REGEX
```

The regex to validate message.body against

#### 6.1.5.66 EDAM\_MESSAGE\_RECIPIENTS\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MESSAGE_RECIPIENTS_MAX
```

The maximum number of recipients on a MessageThread.

#### 6.1.5.67 EDAM\_MIME\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_MIME_LEN_MAX
```

The maximum length of any MIME type string given to Evernote

#### 6.1.5.68 EDAM\_MIME\_LEN\_MIN

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

The minimum length of any MIME type string given to Evernote

#### 6.1.5.69 EDAM\_MIME\_REGEX

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

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

#### 6.1.5.70 EDAM\_MIME\_TYPE\_AAC

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

Canonical MIME type string for AAC audio resources

#### 6.1.5.71 EDAM\_MIME\_TYPE\_AMR

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

Canonical MIME type string for AMR audio resources

#### 6.1.5.72 EDAM\_MIME\_TYPE\_BMP

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_BMP
```

Canonical MIME type string for BMP image resources

#### 6.1.5.73 EDAM\_MIME\_TYPE\_DEFAULT

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

MIME type used for attachments of an unspecified type

#### 6.1.5.74 EDAM\_MIME\_TYPE\_GIF

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

Canonical MIME type string for GIF image resources

#### 6.1.5.75 EDAM\_MIME\_TYPE\_INK

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

Canonical MIME type string for Evernote Ink resources

**6.1.5.76 EDAM\_MIME\_TYPE\_JPEG**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_JPEG`

Canonical MIME type string for JPEG image resources

**6.1.5.77 EDAM\_MIME\_TYPE\_M4A**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_M4A`

Canonical MIME type string for MP4 audio resources

**6.1.5.78 EDAM\_MIME\_TYPE\_MP3**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_MP3`

Canonical MIME type string for MP3 audio resources

**6.1.5.79 EDAM\_MIME\_TYPE\_MP4\_VIDEO**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_MP4_VIDEO`

Canonical MIME type string for MP4 video resources

**6.1.5.80 EDAM\_MIME\_TYPE\_PDF**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_PDF`

Canonical MIME type string for PDF resources

**6.1.5.81 EDAM\_MIME\_TYPE\_PNG**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_PNG`

Canonical MIME type string for PNG image resources

**6.1.5.82 EDAM\_MIME\_TYPE\_TIFF**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_TIFF`

Canonical MIME type string for TIFF image resources

**6.1.5.83 EDAM\_MIME\_TYPE\_WAV**

`QEVERCLOUD_EXPORT const QString qevercloud::EDAM_MIME_TYPE_WAV`

Canonical MIME type string for WAV audio resources

#### 6.1.5.84 EDAM\_MIME\_TYPES

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.85 EDAM\_NOTE\_BUSINESS\_SHARED\_NOTE\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_NOTE_BUSINESS_SHARED_NOTE_MAX
```

Maximum number of [SharedNote](#) records per business note

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

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

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

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

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

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

#### 6.1.5.88 EDAM\_NOTE\_CONTENT\_CLASS\_REGEX

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

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

#### 6.1.5.89 EDAM\_NOTE\_CONTENT\_LEN\_MAX

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

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

#### 6.1.5.90 EDAM\_NOTE\_CONTENT\_LEN\_MIN

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

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

#### 6.1.5.91 EDAM\_NOTE\_LOCK\_VIEWERS\_NOTES\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_NOTE_LOCK_VIEWERS_NOTES_MAX
```

The maximum number of separate notes that may be queried in a single call to `NoteStore.getViewersForNotes`.

#### 6.1.5.92 EDAM\_NOTE\_PERSONAL\_SHARED\_NOTE\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_NOTE_PERSONAL_SHARED_NOTE_MAX
```

Maximum number of [SharedNote](#) records per personal note

#### 6.1.5.93 EDAM\_NOTE\_RESOURCES\_MAX

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

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

#### 6.1.5.94 EDAM\_NOTE\_SIZE\_MAX\_FREE

```
QEVERCLOUD_EXPORT const 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).

#### 6.1.5.95 EDAM\_NOTE\_SIZE\_MAX\_PREMIUM

```
QEVERCLOUD_EXPORT const 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).

#### 6.1.5.96 EDAM\_NOTE\_SOURCE\_MAIL\_CLIP

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

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

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

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.98 EDAM\_NOTE\_SOURCE\_WEB\_CLIP

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

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

#### 6.1.5.99 EDAM\_NOTE\_SOURCE\_WEB\_CLIP\_SIMPLIFIED

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_NOTE_SOURCE_WEB_CLIP_SIMPLIFIED
```

Standardized value for the 'source' NoteAttribute for notes that were clipped using the "simplified article" function of the clipper.

#### 6.1.5.100 EDAM\_NOTE\_TAGS\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_NOTE_TAGS_MAX
```

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

#### 6.1.5.101 EDAM\_NOTE\_TITLE\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_NOTE_TITLE_LEN_MAX
```

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

#### 6.1.5.102 EDAM\_NOTE\_TITLE\_LEN\_MIN

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_NOTE_TITLE_LEN_MIN
```

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

#### 6.1.5.103 EDAM\_NOTE\_TITLE\_QUALITY\_HIGH

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_HIGH
```

A [NoteAttributes.noteTitleQuality](#) value indicating that the quality of an automatically generated note title is high. Examples of high quality titles include those based on a scanned business card, such as "John Doe - Scanned Business Card".

#### 6.1.5.104 EDAM\_NOTE\_TITLE\_QUALITY\_LOW

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_LOW
```

A [NoteAttributes.noteTitleQuality](#) value indicating that the quality of an automatically generated note title is low. Examples of low quality titles include those based on a note's type and location, such as "Snapshot from 123 Sesame Street in New York".

#### 6.1.5.105 EDAM\_NOTE\_TITLE\_QUALITY\_MEDIUM

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_MEDIUM
```

A [NoteAttributes.noteTitleQuality](#) value indicating that the quality of an automatically generated note title is medium. Examples of medium quality titles include those based on a calendar entry, such as "Note from Weekly Staff Meeting".

#### 6.1.5.106 EDAM\_NOTE\_TITLE\_QUALITY\_UNTITLED

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_UNTITLED
```

A [NoteAttributes.noteTitleQuality](#) value indicating that a note has no meaningful title, only a placeholder value such as "Untitled Note".

#### 6.1.5.107 EDAM\_NOTE\_TITLE\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.108 EDAM\_NOTEBOOK\_BUSINESS\_SHARED\_NOTEBOOK\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_NOTEBOOK_BUSINESS_SHARED_NOTEBOOK_MAX
```

Maximum number of shared notebooks per business notebook

#### 6.1.5.109 EDAM\_NOTEBOOK\_NAME\_LEN\_MAX

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

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

#### 6.1.5.110 EDAM\_NOTEBOOK\_NAME\_LEN\_MIN

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

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

#### 6.1.5.111 EDAM\_NOTEBOOK\_NAME\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.112 EDAM\_NOTEBOOK\_PERSONAL\_SHARED\_NOTEBOOK\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_NOTEBOOK_PERSONAL_SHARED_NOTEBOOK_MAX
```

Maximum number of shared notebooks per personal notebook

#### 6.1.5.113 EDAM\_NOTEBOOK\_STACK\_LEN\_MAX

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

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

#### 6.1.5.114 EDAM\_NOTEBOOK\_STACK\_LEN\_MIN

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

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

#### 6.1.5.115 EDAM\_NOTEBOOK\_STACK\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.116 EDAM\_OPEN\_ID\_ACCESS\_TOKEN\_MAX

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_OPEN_ID_ACCESS_TOKEN_MAX
```

Maximum length for OpenID token. There is no official enforced limit. The length of the Token ID depends on the provider. 1000 seems to be the safest value at this time.

#### 6.1.5.117 EDAM\_PREFERENCE\_BUSINESS\_DEFAULT\_NOTEBOOK

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_PREFERENCE_BUSINESS_DEFAULT_NOTEBOOK
```

The name of the preferences entry that contains the notebook GUID (not the linked notebook) of the default business notebook. It must be in the format `EDAM_GUID_REGEX`. If a default business notebook is not set and the user is a business user the user should be prompted to set the default business notebook. The default business notebook must be a read/write notebook. Whenever the default business notebook guid is used, it must be revalidated as a writable notebook. If it is not valid, the user should be re-prompted to set the value. This value is used by clients only.



#### 6.1.5.118 EDAM\_PREFERENCE\_BUSINESS\_QUICKNOTE

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_PREFERENCE_BUSINESS_QUICKNOTE
```

The name of the preferences entry that contains a boolean indicating that default quicknotes should go into a business notebook. The EDAM\_PREFERENCE\_BUSINESS\_DEFAULT\_NOTEBOOK must be set correctly for this preference to be honored. The quicknote preferences should only be set to "true", if quicknote should use a business notebook. Any value other than "true" (or the omission of a value) should be treated as "false". In this case, quicknotes should be created in the user's personal default notebook. The interface should not allow users to set quicknote to a business notebook without a valid default business notebook selected, however, clients should handle the edge case of an invalid business notebook guid. If a user stops being a business user or does not have write access to any business notebooks the quicknote preference should be ignored.

#### 6.1.5.119 EDAM\_PREFERENCE\_NAME\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PREFERENCE_NAME_LEN_MAX
```

Maximum length of a preference name

#### 6.1.5.120 EDAM\_PREFERENCE\_NAME\_LEN\_MIN

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PREFERENCE_NAME_LEN_MIN
```

Minimum length of a preference name

#### 6.1.5.121 EDAM\_PREFERENCE\_NAME\_REGEX

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

A preference name must match this regex.

#### 6.1.5.122 EDAM\_PREFERENCE\_ONLY\_ONE\_VALUE\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PREFERENCE_ONLY_ONE_VALUE_LEN_MAX
```

The maximum length of a preference value if you only use one value per preference rather than up to EDAM\_MAX\_VALUES\_PER\_PREFERENCE. This option is useful if you want a single string that is larger than EDAM\_PREFERENCE\_VALUE\_LEN\_MAX and would otherwise need to split the string into multiple pieces to store it.

#### 6.1.5.123 EDAM\_PREFERENCE\_ONLY\_ONE\_VALUE\_REGEX

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_PREFERENCE_ONLY_ONE_VALUE_REGEX
```

A preference value must match this regex if you are using a single value for a preference.

#### 6.1.5.124 EDAM\_PREFERENCE\_SHORTCUTS

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

The name of the preferences entry that contains shortcuts.

#### 6.1.5.125 EDAM\_PREFERENCE\_SHORTCUTS\_MAX\_VALUES

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PREFERENCE_SHORTCUTS_MAX_VALUES
```

The maximum number of shortcuts that a user may have.

#### 6.1.5.126 EDAM\_PREFERENCE\_VALUE\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PREFERENCE_VALUE_LEN_MAX
```

Maximum length of a preference value

#### 6.1.5.127 EDAM\_PREFERENCE\_VALUE\_LEN\_MIN

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PREFERENCE_VALUE_LEN_MIN
```

Minimum length of a preference value

#### 6.1.5.128 EDAM\_PREFERENCE\_VALUE\_REGEX

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

A preference value must match this regex if you are using more than a single value for a preference.

#### 6.1.5.129 EDAM\_PROMOTION\_ID\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PROMOTION_ID_LEN_MAX
```

The maximum length of a promotion ID in unicode characters.

#### 6.1.5.130 EDAM\_PROMOTION\_ID\_REGEX

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_PROMOTION_ID_REGEX
```

The regex to validate promotion IDs against.

#### 6.1.5.131 EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_PUBLISHING_DESCRIPTION_LEN_MAX
```

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

#### 6.1.5.132 EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MIN

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

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

#### 6.1.5.133 EDAM\_PUBLISHING\_DESCRIPTION\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.134 EDAM\_PUBLISHING\_URI\_LEN\_MAX

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

The maximum length of a public notebook URI component

#### 6.1.5.135 EDAM\_PUBLISHING\_URI\_LEN\_MIN

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

The minimum length of a public notebook URI component

#### 6.1.5.136 EDAM\_PUBLISHING\_URI\_PROHIBITED

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

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

#### 6.1.5.137 EDAM\_PUBLISHING\_URI\_REGEX

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

A public notebook URI component must match the provided pattern

#### 6.1.5.138 EDAM\_RELATED\_MAX\_EXPERTS

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_RELATED_MAX_EXPERTS
```

The maximum number of experts that will be returned from a `findRelated()` query

#### 6.1.5.139 EDAM\_RELATED\_MAX\_NOTEBOOKS

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

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

#### 6.1.5.140 EDAM\_RELATED\_MAX\_NOTES

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

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

#### 6.1.5.141 EDAM\_RELATED\_MAX\_RELATED\_CONTENT

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_RELATED_MAX_RELATED_CONTENT
```

The maximum number of related content snippets that will be returned from a findRelated() query.

#### 6.1.5.142 EDAM\_RELATED\_MAX\_TAGS

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

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

#### 6.1.5.143 EDAM\_RELATED\_PLAINTEXT\_LEN\_MAX

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

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

#### 6.1.5.144 EDAM\_RELATED\_PLAINTEXT\_LEN\_MIN

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

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

#### 6.1.5.145 EDAM\_RESOURCE\_SIZE\_MAX\_FREE

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

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

#### 6.1.5.146 EDAM\_RESOURCE\_SIZE\_MAX\_PREMIUM

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

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

**6.1.5.147 EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MAX**

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

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

**6.1.5.148 EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MIN**

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

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

**6.1.5.149 EDAM\_SAVED\_SEARCH\_NAME\_REGEX**

```
QEVERCLOUD_EXPORT const 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.

**6.1.5.150 EDAM\_SEARCH\_QUERY\_LEN\_MAX**

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

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

**6.1.5.151 EDAM\_SEARCH\_QUERY\_LEN\_MIN**

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

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

**6.1.5.152 EDAM\_SEARCH\_QUERY\_REGEX**

```
QEVERCLOUD_EXPORT const 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.

**6.1.5.153 EDAM\_SEARCH\_SUGGESTIONS\_MAX**

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

Maximum number of search suggestions that can be returned

**6.1.5.154 EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MAX**

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

Maximum length of the search suggestion prefix

**6.1.5.155 EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MIN**

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

Minimum length of the search suggestion prefix

**6.1.5.156 EDAM\_SNIPPETS\_NOTES\_MAX**

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_SNIPPETS_NOTES_MAX
```

The maximum number of note snippets you can retrieve in a single request

**6.1.5.157 EDAM\_SOURCE\_APPLICATION\_ANDROID\_SHARE\_EXTENSION**

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_APPLICATION_ANDROID_SHARE_EXTENSION
```

The [NoteAttributes.sourceApplication](#) value used for notes captured with the Android share extension.

**6.1.5.158 EDAM\_SOURCE\_APPLICATION\_EN\_SCANSNAP**

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_APPLICATION_EN_SCANSNAP
```

The [NoteAttributes.sourceApplication](#) value used for notes captured by PFU ScanSnap Evernote Edition.

**6.1.5.159 EDAM\_SOURCE\_APPLICATION\_EWC**

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_APPLICATION_EWC
```

The [NoteAttributes.sourceApplication](#) value used for notes captured with the Embedded Web Clipper.

**6.1.5.160 EDAM\_SOURCE\_APPLICATION\_IOS\_SHARE\_EXTENSION**

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_APPLICATION_IOS_SHARE_EXTENSION
```

The [NoteAttributes.sourceApplication](#) value used for notes captured with the iOS share extension.

**6.1.5.161 EDAM\_SOURCE\_APPLICATION\_MOLESKINE**

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_APPLICATION_MOLESKINE
```

The [NoteAttributes.sourceApplication](#) value used for notes captured by the Moleskine page camera.

#### 6.1.5.162 EDAM\_SOURCE\_APPLICATION\_POSTIT

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_APPLICATION_POSTIT
```

The [NoteAttributes.sourceApplication](#) value used for notes captured by the Post-it camera.

#### 6.1.5.163 EDAM\_SOURCE\_APPLICATION\_WEB\_CLIPPER

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_APPLICATION_WEB_CLIPPER
```

The [NoteAttributes.sourceApplication](#) value used for notes captured with the Evernote Web Clipper.

#### 6.1.5.164 EDAM\_SOURCE\_OUTLOOK\_CLIPPER

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_SOURCE_OUTLOOK_CLIPPER
```

The [NoteAttributes.source](#) value used for notes captured by the Microsoft Outlook clipper.

#### 6.1.5.165 EDAM\_TAG\_NAME\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_TAG_NAME_LEN_MAX
```

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

#### 6.1.5.166 EDAM\_TAG\_NAME\_LEN\_MIN

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_TAG_NAME_LEN_MIN
```

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

#### 6.1.5.167 EDAM\_TAG\_NAME\_REGEX

```
QEVERCLOUD_EXPORT const 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.

#### 6.1.5.168 EDAM\_TIMEZONE\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_TIMEZONE_LEN_MAX
```

The maximum length of a timezone specification string

#### 6.1.5.169 EDAM\_TIMEZONE\_LEN\_MIN

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

The minimum length of a timezone specification string

#### 6.1.5.170 EDAM\_TIMEZONE\_REGEX

```
QEVERCLOUD_EXPORT const 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

#### 6.1.5.171 EDAM\_USER\_LINKED\_NOTEBOOK\_MAX

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

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

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

```
QEVERCLOUD_EXPORT const 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".

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

```
QEVERCLOUD_EXPORT const 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.

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

```
QEVERCLOUD_EXPORT const 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.



**6.1.5.175 EDAM\_USER\_NAME\_LEN\_MAX**

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

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

**6.1.5.176 EDAM\_USER\_NAME\_LEN\_MIN**

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

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

**6.1.5.177 EDAM\_USER\_NAME\_REGEX**

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

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

**6.1.5.178 EDAM\_USER\_NOTEBOOKS\_MAX**

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

Maximum number of Notebooks per user

**6.1.5.179 EDAM\_USER\_NOTES\_MAX**

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

Maximum number of Notes per user

**6.1.5.180 EDAM\_USER\_PASSWORD\_LEN\_MAX**

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

The maximum length of an Evernote user password

**6.1.5.181 EDAM\_USER\_PASSWORD\_LEN\_MIN**

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

The minimum length of an Evernote user password

**6.1.5.182 EDAM\_USER\_PASSWORD\_REGEX**

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

Evernote user passwords must match this regular expression

**6.1.5.183 EDAM\_USER\_PROFILE\_PHOTO\_MAX\_BYTES**

```
QEVERCLOUD_EXPORT const qint32 qevercloud::EDAM_USER_PROFILE_PHOTO_MAX_BYTES
```

Maximum user profile photo size, in bytes, that clients may send to the service. Photos may be resized before being stored on the service.

**6.1.5.184 EDAM\_USER\_RECENT\_MAILED\_ADDRESSES\_MAX**

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

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

**6.1.5.185 EDAM\_USER\_SAVED\_SEARCHES\_MAX**

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

Maximum number of SavedSearches per account

**6.1.5.186 EDAM\_USER\_TAGS\_MAX**

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

Maximum number of Tags per account

**6.1.5.187 EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS**

```
QEVERCLOUD_EXPORT const 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.

**6.1.5.188 EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_FIRST\_MONTH**

```
QEVERCLOUD_EXPORT const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS_FIRST_MONTH
```

The number of bytes of new data that may be uploaded to new business account during the first month. 50GB.

**6.1.5.189 EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_NEXT\_MONTH**

```
QEVERCLOUD_EXPORT const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS_NEXT_MONTH
```

The number of bytes of new data that may be uploaded to a business account for the next month. 20GB.

**6.1.5.190 EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_PER\_USER**

```
QEVERCLOUD_EXPORT const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS_PER_USER
```

The number of bytes of new data that may be uploaded to a Business for each member of the business per month. The total bytes available can be determined by multiplying this with the number of business users.

**6.1.5.191 EDAM\_USER\_UPLOAD\_LIMIT\_FREE**

```
QEVERCLOUD_EXPORT const 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.

**6.1.5.192 EDAM\_USER\_UPLOAD\_LIMIT\_PLUS**

```
QEVERCLOUD_EXPORT const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_PLUS
```

The number of bytes of new data that may be uploaded each month to an account at a Plus service level.

**6.1.5.193 EDAM\_USER\_UPLOAD\_LIMIT\_PREMIUM**

```
QEVERCLOUD_EXPORT const 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.

**6.1.5.194 EDAM\_USER\_UPLOAD\_SURVEY\_THRESHOLD**

```
QEVERCLOUD_EXPORT const qint64 qevercloud::EDAM_USER_UPLOAD_SURVEY_THRESHOLD
```

The number of bytes of new data uploaded in a monthly quota cycle at which point users should be prompted with a survey to gather information on how they are using Evernote.

**6.1.5.195 EDAM\_USER\_USERNAME\_LEN\_MAX**

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

The maximum length of an Evernote username

**6.1.5.196 EDAM\_USER\_USERNAME\_LEN\_MIN**

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

The minimum length of an Evernote username

#### 6.1.5.197 EDAM\_USER\_USERNAME\_REGEX

```
QEVERCLOUD_EXPORT const 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"

#### 6.1.5.198 EDAM\_USER\_WORKSPACES\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_USER_WORKSPACES_MAX
```

Maximum number of Workspaces per user

#### 6.1.5.199 EDAM\_VAT\_REGEX

```
QEVERCLOUD_EXPORT const 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/re>

#### 6.1.5.200 EDAM\_VERSION\_MAJOR

```
QEVERCLOUD_EXPORT const quint16 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.

#### 6.1.5.201 EDAM\_VERSION\_MINOR

```
QEVERCLOUD_EXPORT const quint16 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.

#### 6.1.5.202 EDAM\_WORKSPACE\_DESCRIPTION\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_WORKSPACE_DESCRIPTION_LEN_MAX
```

The maximum length of a `Workspace.description`, in Unicode characters

#### 6.1.5.203 EDAM\_WORKSPACE\_NAME\_LEN\_MAX

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_WORKSPACE_NAME_LEN_MAX
```

The maximum length of a `Workspace.name`, in Unicode characters

#### 6.1.5.204 EDAM\_WORKSPACE\_NAME\_LEN\_MIN

```
QEVERCLOUD_EXPORT const quint32 qevercloud::EDAM_WORKSPACE_NAME_LEN_MIN
```

The minimum length of a Workspace.name, in Unicode characters

#### 6.1.5.205 EDAM\_WORKSPACE\_NAME\_REGEX

```
QEVERCLOUD_EXPORT const QString qevercloud::EDAM_WORKSPACE_NAME_REGEX
```

All Workspace.name fields must match this pattern. This excludes control chars or line/paragraph separators. The string may not begin or end with whitespace.

#### 6.1.5.206 EverCloudExceptionData

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



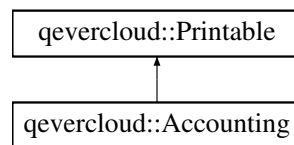
## Chapter 7

# Class Documentation

### 7.1 qevercloud::Accounting Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::Accounting:



#### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [Accounting](#) &other) const
- bool [operator!=](#) (const [Accounting](#) &other) const

#### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [Timestamp](#) > [uploadLimitEnd](#)
- [Optional](#)< [qint64](#) > [uploadLimitNextMonth](#)
- [Optional](#)< [PremiumOrderStatus](#) > [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](#) > [businessRole](#)
- [Optional](#)< [qint32](#) > [unitDiscount](#)
- [Optional](#)< [Timestamp](#) > [nextChargeDate](#)
- [Optional](#)< [qint32](#) > [availablePoints](#)

### 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]
```

#### 7.1.2.3 `print()`

```
virtual void qevercloud::Accounting::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.1.3 Member Data Documentation



#### 7.1.3.1 availablePoints

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

NOT DOCUMENTED

#### 7.1.3.2 businessId

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

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

#### 7.1.3.3 businessName

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

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

#### 7.1.3.4 businessRole

`Optional< BusinessUserRole > qevercloud::Accounting::businessRole`

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

#### 7.1.3.5 currency

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

ISO 4217 currency code

#### 7.1.3.6 lastFailedCharge

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

Date the last time a charge was attempted and failed.

#### 7.1.3.7 lastFailedChargeReason

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

Reason provided for the charge failure

#### 7.1.3.8 lastRequestedCharge

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

Date charge last attempted

#### 7.1.3.9 lastSuccessfulCharge

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

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

#### 7.1.3.10 localData

`EverCloudLocalData qevercloud::Accounting::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.1.3.11 nextChargeDate

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

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

#### 7.1.3.12 nextPaymentDue

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

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

#### 7.1.3.13 premiumCommerceService

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

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

#### 7.1.3.14 premiumLockUntil

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

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

#### 7.1.3.15 premiumOrderNumber

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

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

#### 7.1.3.16 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.

#### 7.1.3.17 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).

#### 7.1.3.18 premiumServiceStatus

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

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

#### 7.1.3.19 premiumSubscriptionNumber

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

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

#### 7.1.3.20 unitDiscount

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

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

#### 7.1.3.21 unitPrice

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

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

#### 7.1.3.22 updated

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

The date any modification where made to this record.

#### 7.1.3.23 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.

### 7.1.3.24 uploadLimitNextMonth

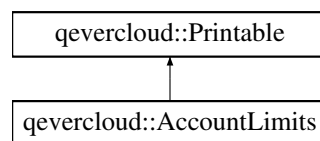
```
Optional< qint64 > qevercloud::Accounting::uploadLimitNextMonth
```

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::AccountLimits Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::AccountLimits:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [AccountLimits](#) &other) const
- bool [operator!=](#) (const [AccountLimits](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< qint32 > [userMailLimitDaily](#)
- [Optional](#)< qint64 > [noteSizeMax](#)
- [Optional](#)< qint64 > [resourceSizeMax](#)
- [Optional](#)< qint32 > [userLinkedNotebookMax](#)
- [Optional](#)< qint64 > [uploadLimit](#)
- [Optional](#)< qint32 > [userNoteCountMax](#)
- [Optional](#)< qint32 > [userNotebookCountMax](#)
- [Optional](#)< qint32 > [userTagCountMax](#)
- [Optional](#)< qint32 > [noteTagCountMax](#)
- [Optional](#)< qint32 > [userSavedSearchesMax](#)
- [Optional](#)< qint32 > [noteResourceCountMax](#)

### 7.2.1 Detailed Description

This structure is used to provide account limits that are in effect for this user.

### 7.2.2 Member Function Documentation

### 7.2.2.1 operator!=(())

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

### 7.2.2.2 operator==(())

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

### 7.2.2.3 print()

```
virtual void qevercloud::AccountLimits::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.2.3 Member Data Documentation

### 7.2.3.1 localData

[EverCloudLocalData](#) qevercloud::AccountLimits::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.2.3.2 noteResourceCountMax

[Optional](#)< qint32 > qevercloud::AccountLimits::noteResourceCountMax

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

### 7.2.3.3 noteSizeMax

[Optional](#)< qint64 > qevercloud::AccountLimits::noteSizeMax

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

#### 7.2.3.4 noteTagCountMax

`Optional< qint32 > qevercloud::AccountLimits::noteTagCountMax`

Maximum number of Tags per [Note](#)

#### 7.2.3.5 resourceSizeMax

`Optional< qint64 > qevercloud::AccountLimits::resourceSizeMax`

Maximum size of a resource, in bytes

#### 7.2.3.6 uploadLimit

`Optional< qint64 > qevercloud::AccountLimits::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.

#### 7.2.3.7 userLinkedNotebookMax

`Optional< qint32 > qevercloud::AccountLimits::userLinkedNotebookMax`

Maximum number of linked notebooks per account.

#### 7.2.3.8 userMailLimitDaily

`Optional< qint32 > qevercloud::AccountLimits::userMailLimitDaily`

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

#### 7.2.3.9 userNotebookCountMax

`Optional< qint32 > qevercloud::AccountLimits::userNotebookCountMax`

Maximum number of Notebooks per user

#### 7.2.3.10 userNoteCountMax

`Optional< qint32 > qevercloud::AccountLimits::userNoteCountMax`

Maximum number of Notes per user

#### 7.2.3.11 userSavedSearchesMax

`Optional< quint32 > qevercloud::AccountLimits::userSavedSearchesMax`

Maximum number of SavedSearches per account

#### 7.2.3.12 userTagCountMax

`Optional< quint32 > qevercloud::AccountLimits::userTagCountMax`

Maximum number of Tags per account

## 7.3 qevercloud::IDurableService::AsyncRequest Struct Reference

```
#include <DurableService.h>
```

### Public Member Functions

- [AsyncRequest](#) (const char \*name, QString description, [AsyncServiceCall](#) &&call)

### Public Attributes

- const char \* [m\\_name](#)
- QString [m\\_description](#)
- [AsyncServiceCall](#) [m\\_call](#)

### 7.3.1 Constructor & Destructor Documentation

#### 7.3.1.1 AsyncRequest()

```
qevercloud::IDurableService::AsyncRequest::AsyncRequest (
    const char * name,
    QString description,
    AsyncServiceCall && call ) [inline]
```

### 7.3.2 Member Data Documentation

### 7.3.2.1 m\_call

```
AsyncServiceCall qevercloud::IDurableService::AsyncRequest::m_call
```

### 7.3.2.2 m\_description

```
QString qevercloud::IDurableService::AsyncRequest::m_description
```

### 7.3.2.3 m\_name

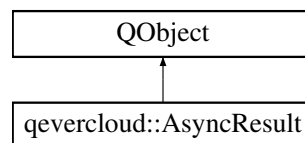
```
const char* qevercloud::IDurableService::AsyncRequest::m_name
```

## 7.4 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, [EverCloudExceptionDataPtr](#) error, [IRequestContextPtr](#) ctx)  
*Emitted upon asynchronous call completion.*

### Public Member Functions

- [AsyncResult](#) (QString url, QByteArray postData, [IRequestContextPtr](#) ctx, [ReadFunctionType](#) read↵  
Function=[AsyncResult::asls](#), bool autoDelete=true, QObject \*parent=nullptr)
- [AsyncResult](#) (QNetworkRequest request, QByteArray postData, [IRequestContextPtr](#) ctx, [ReadFunctionType](#)  
readFunction=[AsyncResult::asls](#), bool autoDelete=true, QObject \*parent=nullptr)
- [AsyncResult](#) (QVariant result, [EverCloudExceptionDataPtr](#) error, [IRequestContextPtr](#) ctx, bool auto↵  
Delete=true, QObject \*parent=nullptr)
- [~AsyncResult](#) ()
- bool [waitForFinished](#) (int timeout=-1)  
*Wait for asynchronous operation to complete.*



## Static Public Member Functions

- static QVariant [asIs](#) (QByteArray replyData)

## Friends

- class [DurableService](#)

### 7.4.1 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,
        EverCloudExceptionDataPtr error,
        IRequestContextPtr ctx)
    {
        if (error) {
            // do something in case of an error
        }
        else {
            Note note = result.value<Note>();
            // process returned result
        }
    }
});
```

### 7.4.2 Member Typedef Documentation

#### 7.4.2.1 ReadFunctionType

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

### 7.4.3 Constructor & Destructor Documentation

#### 7.4.3.1 AsyncResult() [1/3]

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

#### 7.4.3.2 AsyncResult() [2/3]

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

#### 7.4.3.3 AsyncResult() [3/3]

```
qevercloud::AsyncResult::AsyncResult (
    QVariant result,
    EverCloudExceptionDataPtr error,
    IRequestContextPtr ctx,
    bool autoDelete = true,
    QObject * parent = nullptr )
```

Constructor accepting already prepared value and/or exception, for use in tests

#### 7.4.3.4 ~AsyncResult()

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

### 7.4.4 Member Function Documentation

#### 7.4.4.1 asIs()

```
static QVariant qevercloud::AsyncResult::asIs (
    QByteArray replyData ) [static]
```

#### 7.4.4.2 finished

```
void qevercloud::AsyncResult::finished (
    QVariant result,
    EverCloudExceptionDataPtr error,
    IRequestContextPtr ctx ) [signal]
```

Emitted upon asynchronous call completion.

## Parameters

<i>result</i>	Request result
<i>error</i>	Non-nullptr in case of an error. See <a href="#">EverCloudExceptionData</a> for more details
<i>ctx</i>	Request context used to make the request

[AsyncResult](#) deletes itself after emitting this signal (if `autoDelete` parameter passed to its constructor was set to true). You don't have to manage it's lifetime explicitly.

## 7.4.4.3 waitFinished()

```
bool qevercloud::AsyncResult::waitFinished (
    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 successfully, false in case of the timeout

## 7.4.5 Friends And Related Function Documentation

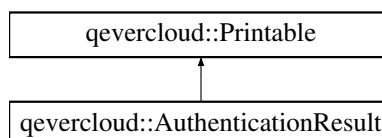
## 7.4.5.1 DurableService

```
friend class DurableService [friend]
```

## 7.5 qevercloud::AuthenticationResult Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::AuthenticationResult`:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [AuthenticationResult](#) &other) const
- bool [operator!=](#) (const [AuthenticationResult](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Timestamp](#) [currentTime](#) = 0
- [QString](#) [authenticationToken](#)
- [Timestamp](#) [expiration](#) = 0
- [Optional](#)< [User](#) > [user](#)
- [Optional](#)< [PublicUserInfo](#) > [publicUserInfo](#)
- [Optional](#)< [QString](#) > [noteStoreUrl](#)
- [Optional](#)< [QString](#) > [webApiUrlPrefix](#)
- [Optional](#)< bool > [secondFactorRequired](#)
- [Optional](#)< [QString](#) > [secondFactorDeliveryHint](#)
- [Optional](#)< [UserUrls](#) > [urls](#)

### 7.5.1 Detailed Description

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

### 7.5.2 Member Function Documentation

#### 7.5.2.1 [operator!=\(\)](#)

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

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

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

#### 7.5.2.3 [print\(\)](#)

```
virtual void qevercloud::AuthenticationResult::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.5.3 Member Data Documentation

#### 7.5.3.1 authenticationToken

`QString qevercloud::AuthenticationResult::authenticationToken`

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

#### 7.5.3.2 currentTime

`Timestamp qevercloud::AuthenticationResult::currentTime = 0`

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

#### 7.5.3.3 expiration

`Timestamp qevercloud::AuthenticationResult::expiration = 0`

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.

#### 7.5.3.4 localData

`EverCloudLocalData qevercloud::AuthenticationResult::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.5.3.5 noteStoreUrl

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

DEPRECATED - Client applications should use `urls.noteStoreUrl`.

#### 7.5.3.6 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.

### 7.5.3.7 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.

### 7.5.3.8 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`.

### 7.5.3.9 urls

```
Optional< UserUrls > qevercloud::AuthenticationResult::urls
```

This structure will contain all of the URLs that clients need to make requests to the Evernote service on behalf of the authenticated [User](#).

### 7.5.3.10 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.

### 7.5.3.11 webApiUrlPrefix

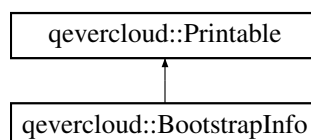
```
Optional< QString > qevercloud::AuthenticationResult::webApiUrlPrefix
```

DEPRECATED - Client applications should use `urls.webApiUrlPrefix`.

## 7.6 qevercloud::BootstrapInfo Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::BootstrapInfo`:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [BootstrapInfo](#) &other) const
- bool [operator!=](#) (const [BootstrapInfo](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- QList< [BootstrapProfile](#) > [profiles](#)

### 7.6.1 Detailed Description

This structure describes a collection of bootstrap profiles.

### 7.6.2 Member Function Documentation

#### 7.6.2.1 [operator!=\(\)](#)

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

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

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

#### 7.6.2.3 [print\(\)](#)

```
virtual void qevercloud::BootstrapInfo::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.6.3 Member Data Documentation

### 7.6.3.1 localData

`EverCloudLocalData` `qevercloud::BootstrapInfo::localData`

See the declaration of `EverCloudLocalData` for details

### 7.6.3.2 profiles

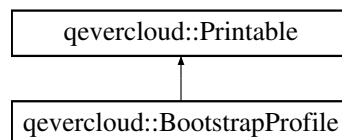
`QList< BootstrapProfile >` `qevercloud::BootstrapInfo::profiles`

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

## 7.7 qevercloud::BootstrapProfile Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::BootstrapProfile`:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `BootstrapProfile` &other) const
- bool `operator!=` (const `BootstrapProfile` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- `QString` `name`
- `BootstrapSettings` `settings`

### 7.7.1 Detailed Description

This structure describes a collection of bootstrap settings.

### 7.7.2 Member Function Documentation



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

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

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

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

#### 7.7.2.3 print()

```
virtual void qevercloud::BootstrapProfile::print (  
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.7.3 Member Data Documentation

#### 7.7.3.1 localData

[EverCloudLocalData](#) qevercloud::BootstrapProfile::localData

See the declaration of [EverCloudLocalData](#) for details

#### 7.7.3.2 name

QString qevercloud::BootstrapProfile::name

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

#### 7.7.3.3 settings

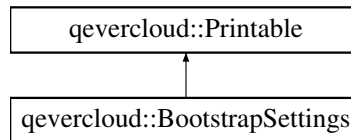
[BootstrapSettings](#) qevercloud::BootstrapProfile::settings

The settings for this profile.

## 7.8 qevercloud::BootstrapSettings Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::BootstrapSettings:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [BootstrapSettings](#) &other) const
- bool [operator!=](#) (const [BootstrapSettings](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- 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](#)
- [Optional](#)< bool > [enableGoogle](#)

### 7.8.1 Detailed Description

This structure describes a collection of bootstrap settings.

### 7.8.2 Member Function Documentation

#### 7.8.2.1 [operator!=\(\)](#)

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

### 7.8.2.2 operator==( )

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

### 7.8.2.3 print()

```
virtual void qevercloud::BootstrapSettings::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.8.3 Member Data Documentation

### 7.8.3.1 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.

### 7.8.3.2 enableFacebookSharing

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

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

### 7.8.3.3 enableGiftSubscriptions

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

Whether the client application should enable gift subscriptions.

### 7.8.3.4 enableGoogle

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

Whether the client application should enable authentication with Google, for example to allow integration with a user's Gmail contacts.

#### 7.8.3.5 enableLinkedInSharing

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

NOT DOCUMENTED

#### 7.8.3.6 enablePublicNotebooks

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

NOT DOCUMENTED

#### 7.8.3.7 enableSharedNotebooks

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

Whether the client application should enable shared notebooks.

#### 7.8.3.8 enableSingleNoteSharing

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

Whether the client application should enable single note sharing.

#### 7.8.3.9 enableSponsoredAccounts

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

Whether the client application should enable sponsored accounts.

#### 7.8.3.10 enableSupportTickets

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

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

#### 7.8.3.11 enableTwitterSharing

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

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

#### 7.8.3.12 localData

`EverCloudLocalData qevercloud::BootstrapSettings::localData`

See the declaration of [EverCloudLocalData](#) for details

## 7.8.3.13 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`.

## 7.8.3.14 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"`.

## 7.8.3.15 supportUrl

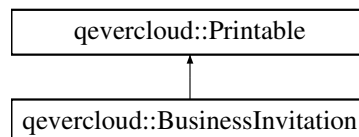
```
QString qevercloud::BootstrapSettings::supportUrl
```

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

## 7.9 qevercloud::BusinessInvitation Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::BusinessInvitation`:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [BusinessInvitation](#) &other) const
- bool [operator!=](#) (const [BusinessInvitation](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [qint32](#) > [businessId](#)
- [Optional](#)< [QString](#) > [email](#)
- [Optional](#)< [BusinessUserRole](#) > [role](#)
- [Optional](#)< [BusinessInvitationStatus](#) > [status](#)
- [Optional](#)< [UserID](#) > [requesterId](#)
- [Optional](#)< [bool](#) > [fromWorkChat](#)
- [Optional](#)< [Timestamp](#) > [created](#)
- [Optional](#)< [Timestamp](#) > [mostRecentReminder](#)

### 7.9.1 Detailed Description

A structure describing an invitation to join a business account.

### 7.9.2 Member Function Documentation

#### 7.9.2.1 `operator!=( )`

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

#### 7.9.2.2 `operator==( )`

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

#### 7.9.2.3 `print()`

```
virtual void qevercloud::BusinessInvitation::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.9.3 Member Data Documentation

#### 7.9.3.1 `businessId`

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

The ID of the business to which the invitation grants access.

#### 7.9.3.2 `created`

```
Optional< Timestamp > qevercloud::BusinessInvitation::created
```

The timestamp at which this invitation was created.

#### 7.9.3.3 email

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

The email address that was invited to join the business.

#### 7.9.3.4 fromWorkChat

```
Optional< bool > qevercloud::BusinessInvitation::fromWorkChat
```

If this invitation was created implicitly via a WorkChat, this field will be true.

#### 7.9.3.5 localData

```
EverCloudLocalData qevercloud::BusinessInvitation::localData
```

See the declaration of [EverCloudLocalData](#) for details

#### 7.9.3.6 mostRecentReminder

```
Optional< Timestamp > qevercloud::BusinessInvitation::mostRecentReminder
```

The timestamp at which the most recent reminder was sent.

#### 7.9.3.7 requesterId

```
Optional< UserID > qevercloud::BusinessInvitation::requesterId
```

For invitations that were initially requested by a non-admin member of the business, this field specifies the user ID of the requestor. For all other invitations, this field will be unset.

#### 7.9.3.8 role

```
Optional< BusinessUserRole > qevercloud::BusinessInvitation::role
```

The role to grant the user after the invitation is accepted.

#### 7.9.3.9 status

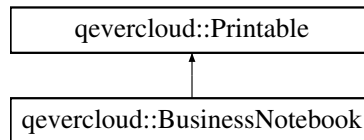
```
Optional< BusinessInvitationStatus > qevercloud::BusinessInvitation::status
```

The status of the invitation.

## 7.10 qevercloud::BusinessNotebook Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::BusinessNotebook:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [BusinessNotebook](#) &other) const
- bool [operator!=](#) (const [BusinessNotebook](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< QString > [notebookDescription](#)
- [Optional](#)< [SharedNotebookPrivilegeLevel](#) > [privilege](#)
- [Optional](#)< bool > [recommended](#)

#### 7.10.1 Detailed Description

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

#### 7.10.2 Member Function Documentation

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

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

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

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



## 7.10.2.3 print()

```
virtual void qevercloud::BusinessNotebook::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.10.3 Member Data Documentation

## 7.10.3.1 localData

[EverCloudLocalData](#) qevercloud::BusinessNotebook::localData

See the declaration of [EverCloudLocalData](#) for details

## 7.10.3.2 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

## 7.10.3.3 privilege

[Optional](#)< [SharedNotebookPrivilegeLevel](#) > qevercloud::BusinessNotebook::privilege

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

## 7.10.3.4 recommended

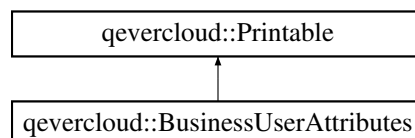
[Optional](#)< bool > qevercloud::BusinessNotebook::recommended

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

## 7.11 qevercloud::BusinessUserAttributes Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::BusinessUserAttributes:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [BusinessUserAttributes](#) &other) const
- bool [operator!=](#) (const [BusinessUserAttributes](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QString](#) > [title](#)
- [Optional](#)< [QString](#) > [location](#)
- [Optional](#)< [QString](#) > [department](#)
- [Optional](#)< [QString](#) > [mobilePhone](#)
- [Optional](#)< [QString](#) > [linkedInProfileUrl](#)
- [Optional](#)< [QString](#) > [workPhone](#)
- [Optional](#)< [Timestamp](#) > [companyStartDate](#)

### 7.11.1 Detailed Description

A structure holding the optional attributes associated with users in a business.

### 7.11.2 Member Function Documentation

#### 7.11.2.1 [operator!=\(\)](#)

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

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

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

#### 7.11.2.3 [print\(\)](#)

```
virtual void qevercloud::BusinessUserAttributes::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.11.3 Member Data Documentation

#### 7.11.3.1 companyStartDate

`Optional< Timestamp > qevercloud::BusinessUserAttributes::companyStartDate`

The date on which the user started working at their company.

#### 7.11.3.2 department

`Optional< QString > qevercloud::BusinessUserAttributes::department`

Free form text of the department this user belongs to.

#### 7.11.3.3 linkedInProfileUrl

`Optional< QString > qevercloud::BusinessUserAttributes::linkedInProfileUrl`

URL to user's public LinkedIn profile page. This should only contain the portion relative to the base LinkedIn URL. For example: "/pub/john-smith/".

#### 7.11.3.4 localData

`EverCloudLocalData qevercloud::BusinessUserAttributes::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.11.3.5 location

`Optional< QString > qevercloud::BusinessUserAttributes::location`

City, State (for US) or City / Province for other countries

#### 7.11.3.6 mobilePhone

`Optional< QString > qevercloud::BusinessUserAttributes::mobilePhone`

User's mobile phone number. Stored as plain text without any formatting.

#### 7.11.3.7 title

`Optional< QString > qevercloud::BusinessUserAttributes::title`

Free form text of this user's title in the business

### 7.11.3.8 workPhone

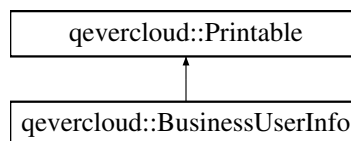
`Optional< QString > qevercloud::BusinessUserAttributes::workPhone`

User's work phone number. Stored as plain text without any formatting.

## 7.12 qevercloud::BusinessUserInfo Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::BusinessUserInfo:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `BusinessUserInfo` &other) const
- bool `operator!=` (const `BusinessUserInfo` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< qint32 >` `businessId`
- `Optional< QString >` `businessName`
- `Optional< BusinessUserRole >` `role`
- `Optional< QString >` `email`
- `Optional< Timestamp >` `updated`

### 7.12.1 Detailed Description

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

### 7.12.2 Member Function Documentation

#### 7.12.2.1 `operator!=()`

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

### 7.12.2.2 operator==( )

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

### 7.12.2.3 print( )

```
virtual void qevercloud::BusinessUserInfo::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.12.3 Member Data Documentation

### 7.12.3.1 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.

### 7.12.3.2 businessName

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

NOT DOCUMENTED

### 7.12.3.3 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.

### 7.12.3.4 localData

[EverCloudLocalData](#) qevercloud::BusinessUserInfo::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.12.3.5 role

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

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

### 7.12.3.6 updated

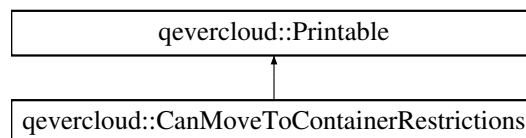
```
Optional< Timestamp > qevercloud::BusinessUserInfo::updated
```

Last time the business user or business user attributes were updated.

## 7.13 qevercloud::CanMoveToContainerRestrictions Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::CanMoveToContainerRestrictions:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [CanMoveToContainerRestrictions](#) &other) const
- bool [operator!=](#) (const [CanMoveToContainerRestrictions](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional< CanMoveToContainerStatus >](#) [canMoveToContainer](#)

### 7.13.1 Detailed Description

Specifies if the client can move a [Notebook](#) to a Workspace.

### 7.13.2 Member Function Documentation

## 7.13.2.1 operator!=(())

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

## 7.13.2.2 operator==(())

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

## 7.13.2.3 print()

```
virtual void qevercloud::CanMoveToContainerRestrictions::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.13.3 Member Data Documentation

## 7.13.3.1 canMoveToContainer

```
Optional< CanMoveToContainerStatus > qevercloud::CanMoveToContainerRestrictions::canMoveTo↵
Container
```

NOT DOCUMENTED

## 7.13.3.2 localData

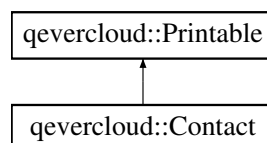
```
EverCloudLocalData qevercloud::CanMoveToContainerRestrictions::localData
```

See the declaration of [EverCloudLocalData](#) for details

## 7.14 qevercloud::Contact Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::Contact:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [Contact](#) &other) const
- bool [operator!=](#) (const [Contact](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QString](#) > [name](#)
- [Optional](#)< [QString](#) > [id](#)
- [Optional](#)< [ContactType](#) > [type](#)
- [Optional](#)< [QString](#) > [photoUrl](#)
- [Optional](#)< [Timestamp](#) > [photoLastUpdated](#)
- [Optional](#)< [QByteArray](#) > [messagingPermit](#)
- [Optional](#)< [Timestamp](#) > [messagingPermitExpires](#)

### 7.14.1 Detailed Description

A structure that represents contact information. [Note](#) this does not necessarily correspond to an Evernote user.

### 7.14.2 Member Function Documentation

#### 7.14.2.1 [operator!=\(\)](#)

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

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

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

#### 7.14.2.3 [print\(\)](#)

```
virtual void qevercloud::Contact::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).



### 7.14.3 Member Data Documentation

#### 7.14.3.1 id

`Optional< QString > qevercloud::Contact::id`

A unique identifier for this ContactType.

#### 7.14.3.2 localData

`EverCloudLocalData qevercloud::Contact::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.14.3.3 messagingPermit

`Optional< QByteArray > qevercloud::Contact::messagingPermit`

This field will only be filled by the service when it is giving a [Contact](#) record to a client, and that client does not normally have enough permission to send a new message to the person represented through this [Contact](#). In that case, this whole [Contact](#) record could be used to send a new Message to the [Contact](#), and the service will inspect this permit to confirm that operation was allowed.

#### 7.14.3.4 messagingPermitExpires

`Optional< Timestamp > qevercloud::Contact::messagingPermitExpires`

If this field is set, then this (whole) [Contact](#) record may be used in calls to `sendMessage` until this time. After that time, those calls may be rejected by the service if the caller does not have direct permission to initiate a message with the represented Evernote user.

#### 7.14.3.5 name

`Optional< QString > qevercloud::Contact::name`

The displayable name of this contact. This field is filled in by the service and is read-only to clients.

#### 7.14.3.6 photoLastUpdated

`Optional< Timestamp > qevercloud::Contact::photoLastUpdated`

timestamp when the profile photo at 'photoUrl' was last updated. This field will be null if the user has never set a profile photo. This field is filled in by the service and is read-only to clients.

### 7.14.3.7 photoUrl

```
Optional< QString > qevercloud::Contact::photoUrl
```

A URL of a profile photo representing this [Contact](#). This field is filled in by the service and is read-only to clients.

### 7.14.3.8 type

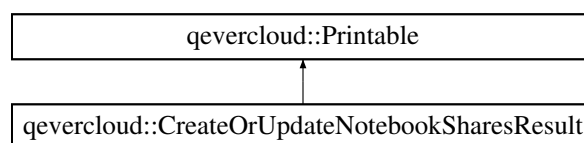
```
Optional< ContactType > qevercloud::Contact::type
```

What service does this contact come from?

## 7.15 qevercloud::CreateOrUpdateNotebookSharesResult Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::CreateOrUpdateNotebookSharesResult:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [CreateOrUpdateNotebookSharesResult](#) &other) const
- bool [operator!=](#) (const [CreateOrUpdateNotebookSharesResult](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< qint32 > [updateSequenceNum](#)
- [Optional](#)< QList< [SharedNotebook](#) > > [matchingShares](#)

### Properties

- [Optional](#) QList

### 7.15.1 Detailed Description

A structure containing the results of a call to `createOrUpdateNotebookShares`.

## 7.15.2 Member Function Documentation

### 7.15.2.1 operator!=(())

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

### 7.15.2.2 operator==(())

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

### 7.15.2.3 print()

```
virtual void qevercloud::CreateOrUpdateNotebookSharesResult::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.15.3 Member Data Documentation

### 7.15.3.1 localData

[EverCloudLocalData](#) qevercloud::CreateOrUpdateNotebookSharesResult::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.15.3.2 matchingShares

[Optional](#)<[QList](#)<[SharedNotebook](#)> > qevercloud::CreateOrUpdateNotebookSharesResult::matching↔  
Shares

A list of [SharedNotebook](#) records that match the desired recipients. These records may have been either created or updated by the call to createOrUpdateNotebookShares, or they may have been at the desired privilege privilege level prior to the call.

### 7.15.3.3 updateSequenceNum

`Optional< qint32 > qevercloud::CreateOrUpdateNotebookSharesResult::updateSequenceNum`

The USN of the notebook after the call.

## 7.15.4 Property Documentation

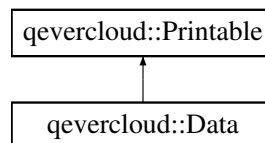
### 7.15.4.1 QList

`Optional qevercloud::CreateOrUpdateNotebookSharesResult::QList`

## 7.16 qevercloud::Data Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::Data`:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `Data` &other) const
- bool `operator!=` (const `Data` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< QByteArray >` `bodyHash`
- `Optional< qint32 >` `size`
- `Optional< QByteArray >` `body`

### 7.16.1 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.

## 7.16.2 Member Function Documentation

### 7.16.2.1 operator!=(())

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

### 7.16.2.2 operator==(())

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

### 7.16.2.3 print()

```
virtual void qevercloud::Data::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.16.3 Member Data Documentation

### 7.16.3.1 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.

### 7.16.3.2 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)

### 7.16.3.3 localData

[EverCloudLocalData](#) `qevercloud::Data::localData`

See the declaration of [EverCloudLocalData](#) for details

### 7.16.3.4 size

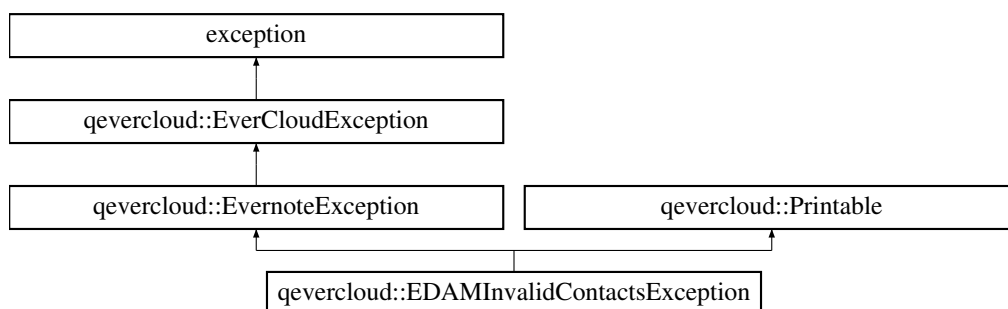
[Optional](#)< `qint32` > `qevercloud::Data::size`

The length, in bytes, of the data body.

## 7.17 qevercloud::EDAMInvalidContactsException Class Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::EDAMInvalidContactsException`:



### Public Member Functions

- [EDAMInvalidContactsException](#) ()
- virtual [~EDAMInvalidContactsException](#) () noexcept override
- [EDAMInvalidContactsException](#) (const [EDAMInvalidContactsException](#) &other)
- const char \* [what](#) () const noexcept override
- virtual [EverCloudExceptionDataPtr](#) [exceptionData](#) () const override
- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [EDAMInvalidContactsException](#) &other) const
- bool [operator!=](#) (const [EDAMInvalidContactsException](#) &other) const

### Public Attributes

- [QList](#)< [Contact](#) > [contacts](#)
- [Optional](#)< [QString](#) > [parameter](#)
- [Optional](#)< [QList](#)< [EDAMInvalidContactReason](#) > > [reasons](#)

### Properties

- [Optional](#) [QList](#)

## Additional Inherited Members

### 7.17.1 Detailed Description

An exception thrown when the provided Contacts fail validation. For instance, email domains could be invalid, phone numbers might not be valid for SMS, etc.

We will not provide individual reasons for each [Contact](#)'s validation failure. The presence of the [Contact](#) in this exception means that the user must figure out how to take appropriate action to fix this [Contact](#).

**contacts** The list of Contacts that are considered invalid by the service

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

**reasons** If supplied, the list of reasons why the server considered a contact invalid, matching, in order, the list returned in the contacts field.

### 7.17.2 Constructor & Destructor Documentation

#### 7.17.2.1 EDAMInvalidContactsException() [1/2]

```
qevercloud::EDAMInvalidContactsException::EDAMInvalidContactsException ( )
```

#### 7.17.2.2 ~EDAMInvalidContactsException()

```
virtual qevercloud::EDAMInvalidContactsException::~~EDAMInvalidContactsException ( ) [override],  
[virtual], [noexcept]
```

#### 7.17.2.3 EDAMInvalidContactsException() [2/2]

```
qevercloud::EDAMInvalidContactsException::EDAMInvalidContactsException (   
    const EDAMInvalidContactsException & other )
```

### 7.17.3 Member Function Documentation

#### 7.17.3.1 exceptionData()

```
virtual EverCloudExceptionDataPtr qevercloud::EDAMInvalidContactsException::exceptionData ( )  
const [override], [virtual]
```

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

#### 7.17.3.2 operator!=(())

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

#### 7.17.3.3 operator==(())

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

#### 7.17.3.4 print()

```
virtual void qevercloud::EDAMInvalidContactsException::print (   
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

#### 7.17.3.5 what()

```
const char* qevercloud::EDAMInvalidContactsException::what ( ) const [override], [virtual],  
[noexcept]
```

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

### 7.17.4 Member Data Documentation

#### 7.17.4.1 contacts

```
QList< Contact > qevercloud::EDAMInvalidContactsException::contacts
```



## 7.17.4.2 parameter

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

## 7.17.4.3 reasons

`Optional< QList< EDAMInvalidContactReason > > qevercloud::EDAMInvalidContactsException::reasons`

## 7.17.5 Property Documentation

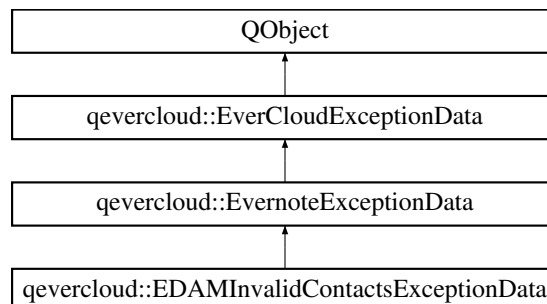
## 7.17.5.1 QList

`Optional qevercloud::EDAMInvalidContactsException::QList`

## 7.18 qevercloud::EDAMInvalidContactsExceptionData Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::EDAMInvalidContactsExceptionData:



## Public Member Functions

- `EDAMInvalidContactsExceptionData` (`QList< Contact > contacts`, `Optional< QString > parameter`, `Optional< QList< EDAMInvalidContactReason > > reasons`)
- virtual void `throwException` () const override

## Protected Attributes

- `QList< Contact > m_contacts`
- `Optional< QString > m_parameter`
- `Optional< QList< EDAMInvalidContactReason > > m_reasons`

## Additional Inherited Members

### 7.18.1 Detailed Description

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

### 7.18.2 Constructor & Destructor Documentation

#### 7.18.2.1 EDAMInvalidContactsExceptionData()

```
qevercloud::EDAMInvalidContactsExceptionData::EDAMInvalidContactsExceptionData (
    QList< Contact > contacts,
    Optional< QString > parameter,
    Optional< QList< EDAMInvalidContactReason > > reasons ) [explicit]
```

### 7.18.3 Member Function Documentation

#### 7.18.3.1 throwException()

```
virtual void qevercloud::EDAMInvalidContactsExceptionData::throwException ( ) const [override],
[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::EvernoteExceptionData](#).

### 7.18.4 Member Data Documentation

#### 7.18.4.1 m\_contacts

```
QList<Contact> qevercloud::EDAMInvalidContactsExceptionData::m_contacts [protected]
```

#### 7.18.4.2 m\_parameter

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

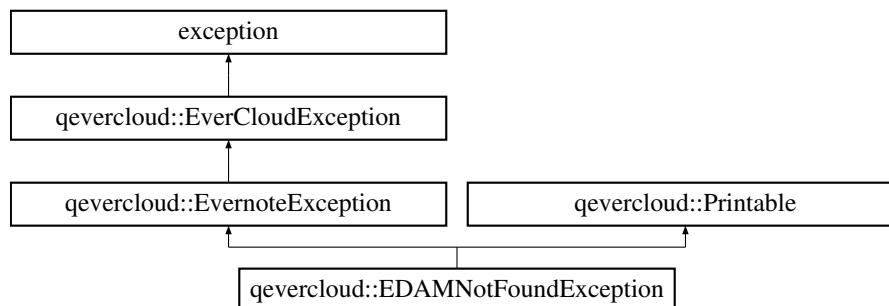
## 7.18.4.3 m\_reasons

```
Optional<QList<EDAMInvalidContactReason> > qevercloud::EDAMInvalidContactsExceptionData::m_↔
reasons [protected]
```

## 7.19 qevercloud::EDAMNotFoundException Class Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::EDAMNotFoundException:



## Public Member Functions

- [EDAMNotFoundException](#) ()
- virtual [~EDAMNotFoundException](#) () noexcept override
- [EDAMNotFoundException](#) (const [EDAMNotFoundException](#) &other)
- const char \* [what](#) () const noexcept override
- virtual [EverCloudExceptionDataPtr exceptionData](#) () const override
- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [EDAMNotFoundException](#) &other) const
- bool [operator!=](#) (const [EDAMNotFoundException](#) &other) const

## Public Attributes

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

## Additional Inherited Members

## 7.19.1 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.

## 7.19.2 Constructor & Destructor Documentation

### 7.19.2.1 EDAMNotFoundException() [1/2]

```
qevercloud::EDAMNotFoundException::EDAMNotFoundException ( )
```

### 7.19.2.2 ~EDAMNotFoundException()

```
virtual qevercloud::EDAMNotFoundException::~~EDAMNotFoundException ( ) [override], [virtual],  
[noexcept]
```

### 7.19.2.3 EDAMNotFoundException() [2/2]

```
qevercloud::EDAMNotFoundException::EDAMNotFoundException (   
    const EDAMNotFoundException & other )
```

## 7.19.3 Member Function Documentation

### 7.19.3.1 exceptionData()

```
virtual EverCloudExceptionDataPtr qevercloud::EDAMNotFoundException::exceptionData ( ) const  
[override], [virtual]
```

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

### 7.19.3.2 operator!=( )

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

## 7.19.3.3 operator==( )

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

## 7.19.3.4 print()

```
virtual void qevercloud::EDAMNotFoundException::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.19.3.5 what()

```
const char* qevercloud::EDAMNotFoundException::what ( ) const [override], [virtual], [noexcept]
```

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

## 7.19.4 Member Data Documentation

## 7.19.4.1 identifier

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

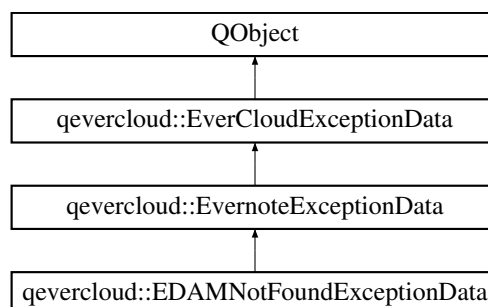
## 7.19.4.2 key

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

## 7.20 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 override

## Protected Attributes

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

## Additional Inherited Members

### 7.20.1 Detailed Description

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

### 7.20.2 Constructor & Destructor Documentation

#### 7.20.2.1 [EDAMNotFoundExceptionData](#)()

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

### 7.20.3 Member Function Documentation

#### 7.20.3.1 [throwException](#)()

```
virtual void qevercloud::EDAMNotFoundExceptionData::throwException ( ) const [override],
[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::EvernoteExceptionData](#).

### 7.20.4 Member Data Documentation

## 7.20.4.1 m\_identifier

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

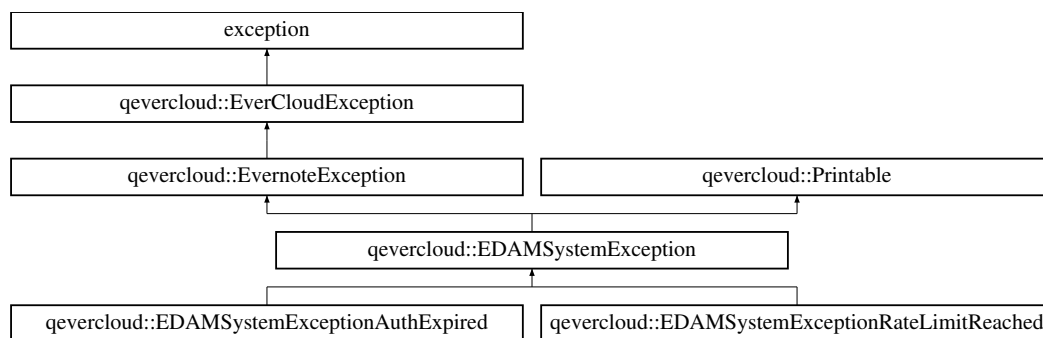
## 7.20.4.2 m\_key

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

## 7.21 qevercloud::EDAMSystemException Class Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::EDAMSystemException:



## Public Member Functions

- [EDAMSystemException](#) ()
- virtual [~EDAMSystemException](#) () noexcept override
- [EDAMSystemException](#) (const [EDAMSystemException](#) &other)
- const char \* [what](#) () const noexcept override
- virtual [EverCloudExceptionDataPtr exceptionData](#) () const override
- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [EDAMSystemException](#) &other) const
- bool [operator!=](#) (const [EDAMSystemException](#) &other) const

## Public Attributes

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

## 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 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 `EDAMError↵ Code`.

**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`,

### 7.21.2 Constructor & Destructor Documentation

#### 7.21.2.1 `EDAMSystemException()` [1/2]

```
qevercloud::EDAMSystemException::EDAMSystemException ( )
```

#### 7.21.2.2 `~EDAMSystemException()`

```
virtual qevercloud::EDAMSystemException::~~EDAMSystemException ( ) [override], [virtual],  
[noexcept]
```

#### 7.21.2.3 `EDAMSystemException()` [2/2]

```
qevercloud::EDAMSystemException::EDAMSystemException (   
    const EDAMSystemException & other )
```

### 7.21.3 Member Function Documentation



### 7.21.3.1 exceptionData()

```
virtual EverCloudExceptionDataPtr qevercloud::EDAMSystemException::exceptionData ( ) const  
[override], [virtual]
```

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

Reimplemented in [qevercloud::EDAMSystemExceptionAuthExpired](#), and [qevercloud::EDAMSystemException←  
RateLimitReached](#).

### 7.21.3.2 operator!=(())

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

### 7.21.3.3 operator==(())

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

### 7.21.3.4 print()

```
virtual void qevercloud::EDAMSystemException::print (   
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.21.3.5 what()

```
const char* qevercloud::EDAMSystemException::what ( ) const [override], [virtual], [noexcept]
```

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

## 7.21.4 Member Data Documentation

#### 7.21.4.1 errorCode

`EDAMErrorCode qevercloud::EDAMSystemException::errorCode`

#### 7.21.4.2 message

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

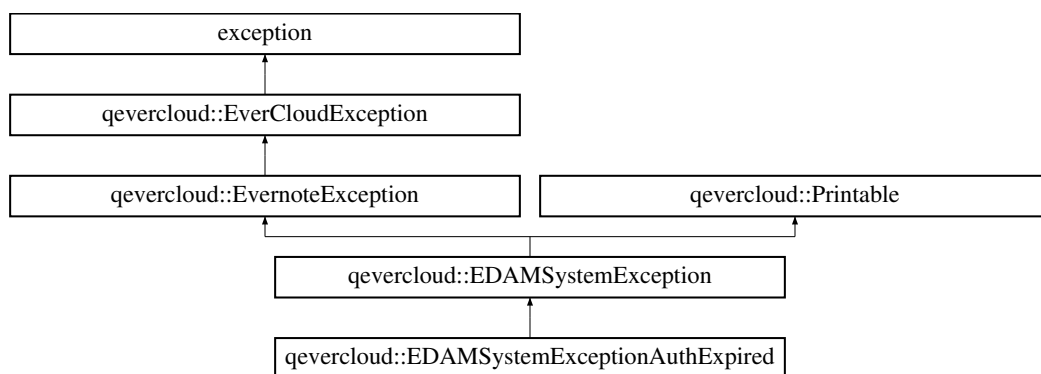
#### 7.21.4.3 rateLimitDuration

`Optional< qint32 > qevercloud::EDAMSystemException::rateLimitDuration`

## 7.22 qevercloud::EDAMSystemExceptionAuthExpired Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionAuthExpired:



### Public Member Functions

- virtual `EverCloudExceptionDataPtr exceptionData ()` const override

### Additional Inherited Members

#### 7.22.1 Detailed Description

`EDAMSystemException` for `errorCode = AUTH_EXPIRED`

## 7.22.2 Member Function Documentation

### 7.22.2.1 exceptionData()

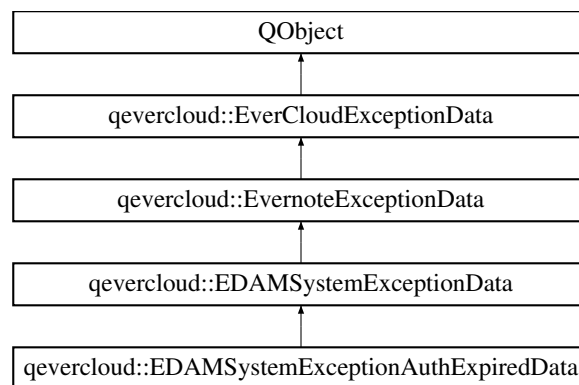
```
virtual EverCloudExceptionDataPtr qevercloud::EDAMSystemExceptionAuthExpired::exceptionData (
) const \[override\], \[virtual\]
```

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

## 7.23 qevercloud::EDAMSystemExceptionAuthExpiredData Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionAuthExpiredData:



### Public Member Functions

- [EDAMSystemExceptionAuthExpiredData](#) (QString error, [EDAMErrorCode](#) errorCode, [Optional](#)< QString > message, [Optional](#)< qint32 > rateLimitDuration)
- virtual void [throwException](#) () const override

### Additional Inherited Members

### 7.23.1 Detailed Description

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

### 7.23.2 Constructor & Destructor Documentation

### 7.23.2.1 EDAMSystemExceptionAuthExpiredData()

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

## 7.23.3 Member Function Documentation

### 7.23.3.1 throwException()

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

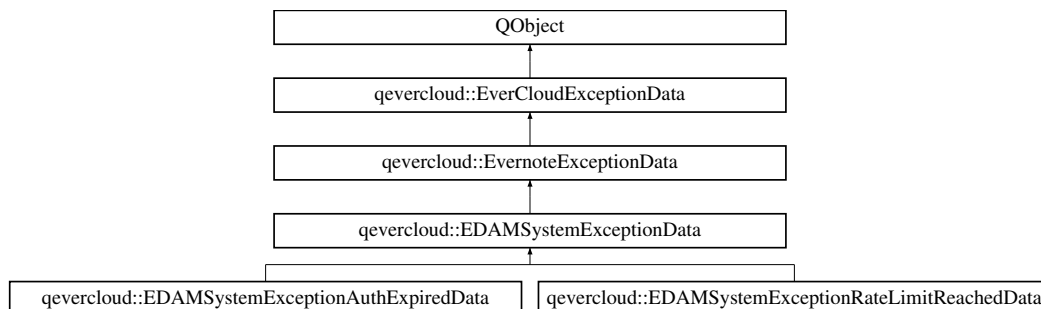
If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant than call this function. Do not use `throw` statement, it's not polymorphic.

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

## 7.24 qevercloud::EDAMSystemExceptionData Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for `qevercloud::EDAMSystemExceptionData`:



### Public Member Functions

- [EDAMSystemExceptionData](#) (QString err, [EDAMErrorCode](#) errorCode, [Optional](#)< QString > message, [Optional](#)< qint32 > rateLimitDuration)
- virtual void [throwException](#) ( ) const override

### Protected Attributes

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

## Additional Inherited Members

### 7.24.1 Detailed Description

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

### 7.24.2 Constructor & Destructor Documentation

#### 7.24.2.1 EDAMSystemExceptionData()

```
qevercloud::EDAMSystemExceptionData::EDAMSystemExceptionData (
    QString err,
    EDAMErrorCode errorCode,
    Optional< QString > message,
    Optional< quint32 > rateLimitDuration ) [explicit]
```

### 7.24.3 Member Function Documentation

#### 7.24.3.1 throwException()

```
virtual void qevercloud::EDAMSystemExceptionData::throwException ( ) const [override], [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::EvernoteExceptionData](#).

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

### 7.24.4 Member Data Documentation

#### 7.24.4.1 m\_errorCode

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

#### 7.24.4.2 m\_message

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

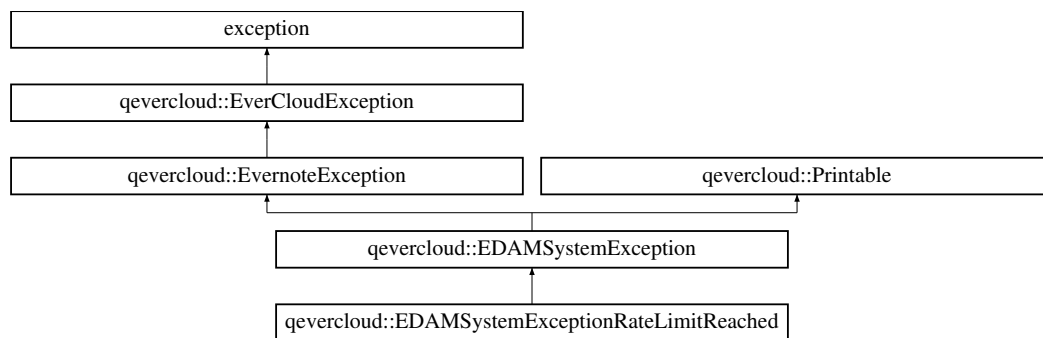
#### 7.24.4.3 m\_rateLimitDuration

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

## 7.25 qevercloud::EDAMSystemExceptionRateLimitReached Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionRateLimitReached:



### Public Member Functions

- virtual `EverCloudExceptionDataPtr exceptionData ()` const override

### Additional Inherited Members

#### 7.25.1 Detailed Description

`EDAMSystemException` for `errorCode = RATE_LIMIT_REACHED`

#### 7.25.2 Member Function Documentation

##### 7.25.2.1 exceptionData()

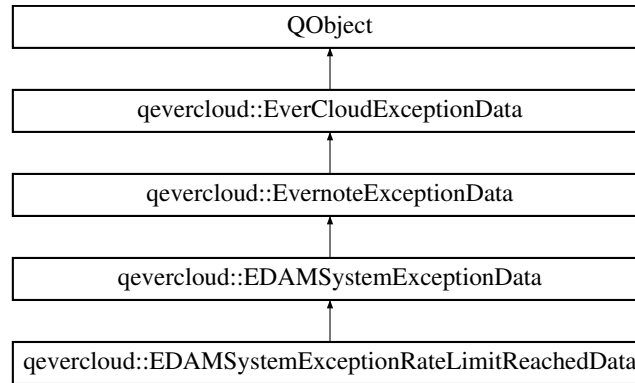
virtual `EverCloudExceptionDataPtr qevercloud::EDAMSystemExceptionRateLimitReached::exceptionData ()` const [override], [virtual]

Reimplemented from `qevercloud::EDAMSystemException`.

## 7.26 qevercloud::EDAMSystemExceptionRateLimitReachedData Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::EDAMSystemExceptionRateLimitReachedData:



### Public Member Functions

- [EDAMSystemExceptionRateLimitReachedData](#) (QString error, [EDAMErrorCode](#) errorCode, [Optional](#)< QString > message, [Optional](#)< qint32 > rateLimitDuration)
- virtual void [throwException](#) () const override

### Additional Inherited Members

#### 7.26.1 Detailed Description

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

#### 7.26.2 Constructor & Destructor Documentation

##### 7.26.2.1 EDAMSystemExceptionRateLimitReachedData()

```

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

#### 7.26.3 Member Function Documentation

### 7.26.3.1 `throwException()`

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

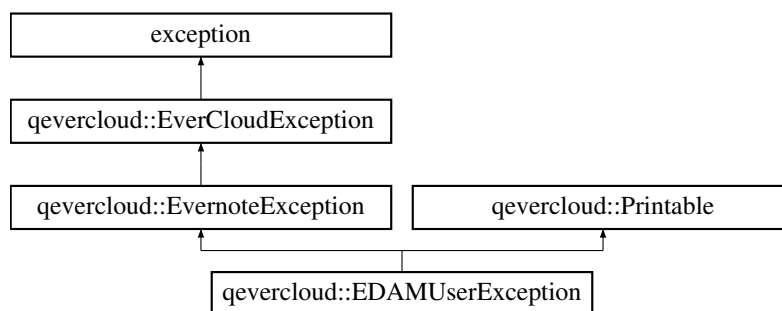
If you want to throw an exception that corresponds to a received [EverCloudExceptionData](#) descendant then call this function. Do not use `throw` statement, it's not polymorphic.

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

## 7.27 `qevercloud::EDAMUserException` Class Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::EDAMUserException`:



### Public Member Functions

- [EDAMUserException](#) ()
- virtual [~EDAMUserException](#) () noexcept override
- [EDAMUserException](#) (const [EDAMUserException](#) &other)
- const char \* [what](#) () const noexcept override
- virtual [EverCloudExceptionDataPtr exceptionData](#) () const override
- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [EDAMUserException](#) &other) const
- bool [operator!=](#) (const [EDAMUserException](#) &other) const

### Public Attributes

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



## Additional Inherited Members

### 7.27.1 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 EDAMError↵ Code.

**parameter:** If the error applied to a particular input parameter, this will indicate which parameter. For some errors (USER\_NOT\_ASSOCIATED, USER\_NOT\_REGISTERED, SSO\_AUTHENTICATION\_REQUIRED), this is the user's email.

### 7.27.2 Constructor & Destructor Documentation

#### 7.27.2.1 EDAMUserException() [1/2]

```
qevercloud::EDAMUserException::EDAMUserException ( )
```

#### 7.27.2.2 ~EDAMUserException()

```
virtual qevercloud::EDAMUserException::~~EDAMUserException ( ) [override], [virtual], [noexcept]
```

#### 7.27.2.3 EDAMUserException() [2/2]

```
qevercloud::EDAMUserException::EDAMUserException (
    const EDAMUserException & other )
```

### 7.27.3 Member Function Documentation

### 7.27.3.1 exceptionData()

```
virtual EverCloudExceptionDataPtr qevercloud::EDAMUserException::exceptionData ( ) const [override],  
[virtual]
```

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

### 7.27.3.2 operator!=(())

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

### 7.27.3.3 operator==(())

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

### 7.27.3.4 print()

```
virtual void qevercloud::EDAMUserException::print (   
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.27.3.5 what()

```
const char* qevercloud::EDAMUserException::what ( ) const [override], [virtual], [noexcept]
```

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

## 7.27.4 Member Data Documentation

### 7.27.4.1 errorCode

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

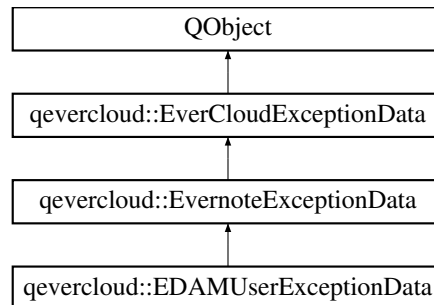
## 7.27.4.2 parameter

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

## 7.28 qevercloud::EDAMUserExceptionData Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::EDAMUserExceptionData:



## Public Member Functions

- [EDAMUserExceptionData](#) (QString error, [EDAMErrorCode](#) errorCode, [Optional< QString >](#) parameter)
- virtual void [throwException](#) () const override

## Protected Attributes

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

## Additional Inherited Members

## 7.28.1 Detailed Description

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

## 7.28.2 Constructor &amp; Destructor Documentation

## 7.28.2.1 EDAMUserExceptionData()

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

## 7.28.3 Member Function Documentation

### 7.28.3.1 `throwException()`

```
virtual void qevercloud::EDAMUserExceptionData::throwException ( ) const [override], [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::EvernoteExceptionData](#).

## 7.28.4 Member Data Documentation

### 7.28.4.1 `m_errorCode`

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

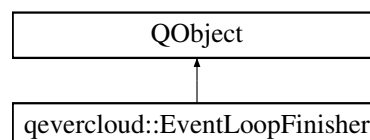
### 7.28.4.2 `m_parameter`

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

## 7.29 `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](#) ()

## 7.29.1 Constructor & Destructor Documentation

### 7.29.1.1 EventLoopFinisher()

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

### 7.29.1.2 ~EventLoopFinisher()

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

## 7.29.2 Member Function Documentation

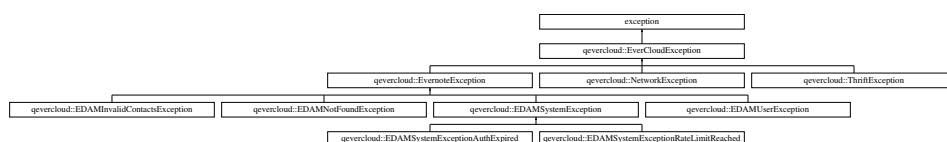
### 7.29.2.1 stopEventLoop

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

## 7.30 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)
- virtual [~EverCloudException](#) () noexcept override
- virtual const char \* [what](#) () const noexcept override
- virtual std::shared\_ptr< [EverCloudExceptionData](#) > [exceptionData](#) () const

## Protected Attributes

- QByteArray [m\\_error](#)

### 7.30.1 Detailed Description

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

### 7.30.2 Constructor & Destructor Documentation

#### 7.30.2.1 [EverCloudException\(\)](#) [1/4]

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

#### 7.30.2.2 [EverCloudException\(\)](#) [2/4]

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

#### 7.30.2.3 [EverCloudException\(\)](#) [3/4]

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

#### 7.30.2.4 [EverCloudException\(\)](#) [4/4]

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

## 7.30.2.5 ~EverCloudException()

```
virtual qevercloud::EverCloudException::~~EverCloudException ( ) [override], [virtual], [noexcept]
```

## 7.30.3 Member Function Documentation

## 7.30.3.1 exceptionData()

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

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

## 7.30.3.2 what()

```
virtual const char* qevercloud::EverCloudException::what ( ) const [override], [virtual],  
[noexcept]
```

Reimplemented in [qevercloud::EDAMInvalidContactsException](#), [qevercloud::EDAMNotFoundException](#), [qevercloud::EDAMSystemException](#), [qevercloud::EDAMUserException](#), [qevercloud::ThriftException](#), and [qevercloud::NetworkException](#).

## 7.30.4 Member Data Documentation

## 7.30.4.1 m\_error

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

## 7.31 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](#)

### 7.31.1 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, EverCloudExceptionData error)
    {
        if (!error.isNull())
        {
            auto errorNotFound =
                std::dynamic_pointer_cast<EDAMNotFoundException>(
                    error);

            auto errorUser =
                std::dynamic_pointer_cast<EDAMUserExceptionData>(
                    error);

            auto errorSystem =
                std::dynamic_pointer_cast<EDAMSystemExceptionData>(
                    error);

            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
    }
});

```

## 7.31.2 Constructor & Destructor Documentation

### 7.31.2.1 EverCloudExceptionData()

```

qevercloud::EverCloudExceptionData::EverCloudExceptionData (
    QString error ) [explicit]

```

## 7.31.3 Member Function Documentation

### 7.31.3.1 throwException()

```

virtual void qevercloud::EverCloudExceptionData::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 in [qevercloud::EDAMSystemExceptionAuthExpiredData](#), [qevercloud::EDAMSystemExceptionRateLimitReachedData](#), [qevercloud::EDAMInvalidContactsExceptionData](#), [qevercloud::EDAMNotFoundExceptionData](#), [qevercloud::EvernoteExceptionData](#), [qevercloud::EDAMSystemExceptionData](#), [qevercloud::EDAMUserExceptionData](#), [qevercloud::ThriftExceptionData](#), and [qevercloud::NetworkExceptionData](#).

## 7.31.4 Member Data Documentation

### 7.31.4.1 errorMessage

```

QString qevercloud::EverCloudExceptionData::errorMessage

```

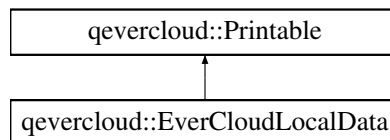
Contains an error message. It's the `std::exception::what()` counterpart.

## 7.32 qevercloud::EverCloudLocalData Class Reference

The [EverCloudLocalData](#) class contains several data elements which are not synchronized with Evernote service but which are nevertheless useful in applications using QEverCloud to implement feature rich full sync Evernote clients. Values of this class' types are contained within QEverCloud types corresponding to actual Evernote API types.

```
#include <Types.h>
```

Inheritance diagram for qevercloud::EverCloudLocalData:



### Public Types

- using [Dict](#) = QHash< QString, QVariant >

### Public Member Functions

- [EverCloudLocalData](#) ()
- virtual [~EverCloudLocalData](#) () noexcept override
- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [EverCloudLocalData](#) &other) const
- bool [operator!=](#) (const [EverCloudLocalData](#) &other) const

### Public Attributes

- QString [id](#)  
*id property can be used as a local unique identifier for any data item before it has been synchronized with Evernote and thus before it can be identified using its guid.*
- bool [dirty](#) = false  
*dirty property can be used to keep track which objects have been modified locally and thus need to be synchronized with Evernote service*
- bool [local](#) = false  
*local property can be used to keep track which data items are meant to be local only and thus never be synchronized with Evernote service*
- bool [favorited](#) = false  
*favorited property can be used to keep track which data items were favorited in the client. Unfortunately, Evernote has never provided a way to synchronize such property between different clients*
- QHash< QString, QVariant > [dict](#)  
*dict can be used for storage of any other auxiliary values associated with objects of QEverCloud types*

### Properties

- [Dict dict](#)

### 7.32.1 Detailed Description

The [EverCloudLocalData](#) class contains several data elements which are not synchronized with Evernote service but which are nevertheless useful in applications using QEverCloud to implement feature rich full sync Evernote clients. Values of this class' types are contained within QEverCloud types corresponding to actual Evernote API types.

### 7.32.2 Member Typedef Documentation

#### 7.32.2.1 Dict

```
using qevercloud::EverCloudLocalData::Dict = QHash<QString, QVariant>
```

### 7.32.3 Constructor & Destructor Documentation

#### 7.32.3.1 EverCloudLocalData()

```
qevercloud::EverCloudLocalData::EverCloudLocalData ( )
```

#### 7.32.3.2 ~EverCloudLocalData()

```
virtual qevercloud::EverCloudLocalData::~~EverCloudLocalData ( ) [override], [virtual], [noexcept]
```

### 7.32.4 Member Function Documentation

#### 7.32.4.1 operator!=( )

```
bool qevercloud::EverCloudLocalData::operator!= (
    const EverCloudLocalData & other ) const
```

#### 7.32.4.2 operator==( )

```
bool qevercloud::EverCloudLocalData::operator== (
    const EverCloudLocalData & other ) const
```

#### 7.32.4.3 print()

```
virtual void qevercloud::EverCloudLocalData::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.32.5 Member Data Documentation

#### 7.32.5.1 dict

```
QHash<QString, QVariant> qevercloud::EverCloudLocalData::dict
```

dict can be used for storage of any other auxiliary values associated with objects of QEverCloud types

#### 7.32.5.2 dirty

```
bool qevercloud::EverCloudLocalData::dirty = false
```

dirty property can be used to keep track which objects have been modified locally and thus need to be synchronized with Evernote service

#### 7.32.5.3 favorited

```
bool qevercloud::EverCloudLocalData::favorited = false
```

favorited property can be used to keep track which data items were favorited in the client. Unfortunately, Evernote has never provided a way to synchronize such property between different clients

## 7.32.5.4 id

```
QString qevercloud::EverCloudLocalData::id
```

id property can be used as a local unique identifier for any data item before it has been synchronized with Evernote and thus before it can be identified using its guid.

id property is generated automatically on [EverCloudLocalData](#) construction for convenience but can be overridden manually

## 7.32.5.5 local

```
bool qevercloud::EverCloudLocalData::local = false
```

local property can be used to keep track which data items are meant to be local only and thus never be synchronized with Evernote service

## 7.32.6 Property Documentation

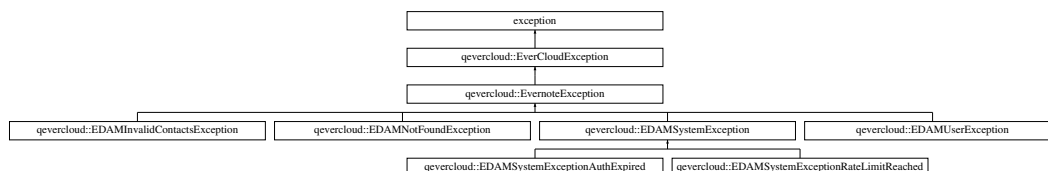
## 7.32.6.1 dict

```
Dict qevercloud::EverCloudLocalData::dict
```

## 7.33 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 [EverCloudExceptionDataPtr exceptionData](#) () const override

## Additional Inherited Members

### 7.33.1 Detailed Description

All exception sent by Evernote servers (as opposed to other error conditions, for example http errors) are descendants of this class.

### 7.33.2 Constructor & Destructor Documentation

#### 7.33.2.1 `EvernoteException()` [1/4]

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

#### 7.33.2.2 `EvernoteException()` [2/4]

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

#### 7.33.2.3 `EvernoteException()` [3/4]

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

#### 7.33.2.4 `EvernoteException()` [4/4]

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

### 7.33.3 Member Function Documentation

## 7.33.3.1 exceptionData()

```
virtual EverCloudExceptionDataPtr qevercloud::EvernoteException::exceptionData ( ) const [override],
[virtual]
```

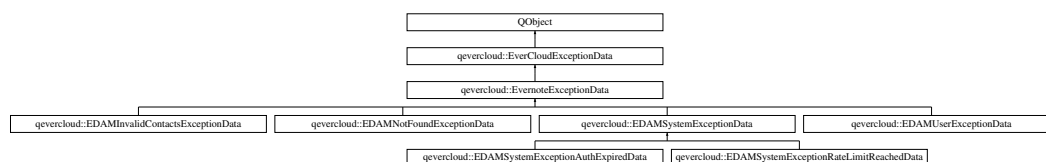
Reimplemented from [qevercloud::EverCloudException](#).

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

## 7.34 qevercloud::EvernoteExceptionData Class Reference

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

Inheritance diagram for qevercloud::EvernoteExceptionData:



## Public Member Functions

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

## Additional Inherited Members

## 7.34.1 Detailed Description

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

## 7.34.2 Constructor &amp; Destructor Documentation

## 7.34.2.1 EvernoteExceptionData()

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

## 7.34.3 Member Function Documentation

### 7.34.3.1 `throwException()`

```
virtual void qevercloud::EvernoteExceptionData::throwException ( ) const [override], [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](#).

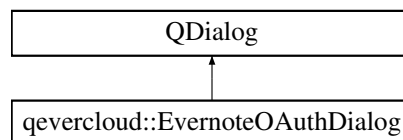
Reimplemented in [qevercloud::EDAMSystemExceptionAuthExpiredData](#), [qevercloud::EDAMSystemExceptionRateLimitReachedData](#), [qevercloud::EDAMInvalidContactsExceptionData](#), [qevercloud::EDAMNotFoundExceptionData](#), [qevercloud::EDAMSystemExceptionData](#), and [qevercloud::EDAMUserExceptionData](#).

## 7.35 `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

- using [OAuthResult](#) = [EvernoteOAuthWebView::OAuthResult](#)

### Public Member Functions

- [EvernoteOAuthDialog](#) (QString consumerKey, QString consumerSecret, QString host=QString::Literal("www.evernote.com"), QWidget \*parent=Q\_NULLPTR)
- virtual [~EvernoteOAuthDialog](#) () override
- void [setWebViewSizeHint](#) (QSize sizeHint)
- bool [isSucceeded](#) () const
- QString [oauthError](#) () const
- [OAuthResult](#) [oauthResult](#) () const
- virtual int [exec](#) () override
- virtual void [open](#) () override



### 7.35.1 Detailed Description

Authorizes your app with the Evernote service by means of OAuth authentication.

Intended usage:

```
#include <QEverCloudOAuth.h>

EvernoteOAuthDialog d(myConsumerKey, myConsumerSecret);
if(d.exec() == QDialog::Accepted) {
    EvernoteOAuthDialog::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.

### 7.35.2 Member Typedef Documentation

#### 7.35.2.1 OAuthResult

```
using qevercloud::EvernoteOAuthDialog::OAuthResult = EvernoteOAuthWebView::OAuthResult
```

### 7.35.3 Constructor & Destructor Documentation

#### 7.35.3.1 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

## 7.35.3.2 ~EvernoteOAuthDialog()

```
virtual qevercloud::EvernoteOAuthDialog::~EvernoteOAuthDialog ( ) [override], [virtual]
```

## 7.35.4 Member Function Documentation

## 7.35.4.1 exec()

```
virtual int qevercloud::EvernoteOAuthDialog::exec ( ) [override], [virtual]
```

## Returns

QDialog::Accepted on a successful authentication.

## 7.35.4.2 isSucceeded()

```
bool qevercloud::EvernoteOAuthDialog::isSucceeded ( ) const
```

## Returns

true in case of a successful authentication. You probably better check [exec\(\)](#) return value instead.

## 7.35.4.3 oauthError()

```
QString qevercloud::EvernoteOAuthDialog::oauthError ( ) const
```

## Returns

In case of an authentication error may return some information about the error.

## 7.35.4.4 oauthResult()

```
OAuthResult qevercloud::EvernoteOAuthDialog::oauthResult ( ) const
```

## Returns

the result of a successful authentication.

## 7.35.4.5 open()

```
virtual void qevercloud::EvernoteOAuthDialog::open ( ) [override], [virtual]
```

Shows the dialog as a window modal dialog, returning immediately.

## 7.35.4.6 setWebViewSizeHint()

```
void qevercloud::EvernoteOAuthDialog::setWebViewSizeHint (
    QSize sizeHint )
```

The dialog adjusts its initial size automatically based on the contained QWebView preferred size. Use this method to set the size.

## Parameters

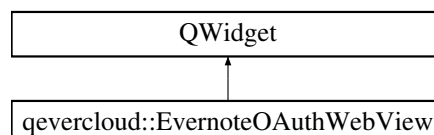
<i>sizeHint</i>	will be used as the preferred size of the contained QWebView.
-----------------	---------------------------------------------------------------

## 7.36 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, const quint64 timeout←Msec=30000)
- bool [isSucceeded](#) () const
- QString [oauthError](#) () const
- [OAuthResult](#) [oauthResult](#) () const
- void [setSizeHint](#) (QSize [sizeHint](#))
- QSize [sizeHint](#) () const override

### 7.36.1 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 [OAuth.h](#) header.

By default [EvernoteOAuthWebView](#) 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.

### 7.36.2 Constructor & Destructor Documentation

#### 7.36.2.1 EvernoteOAuthWebView()

```
qevercloud::EvernoteOAuthWebView::EvernoteOAuthWebView (
    QWidget * parent = Q_NULLPTR )
```

### 7.36.3 Member Function Documentation

#### 7.36.3.1 authenticate()

```
void qevercloud::EvernoteOAuthWebView::authenticate (
    QString host,
    QString consumerKey,
    QString consumerSecret,
    const quint64 timeoutMsec = 30000 )
```

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>• "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
<i>timeoutMsec</i>	Timeout for network requests in milliseconds

## 7.36.3.2 authenticationFailed

```
void qevercloud::EvernoteOAuthWebView::authenticationFailed ( ) [signal]
```

Emitted when the OAuth sequence is finished with a failure. Some error info may be available with `errorText()`.

## 7.36.3.3 authenticationFinished

```
void qevercloud::EvernoteOAuthWebView::authenticationFinished (
    bool success ) [signal]
```

Emitted when the OAuth sequence started with `authenticate()` call is finished

## 7.36.3.4 authenticationSucceeded

```
void qevercloud::EvernoteOAuthWebView::authenticationSucceeded ( ) [signal]
```

Emitted when the OAuth sequence is successfully finished. Call `oauthResult()` to get the data.

## 7.36.3.5 isSucceeded()

```
bool qevercloud::EvernoteOAuthWebView::isSucceeded ( ) const
```

## Returns

true if the last call to authenticate resulted in a successful authentication.

## 7.36.3.6 oauthError()

```
QString qevercloud::EvernoteOAuthWebView::oauthError ( ) const
```

## Returns

error message resulted from the last call to authenticate

### 7.36.3.7 oauthResult()

```
OAuthResult qevercloud::EvernoteOAuthWebView::oauthResult ( ) const
```

#### Returns

the result of the last authentication, i.e. [authenticate\(\)](#) call.

### 7.36.3.8 setSizeHint()

```
void qevercloud::EvernoteOAuthWebView::setSizeHint (
    QSize sizeHint )
```

The method is useful to specify default size for a EverOAuthWebView.

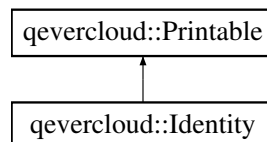
### 7.36.3.9 sizeHint()

```
QSize qevercloud::EvernoteOAuthWebView::sizeHint ( ) const [override]
```

## 7.37 qevercloud::Identity Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::Identity:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [Identity](#) &other) const
- bool [operator!=](#) (const [Identity](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [IdentityID](#) [id](#) = 0
- [Optional](#)< [Contact](#) > [contact](#)
- [Optional](#)< [UserID](#) > [userId](#)
- [Optional](#)< bool > [deactivated](#)
- [Optional](#)< bool > [sameBusiness](#)
- [Optional](#)< bool > [blocked](#)
- [Optional](#)< bool > [userConnected](#)
- [Optional](#)< [MessageEventID](#) > [eventId](#)

### 7.37.1 Detailed Description

An object that represents the relationship between a [Contact](#) that possibly belongs to an Evernote [User](#).

### 7.37.2 Member Function Documentation

#### 7.37.2.1 operator!=(())

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

#### 7.37.2.2 operator==(())

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

#### 7.37.2.3 print()

```
virtual void qevercloud::Identity::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.37.3 Member Data Documentation

#### 7.37.3.1 blocked

```
Optional< bool > qevercloud::Identity::blocked
```

Has the caller blocked the Evernote user this [Identity](#) represents?

#### 7.37.3.2 contact

```
Optional< Contact > qevercloud::Identity::contact
```

NOT DOCUMENTED

### 7.37.3.3 deactivated

```
Optional< bool > qevercloud::Identity::deactivated
```

Indicates that the contact for this identity is no longer active and should not be used when creating new threads using `Destination.recipients`, unless you know of another [Identity](#) instance with the same contact information that is active. If you are connected to the user (see `userConnected`), you can still create threads using their Evernote-type contact.

### 7.37.3.4 eventId

```
Optional< MessageEventID > qevercloud::Identity::eventId
```

A server-assigned sequence number for the events in the messages subsystem.

### 7.37.3.5 id

```
IdentityID qevercloud::Identity::id = 0
```

The unique identifier for this mapping.

### 7.37.3.6 localData

```
EverCloudLocalData qevercloud::Identity::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.37.3.7 sameBusiness

```
Optional< bool > qevercloud::Identity::sameBusiness
```

Does this [Identity](#) belong to someone who is in the same business as the caller?

### 7.37.3.8 userConnected

```
Optional< bool > qevercloud::Identity::userConnected
```

Indicates that the caller is "connected" to the user of this identity via this identity. When you have a connection via an identity, you should always create new threads using the Evernote-type contact (see `ContactType`) using the `userId` field from a connected [Identity](#). On the Evernote service, the Evernote-type contact is the most durable. Phone numbers and e-mail addresses can get re-assigned but your Evernote account user ID will remain the same. A connection exists when both of you are in the same business or the user has replied to a thread that you are on. When connected, you will also get to see more information about the user who has claimed the identity. **Note** that you are never connected to yourself since you won't be sending messages to yourself, but you will obviously see your own profile information.



7.37.3.9 `userId`

```
Optional< UserID > qevercloud::Identity::userId
```

The Evernote [User](#) id that is connected to the [Contact](#). May be unset if this identity has not yet been claimed, or the caller is not connected to this identity.

7.38 `qevercloud::IDurableService` Class Reference

```
#include <DurableService.h>
```

## Classes

- struct [AsyncRequest](#)
- struct [SyncRequest](#)

## Public Types

- using [SyncResult](#) = std::pair< QVariant, [EverCloudExceptionDataPtr](#) >
- using [SyncServiceCall](#) = std::function< [SyncResult](#)([IRequestContextPtr](#))>
- using [AsyncServiceCall](#) = std::function< [AsyncResult](#) \*([IRequestContextPtr](#))>

## Public Member Functions

- virtual [SyncResult](#) [executeSyncRequest](#) ([SyncRequest](#) &&syncRequest, [IRequestContextPtr](#) ctx)=0
- virtual [AsyncResult](#) \* [executeAsyncRequest](#) ([AsyncRequest](#) &&asyncRequest, [IRequestContextPtr](#) ctx)=0

## 7.38.1 Member Typedef Documentation

7.38.1.1 `AsyncServiceCall`

```
using qevercloud::IDurableService::AsyncServiceCall = std::function<AsyncResult*(IRequestContextPtr)>
```

7.38.1.2 `SyncResult`

```
using qevercloud::IDurableService::SyncResult = std::pair<QVariant, EverCloudExceptionDataPtr>
```

### 7.38.1.3 SyncServiceCall

```
using qevercloud::IDurableService::SyncServiceCall = std::function<SyncResult (IRequestContext↵
Ptr)>
```

## 7.38.2 Member Function Documentation

### 7.38.2.1 executeAsyncRequest()

```
virtual AsyncResult* qevercloud::IDurableService::executeAsyncRequest (
    AsyncRequest && asyncRequest,
    IRequestContextPtr ctx ) [pure virtual]
```

### 7.38.2.2 executeSyncRequest()

```
virtual SyncResult qevercloud::IDurableService::executeSyncRequest (
    SyncRequest && syncRequest,
    IRequestContextPtr ctx ) [pure virtual]
```

## 7.39 qevercloud::ILogger Class Reference

```
#include <Log.h>
```

### Public Member Functions

- virtual bool [shouldLog](#) (const [LogLevel](#) level, const char \*component) const =0
- virtual void [log](#) (const [LogLevel](#) level, const char \*component, const char \*fileName, const quint32 line↵
Number, const qint64 timestamp, const QString &message)=0
- virtual void [setLevel](#) (const [LogLevel](#) level)=0
- virtual [LogLevel](#) level () const =0

### 7.39.1 Member Function Documentation

#### 7.39.1.1 level()

```
virtual LogLevel qevercloud::ILogger::level ( ) const [pure virtual]
```

## 7.39.1.2 log()

```
virtual void qevercloud::ILogger::log (
    const LogLevel level,
    const char * component,
    const char * fileName,
    const quint32 lineNumber,
    const qint64 timestamp,
    const QString & message ) [pure virtual]
```

## 7.39.1.3 setLevel()

```
virtual void qevercloud::ILogger::setLevel (
    const LogLevel level ) [pure virtual]
```

## 7.39.1.4 shouldLog()

```
virtual bool qevercloud::ILogger::shouldLog (
    const LogLevel level,
    const char * component ) const [pure virtual]
```

## 7.40 qevercloud::InkNoteImageDownloader Class Reference

the [InkNoteImageDownloader](#) class is for downloading the images of ink notes which can be created with the official Evernote client on Windows (only with it, at least at the time of this writing).

```
#include <InkNoteImageDownloader.h>
```

## Public Member Functions

- [InkNoteImageDownloader](#) ()  
*Default constructor.*
- [InkNoteImageDownloader](#) (QString host, QString shardId, QString authenticationToken, int width, int height)  
*Constructs [InkNoteImageDownloader](#).*
- virtual [~InkNoteImageDownloader](#) ()
- [InkNoteImageDownloader](#) & [setHost](#) (QString host)
- [InkNoteImageDownloader](#) & [setShardId](#) (QString shardId)
- [InkNoteImageDownloader](#) & [setAuthenticationToken](#) (QString authenticationToken)
- [InkNoteImageDownloader](#) & [setWidth](#) (int width)
- [InkNoteImageDownloader](#) & [setHeight](#) (int height)
- QByteArray [download](#) (Guid guid, const bool isPublic=false, const qint64 timeoutMsec=30000)  
*Downloads the image for the ink note.*

### 7.40.1 Detailed Description

the [InkNoteImageDownloader](#) class is for downloading the images of ink notes which can be created with the official Evernote client on Windows (only with it, at least at the time of this writing).

On all other platforms the most one can get instead of the actual ink note is its non-editable image. This class retrieves just these, exclusively in PNG format.

NOTE: almost the entirety of this class' content represents an ad-hoc solution to a completely undocumented feature of Evernote service. A very small glimpse of information was once available on Evernote forums but it's deleted now... I collected an even smaller glimpse of information in this SO question: <https://stackoverflow.com/q/39179012/1217285>. For all practical purposes it is the only piece of information on this feature in existence now.

### 7.40.2 Constructor & Destructor Documentation

#### 7.40.2.1 InkNoteImageDownloader() [1/2]

```
qevercloud::InkNoteImageDownloader::InkNoteImageDownloader ( )
```

Default constructor.

host, shardId, authenticationToken, width, height have to be specified before calling [download](#) or createPostRequest

#### 7.40.2.2 InkNoteImageDownloader() [2/2]

```
qevercloud::InkNoteImageDownloader::InkNoteImageDownloader (
    QString host,
    QString shardId,
    QString authenticationToken,
    int width,
    int height )
```

Constructs [InkNoteImageDownloader](#).

#### Parameters

<i>host</i>	www.evernote.com or sandbox.evernote.com
<i>shardId</i>	You can get the value from UserStore service or as a result of an authentication.
<i>authenticationToken</i>	For working private ink notes you must supply a valid authentication token. For public resources the value specified is not used.
<i>width</i>	Width of the ink note's resource
<i>height</i>	Height of the ink note's resource

## 7.40.2.3 ~InkNoteImageDownloader()

```
virtual qevercloud::InkNoteImageDownloader::~InkNoteImageDownloader ( ) [virtual]
```

## 7.40.3 Member Function Documentation

## 7.40.3.1 download()

```
QByteArray qevercloud::InkNoteImageDownloader::download (
    Guid guid,
    const bool isPublic = false,
    const qint64 timeoutMsec = 30000 )
```

Downloads the image for the ink note.

Unlike other pieces of QEverCloud API, downloading of ink note images is currently synchronous only. The reason for that is that [AsyncResult](#) is bounded to a single `QNetworkRequest` object but downloading of the ink note image might take multiple requests for several ink note image's vertical stripes which are then merged together to form a single image. Downloading the entire ink note's image via a single request works sometimes but sometimes Evernote replies to such request with messed up data which cannot be loaded into a `QImage`. The reason for that behaviour is unknown at the moment, and, given the state of official documentation

- missing - it is likely to stay so. if someone has an idea how to make it more reliable, please let me know.

## Parameters

<i>guid</i>	The guid of the ink note's resource
<i>isPublic</i>	Specify true for public ink notes. In this case authentication token is not sent to with the request as it should be according to the docs.
<i>timeoutMsec</i>	Timeout for download request in milliseconds

## Returns

downloaded data.

## 7.40.3.2 setAuthenticationToken()

```
InkNoteImageDownloader& qevercloud::InkNoteImageDownloader::setAuthenticationToken (
    QString authenticationToken )
```

## Parameters

<i>authenticationToken</i>	For working private ink notes you must supply a valid authentication token. For public resources the value specified is not used.
----------------------------	-----------------------------------------------------------------------------------------------------------------------------------

#### 7.40.3.3 setHeight()

```
InkNoteImageDownloader& qevercloud::InkNoteImageDownloader::setHeight (
    int height )
```

##### Parameters

<i>height</i>	Height of the ink note's resource
---------------	-----------------------------------

#### 7.40.3.4 setHost()

```
InkNoteImageDownloader& qevercloud::InkNoteImageDownloader::setHost (
    QString host )
```

##### Parameters

<i>host</i>	www.evernote.com or sandbox.evernote.com
-------------	------------------------------------------

#### 7.40.3.5 setShardId()

```
InkNoteImageDownloader& qevercloud::InkNoteImageDownloader::setShardId (
    QString shardId )
```

##### Parameters

<i>shardId</i>	You can get the value from UserStore service or as a result of an authentication.
----------------	-----------------------------------------------------------------------------------

#### 7.40.3.6 setWidth()

```
InkNoteImageDownloader& qevercloud::InkNoteImageDownloader::setWidth (
    int width )
```

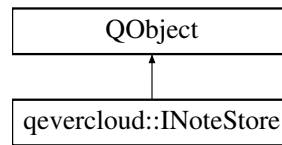
##### Parameters

<i>width</i>	Width of the ink note's resource
--------------	----------------------------------

## 7.41 qevercloud::INoteStore Class Reference

```
#include <Services.h>
```

Inheritance diagram for qevercloud::INoteStore:



### Public Member Functions

- virtual QString [noteStoreUrl](#) () const =0
- virtual void [setNoteStoreUrl](#) (QString url)=0
- virtual [SyncState](#) [getSyncState](#) (IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [getSyncStateAsync](#) (IRequestContextPtr ctx={})=0
- virtual [SyncChunk](#) [getFilteredSyncChunk](#) (qint32 afterUSN, qint32 maxEntries, const [SyncChunkFilter](#) &filter, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [getFilteredSyncChunkAsync](#) (qint32 afterUSN, qint32 maxEntries, const [SyncChunkFilter](#) &filter, IRequestContextPtr ctx={})=0
- virtual [SyncState](#) [getLinkedNotebookSyncState](#) (const [LinkedNotebook](#) &linkedNotebook, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [getLinkedNotebookSyncStateAsync](#) (const [LinkedNotebook](#) &linkedNotebook, IRequestContextPtr ctx={})=0
- virtual [SyncChunk](#) [getLinkedNotebookSyncChunk](#) (const [LinkedNotebook](#) &linkedNotebook, qint32 afterUSN, qint32 maxEntries, bool fullSyncOnly, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [getLinkedNotebookSyncChunkAsync](#) (const [LinkedNotebook](#) &linkedNotebook, qint32 afterUSN, qint32 maxEntries, bool fullSyncOnly, IRequestContextPtr ctx={})=0
- virtual QList< [Notebook](#) > [listNotebooks](#) (IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [listNotebooksAsync](#) (IRequestContextPtr ctx={})=0
- virtual QList< [Notebook](#) > [listAccessibleBusinessNotebooks](#) (IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [listAccessibleBusinessNotebooksAsync](#) (IRequestContextPtr ctx={})=0
- virtual [Notebook](#) [getNotebook](#) (Guid guid, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [getNotebookAsync](#) (Guid guid, IRequestContextPtr ctx={})=0
- virtual [Notebook](#) [getDefaultNotebook](#) (IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [getDefaultNotebookAsync](#) (IRequestContextPtr ctx={})=0
- virtual [Notebook](#) [createNotebook](#) (const [Notebook](#) &notebook, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [createNotebookAsync](#) (const [Notebook](#) &notebook, IRequestContextPtr ctx={})=0
- virtual qint32 [updateNotebook](#) (const [Notebook](#) &notebook, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [updateNotebookAsync](#) (const [Notebook](#) &notebook, IRequestContextPtr ctx={})=0
- virtual qint32 [expungeNotebook](#) (Guid guid, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [expungeNotebookAsync](#) (Guid guid, IRequestContextPtr ctx={})=0
- virtual QList< [Tag](#) > [listTags](#) (IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [listTagsAsync](#) (IRequestContextPtr ctx={})=0
- virtual QList< [Tag](#) > [listTagsByNotebook](#) (Guid notebookGuid, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [listTagsByNotebookAsync](#) (Guid notebookGuid, IRequestContextPtr ctx={})=0
- virtual [Tag](#) [getTag](#) (Guid guid, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [getTagAsync](#) (Guid guid, IRequestContextPtr ctx={})=0
- virtual [Tag](#) [createTag](#) (const [Tag](#) &tag, IRequestContextPtr ctx={})=0
- virtual [AsyncResult](#) \* [createTagAsync](#) (const [Tag](#) &tag, IRequestContextPtr ctx={})=0
- virtual qint32 [updateTag](#) (const [Tag](#) &tag, IRequestContextPtr ctx={})=0

- virtual [AsyncResult](#) \* [updateTagAsync](#) (const [Tag](#) &tag, [IRequestContextPtr](#) ctx={})=0
- virtual void [untagAll](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [untagAllAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [expungeTag](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [expungeTagAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [QList](#)< [SavedSearch](#) > [listSearches](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [listSearchesAsync](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [SavedSearch](#) [getSearch](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getSearchAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [SavedSearch](#) [createSearch](#) (const [SavedSearch](#) &search, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [createSearchAsync](#) (const [SavedSearch](#) &search, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [updateSearch](#) (const [SavedSearch](#) &search, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [updateSearchAsync](#) (const [SavedSearch](#) &search, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [expungeSearch](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [expungeSearchAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [findNoteOffset](#) (const [NoteFilter](#) &filter, [Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [findNoteOffsetAsync](#) (const [NoteFilter](#) &filter, [Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [NotesMetadataList](#) [findNotesMetadata](#) (const [NoteFilter](#) &filter, [qint32](#) offset, [qint32](#) maxNotes, const [NotesMetadataResultSpec](#) &resultSpec, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [findNotesMetadataAsync](#) (const [NoteFilter](#) &filter, [qint32](#) offset, [qint32](#) maxNotes, const [NotesMetadataResultSpec](#) &resultSpec, [IRequestContextPtr](#) ctx={})=0
- virtual [NoteCollectionCounts](#) [findNoteCounts](#) (const [NoteFilter](#) &filter, bool withTrash, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [findNoteCountsAsync](#) (const [NoteFilter](#) &filter, bool withTrash, [IRequestContextPtr](#) ctx={})=0
- virtual [Note](#) [getNoteWithResultSpec](#) ([Guid](#) guid, const [NoteResultSpec](#) &resultSpec, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteWithResultSpecAsync](#) ([Guid](#) guid, const [NoteResultSpec](#) &resultSpec, [IRequestContextPtr](#) ctx={})=0
- virtual [Note](#) [getNote](#) ([Guid](#) guid, bool withContent, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteAsync](#) ([Guid](#) guid, bool withContent, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [LazyMap](#) [getNoteApplicationData](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteApplicationDataAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [QString](#) [getNoteApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [setNoteApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [setNoteApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [unsetNoteApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [unsetNoteApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [QString](#) [getNoteContent](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteContentAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [QString](#) [getNoteSearchText](#) ([Guid](#) guid, bool noteOnly, bool tokenizeForIndexing, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteSearchTextAsync](#) ([Guid](#) guid, bool noteOnly, bool tokenizeForIndexing, [IRequestContextPtr](#) ctx={})=0
- virtual [QString](#) [getResourceSearchText](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceSearchTextAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [QStringList](#) [getNoteTagNames](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteTagNamesAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0



- virtual [Note](#) [createNote](#) (const [Note](#) &note, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [createNoteAsync](#) (const [Note](#) &note, [IRequestContextPtr](#) ctx={})=0
- virtual [Note](#) [updateNote](#) (const [Note](#) &note, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [updateNoteAsync](#) (const [Note](#) &note, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [deleteNote](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [deleteNoteAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [expungeNote](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [expungeNoteAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [Note](#) [copyNote](#) ([Guid](#) noteGuid, [Guid](#) toNotebookGuid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [copyNoteAsync](#) ([Guid](#) noteGuid, [Guid](#) toNotebookGuid, [IRequestContextPtr](#) ctx={})=0
- virtual [QList](#)< [NoteVersionId](#) > [listNoteVersions](#) ([Guid](#) noteGuid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [listNoteVersionsAsync](#) ([Guid](#) noteGuid, [IRequestContextPtr](#) ctx={})=0
- virtual [Note](#) [getNoteVersion](#) ([Guid](#) noteGuid, [qint32](#) updateSequenceNum, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNoteVersionAsync](#) ([Guid](#) noteGuid, [qint32](#) updateSequenceNum, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [Resource](#) [getResource](#) ([Guid](#) guid, bool withData, bool withRecognition, bool withAttributes, bool withAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceAsync](#) ([Guid](#) guid, bool withData, bool withRecognition, bool withAttributes, bool withAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [LazyMap](#) [getResourceApplicationData](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceApplicationDataAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [QString](#) [getResourceApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [setResourceApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [setResourceApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [QString](#) value, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [unsetResourceApplicationDataEntry](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [unsetResourceApplicationDataEntryAsync](#) ([Guid](#) guid, [QString](#) key, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [updateResource](#) (const [Resource](#) &resource, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [updateResourceAsync](#) (const [Resource](#) &resource, [IRequestContextPtr](#) ctx={})=0
- virtual [QByteArray](#) [getResourceData](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceDataAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [Resource](#) [getResourceByHash](#) ([Guid](#) noteGuid, [QByteArray](#) contentHash, bool withData, bool withRecognition, bool withAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceByHashAsync](#) ([Guid](#) noteGuid, [QByteArray](#) contentHash, bool withData, bool withRecognition, bool withAlternateData, [IRequestContextPtr](#) ctx={})=0
- virtual [QByteArray](#) [getResourceRecognition](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceRecognitionAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [QByteArray](#) [getResourceAlternateData](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceAlternateDataAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [ResourceAttributes](#) [getResourceAttributes](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getResourceAttributesAsync](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx={})=0
- virtual [Notebook](#) [getPublicNotebook](#) ([UserID](#) userId, [QString](#) publicUri, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getPublicNotebookAsync](#) ([UserID](#) userId, [QString](#) publicUri, [IRequestContextPtr](#) ctx={})=0
- virtual [SharedNotebook](#) [shareNotebook](#) (const [SharedNotebook](#) &sharedNotebook, [QString](#) message, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [shareNotebookAsync](#) (const [SharedNotebook](#) &sharedNotebook, [QString](#) message, [IRequestContextPtr](#) ctx={})=0
- virtual [CreateOrUpdateNotebookSharesResult](#) [createOrUpdateNotebookShares](#) (const [NotebookShareTemplate](#) &shareTemplate, [IRequestContextPtr](#) ctx={})=0

- virtual [AsyncResult](#) \* [createOrUpdateNotebookSharesAsync](#) (const [NotebookShareTemplate](#) &shareTemplate, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [updateSharedNotebook](#) (const [SharedNotebook](#) &sharedNotebook, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [updateSharedNotebookAsync](#) (const [SharedNotebook](#) &sharedNotebook, [IRequestContextPtr](#) ctx={})=0
- virtual [Notebook](#) [setNotebookRecipientSettings](#) (QString notebookGuid, const [NotebookRecipientSettings](#) &recipientSettings, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [setNotebookRecipientSettingsAsync](#) (QString notebookGuid, const [NotebookRecipientSettings](#) &recipientSettings, [IRequestContextPtr](#) ctx={})=0
- virtual [QList](#)< [SharedNotebook](#) > [listSharedNotebooks](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [listSharedNotebooksAsync](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [LinkedNotebook](#) [createLinkedNotebook](#) (const [LinkedNotebook](#) &linkedNotebook, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [createLinkedNotebookAsync](#) (const [LinkedNotebook](#) &linkedNotebook, [IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [updateLinkedNotebook](#) (const [LinkedNotebook](#) &linkedNotebook, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [updateLinkedNotebookAsync](#) (const [LinkedNotebook](#) &linkedNotebook, [IRequestContextPtr](#) ctx={})=0
- virtual [QList](#)< [LinkedNotebook](#) > [listLinkedNotebooks](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [listLinkedNotebooksAsync](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [qint32](#) [expungeLinkedNotebook](#) (Guid guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [expungeLinkedNotebookAsync](#) (Guid guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AuthenticationResult](#) [authenticateToSharedNotebook](#) (QString shareKeyOrGlobalId, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [authenticateToSharedNotebookAsync](#) (QString shareKeyOrGlobalId, [IRequestContextPtr](#) ctx={})=0
- virtual [SharedNotebook](#) [getSharedNotebookByAuth](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getSharedNotebookByAuthAsync](#) ([IRequestContextPtr](#) ctx={})=0
- virtual void [emailNote](#) (const [NoteEmailParameters](#) &parameters, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [emailNoteAsync](#) (const [NoteEmailParameters](#) &parameters, [IRequestContextPtr](#) ctx={})=0
- virtual QString [shareNote](#) (Guid guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [shareNoteAsync](#) (Guid guid, [IRequestContextPtr](#) ctx={})=0
- virtual void [stopSharingNote](#) (Guid guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [stopSharingNoteAsync](#) (Guid guid, [IRequestContextPtr](#) ctx={})=0
- virtual [AuthenticationResult](#) [authenticateToSharedNote](#) (QString guid, QString noteKey, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [authenticateToSharedNoteAsync](#) (QString guid, QString noteKey, [IRequestContextPtr](#) ctx={})=0
- virtual [RelatedResult](#) [findRelated](#) (const [RelatedQuery](#) &query, const [RelatedResultSpec](#) &resultSpec, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [findRelatedAsync](#) (const [RelatedQuery](#) &query, const [RelatedResultSpec](#) &resultSpec, [IRequestContextPtr](#) ctx={})=0
- virtual [UpdateNoteIfUsnMatchesResult](#) [updateNoteIfUsnMatches](#) (const [Note](#) &note, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [updateNoteIfUsnMatchesAsync](#) (const [Note](#) &note, [IRequestContextPtr](#) ctx={})=0
- virtual [ManageNotebookSharesResult](#) [manageNotebookShares](#) (const [ManageNotebookSharesParameters](#) &parameters, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [manageNotebookSharesAsync](#) (const [ManageNotebookSharesParameters](#) &parameters, [IRequestContextPtr](#) ctx={})=0
- virtual [ShareRelationships](#) [getNotebookShares](#) (QString notebookGuid, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getNotebookSharesAsync](#) (QString notebookGuid, [IRequestContextPtr](#) ctx={})=0

## Protected Member Functions

- [INoteStore](#) (QObject \*parent)

### 7.41.1 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.

Most functions take an "authenticationToken" parameter, which is the value returned by the UserStore which permits access to the account.

Calls which require an authenticationToken may throw an [EDAMUserException](#) for the following reasons:

- DATA\_REQUIRED "authenticationToken" - token is empty
- BAD\_DATA\_FORMAT "authenticationToken" - token is malformed
- INVALID\_AUTH "authenticationToken" - token signature is invalid
- AUTH\_EXPIRED "authenticationToken" - token has expired or been revoked
- PERMISSION\_DENIED "authenticationToken" - token does not grant permission to perform the requested action
- BUSINESS\_SECURITY\_LOGIN\_REQUIRED "sso" - the user is a member of a business that requires single sign-on, and must complete SSO before accessing business content.

### 7.41.2 Constructor & Destructor Documentation

#### 7.41.2.1 INoteStore()

```
qevercloud::INoteStore::INoteStore (
    QObject * parent ) [inline], [protected]
```

### 7.41.3 Member Function Documentation

#### 7.41.3.1 authenticateToSharedNote()

```
virtual AuthenticationResult qevercloud::INoteStore::authenticateToSharedNote (
    QString guid,
    QString noteKey,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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="#">EDAMUserException</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="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "guid" - the note with that GUID is not found</li> </ul>
<a href="#">EDAMSystemException</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>

7.41.3.2 [authenticateToSharedNoteAsync\(\)](#)

```
virtual AsyncResult* qevercloud::INoteStore::authenticateToSharedNoteAsync (
    QString guid,
    QString noteKey,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [authenticateToSharedNote](#)

7.41.3.3 [authenticateToSharedNotebook\(\)](#)

```
virtual AuthenticationResult qevercloud::INoteStore::authenticateToSharedNotebook (
    QString shareKeyOrGlobalId,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>shareKeyOr↔ GlobalId</i>	<p>May be one of the following:</p> <ul style="list-style-type: none"> <li>• A share key for a shared notebook that was granted to some recipient. Must be used if you are joining a notebook unless it was shared via <code>createOrUpdateNotebookShares</code>. Share keys are delivered out-of-band and are generally not available to clients. For security reasons, share keys may be invalidated at the discretion of the service.</li> <li>• The shared notebook global identifier. May be used to access a notebook that is already joined.</li> <li>• The <a href="#">Notebook</a> GUID. May be used to access a notebook that was already joined, or to access a notebook that was shared with the recipient via <code>createOrUpdateNotebookShares</code>.</li> </ul>
<i>authenticationToken</i>	<p>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. If this string is empty, the service will attempt to authenticate to the shared notebook without any logged in user.</p>

## Exceptions

<a href="#">EDAMSystemException</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="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "SharedNotebook.id" - the shared notebook no longer exists</li> </ul>
<a href="#">EDAMUserException</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>

## 7.41.3.4 authenticateToSharedNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::authenticateToSharedNotebookAsync (
    QString shareKeyOrGlobalId,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [authenticateToSharedNotebook](#)

## 7.41.3.5 copyNote()

```
virtual Note qevercloud::INoteStore::copyNote (
    Guid noteGuid,
```

```

    Guid toNotebookGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]

```

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.

If the original note has been shared and has [SharedNote](#) records, the shares are NOT copied.

#### 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>

#### 7.41.3.6 copyNoteAsync()

```

virtual AsyncResult* qevercloud::INoteStore::copyNoteAsync (
    Guid noteGuid,
    Guid toNotebookGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]

```

Asynchronous version of [copyNote](#)

#### 7.41.3.7 createLinkedNotebook()

```

virtual LinkedNotebook qevercloud::INoteStore::createLinkedNotebook (
    const LinkedNotebook & linkedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]

```

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>• DATA_REQUIRED "LinkedNotebook.shareName" - missing shareName</li> <li>• BAD_DATA_FORMAT "LinkedNotebook.name" - invalid shareName 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>• DATA_REQUIRED "LinkedNotebook.shardId" - if private notebook but shard id not provided</li> <li>• BAD_DATA_FORMAT "LinkedNotebook.stack" - invalid stack name length or pattern</li> </ul>
<a href="#">EDAMSystemException</a>	<ul style="list-style-type: none"> <li>• BAD_DATA_FORMAT "LinkedNotebook.sharedNotebookGlobalId" - if a bad global identifier was set on a private notebook</li> </ul>

## 7.41.3.8 createLinkedNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::createLinkedNotebookAsync (
    const LinkedNotebook & linkedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [createLinkedNotebook](#)

## 7.41.3.9 createNote()

```
virtual Note qevercloud::INoteStore::createNote (
    const Note & note,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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. 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 (except Recognition [Resource](#) body, for which the behavior is unspecified).

## Exceptions

<a href="#">EDAMUserException</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="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Note.notebookGuid" - not found, by GUID</li> </ul>



## 7.41.3.10 createNoteAsync()

```
virtual AsyncResult* qevercloud::INoteStore::createNoteAsync (
    const Note & note,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [createNote](#)

## 7.41.3.11 createNotebook()

```
virtual Notebook qevercloud::INoteStore::createNotebook (
    const Notebook & notebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>• DATA_REQUIRED "Notebook" - notebook parameter was null</li><li>• PERMISSION_DENIED "Notebook.defaultNotebook" - if the 'defaultNotebook' field is set to 'true' for a <a href="#">Notebook</a> that is not owned by the user identified by the passed authenticationToken.</li><li>• LIMIT_REACHED "Notebook" - at max number of notebooks</li></ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"><li>• "Workspace.guid" - if workspaceGuid set and no Workspace exists for the GUID</li></ul>

### 7.41.3.12 createNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::createNotebookAsync (
    const Notebook & notebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [createNotebook](#)

### 7.41.3.13 createOrUpdateNotebookShares()

```
virtual CreateOrUpdateNotebookSharesResult qevercloud::INoteStore::createOrUpdateNotebook↵
Shares (
    const NotebookShareTemplate & shareTemplate,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Share a notebook by a messaging thread ID or a list of contacts. This function is intended to be used in conjunction with Evernote messaging, and as such does not notify the recipient that a notebook has been shared with them.

Sharing with a subset of participants on a thread is accomplished by specifying both a thread ID and a list of contacts. This ensures that even if those contacts are on the thread under a deactivated identity, the correct user (the one who has the given contact on the thread) receives the share.

#### Parameters

<i>authenticationToken</i>	An authentication token that grants the caller permission to share the notebook. This should be an owner token if the notebook is owned by the caller. If the notebook is a business notebook to which the caller has full access, this should be their business authentication token. If the notebook is a shared (non-business) notebook to which the caller has full access, this should be the shared notebook authentication token returned by <code>NoteStore.authenticateToNotebook</code> .
<i>shareTemplate</i>	Specifies the GUID of the notebook to be shared, the privilege at which the notebook should be shared, and the recipient information.

#### Returns

A structure containing the USN of the [Notebook](#) after the change and a list of created or updated Shared↵  
Notebooks.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "Notebook.guid" - if no notebook GUID was specified</li> <li>• BAD_DATA_FORMAT "Notebook.guid" - if shareTemplate.notebookGuid is not a valid GUID</li> <li>• DATA_REQUIRED "shareTemplate" - if the shareTemplate parameter was missing</li> <li>• DATA_REQUIRED "NotebookShareTemplate.privilege" - if no privilege was specified</li> <li>• DATA_CONFLICT "NotebookShareTemplate.privilege" - if the specified privilege is not allowed.</li> <li>• DATA_REQUIRED "NotebookShareTemplate.recipients" - if no recipients were specified, either by thread ID or as a list of contacts</li> <li>• LIMIT_REACHED "SharedNotebook" - if the notebook has reached its maximum number of shares</li> </ul>
<a href="#"><i>EDAMInvalidContactsException</i></a>	<ul style="list-style-type: none"> <li>• "NotebookShareTemplate.recipients" - if one or more of the recipients specified in shareTemplate.recipients was not syntactically valid, or if attempting to share a notebook with an Evernote identity that the sharer does not have a connection to. The exception will specify which recipients were invalid.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Notebook.guid" - if no notebook with the specified GUID was found</li> <li>• "NotebookShareTemplate.recipientThreadId" - if the recipient thread ID was specified, but no thread with that ID exists</li> </ul>

## 7.41.3.14 createOrUpdateNotebookSharesAsync()

```
virtual AsyncResult* qevercloud::INoteStore::createOrUpdateNotebookSharesAsync (
    const NotebookShareTemplate & shareTemplate,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [createOrUpdateNotebookShares](#)

## 7.41.3.15 createSearch()

```
virtual SavedSearch qevercloud::INoteStore::createSearch (
    const SavedSearch & search,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
-----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.41.3.16 createSearchAsync()

```
virtual AsyncResult* qevercloud::INoteStore::createSearchAsync (
    const SavedSearch & search,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [createSearch](#)

## 7.41.3.17 createTag()

```
virtual Tag qevercloud::INoteStore::createTag (
    const Tag & tag,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.18 createTagAsync()

```
virtual AsyncResult* qevercloud::INoteStore::createTagAsync (
    const Tag & tag,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [createTag](#)

## 7.41.3.19 deleteNote()

```
virtual qint32 qevercloud::INoteStore::deleteNote (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
---------------------------------------	------------------------------------------------------------------------------------

## 7.41.3.20 deleteNoteAsync()

```
virtual AsyncResult* qevercloud::INoteStore::deleteNoteAsync (  
    Guid guid,  
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [deleteNote](#)

## 7.41.3.21 emailNote()

```
virtual void qevercloud::INoteStore::emailNote (  
    const NoteEmailParameters & parameters,  
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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="#"><i>EDAMUserException</i></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="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

## 7.41.3.22 emailNoteAsync()

```
virtual AsyncResult* qevercloud::INoteStore::emailNoteAsync (
    const NoteEmailParameters & parameters,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [emailNote](#)

## 7.41.3.23 expungeLinkedNotebook()

```
virtual qint32 qevercloud::INoteStore::expungeLinkedNotebook (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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.
-------------	-----------------------------------------------------------------------------------------------------------------------------

#### 7.41.3.24 expungeLinkedNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::expungeLinkedNotebookAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [expungeLinkedNotebook](#)

#### 7.41.3.25 expungeNote()

```
virtual qint32 qevercloud::INoteStore::expungeNote (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

#### 7.41.3.26 expungeNoteAsync()

```
virtual AsyncResult* qevercloud::INoteStore::expungeNoteAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [expungeNote](#)



#### 7.41.3.27 expungeNotebook()

```
virtual qint32 qevercloud::INoteStore::expungeNotebook (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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="#">EDAMUserException</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>
-----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### 7.41.3.28 expungeNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::expungeNotebookAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [expungeNotebook](#)

#### 7.41.3.29 expungeSearch()

```
virtual qint32 qevercloud::INoteStore::expungeSearch (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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="#"><i>EDAMUserException</i></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="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "SavedSearch.guid" - not found, by GUID</li> </ul>

## 7.41.3.30 expungeSearchAsync()

```
virtual AsyncResult\* qevercloud::INoteStore::expungeSearchAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [expungeSearch](#)

## 7.41.3.31 expungeTag()

```
virtual qint32 qevercloud::INoteStore::expungeTag (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Permanently deletes the tag with the provided GUID, if present.

NOTE: This function is not generally 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>

## 7.41.3.32 expungeTagAsync()

```
virtual AsyncResult* qevercloud::INoteStore::expungeTagAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [expungeTag](#)

## 7.41.3.33 findNoteCounts()

```
virtual NoteCollectionCounts qevercloud::INoteStore::findNoteCounts (
    const NoteFilter & filter,
    bool withTrash,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Notebook.guid" - not found, by GUID</li> </ul>
----------------------------------------------	------------------------------------------------------------------------------------------

## 7.41.3.34 findNoteCountsAsync()

```
virtual AsyncResult* qevercloud::INoteStore::findNoteCountsAsync (
    const NoteFilter & filter,
    bool withTrash,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [findNoteCounts](#)

## 7.41.3.35 findNoteOffset()

```
virtual qint32 qevercloud::INoteStore::findNoteOffset (
    const NoteFilter & filter,
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.36 findNoteOffsetAsync()

```
virtual AsyncResult* qevercloud::INoteStore::findNoteOffsetAsync (
    const NoteFilter & filter,
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [findNoteOffset](#)

## 7.41.3.37 findNotesMetadata()

```
virtual NotesMetadataList qevercloud::INoteStore::findNotesMetadata (
    const NoteFilter & filter,
    qint32 offset,
    qint32 maxNotes,
    const NotesMetadataResultSpec & resultSpec,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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.

## Parameters

<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. Currently, the service will not return more than 250 notes in a single request, but this number may change in the future.
<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. The `Notes.sharedNotes` field will not be set.

## 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>

## 7.41.3.38 findNotesMetadataAsync()

```
virtual AsyncResult* qevercloud::INoteStore::findNotesMetadataAsync (
    const NoteFilter & filter,
    qint32 offset,
    qint32 maxNotes,
    const NotesMetadataResultSpec & resultSpec,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [findNotesMetadata](#)

## 7.41.3.39 findRelated()

```
virtual RelatedResult qevercloud::INoteStore::findRelated (
    const RelatedQuery & query,
    const RelatedResultSpec & resultSpec,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Identify related entities on the service, such as notes, notebooks, tags and users in a business 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="#"><i>EDAMUserException</i></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>PERMISSION_DENIED "authenticationToken" - If the caller has requested to findExperts in the context of a non business user (i.e. The authenticationToken is not a business auth token).</li> <li>DATA_REQUIRED "RelatedResultSpec" - If you did not not set any values in the result spec.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></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>

## 7.41.3.40 findRelatedAsync()

```
virtual AsyncResult* qevercloud::INoteStore::findRelatedAsync (
    const RelatedQuery & query,
    const RelatedResultSpec & resultSpec,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [findRelated](#)

7.41.3.41 `getDefaultNotebook()`

```
virtual Notebook qevercloud::INoteStore::getDefaultNotebook (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns the notebook that should be used to store new notes in the user's account when no other notebooks are specified.

7.41.3.42 `getDefaultNotebookAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getDefaultNotebookAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getDefaultNotebook](#)

7.41.3.43 `getFilteredSyncChunk()`

```
virtual SyncChunk qevercloud::INoteStore::getFilteredSyncChunk (
    qint32 afterUSN,
    qint32 maxEntries,
    const SyncChunkFilter & filter,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
-----------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------



## 7.41.3.44 getFilteredSyncChunkAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getFilteredSyncChunkAsync (
    qint32 afterUSN,
    qint32 maxEntries,
    const SyncChunkFilter & filter,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getFilteredSyncChunk](#)

## 7.41.3.45 getLinkedNotebookSyncChunk()

```
virtual SyncChunk qevercloud::INoteStore::getLinkedNotebookSyncChunk (
    const LinkedNotebook & linkedNotebook,
    qint32 afterUSN,
    qint32 maxEntries,
    bool fullSyncOnly,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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. shareKey) or a public notebook (e.g. username, uri)
<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>

7.41.3.46 `getLinkedNotebookSyncChunkAsync()`

```
virtual AsyncResult\* qevercloud::INoteStore::getLinkedNotebookSyncChunkAsync (
    const LinkedNotebook & linkedNotebook,
    qint32 afterUSN,
    qint32 maxEntries,
    bool fullSyncOnly,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getLinkedNotebookSyncChunk](#)

7.41.3.47 `getLinkedNotebookSyncState()`

```
virtual SyncState qevercloud::INoteStore::getLinkedNotebookSyncState (
    const LinkedNotebook & linkedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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 <code>authenticationToken</code> for the guest who has received the invitation to the share. (I.e. this should not be the result of <code>NoteStore.authenticateToSharedNotebook</code> )
<i>linkedNotebook</i>	This structure should contain identifying information and permissions to access the notebook in question.

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>DATA_REQUIRED "LinkedNotebook.username" - The username field must be populated with the current username of the owner of the notebook for which you are obtaining sync state.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>"LinkedNotebook.username" - If the <a href="#">LinkedNotebook.username</a> field does not correspond to a current user on the service.</li> </ul>
<a href="#"><i>SystemException</i></a>	<ul style="list-style-type: none"> <li>SHARD_UNAVAILABLE - If the provided <a href="#">LinkedNotebook.username</a> corresponds to a user whose account is on a shard other than that on which this method was invoked.</li> </ul>

## 7.41.3.48 getLinkedNotebookSyncStateAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getLinkedNotebookSyncStateAsync (
    const LinkedNotebook & linkedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getLinkedNotebookSyncState](#)

## 7.41.3.49 getNote()

```
virtual Note qevercloud::INoteStore::getNote (
    Guid guid,
    bool withContent,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

DEPRECATED. See [getNoteWithResultSpec](#).

This function is equivalent to [getNoteWithResultSpec](#), with each of the boolean parameters mapping to the equivalent field of a [NoteResultSpec](#). The [Note.sharedNotes](#) field is never populated on the returned note. To get a note with its shares, use [getNoteWithResultSpec](#).

## 7.41.3.50 getNoteApplicationData()

```
virtual LazyMap qevercloud::INoteStore::getNoteApplicationData (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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.

**7.41.3.51** `getNoteApplicationDataAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getNoteApplicationDataAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNoteApplicationData](#)

**7.41.3.52** `getNoteApplicationDataEntry()`

```
virtual QString qevercloud::INoteStore::getNoteApplicationDataEntry (
    Guid guid,
    QString key,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
---------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------

**7.41.3.53** `getNoteApplicationDataEntryAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getNoteApplicationDataEntryAsync (
    Guid guid,
    QString key,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNoteApplicationDataEntry](#)

**7.41.3.54** `getNoteAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getNoteAsync (
    Guid guid,
    bool withContent,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNote](#)

#### 7.41.3.55 getNotebook()

```
virtual Notebook qevercloud::INoteStore::getNotebook (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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

<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>PERMISSION_DENIED "Notebook" - private notebook, user doesn't own</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>"Notebook.guid" - tag not found, by GUID</li> </ul>

7.41.3.56 `getNotebookAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getNotebookAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNotebook](#)

7.41.3.57 `getNotebookShares()`

```
virtual ShareRelationships qevercloud::INoteStore::getNotebookShares (
    QString notebookGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Return the share relationships for the given notebook, including both the invitations and the memberships.

**Note:** Beta method! This method is currently intended for limited use by Evernote clients that have discussed using this routine with the platform team.

7.41.3.58 `getNotebookSharesAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getNotebookSharesAsync (
    QString notebookGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNotebookShares](#)

7.41.3.59 `getNoteContent()`

```
virtual QString qevercloud::INoteStore::getNoteContent (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

7.41.3.60 `getNoteContentAsync()`

```
virtual AsyncResult\* qevercloud::INoteStore::getNoteContentAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNoteContent](#)

7.41.3.61 `getNoteSearchText()`

```
virtual QString qevercloud::INoteStore::getNoteSearchText (
    Guid guid,
    bool noteOnly,
    bool tokenizeForIndexing,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID</li> </ul>

7.41.3.62 `getNoteSearchTextAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getNoteSearchTextAsync (
    Guid guid,
    bool noteOnly,
    bool tokenizeForIndexing,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNoteSearchText](#)

7.41.3.63 `getNoteTagNames()`

```
virtual QStringList qevercloud::INoteStore::getNoteTagNames (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

7.41.3.64 `getNoteTagNamesAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getNoteTagNamesAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```



Asynchronous version of [getNoteTagNames](#)

### 7.41.3.65 getNoteVersion()

```
virtual Note qevercloud::INoteStore::getNoteVersion (
    Guid noteGuid,
    qint32 updateSequenceNum,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>DATA_REQUIRED "Note.guid" - if GUID is null or empty string.</li> <li>BAD_DATA_FORMAT "Note.guid" - if GUID is not of correct length.</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>

### 7.41.3.66 getNoteVersionAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getNoteVersionAsync (
    Guid noteGuid,
    qint32 updateSequenceNum,
    bool withResourcesData,
    bool withResourcesRecognition,
```

```
bool withResourcesAlternateData,
 IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNoteVersion](#)

#### 7.41.3.67 getNoteWithResultSpec()

```
virtual Note qevercloud::INoteStore::getNoteWithResultSpec (
    Guid guid,
    const NoteResultSpec & resultSpec,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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 content depends on whether it is explicitly requested in resultSpec parameter. 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>authenticationToken</i>	An authentication token that grants the caller access to the requested note.
<i>guid</i>	The GUID of the note to be retrieved.
<i>resultSpec</i>	A structure specifying the fields of the note that the caller would like to get.

##### 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>

#### 7.41.3.68 getNoteWithResultSpecAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getNoteWithResultSpecAsync (
    Guid guid,
    const NoteResultSpec & resultSpec,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getNoteWithResultSpec](#)

#### 7.41.3.69 getPublicNotebook()

```
virtual Notebook qevercloud::INoteStore::getPublicNotebook (
    UserID userId,
```

```
QString publicUri,
 IRequestContextPtr ctx = {} ) [pure virtual]
```

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 <code>UserStore.getPublicUserInfo</code>
<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>

#### 7.41.3.70 getPublicNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getPublicNotebookAsync (
    UserID userId,
    QString publicUri,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getPublicNotebook](#)

#### 7.41.3.71 getResource()

```
virtual Resource qevercloud::INoteStore::getResource (
    Guid guid,
    bool withData,
    bool withRecognition,
    bool withAttributes,
    bool withAlternateData,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.72 getResourceAlternateData()

```
virtual QByteArray qevercloud::INoteStore::getResourceAlternateData (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.73 getResourceAlternateDataAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceAlternateDataAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceAlternateData](#)

## 7.41.3.74 getResourceApplicationData()

```
virtual LazyMap qevercloud::INoteStore::getResourceApplicationData (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Get all of the application data for the [Resource](#) identified by GUID, with values returned within the [LazyMap](#) full↔Map 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.

## 7.41.3.75 getResourceApplicationDataAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceApplicationDataAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceApplicationData](#)

## 7.41.3.76 getResourceApplicationDataEntry()

```
virtual QString qevercloud::INoteStore::getResourceApplicationDataEntry (
    Guid guid,
    QString key,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
---------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.41.3.77 getResourceApplicationDataEntryAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceApplicationDataEntryAsync (
    Guid guid,
```

```
QString key,
IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceApplicationDataEntry](#)

#### 7.41.3.78 getResourceAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceAsync (
    Guid guid,
    bool withData,
    bool withRecognition,
    bool withAttributes,
    bool withAlternateData,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResource](#)

#### 7.41.3.79 getResourceAttributes()

```
virtual ResourceAttributes qevercloud::INoteStore::getResourceAttributes (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

#### 7.41.3.80 getResourceAttributesAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceAttributesAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceAttributes](#)

## 7.41.3.81 getResourceByHash()

```
virtual Resource qevercloud::INoteStore::getResourceByHash (
    Guid noteGuid,
    QByteArray contentHash,
    bool withData,
    bool withRecognition,
    bool withAlternateData,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.82 getResourceByHashAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceByHashAsync (
    Guid noteGuid,
    QByteArray contentHash,
    bool withData,
    bool withRecognition,
    bool withAlternateData,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceByHash](#)

7.41.3.83 `getResourceData()`

```
virtual QByteArray qevercloud::INoteStore::getResourceData (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

7.41.3.84 `getResourceDataAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::getResourceDataAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceData](#)

7.41.3.85 `getResourceRecognition()`

```
virtual QByteArray qevercloud::INoteStore::getResourceRecognition (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.86 getResourceRecognitionAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceRecognitionAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceRecognition](#)

## 7.41.3.87 getResourceSearchText()

```
virtual QString qevercloud::INoteStore::getResourceSearchText (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

### 7.41.3.88 getResourceSearchTextAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getResourceSearchTextAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getResourceSearchText](#)

### 7.41.3.89 getSearch()

```
virtual SavedSearch qevercloud::INoteStore::getSearch (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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="#"><i>EDAMUserException</i></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="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>"SavedSearch.guid" - not found, by GUID</li> </ul>

### 7.41.3.90 getSearchAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getSearchAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getSearch](#)

### 7.41.3.91 getSharedNotebookByAuth()

```
virtual SharedNotebook qevercloud::INoteStore::getSharedNotebookByAuth (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.92 getSharedNotebookByAuthAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getSharedNotebookByAuthAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getSharedNotebookByAuth](#)

## 7.41.3.93 getSyncState()

```
virtual SyncState qevercloud::INoteStore::getSyncState (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asks the NoteStore to provide information about the status of the user account corresponding to the provided authentication token.

## 7.41.3.94 getSyncStateAsync()

```
virtual AsyncResult* qevercloud::INoteStore::getSyncStateAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getSyncState](#)

## 7.41.3.95 getTag()

```
virtual Tag qevercloud::INoteStore::getTag (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

7.41.3.96 `getTagAsync()`

```
virtual AsyncResult\* qevercloud::INoteStore::getTagAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getTag](#)

7.41.3.97 `listAccessibleBusinessNotebooks()`

```
virtual QList<Notebook> qevercloud::INoteStore::listAccessibleBusinessNotebooks (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns a list of all the notebooks in a business that the user has permission to access, regardless of whether the user has joined them. This includes notebooks that have been shared with the entire business as well as notebooks that have been shared directly with the user.

## Parameters

<i>authenticationToken</i>	A business authentication token obtained by calling <code>UserStore.authenticateToBusiness</code> .
----------------------------	-----------------------------------------------------------------------------------------------------

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>INVALID_AUTH "authenticationToken" - if the authentication token is not a business auth token.</li> </ul>
------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------

**7.41.3.98 listAccessibleBusinessNotebooksAsync()**

```
virtual AsyncResult* qevercloud::INoteStore::listAccessibleBusinessNotebooksAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listAccessibleBusinessNotebooks](#)

**7.41.3.99 listLinkedNotebooks()**

```
virtual QList<LinkedNotebook> qevercloud::INoteStore::listLinkedNotebooks (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns a list of linked notebooks

**7.41.3.100 listLinkedNotebooksAsync()**

```
virtual AsyncResult* qevercloud::INoteStore::listLinkedNotebooksAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listLinkedNotebooks](#)

**7.41.3.101 listNotebooks()**

```
virtual QList<Notebook> qevercloud::INoteStore::listNotebooks (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns a list of all of the notebooks in the account.

**7.41.3.102 listNotebooksAsync()**

```
virtual AsyncResult* qevercloud::INoteStore::listNotebooksAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listNotebooks](#)

**7.41.3.103 listNoteVersions()**

```
virtual QList<NoteVersionId> qevercloud::INoteStore::listNoteVersions (
    Guid noteGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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. This call is only available for notes in Premium accounts. (I.e. access to past versions of Notes is a Premium-only feature.)

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "Note.guid" - if GUID is null or empty string.</li> <li>• BAD_DATA_FORMAT "Note.guid" - if GUID is not of correct length.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "Note.guid" - not found, by GUID.</li> </ul>

## 7.41.3.104 listNoteVersionsAsync()

```
virtual AsyncResult* qevercloud::INoteStore::listNoteVersionsAsync (
    Guid noteGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listNoteVersions](#)

## 7.41.3.105 listSearches()

```
virtual QList<SavedSearch> qevercloud::INoteStore::listSearches (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns a list of the searches in the account. Evernote does not support the undeletion of searches, so this will only include active searches.

## 7.41.3.106 listSearchesAsync()

```
virtual AsyncResult* qevercloud::INoteStore::listSearchesAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listSearches](#)

## 7.41.3.107 listSharedNotebooks()

```
virtual QList<SharedNotebook> qevercloud::INoteStore::listSharedNotebooks (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Lists the collection of shared notebooks for all notebooks in the users account.

## Returns

The list of all SharedNotebooks for the user

#### 7.41.3.108 listSharedNotebooksAsync()

```
virtual AsyncResult* qevercloud::INoteStore::listSharedNotebooksAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listSharedNotebooks](#)

#### 7.41.3.109 listTags()

```
virtual QList<Tag> qevercloud::INoteStore::listTags (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns a list of the tags in the account. Evernote does not support the undeletion of tags, so this will only include active tags.

#### 7.41.3.110 listTagsAsync()

```
virtual AsyncResult* qevercloud::INoteStore::listTagsAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listTags](#)

#### 7.41.3.111 listTagsByNotebook()

```
virtual QList<Tag> qevercloud::INoteStore::listTagsByNotebook (
    Guid notebookGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
---------------------------------------	----------------------------------------------------------------------------------------------

#### 7.41.3.112 listTagsByNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::listTagsByNotebookAsync (
    Guid notebookGuid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listTagsByNotebook](#)

7.41.3.113 `manageNotebookShares()`

```
virtual ManageNotebookSharesResult qevercloud::INoteStore::manageNotebookShares (
    const ManageNotebookSharesParameters & parameters,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Manage invitations and memberships associated with a given notebook.

**Note:** Beta method! This method is currently intended for limited use by Evernote clients that have discussed using this routine with the platform team.

## Parameters

<i>parameters</i>	A structure containing all parameters for the updates. See the structure documentation for details.
-------------------	-----------------------------------------------------------------------------------------------------

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li><a href="#"><code>EDAMErrorCode.LIMIT_REACHED</code></a> "SharedNotebook" - Trying to share a notebook while the notebook already has <code>EDAM_NOTEBOOK_SHARED_NOTEBOOK_MAX</code> shares.</li> </ul>
------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

7.41.3.114 `manageNotebookSharesAsync()`

```
virtual AsyncResult* qevercloud::INoteStore::manageNotebookSharesAsync (
    const ManageNotebookSharesParameters & parameters,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [`manageNotebookShares`](#)

7.41.3.115 `noteStoreUrl()`

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

7.41.3.116 `setNoteApplicationDataEntry()`

```
virtual quint32 qevercloud::INoteStore::setNoteApplicationDataEntry (
    Guid guid,
    QString key,
    QString value,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Update, or create, an entry in the `applicationData` map for the note identified by `guid`.



#### 7.41.3.117 setNoteApplicationDataEntryAsync()

```
virtual AsyncResult* qevercloud::INoteStore::setNoteApplicationDataEntryAsync (
    Guid guid,
    QString key,
    QString value,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [setNoteApplicationDataEntry](#)

#### 7.41.3.118 setNotebookRecipientSettings()

```
virtual Notebook qevercloud::INoteStore::setNotebookRecipientSettings (
    QString notebookGuid,
    const NotebookRecipientSettings & recipientSettings,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Set values for the recipient settings associated with a notebook share. Only the recipient of the share can update their recipient settings.

If you do *not* wish to, or cannot, change one of the recipient settings fields, you must leave that field unset in recipientSettings. This method will skip that field for updates and attempt to leave the existing value as it is.

If recipientSettings.inMyList is false, both reminderNotifyInApp and reminderNotifyEmail will be either left as null or converted to false (if currently true).

To unset a notebook's stack, pass in the empty string for the stack field.

##### Parameters

<i>authenticationToken</i>	The owner authentication token for the recipient of the share.
----------------------------	----------------------------------------------------------------

##### Returns

The updated [Notebook](#) with the new recipient settings. **Note** that some of the recipient settings may differ from what was requested. Clients should update their state based on this return value.

##### Exceptions

<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"><li><a href="#">Notebook.guid</a> - Thrown if the service does not have a notebook record with the notebookGuid on the given shard.</li><li>Publishing.publishState - Thrown if the business notebook is not shared with the user and is also not published to their business.</li></ul>
---------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• PERMISSION_DENIED "authenticationToken" - If the owner of the given token is not allowed to set recipient settings on the specified notebook.</li> <li>• DATA_CONFLICT "recipientSettings.reminderNotifyEmail" - Setting reminderNotifyEmail is allowed only for notebooks which belong to the same business as the user.</li> <li>• DATA_CONFLICT "recipientSettings.inMyList" - If the request is setting inMyList to false and any of reminder* settings to true.</li> </ul>
------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

7.41.3.119 **setNotebookRecipientSettingsAsync()**

```
virtual AsyncResult* qevercloud::INoteStore::setNotebookRecipientSettingsAsync (
    QString notebookGuid,
    const NotebookRecipientSettings & recipientSettings,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [setNotebookRecipientSettings](#)

7.41.3.120 **setNoteStoreUrl()**

```
virtual void qevercloud::INoteStore::setNoteStoreUrl (
    QString url ) [pure virtual]
```

7.41.3.121 **setResourceApplicationDataEntry()**

```
virtual qint32 qevercloud::INoteStore::setResourceApplicationDataEntry (
    Guid guid,
    QString key,
    QString value,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Update, or create, an entry in the applicationData map for the [Resource](#) identified by guid.

7.41.3.122 **setResourceApplicationDataEntryAsync()**

```
virtual AsyncResult* qevercloud::INoteStore::setResourceApplicationDataEntryAsync (
    Guid guid,
    QString key,
    QString value,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [setResourceApplicationDataEntry](#)

## 7.41.3.123 shareNote()

```
virtual QString qevercloud::INoteStore::shareNote (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

If this note is not already shared publicly (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

<i>EDAMUserException</i>	<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>
<i>EDAMNotFoundException</i>	<ul style="list-style-type: none"> <li>"Note.guid" - not found, by GUID</li> </ul>

## 7.41.3.124 shareNoteAsync()

```
virtual AsyncResult* qevercloud::INoteStore::shareNoteAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [shareNote](#)

## 7.41.3.125 shareNotebook()

```
virtual SharedNotebook qevercloud::INoteStore::shareNotebook (
    const SharedNotebook & sharedNotebook,
    QString message,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

for first-party clients. See createOrUpdateNotebookShares.

Share a notebook with an email address, and optionally to a specific recipient. If an existing [SharedNotebook](#) associated with sharedNotebook.notebookGuid is found by recipientUsername or email, then the values of shared↵ Notebook will be used to update the existing record, else a new record will be created.

If recipientUsername is set and there is already a [SharedNotebook](#) for that [Notebook](#) with that recipientUsername and the privileges on the existing notebook are lower, than on this one, this will update the privileges and sharer↔ UserId. If there isn't an existing [SharedNotebook](#) for recipientUsername, this will create and return a shared notebook for that email and recipientUsername. If recipientUsername is not set and there already is a [SharedNotebook](#) for a [Notebook](#) for that email address and the privileges on the existing [SharedNotebook](#) are lower than on this one, this will update the privileges and sharerUserId, and return the updated [SharedNotebook](#). Otherwise, this will create and return a [SharedNotebook](#) for the email address.

If the authenticationToken is a Business auth token, recipientUsername is set and the recipient is in the same business as the business auth token, this method will also auto-join the business user to the [SharedNotebook](#) - that is it will set serviceJoined on the [SharedNotebook](#) and create a [LinkedNotebook](#) on the recipient's account pointing to the [SharedNotebook](#). The [LinkedNotebook](#) creation happens out-of-band, so there will be a delay on the order of half a minute between the [SharedNotebook](#) and [LinkedNotebook](#) creation.

Also handles sending an email to the email addresses: if a [SharedNotebook](#) is being created, this will send the shared notebook invite email, and if a [SharedNotebook](#) already exists, it will send the shared notebook reminder email. Both these emails contain a link to join the notebook. If the notebook is being auto-joined, it sends an email with that information to the recipient.

#### 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>	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 SharedNotebook.allowPreview field must be explicitly set with either a true or false value.
<i>message</i>	The sharer-defined message to put in the email sent out.

#### Returns

The fully populated [SharedNotebook](#) object including the server assigned globalId 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>• DATA_REQUIRED "SharedNotebook.privilege" - if the SharedNotebook.privilegeLevel was not set.</li> <li>• BAD_DATA_FORMAT "SharedNotebook.requireLogin" - if requireLogin was set. requireLogin is deprecated.</li> <li>• BAD_DATA_FORMAT "SharedNotebook.privilegeLevel" - if the SharedNotebook.privilegeLevel field was unset or set to GROUP.</li> <li>• PERMISSION_DENIED "user" - if the email address on the authenticationToken's owner's account is not confirmed.</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> <li>• <a href="#"><i>EDAMErrorCode.LIMIT_REACHED</i></a> "SharedNotebook" - The notebook already has EDAM_NOTEBOOK_SHARED_NOTEBOOK_MAX shares.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• <a href="#"><i>Notebook.guid</i></a> - if the notebookGuid is not a valid GUID for the user.</li> </ul>

## 7.41.3.126 shareNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::shareNotebookAsync (
    const SharedNotebook & sharedNotebook,
    QString message,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [\*shareNotebook\*](#)

## 7.41.3.127 stopSharingNote()

```
virtual void qevercloud::INoteStore::stopSharingNote (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

If this note is shared publicly 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.

This function does not remove individual shares for the note. To remove individual shares, see [\*stopSharingNoteWithRecipients\*](#).

## 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>

## 7.41.3.128 stopSharingNoteAsync()

```
virtual AsyncResult* qevercloud::INoteStore::stopSharingNoteAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [stopSharingNote](#)

## 7.41.3.129 unsetNoteApplicationDataEntry()

```
virtual qint32 qevercloud::INoteStore::unsetNoteApplicationDataEntry (
    Guid guid,
    QString key,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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.

## 7.41.3.130 unsetNoteApplicationDataEntryAsync()

```
virtual AsyncResult* qevercloud::INoteStore::unsetNoteApplicationDataEntryAsync (
    Guid guid,
    QString key,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [unsetNoteApplicationDataEntry](#)

## 7.41.3.131 unsetResourceApplicationDataEntry()

```
virtual qint32 qevercloud::INoteStore::unsetResourceApplicationDataEntry (
    Guid guid,
    QString key,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Remove an entry identified by 'key' from the applicationData map for the [Resource](#) identified by 'guid'.

## 7.41.3.132 unsetResourceApplicationDataEntryAsync()

```
virtual AsyncResult* qevercloud::INoteStore::unsetResourceApplicationDataEntryAsync (
    Guid guid,
    QString key,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [unsetResourceApplicationDataEntry](#)

## 7.41.3.133 untagAll()

```
virtual void qevercloud::INoteStore::untagAll (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.134 untagAllAsync()

```
virtual AsyncResult* qevercloud::INoteStore::untagAllAsync (
    Guid guid,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [untagAll](#)

## 7.41.3.135 updateLinkedNotebook()

```
virtual qint32 qevercloud::INoteStore::updateLinkedNotebook (
    const LinkedNotebook & linkedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

## 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>• DATA_REQUIRED "LinkedNotebook.shareName" - missing shareName</li> <li>• BAD_DATA_FORMAT "LinkedNotebook.shareName" - invalid shareName length or pattern</li> <li>• BAD_DATA_FORMAT "LinkedNotebook.stack" - invalid stack name length or pattern</li> </ul>
-----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.41.3.136 updateLinkedNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::updateLinkedNotebookAsync (
    const LinkedNotebook & linkedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateLinkedNotebook](#)

## 7.41.3.137 updateNote()

```
virtual Note qevercloud::INoteStore::updateNote (
    const Note & note,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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 [Note.sharedNotes](#) field will not be set. 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.



## 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.notebookGuid" - user doesn't own destination</li> <li>• PERMISSION_DENIED "Note.tags" - user doesn't have permission to modify the note's tags. note.tags must be unset.</li> <li>• PERMISSION_DENIED "Note.attributes" - user doesn't have permission to modify the note's attributes. note.attributes must be unset.</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>

## 7.41.3.138 updateNoteAsync()

```
virtual AsyncResult* qevercloud::INoteStore::updateNoteAsync (
    const Note & note,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateNote](#)

## 7.41.3.139 updateNotebook()

```
virtual qint32 qevercloud::INoteStore::updateNotebook (
    const Notebook & notebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Submits notebook changes to the service. The provided data must include the notebook's guid field for identification.

The [Notebook](#) will be moved to the specified Workspace, if a non empty Notebook.workspaceGuid is provided. If an empty Notebook.workspaceGuid is set and the [Notebook](#) is in a Workspace, then it will be removed from the Workspace and a full access [SharedNotebook](#) record will be ensured for the caller. If the caller does not already have a full access share, either the privilege of an existing share will be upgraded or a new share will be created. It is illegal to set a Notebook.workspaceGuid on a Workspace backing [Notebook](#).

## Parameters

<i>notebook</i>	The notebook object containing the requested changes.
-----------------	-------------------------------------------------------

## 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 "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>• DATA_REQUIRED "Notebook" - notebook parameter was null</li> <li>• PERMISSION_DENIED "Notebook.defaultNotebook" - if the 'defaultNotebook' field is set to 'true' for a <a href="#">Notebook</a> that is not owned by the user identified by the passed authenticationToken.</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>• "Notebook.guid" - not found, by GUID</li> <li>• "Workspace.guid" - if a non empty workspaceGuid set and no Workspace exists for the GUID</li> </ul>

## 7.41.3.140 updateNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::updateNotebookAsync (
```

```
const Notebook & notebook,
 IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateNotebook](#)

#### 7.41.3.141 updateNoteIfUsnMatches()

```
virtual UpdateNoteIfUsnMatchesResult qevercloud::INoteStore::updateNoteIfUsnMatches (
    const Note & note,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Perform the same operation as [updateNote\(\)](#) would provided that the update sequence number on the parameter [Note](#) object matches the current update sequence number that the service has for the note. If they do *not* match, then *no* update is performed and the return value will have the current server state in the note field and updated will be false. If the update sequence numbers between the client and server do match, then the note will be updated and the note field of the return value will be returned as it would be for the [updateNote](#) method. This method allows you to check for an update to the note on the service, by another client instance, from when you obtained the note state as a baseline for your edits and the time when you wish to save your edits. If your client can merge the conflict, you can avoid overwriting changes that were saved to the service by the other client.

See the [updateNote](#) method for information on the exceptions and parameters for this method. The only difference is that you must have an update sequence number defined on the note parameter (equal to the USN of the note as synched to the client), and the following additional exceptions might be thrown.

#### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "Note.updateSequenceNum" - If the update sequence number was not provided. This includes a value that is set as 0.</li> <li>• BAD_DATA_FORMAT "Note.updateSequenceNum" - If the note has an update sequence number that is larger than the current server value, which should not happen if your client is working correctly.</li> </ul>
-----------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### 7.41.3.142 updateNoteIfUsnMatchesAsync()

```
virtual AsyncResult* qevercloud::INoteStore::updateNoteIfUsnMatchesAsync (
    const Note & note,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateNoteIfUsnMatches](#)

#### 7.41.3.143 updateResource()

```
virtual qint32 qevercloud::INoteStore::updateResource (
    const Resource & resource,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.144 updateResourceAsync()

```
virtual AsyncResult* qevercloud::INoteStore::updateResourceAsync (
    const Resource & resource,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateResource](#)

## 7.41.3.145 updateSearch()

```
virtual qint32 qevercloud::INoteStore::updateSearch (
    const SavedSearch & search,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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

<i>EDAMUserException</i>	<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>
<i>EDAMNotFoundException</i>	<ul style="list-style-type: none"> <li>"SavedSearch.guid" - not found, by GUID</li> </ul>

## 7.41.3.146 updateSearchAsync()

```
virtual AsyncResult* qevercloud::INoteStore::updateSearchAsync (
    const SavedSearch & search,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateSearch](#)

## 7.41.3.147 updateSharedNotebook()

```
virtual qint32 qevercloud::INoteStore::updateSharedNotebook (
    const SharedNotebook & sharedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

See [createOrUpdateNotebookShares](#) and [manageNotebookShares](#).

## 7.41.3.148 updateSharedNotebookAsync()

```
virtual AsyncResult* qevercloud::INoteStore::updateSharedNotebookAsync (
    const SharedNotebook & sharedNotebook,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateSharedNotebook](#)

#### 7.41.3.149 updateTag()

```
virtual qint32 qevercloud::INoteStore::updateTag (
    const Tag & tag,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>

## 7.41.3.150 updateTagAsync()

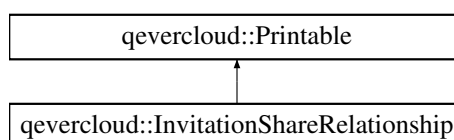
```
virtual AsyncResult* qevercloud::INoteStore::updateTagAsync (
    const Tag & tag,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [updateTag](#)

## 7.42 qevercloud::InvitationShareRelationship Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::InvitationShareRelationship:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [InvitationShareRelationship](#) &other) const
- bool [operator!=](#) (const [InvitationShareRelationship](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) localData
- [Optional](#)< [QString](#) > [displayName](#)
- [Optional](#)< [UserIdentity](#) > [recipientUserIdentity](#)
- [Optional](#)< [ShareRelationshipPrivilegeLevel](#) > [privilege](#)
- [Optional](#)< [UserID](#) > [sharerUserId](#)

### 7.42.1 Detailed Description

Describes an invitation to a person to use their Evernote credentials to become a member of a notebook.

### 7.42.2 Member Function Documentation

#### 7.42.2.1 [operator!=\(\)](#)

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

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

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

#### 7.42.2.3 [print\(\)](#)

```
virtual void qevercloud::InvitationShareRelationship::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.42.3 Member Data Documentation



## 7.42.3.1 displayName

`Optional< QString > qevercloud::InvitationShareRelationship::displayName`

The string that clients should show to users to represent this invitation.

## 7.42.3.2 localData

`EverCloudLocalData qevercloud::InvitationShareRelationship::localData`

See the declaration of [EverCloudLocalData](#) for details

## 7.42.3.3 privilege

`Optional< ShareRelationshipPrivilegeLevel > qevercloud::InvitationShareRelationship::privilege`

The privilege level at which the member will be joined, if it turns out that the member is not already joined at a higher level. **Note** that the `identity` field may not uniquely identify an Evernote [User](#) ID, and so we won't know until the invitation is redeemed whether or not the recipient already has privilege.

## 7.42.3.4 recipientUserIdentity

`Optional< UserIdentity > qevercloud::InvitationShareRelationship::recipientUserIdentity`

Identifies the recipient of the invitation. The user identity type can be either EMAIL, EVERNOTE or IDENTITYID. If the invitation was created using the classic notebook sharing APIs it will be EMAIL. If it was created using the new identity-based notebook sharing APIs it will either be EVERNOTE or IDENTITYID, depending on whether we can map the identity to an Evernote user at the time of creation.

## 7.42.3.5 sharerUserId

`Optional< UserID > qevercloud::InvitationShareRelationship::sharerUserId`

The user id of the user who most recently shared this notebook to this identity. This field is used by the service to convey information to the user, so clients should treat it as read-only.

## 7.43 qevercloud::IRequestContext Class Reference

```
#include <RequestContext.h>
```

## Public Member Functions

- virtual `QUuid requestId () const =0`
- virtual `QString authenticationToken () const =0`
- virtual `qint64 requestTimeout () const =0`
- virtual `bool increaseRequestTimeoutExponentially () const =0`
- virtual `qint64 maxRequestTimeout () const =0`
- virtual `quint32 maxRequestRetryCount () const =0`
- virtual `QList< QNetworkCookie > cookies () const =0`
- virtual `IRequestContext * clone () const =0`
- virtual `~IRequestContext ()=default`

## Friends

- [QEVERCLOUD\\_EXPORT](#) `QTextStream & operator<< (QTextStream &strm, const IRequestContext &ctx)`
- [QEVERCLOUD\\_EXPORT](#) `QDebug & operator<< (QDebug &dbg, const IRequestContext &ctx)`

### 7.43.1 Detailed Description

[IRequestContext](#) carries several request scoped values defining the way request is handled by QEverCloud

### 7.43.2 Constructor & Destructor Documentation

#### 7.43.2.1 `~IRequestContext()`

```
virtual qevercloud::IRequestContext::~IRequestContext ( ) [virtual], [default]
```

### 7.43.3 Member Function Documentation

#### 7.43.3.1 `authenticationToken()`

```
virtual QString qevercloud::IRequestContext::authenticationToken ( ) const [pure virtual]
```

Authentication token to use along with the request

#### 7.43.3.2 `clone()`

```
virtual IRequestContext* qevercloud::IRequestContext::clone ( ) const [pure virtual]
```

Create a new instance of [IRequestContext](#) with all the same parameters as in the source but a distinct id

#### 7.43.3.3 `cookies()`

```
virtual QList<QNetworkCookie> qevercloud::IRequestContext::cookies ( ) const [pure virtual]
```

Cookies to set to QNetworkRequest corresponding to Evernote API call

#### 7.43.3.4 `increaseRequestTimeoutExponentially()`

```
virtual bool qevercloud::IRequestContext::increaseRequestTimeoutExponentially ( ) const [pure virtual]
```

Should request timeout be exponentially increased on retries or not

**7.43.3.5 maxRequestRetryCount()**

```
virtual quint32 qevercloud::IRequestContext::maxRequestRetryCount ( ) const [pure virtual]
```

Max number of attempts to retry a request

**7.43.3.6 maxRequestTimeout()**

```
virtual qint64 qevercloud::IRequestContext::maxRequestTimeout ( ) const [pure virtual]
```

Max request timeout in milliseconds (upper boundary for exponentially increasing timeouts on retries)

**7.43.3.7 requestId()**

```
virtual QUuid qevercloud::IRequestContext::requestId ( ) const [pure virtual]
```

Automatically generated unique identifier for each request

**7.43.3.8 requestTimeout()**

```
virtual qint64 qevercloud::IRequestContext::requestTimeout ( ) const [pure virtual]
```

Request timeout in milliseconds

**7.43.4 Friends And Related Function Documentation****7.43.4.1 operator<< [1/2]**

```
QEVERCLOUD_EXPORT QTextStream& operator<< (
    QTextStream & strm,
    const IRequestContext & ctx ) [friend]
```

**7.43.4.2 operator<< [2/2]**

```
QEVERCLOUD_EXPORT QDebug& operator<< (
    QDebug & dbg,
    const IRequestContext & ctx ) [friend]
```

**7.44 qevercloud::IRetryPolicy Struct Reference**

```
#include <DurableService.h>
```

## Public Member Functions

- virtual bool [shouldRetry](#) (const [EverCloudExceptionDataPtr](#) &exceptionData)=0

### 7.44.1 Member Function Documentation

#### 7.44.1.1 [shouldRetry\(\)](#)

```
virtual bool qevercloud::IRetryPolicy::shouldRetry (
    const EverCloudExceptionDataPtr & exceptionData ) [pure virtual]
```

## 7.45 [qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator](#) Struct Reference

```
#include <Helpers.h>
```

## Public Member Functions

- [iterator](#) (const typename Container::iterator it)
- Container::iterator [operator\\*](#) ()
- [iterator](#) & [operator++](#) ()
- bool [operator!=](#) (const [iterator](#) &other) const

## Public Attributes

- Container::iterator [m\\_iterator](#)

### 7.45.1 Constructor & Destructor Documentation

#### 7.45.1.1 [iterator\(\)](#)

```
template<typename Container >
qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator::iterator (
    const typename Container::iterator it ) [inline]
```

### 7.45.2 Member Function Documentation

## 7.45.2.1 operator!=(())

```
template<typename Container >
bool qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator::operator!= (
    const iterator & other ) const [inline]
```

## 7.45.2.2 operator\*()

```
template<typename Container >
Container::iterator qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator↵
::operator* ( ) [inline]
```

## 7.45.2.3 operator++()

```
template<typename Container >
iterator& qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator::operator++↵
( ) [inline]
```

## 7.45.3 Member Data Documentation

## 7.45.3.1 m\_iterator

```
template<typename Container >
Container::iterator qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator↵
::m_iterator
```

## 7.46 qevercloud::QAssociativeContainerConstReferenceWrapper&lt; Container &gt;::↵ iterator Struct Reference

```
#include <Helpers.h>
```

## Public Member Functions

- iterator (const typename Container::const\_iterator it)
- Container::const\_iterator operator\* ()
- iterator & operator++ ()
- bool operator!= (const iterator &other) const

## Public Attributes

- Container::const\_iterator [m\\_iterator](#)

## 7.46.1 Constructor & Destructor Documentation

### 7.46.1.1 iterator()

```
template<typename Container >  
qevercloud::QAssociativeContainerConstReferenceWrapper< Container >::iterator::iterator (  
    const typename Container::const_iterator it ) [inline]
```

## 7.46.2 Member Function Documentation

### 7.46.2.1 operator!=(())

```
template<typename Container >  
bool qevercloud::QAssociativeContainerConstReferenceWrapper< Container >::iterator::operator!=  
(  
    const iterator & other ) const [inline]
```

### 7.46.2.2 operator\*()

```
template<typename Container >  
Container::const_iterator qevercloud::QAssociativeContainerConstReferenceWrapper< Container  
>::iterator::operator* ( ) [inline]
```

### 7.46.2.3 operator++()

```
template<typename Container >  
iterator& qevercloud::QAssociativeContainerConstReferenceWrapper< Container >::iterator↔  
::operator++ ( ) [inline]
```

## 7.46.3 Member Data Documentation

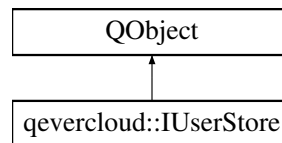
## 7.46.3.1 m\_iterator

```
template<typename Container >
Container::const_iterator qevercloud::QAssociativeContainerConstReferenceWrapper< Container
>::iterator::m_iterator
```

## 7.47 qevercloud::IUserStore Class Reference

```
#include <Services.h>
```

Inheritance diagram for qevercloud::IUserStore:



## Public Member Functions

- virtual QString [userStoreUrl](#) () const =0
- virtual void [setUserStoreUrl](#) (QString url)=0
- virtual bool [checkVersion](#) (QString clientName, qint16 edamVersionMajor=[EDAM\\_VERSION\\_MAJOR](#), qint16 edamVersionMinor=[EDAM\\_VERSION\\_MINOR](#), IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [checkVersionAsync](#) (QString clientName, qint16 edamVersionMajor=[EDAM\\_VERSION\\_MAJOR](#), qint16 edamVersionMinor=[EDAM\\_VERSION\\_MINOR](#), IRequestContextPtr ctx={})=0
- virtual BootstrapInfo [getBootstrapInfo](#) (QString locale, IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [getBootstrapInfoAsync](#) (QString locale, IRequestContextPtr ctx={})=0
- virtual AuthenticationResult [authenticateLongSession](#) (QString username, QString password, QString consumerKey, QString consumerSecret, QString deviceIdIdentifier, QString deviceDescription, bool supportsTwoFactor, IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [authenticateLongSessionAsync](#) (QString username, QString password, QString consumerKey, QString consumerSecret, QString deviceIdIdentifier, QString deviceDescription, bool supportsTwoFactor, IRequestContextPtr ctx={})=0
- virtual AuthenticationResult [completeTwoFactorAuthentication](#) (QString oneTimeCode, QString deviceIdIdentifier, QString deviceDescription, IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [completeTwoFactorAuthenticationAsync](#) (QString oneTimeCode, QString deviceIdIdentifier, QString deviceDescription, IRequestContextPtr ctx={})=0
- virtual void [revokeLongSession](#) (IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [revokeLongSessionAsync](#) (IRequestContextPtr ctx={})=0
- virtual AuthenticationResult [authenticateToBusiness](#) (IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [authenticateToBusinessAsync](#) (IRequestContextPtr ctx={})=0
- virtual User [getUser](#) (IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [getUserAsync](#) (IRequestContextPtr ctx={})=0
- virtual PublicUserInfo [getPublicUserInfo](#) (QString username, IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [getPublicUserInfoAsync](#) (QString username, IRequestContextPtr ctx={})=0
- virtual UserUrls [getUserUrls](#) (IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [getUserUrlsAsync](#) (IRequestContextPtr ctx={})=0
- virtual void [inviteToBusiness](#) (QString emailAddress, IRequestContextPtr ctx={})=0
- virtual AsyncResult \* [inviteToBusinessAsync](#) (QString emailAddress, IRequestContextPtr ctx={})=0
- virtual void [removeFromBusiness](#) (QString emailAddress, IRequestContextPtr ctx={})=0

- virtual [AsyncResult](#) \* [removeFromBusinessAsync](#) (QString emailAddress, [IRequestContextPtr](#) ctx={})=0
- virtual void [updateBusinessUserIdentifier](#) (QString oldEmailAddress, QString newEmailAddress, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [updateBusinessUserIdentifierAsync](#) (QString oldEmailAddress, QString newEmailAddress, [IRequestContextPtr](#) ctx={})=0
- virtual QList< [UserProfile](#) > [listBusinessUsers](#) ([IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [listBusinessUsersAsync](#) ([IRequestContextPtr](#) ctx={})=0
- virtual QList< [BusinessInvitation](#) > [listBusinessInvitations](#) (bool includeRequestedInvitations, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [listBusinessInvitationsAsync](#) (bool includeRequestedInvitations, [IRequestContextPtr](#) ctx={})=0
- virtual [AccountLimits](#) [getAccountLimits](#) ([ServiceLevel](#) serviceLevel, [IRequestContextPtr](#) ctx={})=0
- virtual [AsyncResult](#) \* [getAccountLimitsAsync](#) ([ServiceLevel](#) serviceLevel, [IRequestContextPtr](#) ctx={})=0

## Protected Member Functions

- [IUserStore](#) (QObject \*parent)

### 7.47.1 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
- PERMISSION\_DENIED "authenticationToken" - token does not convey sufficient privileges

### 7.47.2 Constructor & Destructor Documentation

#### 7.47.2.1 IUserStore()

```
qevercloud::IUserStore::IUserStore (
    QObject * parent ) [inline], [protected]
```

### 7.47.3 Member Function Documentation



## 7.47.3.1 authenticateLongSession()

```
virtual AuthenticationResult qevercloud::IUserStore::authenticateLongSession (
    QString username,
    QString password,
    QString consumerKey,
    QString consumerSecret,
    QString deviceIdIdentifier,
    QString deviceDescription,
    bool supportsTwoFactor,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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](https://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](https://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 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>• INVALID_AUTH "businessOnly" - the user is a business-only account</li> <li>• PERMISSION_DENIED "User.active" - user account is closed</li> <li>• PERMISSION_DENIED "User.tooManyFailuresTryAgainLater" - user has failed authentication too often</li> <li>• AUTH_EXPIRED "password" - user password is expired</li> </ul>
-----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.47.3.2 authenticateLongSessionAsync()

```
virtual AsyncResult* qevercloud::IUserStore::authenticateLongSessionAsync (
    QString username,
    QString password,
    QString consumerKey,
    QString consumerSecret,
    QString deviceIdIdentifier,
    QString deviceDescription,
    bool supportsTwoFactor,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [authenticateLongSession](#)

## 7.47.3.3 authenticateToBusiness()

```
virtual AuthenticationResult qevercloud::IUserStore::authenticateToBusiness (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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 <code>NoteStore.authenticateToSharedNotebook</code> or <code>NoteStore.authenticateToSharedNote</code> ) 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="#"><i>EDAMUserException</i></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> <li>• BUSINESS_SECURITY_LOGIN_REQUIRED "sso" - the user must complete single sign-on before authenticating to the business.</li> </ul>
------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.47.3.4 authenticateToBusinessAsync()

```
virtual AsyncResult* qevercloud::IUserStore::authenticateToBusinessAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [authenticateToBusiness](#)

## 7.47.3.5 checkVersion()

```
virtual bool qevercloud::IUserStore::checkVersion (
    QString clientName,
    qint16 edamVersionMajor = EDAM_VERSION_MAJOR,
    qint16 edamVersionMinor = EDAM_VERSION_MINOR,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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.

## 7.47.3.6 checkVersionAsync()

```
virtual AsyncResult* qevercloud::IUserStore::checkVersionAsync (
    QString clientName,
    qint16 edamVersionMajor = EDAM\_VERSION\_MAJOR,
    qint16 edamVersionMinor = EDAM\_VERSION\_MINOR,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [checkVersion](#)

## 7.47.3.7 completeTwoFactorAuthentication()

```
virtual AuthenticationResult qevercloud::IUserStore::completeTwoFactorAuthentication (
    QString oneTimeCode,
    QString deviceIdIdentifier,
    QString deviceDescription,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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 <code>UserStore.authenticate</code> or <code>UserStore.authenticateLongSession</code> 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 <code>authenticateLongSession</code> .
<i>deviceDescription</i>	See the corresponding parameter in <code>authenticateLongSession</code> .

## 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 "deviceIdIdentifier" - deviceIdIdentifier is not valid</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>
-----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.47.3.8 completeTwoFactorAuthenticationAsync()

```
virtual AsyncResult* qevercloud::IUserStore::completeTwoFactorAuthenticationAsync (
    QString oneTimeCode,
    QString deviceIdIdentifier,
    QString deviceDescription,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [completeTwoFactorAuthentication](#)

## 7.47.3.9 getAccountLimits()

```
virtual AccountLimits qevercloud::IUserStore::getAccountLimits (
    ServiceLevel serviceLevel,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Retrieve the standard account limits for a given service level. This should only be called when necessary, e.g. to determine if a higher level is available should the user upgrade, and should be cached for long periods (e.g. 30 days) as the values are not expected to fluctuate frequently.

## Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "serviceLevel" - serviceLevel is null</li> </ul>
-----------------------------------	---------------------------------------------------------------------------------------------------------

#### 7.47.3.10 `getAccountLimitsAsync()`

```
virtual AsyncResult* qevercloud::IUserStore::getAccountLimitsAsync (
    ServiceLevel serviceLevel,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getAccountLimits](#)

#### 7.47.3.11 `getBootstrapInfo()`

```
virtual BootstrapInfo qevercloud::IUserStore::getBootstrapInfo (
    QString locale,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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.

#### 7.47.3.12 `getBootstrapInfoAsync()`

```
virtual AsyncResult* qevercloud::IUserStore::getBootstrapInfoAsync (
    QString locale,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getBootstrapInfo](#)

#### 7.47.3.13 `getPublicUserInfo()`

```
virtual PublicUserInfo qevercloud::IUserStore::getPublicUserInfo (
    QString username,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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>
-----------------------------------	------------------------------------------------------------------------------------------------

## 7.47.3.14 getPublicUserInfoAsync()

```
virtual AsyncResult* qevercloud::IUserStore::getPublicUserInfoAsync (
    QString username,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getPublicUserInfo](#)

## 7.47.3.15 getUser()

```
virtual User qevercloud::IUserStore::getUser (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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.

## 7.47.3.16 getUserAsync()

```
virtual AsyncResult* qevercloud::IUserStore::getUserAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getUser](#)

## 7.47.3.17 getUserUrls()

```
virtual UserUrls qevercloud::IUserStore::getUserUrls (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns the URLs that should be used when sending requests to the service on behalf of the account represented by the provided authenticationToken.

This method isn't needed by most clients, who can retrieve the correct set of [UserUrls](#) from the [AuthenticationResult](#) returned from `UserStore::authenticateLongSession()`. This method is typically only needed to look up the correct URLs for an existing long-lived authentication token.

## 7.47.3.18 getUserUrlsAsync()

```
virtual AsyncResult* qevercloud::IUserStore::getUserUrlsAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [getUserUrls](#)

**7.47.3.19 inviteToBusiness()**

```
virtual void qevercloud::IUserStore::inviteToBusiness (
    QString emailAddress,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Invite a user to join an Evernote Business account.

Behavior will depend on the auth token.

1. auth token with privileges to manage Evernote Business membership. "External Provisioning" - The user will receive an email inviting them to join the business. They do not need to have an existing Evernote account. If the user has already been invited, a new invitation email will be sent.
2. business auth token issued to an admin user. Only for first-party clients: "Approve Invitation" - If there has been a request to invite the email, approve it. Invited user will receive email with a link to join business. "Invite User" - If no invitation for the email exists, create an approved invitation for the email. An email will be sent to the emailAddress with a link to join the caller's business. business auth token: "Request Invitation" - If no invitation exists, create a request to invite the user to the business. These requests do not count towards a business' max active user limit.

**Parameters**

<i>authenticationToken</i>	the authentication token with sufficient privileges to manage Evernote Business membership or a business auth token.
<i>emailAddress</i>	the email address of the user to invite to join the Evernote Business account.

**Exceptions**

<i>EDAMUserException</i>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "email" - if no email address was provided</li> <li>• BAD_DATA_FORMAT "email" - if the email address is not well formed</li> <li>• DATA_CONFLICT "BusinessUser.email" - if there is already a user in the business whose business email address matches the specified email address.</li> <li>• LIMIT_REACHED "Business.maxActiveUsers" - if the business has reached its user limit.</li> </ul>
--------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**7.47.3.20 inviteToBusinessAsync()**

```
virtual AsyncResult* qevercloud::IUserStore::inviteToBusinessAsync (
    QString emailAddress,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [inviteToBusiness](#)



### 7.47.3.21 listBusinessInvitations()

```
virtual QList<BusinessInvitation> qevercloud::IUserStore::listBusinessInvitations (
    bool includeRequestedInvitations,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns a list of outstanding invitations to join an Evernote Business account.

Only outstanding invitations are returned by this function. Users who have accepted an invitation and joined a business are listed using listBusinessUsers.

#### Parameters

<i>authenticationToken</i>	An authentication token with sufficient privileges to manage Evernote Business membership.
<i>includeRequestedInvitations</i>	If true, invitations with a status of <a href="#">BusinessInvitationStatus.REQUESTED</a> will be included in the returned list. If false, only invitations with a status of <a href="#">BusinessInvitationStatus.APPROVED</a> will be included.

### 7.47.3.22 listBusinessInvitationsAsync()

```
virtual AsyncResult* qevercloud::IUserStore::listBusinessInvitationsAsync (
    bool includeRequestedInvitations,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listBusinessInvitations](#)

### 7.47.3.23 listBusinessUsers()

```
virtual QList<UserProfile> qevercloud::IUserStore::listBusinessUsers (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Returns a list of active business users in a given business.

Clients are required to cache this information and re-fetch no more than once per day or when they encountered a user ID or username that was not known to them.

To avoid excessive look ups, clients should also track user IDs and usernames that belong to users who are not in the business, since they will not be included in the result.

I.e., when a client encounters a previously unknown user ID as a note's creator, it may query listBusinessUsers to find information about this user. If the user is not in the resulting list, the client should track that fact and not re-query the service the next time that it sees this user on a note.

#### Parameters

<i>authenticationToken</i>	A business authentication token returned by <a href="#">authenticateToBusiness</a> or with sufficient privileges to manage Evernote Business membership.
----------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------

#### 7.47.3.24 listBusinessUsersAsync()

```
virtual AsyncResult* qevercloud::IUserStore::listBusinessUsersAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [listBusinessUsers](#)

#### 7.47.3.25 removeFromBusiness()

```
virtual void qevercloud::IUserStore::removeFromBusiness (
    QString emailAddress,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Remove a user from an Evernote Business account. Once removed, the user will no longer be able to access content within the Evernote Business account.

The email address of the user to remove from the business must match the email address used to invite a user to join the business via `UserStore.inviteToBusiness`. This function will only remove users who were invited by external provisioning

##### Parameters

<i>authenticationToken</i>	An authentication token with sufficient privileges to manage Evernote Business membership.
<i>emailAddress</i>	The email address of the user to remove from the Evernote Business account.

##### Exceptions

<a href="#">EDAMUserException</a>	<ul style="list-style-type: none"> <li>DATA_REQUIRED "email" - if no email address was provided</li> <li>BAD_DATA_FORMAT "email" - The email address is not well formed</li> </ul>
<a href="#">EDAMNotFoundException</a>	<ul style="list-style-type: none"> <li>"email" - If there is no user with the specified email address in the business or that user was not invited via external provisioning.</li> </ul>

#### 7.47.3.26 removeFromBusinessAsync()

```
virtual AsyncResult* qevercloud::IUserStore::removeFromBusinessAsync (
    QString emailAddress,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [removeFromBusiness](#)

## 7.47.3.27 revokeLongSession()

```
virtual void qevercloud::IUserStore::revokeLongSession (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

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="#"><i>EDAMUserException</i></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>
------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.47.3.28 revokeLongSessionAsync()

```
virtual AsyncResult* qevercloud::IUserStore::revokeLongSessionAsync (
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [`revokeLongSession`](#)

## 7.47.3.29 setUserStoreUrl()

```
virtual void qevercloud::IUserStore::setUserStoreUrl (
    QString url ) [pure virtual]
```

## 7.47.3.30 updateBusinessUserIdentifier()

```
virtual void qevercloud::IUserStore::updateBusinessUserIdentifier (
    QString oldEmailAddress,
    QString newEmailAddress,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Update the email address used to uniquely identify an Evernote Business user.

This will update the identifier for a user who was previously invited using `inviteToBusiness`, ensuring that caller and the Evernote service maintain an agreed-upon identifier for a specific user.

For example, the following sequence of calls would invite a user to join a business, update their email address, and then remove the user from the business using the updated email address.

```
inviteToBusiness("foo@bar.com") updateBusinessUserIdentifier("foo@bar.com", "baz@bar.com") removeFromBusiness("baz@bar.com")
```

#### Parameters

<i>authenticationToken</i>	An authentication token with sufficient privileges to manage Evernote Business membership.
<i>oldEmailAddress</i>	The existing email address used to uniquely identify the user.
<i>newEmailAddress</i>	The new email address used to uniquely identify the user.

#### Exceptions

<a href="#"><i>EDAMUserException</i></a>	<ul style="list-style-type: none"> <li>• DATA_REQUIRED "oldEmailAddress" - No old email address was provided</li> <li>• DATA_REQUIRED "newEmailAddress" - No new email address was provided</li> <li>• BAD_DATA_FORMAT "oldEmailAddress" - The old email address is not well formed</li> <li>• BAD_DATA_FORMAT "newEmailAddress" - The new email address is not well formed</li> <li>• DATA_CONFLICT "oldEmailAddress" - The old and new email addresses were the same</li> <li>• DATA_CONFLICT "newEmailAddress" - There is already an invitation or registered user with the provided new email address.</li> <li>• DATA_CONFLICT "invitation.externallyProvisioned" - The user identified by oldEmailAddress was not added via UserStore.inviteToBusiness and therefore cannot be updated.</li> </ul>
<a href="#"><i>EDAMNotFoundException</i></a>	<ul style="list-style-type: none"> <li>• "oldEmailAddress" - If there is no user or invitation with the specified oldEmailAddress in the business.</li> </ul>

#### 7.47.3.31 updateBusinessUserIdentifierAsync()

```
virtual AsyncResult* qevercloud::IUserStore::updateBusinessUserIdentifierAsync (
    QString oldEmailAddress,
    QString newEmailAddress,
    IRequestContextPtr ctx = {} ) [pure virtual]
```

Asynchronous version of [`updateBusinessUserIdentifier`](#)

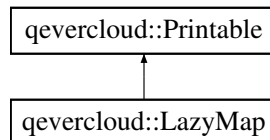
## 7.47.3.32 userStoreUrl()

```
virtual QString qevercloud::IUserStore::userStoreUrl ( ) const [pure virtual]
```

## 7.48 qevercloud::LazyMap Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::LazyMap:



## Public Types

- using [FullMap](#) = QMap< QString, QString >

## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [LazyMap](#) &other) const
- bool [operator!=](#) (const [LazyMap](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QSet](#)< QString > > [keysOnly](#)
- [Optional](#)< QMap< QString, QString > > [fullMap](#)

## Properties

- [Optional](#) [QSet](#)
- [Optional](#)< [FullMap](#) > [fullMap](#)

## 7.48.1 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.

## 7.48.2 Member Typedef Documentation

### 7.48.2.1 FullMap

```
using qevercloud::LazyMap::FullMap = QMap<QString, QString>
```

## 7.48.3 Member Function Documentation

### 7.48.3.1 operator!=(())

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

### 7.48.3.2 operator==(())

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

### 7.48.3.3 print()

```
virtual void qevercloud::LazyMap::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.48.4 Member Data Documentation

### 7.48.4.1 fullMap

```
Optional<QMap<QString, QString> > qevercloud::LazyMap::fullMap
```

The complete map, including all keys and values.

#### 7.48.4.2 keysOnly

```
Optional<QSet<QString> > qevercloud::LazyMap::keysOnly
```

The set of keys for the map. This field is ignored by the server when set.

#### 7.48.4.3 localData

```
EverCloudLocalData qevercloud::LazyMap::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.48.5 Property Documentation

#### 7.48.5.1 fullMap

```
Optional<FullMap> qevercloud::LazyMap::fullMap
```

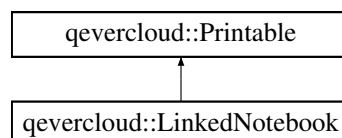
#### 7.48.5.2 QSet

```
Optional qevercloud::LazyMap::QSet
```

## 7.49 qevercloud::LinkedNotebook Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::LinkedNotebook:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [LinkedNotebook](#) &other) const
- bool [operator!=](#) (const [LinkedNotebook](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- [Optional](#)< [QString](#) > `shareName`
- [Optional](#)< [QString](#) > `username`
- [Optional](#)< [QString](#) > `shardId`
- [Optional](#)< [QString](#) > `sharedNotebookGlobalId`
- [Optional](#)< [QString](#) > `uri`
- [Optional](#)< [Guid](#) > `guid`
- [Optional](#)< [qint32](#) > `updateSequenceNum`
- [Optional](#)< [QString](#) > `noteStoreUrl`
- [Optional](#)< [QString](#) > `webApiUrlPrefix`
- [Optional](#)< [QString](#) > `stack`
- [Optional](#)< [qint32](#) > `businessId`

### 7.49.1 Detailed Description

A link in a user's account that refers them to a public or individual shared notebook in another user's account.

### 7.49.2 Member Function Documentation

#### 7.49.2.1 `operator!=( )`

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

#### 7.49.2.2 `operator==( )`

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

#### 7.49.2.3 `print( )`

```
virtual void qevercloud::LinkedNotebook::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.49.3 Member Data Documentation



#### 7.49.3.1 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.

#### 7.49.3.2 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

#### 7.49.3.3 localData

`EverCloudLocalData qevercloud::LinkedNotebook::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.49.3.4 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.

#### 7.49.3.5 shardId

`Optional< QString > qevercloud::LinkedNotebook::shardId`

The shard ID of the notebook if the notebook is not public.

uri The identifier of the public notebook.

#### 7.49.3.6 sharedNotebookGlobalId

`Optional< QString > qevercloud::LinkedNotebook::sharedNotebookGlobalId`

The globally unique identifier (globalId) of the shared notebook that corresponds to the share key, or the GUID of the [Notebook](#) that the linked notebook refers to. This field must be filled in with the [SharedNotebook.globalId](#) or [Notebook.GUID](#) value when creating new LinkedNotebooks. This field replaces the deprecated "shareKey" field.

#### 7.49.3.7 shareName

`Optional< QString > qevercloud::LinkedNotebook::shareName`

The display name of the shared notebook. The link owner can change this.

#### 7.49.3.8 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.

#### 7.49.3.9 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.

#### 7.49.3.10 uri

`Optional< QString > qevercloud::LinkedNotebook::uri`

NOT DOCUMENTED

#### 7.49.3.11 username

`Optional< QString > qevercloud::LinkedNotebook::username`

The username of the user who owns the shared or public notebook.

#### 7.49.3.12 webApiUrlPrefix

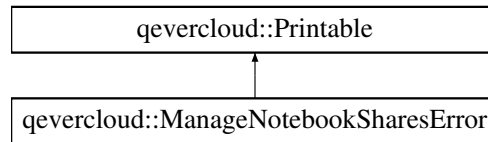
`Optional< QString > qevercloud::LinkedNotebook::webApiUrlPrefix`

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.

## 7.50 qevercloud::ManageNotebookSharesError Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::ManageNotebookSharesError:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [ManageNotebookSharesError](#) &other) const
- bool [operator!=](#) (const [ManageNotebookSharesError](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [UserIdentity](#) > [userIdentity](#)
- [Optional](#)< [EDAMUserException](#) > [userException](#)
- [Optional](#)< [EDAMNotFoundException](#) > [notFoundException](#)

### 7.50.1 Detailed Description

A structure to capture certain errors that occurred during a call to `manageNotebookShares`. That method can be run best-effort, meaning that some change requests can be applied while others fail. **Note** that some errors such as system errors will still fail the entire transaction regardless of running best effort. When some change requests do not succeed, the error conditions are captured in instances of this class, captured by the identity of the share relationship and one of the exception fields.

### 7.50.2 Member Function Documentation

#### 7.50.2.1 `operator!=()`

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

### 7.50.2.2 operator==( )

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

### 7.50.2.3 print()

```
virtual void qevercloud::ManageNotebookSharesError::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.50.3 Member Data Documentation

### 7.50.3.1 localData

[EverCloudLocalData](#) qevercloud::ManageNotebookSharesError::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.50.3.2 notFoundException

[Optional](#)< [EDAMNotFoundException](#) > qevercloud::ManageNotebookSharesError::notFoundException

If the error is represented as an [EDAMNotFoundException](#) that would have otherwise been thrown without best-effort execution. Only one exception field will be set.

### 7.50.3.3 userException

[Optional](#)< [EDAMUserException](#) > qevercloud::ManageNotebookSharesError::userException

If the error is represented as an [EDAMUserException](#) that would have otherwise been thrown without best-effort execution. Only one exception field will be set.

### 7.50.3.4 userIdentity

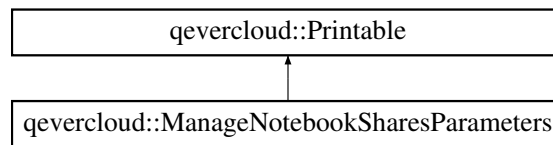
[Optional](#)< [UserIdentity](#) > qevercloud::ManageNotebookSharesError::userIdentity

The identity of the share relationship whose update encountered an error.

## 7.51 qevercloud::ManageNotebookSharesParameters Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::ManageNotebookSharesParameters:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [ManageNotebookSharesParameters](#) &other) const
- bool [operator!=](#) (const [ManageNotebookSharesParameters](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) localData
- [Optional](#)< QString > [notebookGuid](#)
- [Optional](#)< QString > [inviteMessage](#)
- [Optional](#)< QList< [MemberShareRelationship](#) > > [membershipsToUpdate](#)
- [Optional](#)< QList< [InvitationShareRelationship](#) > > [invitationsToCreateOrUpdate](#)
- [Optional](#)< QList< [UserIdentity](#) > > [unshares](#)

### Properties

- [Optional](#) QList

#### 7.51.1 Detailed Description

A structure that captures parameters used by clients to manage the shares for a given notebook via the [manageNotebookShares](#) method.

#### 7.51.2 Member Function Documentation

##### 7.51.2.1 [operator!=\(\)](#)

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

### 7.51.2.2 operator==( )

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

### 7.51.2.3 print( )

```
virtual void qevercloud::ManageNotebookSharesParameters::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.51.3 Member Data Documentation

### 7.51.3.1 invitationsToCreateOrUpdate

```
Optional<QList<InvitationShareRelationship> > qevercloud::ManageNotebookSharesParameters←
::invitationsToCreateOrUpdate
```

The list of invitations to update, as matched by the identity field of the [InvitationShareRelationship](#) instances, or to create if an existing invitation does not exist. This field is not intended to be the full set of invitations on the notebook and should only include those invitations that you wish to create or update. [Note](#) that your invitation could convert into a membership via a service-supported auto-join operation. This happens, for example, when you use an invitation with an Evernote UserID type for a recipient who is a member of the business to which the notebook belongs. [Note](#) that to discover the user IDs for business members, the sharer must also be part of the business.

### 7.51.3.2 inviteMessage

```
Optional< QString > qevercloud::ManageNotebookSharesParameters::inviteMessage
```

If the service sends a message to invitees, this parameter will be used to form the actual message that is sent.

### 7.51.3.3 localData

```
EverCloudLocalData qevercloud::ManageNotebookSharesParameters::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.51.3.4 membershipsToUpdate

```
Optional<QList<MemberShareRelationship> > qevercloud::ManageNotebookSharesParameters::memberships←
ToUpdate
```

The list of existing memberships to update. This field is not intended to be the full set of memberships for the notebook and should only include those already-existing memberships that you actually want to change. If you want to remove shares, see the unshares fields. If you want to create a membership, i.e. auto-join a business user, you can do this via the invitationsToCreateOrUpdate field using an Evernote UserID of a fellow business member (the created invitation is automatically joined by the service, so the client is creating an invitation, not a membership).

## 7.51.3.5 notebookGuid

`Optional< QString > qevercloud::ManageNotebookSharesParameters::notebookGuid`

The GUID of the notebook whose shares are being managed.

## 7.51.3.6 unshares

`Optional< QList< UserIdentity > > qevercloud::ManageNotebookSharesParameters::unshares`

The list of share relationships to expunge from the service. If the user identity is for an Evernote UserID, then matching invitations or memberships will be removed. If it's an e-mail, then e-mail based shared notebook invitations will be removed. If it's for an [Identity](#) ID, then any invitations that match the identity (by identity ID or user ID or e-mail for legacy invitations) will be removed.

## 7.51.4 Property Documentation

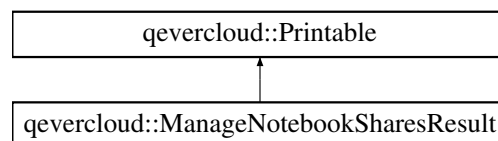
## 7.51.4.1 QList

`Optional qevercloud::ManageNotebookSharesParameters::QList`

## 7.52 qevercloud::ManageNotebookSharesResult Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::ManageNotebookSharesResult:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [ManageNotebookSharesResult](#) &other) const
- bool [operator!=](#) (const [ManageNotebookSharesResult](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- `Optional< QList< ManageNotebookSharesError > >` [errors](#)

## Properties

- [Optional QList](#)

### 7.52.1 Detailed Description

The return value of a call to the `manageNotebookShares` method.

### 7.52.2 Member Function Documentation

#### 7.52.2.1 `operator!=()`

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

#### 7.52.2.2 `operator==()`

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

#### 7.52.2.3 `print()`

```
virtual void qevercloud::ManageNotebookSharesResult::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.52.3 Member Data Documentation

#### 7.52.3.1 `errors`

```
Optional<QList<ManageNotebookSharesError> > qevercloud::ManageNotebookSharesResult::errors
```

If the method completed without throwing exceptions, some errors might still have occurred, and in that case, this field will contain the list of those errors the occurred.



## 7.52.3.2 localData

`EverCloudLocalData` `qevercloud::ManageNotebookSharesResult::localData`

See the declaration of `EverCloudLocalData` for details

## 7.52.4 Property Documentation

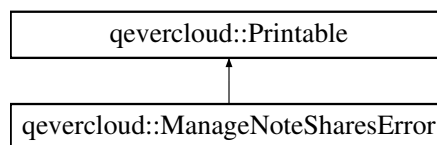
## 7.52.4.1 QList

`Optional` `qevercloud::ManageNotebookSharesResult::QList`

## 7.53 qevercloud::ManageNoteSharesError Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::ManageNoteSharesError`:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `ManageNoteSharesError` &other) const
- bool `operator!=` (const `ManageNoteSharesError` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< IdentityID >` `identityID`
- `Optional< UserID >` `userID`
- `Optional< EDAMUserException >` `userException`
- `Optional< EDAMNotFoundException >` `notFoundException`

## 7.53.1 Detailed Description

Captures errors that occur during a call to `manageNoteShares`. That function can be run best-effort, meaning that some change requests can be applied while others fail. **Note** that some errors such as system exceptions may still cause the entire call to fail.

Only one of the two ID fields will be set on a given error.

Only one of the two exception fields will be set on a given error.

## 7.53.2 Member Function Documentation

### 7.53.2.1 operator!=(())

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

### 7.53.2.2 operator==(())

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

### 7.53.2.3 print()

```
virtual void qevercloud::ManageNoteSharesError::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.53.3 Member Data Documentation

### 7.53.3.1 identityID

```
Optional< IdentityID > qevercloud::ManageNoteSharesError::identityID
```

The identity ID of an outstanding invitation that was not updated due to the error.

### 7.53.3.2 localData

```
EverCloudLocalData qevercloud::ManageNoteSharesError::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.53.3.3 notFoundException

```
Optional< EDAMNotFoundException > qevercloud::ManageNoteSharesError::notFoundException
```

If the error is represented as an [EDAMNotFoundException](#) that would have otherwise been thrown without best-effort execution. The identifier field of the exception will be either "Identity.id" or "User.id", indicating that no existing share could be found for the specified recipient.

## 7.53.3.4 userException

`Optional< EDAMUserException > qevercloud::ManageNoteSharesError::userException`

If the error is represented as an [EDAMUserException](#) that would have otherwise been thrown without best-effort execution.

## 7.53.3.5 userID

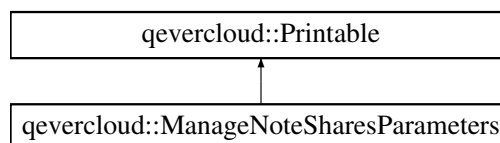
`Optional< UserID > qevercloud::ManageNoteSharesError::userID`

The user ID of an existing membership that was not updated due to the error.

## 7.54 qevercloud::ManageNoteSharesParameters Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::ManageNoteSharesParameters:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [ManageNoteSharesParameters](#) &other) const
- bool [operator!=](#) (const [ManageNoteSharesParameters](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- `Optional< QString >` [noteGuid](#)
- `Optional< QList< NoteMemberShareRelationship > >` [membershipsToUpdate](#)
- `Optional< QList< NoteInvitationShareRelationship > >` [invitationsToUpdate](#)
- `Optional< QList< UserID > >` [membershipsToUnshare](#)
- `Optional< QList< IdentityID > >` [invitationsToUnshare](#)

## Properties

- `Optional` [QList](#)

### 7.54.1 Detailed Description

Captures parameters used by clients to manage the shares for a given note via the `manageNoteShares` function. This is used only to manage the existing memberships and invitations for a note. To invite a new recipient, use `NoteStore.createOrUpdateSharedNotes`.

The only field of an existing membership or invitation that can be updated by this function is the share privilege.

### 7.54.2 Member Function Documentation

#### 7.54.2.1 `operator!=(())`

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

#### 7.54.2.2 `operator==(())`

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

#### 7.54.2.3 `print()`

```
virtual void qevercloud::ManageNoteSharesParameters::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.54.3 Member Data Documentation

#### 7.54.3.1 `invitationsToUnshare`

```
Optional<QList<IdentityID> > qevercloud::ManageNoteSharesParameters::invitationsToUnshare
```

A list of outstanding invitations to expunge from the service.

### 7.54.3.2 invitationsToUpdate

`Optional<QList<NoteInvitationShareRelationship> > qevercloud::ManageNoteSharesParameters::invitationsToUpdate`

The list of outstanding invitations to update, as matched by the identity field of the `NoteInvitationShareRelationship` instances. This field is not meant to be the full set of invitations for the note. Clients should only include those existing invitations that they wish to modify.

### 7.54.3.3 localData

`EverCloudLocalData qevercloud::ManageNoteSharesParameters::localData`

See the declaration of [EverCloudLocalData](#) for details

### 7.54.3.4 membershipsToUnshare

`Optional<QList<UserID> > qevercloud::ManageNoteSharesParameters::membershipsToUnshare`

A list of existing memberships to expunge from the service.

### 7.54.3.5 membershipsToUpdate

`Optional<QList<NoteMemberShareRelationship> > qevercloud::ManageNoteSharesParameters::membershipsToUpdate`

A list of existing memberships to update. This field is not meant to be the full set of memberships for the note. Clients should only include those existing memberships that they wish to modify. To remove an existing membership, see the `unshares` field.

### 7.54.3.6 noteGuid

`Optional<QString> qevercloud::ManageNoteSharesParameters::noteGuid`

The GUID of the note whose shares are being managed.

## 7.54.4 Property Documentation

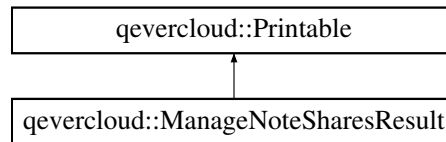
### 7.54.4.1 QList

`Optional qevercloud::ManageNoteSharesParameters::QList`

## 7.55 qevercloud::ManageNoteSharesResult Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::ManageNoteSharesResult:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [ManageNoteSharesResult](#) &other) const
- bool [operator!=](#) (const [ManageNoteSharesResult](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QList](#)< [ManageNoteSharesError](#) > > [errors](#)

### Properties

- [Optional](#) [QList](#)

### 7.55.1 Detailed Description

The return value of a call to the `manageNoteShares` function.

### 7.55.2 Member Function Documentation

#### 7.55.2.1 `operator!=()`

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

### 7.55.2.2 operator==( )

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

### 7.55.2.3 print( )

```
virtual void qevercloud::ManageNoteSharesResult::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.55.3 Member Data Documentation

### 7.55.3.1 errors

```
Optional<QList<ManageNoteSharesError> > qevercloud::ManageNoteSharesResult::errors
```

If the call succeeded without throwing an exception, some errors might still have occurred. In that case, this field will contain the list of errors.

### 7.55.3.2 localData

```
EverCloudLocalData qevercloud::ManageNoteSharesResult::localData
```

See the declaration of [EverCloudLocalData](#) for details

## 7.55.4 Property Documentation

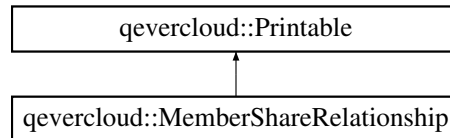
### 7.55.4.1 QList

```
Optional qevercloud::ManageNoteSharesResult::QList
```

## 7.56 qevercloud::MemberShareRelationship Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::MemberShareRelationship:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [MemberShareRelationship](#) &other) const
- bool [operator!=](#) (const [MemberShareRelationship](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QString](#) > [displayName](#)
- [Optional](#)< [UserID](#) > [recipientUserId](#)
- [Optional](#)< [ShareRelationshipPrivilegeLevel](#) > [bestPrivilege](#)
- [Optional](#)< [ShareRelationshipPrivilegeLevel](#) > [individualPrivilege](#)
- [Optional](#)< [ShareRelationshipRestrictions](#) > [restrictions](#)
- [Optional](#)< [UserID](#) > [sharerUserId](#)

### 7.56.1 Detailed Description

Describes the association between a [Notebook](#) and an Evernote [User](#) who is a member of that notebook.

### 7.56.2 Member Function Documentation

#### 7.56.2.1 [operator!=\(\)](#)

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

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

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



### 7.56.2.3 print()

```
virtual void qevercloud::MemberShareRelationship::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.56.3 Member Data Documentation

### 7.56.3.1 bestPrivilege

```
Optional< ShareRelationshipPrivilegeLevel > qevercloud::MemberShareRelationship::bestPrivilege
```

The privilege at which the member can access the notebook, which is the best privilege granted either individually or to a group to which a member belongs, such as a business. This field is used by the service to convey information to the user, so clients should treat it as read-only.

### 7.56.3.2 displayName

```
Optional< QString > qevercloud::MemberShareRelationship::displayName
```

The string that clients should show to users to represent this member.

### 7.56.3.3 individualPrivilege

```
Optional< ShareRelationshipPrivilegeLevel > qevercloud::MemberShareRelationship::individual↔
Privilege
```

The individually granted privilege for the member, which does not take GROUP privileges into account. This value may be unset if only a group-assigned privilege has been granted to the member. This value can be managed by others with sufficient rights using the `manageNotebookShares` method. The valid values that clients should present to users for selection are given via the the 'restrictions' field.

### 7.56.3.4 localData

```
EverCloudLocalData qevercloud::MemberShareRelationship::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.56.3.5 recipientUserId

```
Optional< UserID > qevercloud::MemberShareRelationship::recipientUserId
```

The Evernote [User](#) ID of the recipient of this notebook share.

### 7.56.3.6 restrictions

```
Optional< ShareRelationshipRestrictions > qevercloud::MemberShareRelationship::restrictions
```

The restrictions on which privileges may be individually assigned to the recipient of this share relationship.

### 7.56.3.7 sharerUserId

```
Optional< UserID > qevercloud::MemberShareRelationship::sharerUserId
```

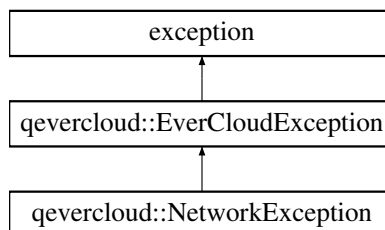
The user id of the user who most recently shared the notebook to this user. This field is currently unset for a [MemberShareRelationship](#) created by joining a notebook that has been published to the business (MemberShareRelationships where the individual privilege is unset). This field is used by the service to convey information to the user, so clients should treat it as read-only.

## 7.57 qevercloud::NetworkException Class Reference

The [NetworkException](#) class represents QNetworkReply level errors.

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::NetworkException:



### Public Member Functions

- [NetworkException](#) ()
- [NetworkException](#) (QNetworkReply::NetworkError error)
- [NetworkException](#) (QNetworkReply::NetworkError error, QString message)
- virtual [~NetworkException](#) () noexcept override
- bool [operator==](#) (const [NetworkException](#) &other) const
- bool [operator!=](#) (const [NetworkException](#) &other) const
- QNetworkReply::NetworkError [type](#) () const
- const char \* [what](#) () const noexcept override
- virtual [EverCloudExceptionDataPtr exceptionData](#) () const override

### Protected Attributes

- QNetworkReply::NetworkError [m\\_type](#)

### 7.57.1 Detailed Description

The [NetworkException](#) class represents QNetworkReply level errors.

### 7.57.2 Constructor & Destructor Documentation

#### 7.57.2.1 NetworkException() [1/3]

```
qevercloud::NetworkException::NetworkException ( )
```

#### 7.57.2.2 NetworkException() [2/3]

```
qevercloud::NetworkException::NetworkException (
    QNetworkReply::NetworkError error )
```

#### 7.57.2.3 NetworkException() [3/3]

```
qevercloud::NetworkException::NetworkException (
    QNetworkReply::NetworkError error,
    QString message )
```

#### 7.57.2.4 ~NetworkException()

```
virtual qevercloud::NetworkException::~~NetworkException ( ) [override], [virtual], [noexcept]
```

### 7.57.3 Member Function Documentation

#### 7.57.3.1 exceptionData()

```
virtual EverCloudExceptionDataPtr qevercloud::NetworkException::exceptionData ( ) const [override],
[virtual]
```

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

### 7.57.3.2 operator!=(())

```
bool qevercloud::NetworkException::operator!= (
    const NetworkException & other ) const
```

### 7.57.3.3 operator==(())

```
bool qevercloud::NetworkException::operator== (
    const NetworkException & other ) const
```

### 7.57.3.4 type()

```
QNetworkReply::NetworkError qevercloud::NetworkException::type ( ) const
```

### 7.57.3.5 what()

```
const char* qevercloud::NetworkException::what ( ) const [override], [virtual], [noexcept]
```

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

## 7.57.4 Member Data Documentation

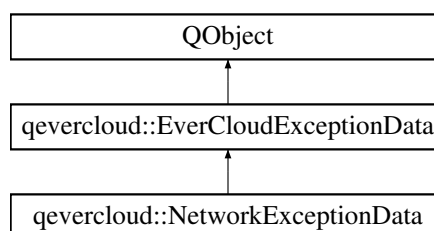
### 7.57.4.1 m\_type

```
QNetworkReply::NetworkError qevercloud::NetworkException::m_type [protected]
```

## 7.58 qevercloud::NetworkExceptionData Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::NetworkExceptionData:



## Public Member Functions

- [NetworkExceptionData](#) (QString error, QNetworkReply::NetworkError type)
- virtual void [throwException](#) () const override

## Protected Attributes

- QNetworkReply::NetworkError [m\\_type](#)

## Additional Inherited Members

### 7.58.1 Detailed Description

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

### 7.58.2 Constructor & Destructor Documentation

#### 7.58.2.1 NetworkExceptionData()

```
qevercloud::NetworkExceptionData::NetworkExceptionData (
    QString error,
    QNetworkReply::NetworkError type ) [explicit]
```

### 7.58.3 Member Function Documentation

#### 7.58.3.1 throwException()

```
virtual void qevercloud::NetworkExceptionData::throwException ( ) const [override], [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](#).

### 7.58.4 Member Data Documentation

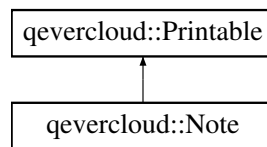
### 7.58.4.1 m\_type

QNetworkReply::NetworkError qevercloud::NetworkExceptionData::m\_type [protected]

## 7.59 qevercloud::Note Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::Note:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [Note](#) &other) const
- bool [operator!=](#) (const [Note](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [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](#)
- [Optional< QList< SharedNote > >](#) [sharedNotes](#)
- [Optional< NoteRestrictions >](#) [restrictions](#)
- [Optional< NoteLimits >](#) [limits](#)

### Properties

- [Optional QList](#)

### 7.59.1 Detailed Description

Represents a single note in the user's account.

### 7.59.2 Member Function Documentation

#### 7.59.2.1 operator!=(())

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

#### 7.59.2.2 operator==(())

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

#### 7.59.2.3 print()

```
virtual void qevercloud::Note::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.59.3 Member Data Documentation

#### 7.59.3.1 active

```
Optional< bool > qevercloud::Note::active
```

If the note is available for normal actions and viewing, this flag will be set to true.

#### 7.59.3.2 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.

### 7.59.3.3 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

### 7.59.3.4 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)

### 7.59.3.5 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](#).

### 7.59.3.6 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.

### 7.59.3.7 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.

### 7.59.3.8 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.create←  
Note()`

Length: EDAM\_GUID\_LEN\_MIN - EDAM\_GUID\_LEN\_MAX

Regex: EDAM\_GUID\_REGEX



#### 7.59.3.9 limits

`Optional< NoteLimits > qevercloud::Note::limits`

NOT DOCUMENTED

#### 7.59.3.10 localData

`EverCloudLocalData qevercloud::Note::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.59.3.11 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

#### 7.59.3.12 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

#### 7.59.3.13 restrictions

`Optional< NoteRestrictions > qevercloud::Note::restrictions`

If this field is set, the user has note-level permissions that may differ from their notebook-level permissions. In this case, the restrictions structure specifies a set of restrictions limiting the actions that a user may take on the note based on their note-level permissions. If this field is unset, then there are no note-specific restrictions. However, a client may still be limited based on the user's notebook permissions.

#### 7.59.3.14 sharedNotes

`Optional< QList< SharedNote > > qevercloud::Note::sharedNotes`

The list of recipients with whom this note has been shared. This field will be unset if the caller has access to the note via the containing notebook, but does not have activity feed permission for that notebook. This field is read-only. Clients may not make changes to a note's sharing state via this field.

#### 7.59.3.15 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

#### 7.59.3.16 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).

#### 7.59.3.17 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

#### 7.59.3.18 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.

#### 7.59.3.19 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.

### 7.59.4 Property Documentation

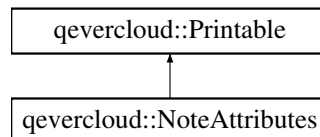
## 7.59.4.1 QList

`Optional qevercloud::Note::QList`

## 7.60 qevercloud::NoteAttributes Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteAttributes:



## Public Types

- using `Classifications` = QMap< QString, QString >

## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NoteAttributes` &other) const
- bool `operator!=` (const `NoteAttributes` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `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`
- `Optional< bool >` `sharedWithBusiness`
- `Optional< Guid >` `conflictSourceNoteGuid`
- `Optional< qint32 >` `noteTitleQuality`

## Properties

- [Optional](#) < [Classifications](#) > [classifications](#)

### 7.60.1 Detailed Description

The list of optional attributes that can be stored on a note.

### 7.60.2 Member Typedef Documentation

#### 7.60.2.1 Classifications

```
using qevercloud::NoteAttributes::Classifications = QMap<QString, QString>
```

### 7.60.3 Member Function Documentation

#### 7.60.3.1 operator"!="()

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

#### 7.60.3.2 operator==( )

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

#### 7.60.3.3 print()

```
virtual void qevercloud::NoteAttributes::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.60.4 Member Data Documentation

#### 7.60.4.1 altitude

`Optional< double > qevercloud::NoteAttributes::altitude`

the altitude where the note was taken

#### 7.60.4.2 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

#### 7.60.4.3 author

`Optional< QString > qevercloud::NoteAttributes::author`

the author of the content of the note

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.60.4.4 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_`.

#### 7.60.4.5 conflictSourceNoteGuid

`Optional< Guid > qevercloud::NoteAttributes::conflictSourceNoteGuid`

If set, this specifies the GUID of a note that caused a sync conflict resulting in the creation of a duplicate note. The duplicated note contains the user's changes that could not be applied as a result of the sync conflict, and uses the `conflictSourceNoteGuid` field to specify the note that caused the conflict. This allows clients to provide a customized user experience for note conflicts.

#### 7.60.4.6 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

#### 7.60.4.7 creatorId

```
Optional< UserID > qevercloud::NoteAttributes::creatorId
```

The numeric user ID of the user who originally created the note.

#### 7.60.4.8 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.

#### 7.60.4.9 lastEditorId

```
Optional< UserID > qevercloud::NoteAttributes::lastEditorId
```

The numeric user ID of the user described in lastEditedBy.

#### 7.60.4.10 latitude

```
Optional< double > qevercloud::NoteAttributes::latitude
```

the latitude where the note was taken

#### 7.60.4.11 localData

`EverCloudLocalData qevercloud::NoteAttributes::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.60.4.12 longitude

`Optional< double > qevercloud::NoteAttributes::longitude`

the longitude where the note was taken

#### 7.60.4.13 noteTitleQuality

`Optional< qint32 > qevercloud::NoteAttributes::noteTitleQuality`

If set, this specifies that the note's title was automatically generated and indicates the likelihood that the generated title is useful for display to the user. If not set, the note's title was manually entered by the user.

Clients MUST set this attribute to one of the following values when the corresponding note's title was not manually entered by the user: EDAM\_NOTE\_TITLE\_QUALITY\_UNTITLED, EDAM\_NOTE\_TITLE\_QUALITY\_LOW, EDAM\_NOTE\_TITLE\_QUALITY\_MEDIUM or EDAM\_NOTE\_TITLE\_QUALITY\_HIGH.

When a user edits a note's title, clients MUST unset this value.

#### 7.60.4.14 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.

#### 7.60.4.15 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.

#### 7.60.4.16 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 reminder↔ DoneTime or reminderTime is set but a reminderOrder is not set, the server will fill in the current server time for the reminderOrder field.

#### 7.60.4.17 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.

#### 7.60.4.18 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.

#### 7.60.4.19 sharedWithBusiness

```
Optional< bool > qevercloud::NoteAttributes::sharedWithBusiness
```

When this flag is set on a business note, any user in that business may view the note if they request it by GUID. This field is read-only by clients. The server will ignore all values set by clients into this field.

To share a note with the business, use `NoteStore.shareNoteWithBusiness` and to stop sharing a note with the business, use `NoteStore.stopSharingNoteWithBusiness`.

#### 7.60.4.20 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



## 7.60.4.21 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

## 7.60.4.22 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

## 7.60.4.23 subjectDate

`Optional< Timestamp > qevercloud::NoteAttributes::subjectDate`

time that the note refers to

## 7.60.5 Property Documentation

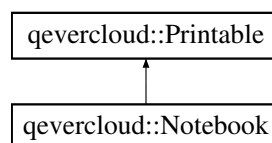
## 7.60.5.1 classifications

`Optional<Classifications> qevercloud::NoteAttributes::classifications`

## 7.61 qevercloud::Notebook Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::Notebook:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `Notebook` &other) const
- bool `operator!=` (const `Notebook` &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- [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`
- [Optional](#)< [NotebookRecipientSettings](#) > `recipientSettings`

## Properties

- [Optional](#) [QList](#)

### 7.61.1 Detailed Description

A unique container for a set of notes.

### 7.61.2 Member Function Documentation

#### 7.61.2.1 `operator!=(())`

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

#### 7.61.2.2 `operator==(())`

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

### 7.61.2.3 print()

```
virtual void qevercloud::Notebook::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.61.3 Member Data Documentation

### 7.61.3.1 businessNotebook

```
Optional< BusinessNotebook > qevercloud::Notebook::businessNotebook
```

If the notebook is part of a business account and has been shared with the entire business, this will contain sharing information. 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 shared with the entire business.

### 7.61.3.2 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.

### 7.61.3.3 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.

### 7.61.3.4 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

#### 7.61.3.5 localData

`EverCloudLocalData` `qevercloud::Notebook::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.61.3.6 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

#### 7.61.3.7 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' or 'businessNotebook' specifications, 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()`.

#### 7.61.3.8 publishing

`Optional< Publishing > qevercloud::Notebook::publishing`

If the [Notebook](#) has been opened for public access, 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()`. [Note](#) that this structure is never populated for business notebooks, see the `businessNotebook` field.

#### 7.61.3.9 recipientSettings

`Optional< NotebookRecipientSettings > qevercloud::Notebook::recipientSettings`

This represents the preferences/settings that a recipient has set for this notebook. These are intended to be changed only by the recipient, and each recipient has their own recipient settings.

#### 7.61.3.10 restrictions

`Optional< NotebookRestrictions > qevercloud::Notebook::restrictions`

NOT DOCUMENTED

#### 7.61.3.11 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.

#### 7.61.3.12 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.

#### 7.61.3.13 sharedNotebookIds

`Optional< QList< qint64 > > qevercloud::Notebook::sharedNotebookIds`

*DEPRECATED* - replaced by sharedNotebooks.

#### 7.61.3.14 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.

#### 7.61.3.15 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.

#### 7.61.3.16 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.

## 7.61.4 Property Documentation

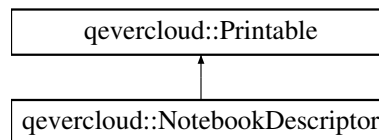
### 7.61.4.1 QList

`Optional` `qevercloud::Notebook::QList`

## 7.62 qevercloud::NotebookDescriptor Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::NotebookDescriptor`:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NotebookDescriptor` &other) const
- bool `operator!=` (const `NotebookDescriptor` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< Guid >` `guid`
- `Optional< QString >` `notebookDisplayName`
- `Optional< QString >` `contactName`
- `Optional< bool >` `hasSharedNotebook`
- `Optional< qint32 >` `joinedUserCount`

### 7.62.1 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`.

### 7.62.2 Member Function Documentation

#### 7.62.2.1 operator!=(())

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

#### 7.62.2.2 operator==(())

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

#### 7.62.2.3 print()

```
virtual void qevercloud::NotebookDescriptor::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.62.3 Member Data Documentation

#### 7.62.3.1 contactName

```
Optional< QString > qevercloud::NotebookDescriptor::contactName
```

The [User.name](#) value of the notebook's "contact".

#### 7.62.3.2 guid

```
Optional< Guid > qevercloud::NotebookDescriptor::guid
```

The unique identifier of the notebook.

#### 7.62.3.3 hasSharedNotebook

```
Optional< bool > qevercloud::NotebookDescriptor::hasSharedNotebook
```

Whether a [SharedNotebook](#) record exists between the calling user and this notebook.

#### 7.62.3.4 joinedUserCount

```
Optional< qint32 > qevercloud::NotebookDescriptor::joinedUserCount
```

The number of users who have joined this notebook.

#### 7.62.3.5 localData

```
EverCloudLocalData qevercloud::NotebookDescriptor::localData
```

See the declaration of [EverCloudLocalData](#) for details

#### 7.62.3.6 notebookDisplayName

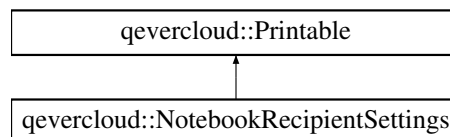
```
Optional< QString > qevercloud::NotebookDescriptor::notebookDisplayName
```

A sequence of characters representing the name of the notebook.

### 7.63 qevercloud::NotebookRecipientSettings Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NotebookRecipientSettings:



#### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NotebookRecipientSettings](#) &other) const
- bool [operator!=](#) (const [NotebookRecipientSettings](#) &other) const

#### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< bool > [reminderNotifyEmail](#)
- [Optional](#)< bool > [reminderNotifyInApp](#)
- [Optional](#)< bool > [inMyList](#)
- [Optional](#)< QString > [stack](#)
- [Optional](#)< [RecipientStatus](#) > [recipientStatus](#)



### 7.63.1 Detailed Description

Settings meant for the recipient of a notebook share.

Some of these 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.

### 7.63.2 Member Function Documentation

#### 7.63.2.1 operator!=(())

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

#### 7.63.2.2 operator==(())

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

#### 7.63.2.3 print()

```
virtual void qevercloud::NotebookRecipientSettings::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.63.3 Member Data Documentation

#### 7.63.3.1 inMyList

```
Optional< bool > qevercloud::NotebookRecipientSettings::inMyList
```

DEPRECATED: Use recipientStatus instead. The notebook is on the recipient's notebook list (formerly, we would say that the recipient has "joined" the notebook)

### 7.63.3.2 `localData`

`EverCloudLocalData` `qevercloud::NotebookRecipientSettings::localData`

See the declaration of `EverCloudLocalData` for details

### 7.63.3.3 `recipientStatus`

`Optional< RecipientStatus >` `qevercloud::NotebookRecipientSettings::recipientStatus`

The notebook is on/off the recipient's notebook list (formerly, we would say that the recipient has "joined" the notebook) and perhaps also their default notebook

### 7.63.3.4 `reminderNotifyEmail`

`Optional< bool >` `qevercloud::NotebookRecipientSettings::reminderNotifyEmail`

Indicates that the user wishes to receive daily e-mail notifications for reminders associated with the 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. This value will initially be unset.

### 7.63.3.5 `reminderNotifyInApp`

`Optional< bool >` `qevercloud::NotebookRecipientSettings::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. This value will initially be unset.

### 7.63.3.6 `stack`

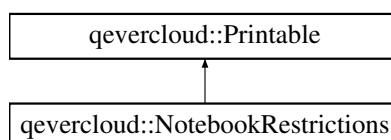
`Optional< QString >` `qevercloud::NotebookRecipientSettings::stack`

The stack the recipient has put this notebook into. See `Notebook.stack` for a definition. Every recipient can have their own stack value for the same notebook.

## 7.64 `qevercloud::NotebookRestrictions` Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::NotebookRestrictions`:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NotebookRestrictions](#) &other) const
- bool [operator!=](#) (const [NotebookRestrictions](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [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](#) > [updateWhichSharedNotebookRestrictions](#)
- [Optional](#)< [SharedNotebookInstanceRestrictions](#) > [expungeWhichSharedNotebookRestrictions](#)
- [Optional](#)< bool > [noShareNotesWithBusiness](#)
- [Optional](#)< bool > [noRenameNotebook](#)
- [Optional](#)< bool > [noSetInMyList](#)
- [Optional](#)< bool > [noChangeContact](#)
- [Optional](#)< [CanMoveToContainerRestrictions](#) > [canMoveToContainerRestrictions](#)
- [Optional](#)< bool > [noSetReminderNotifyEmail](#)
- [Optional](#)< bool > [noSetReminderNotifyInApp](#)
- [Optional](#)< bool > [noSetRecipientSettingsStack](#)
- [Optional](#)< bool > [noCanMoveNote](#)

### 7.64.1 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.

## 7.64.2 Member Function Documentation

### 7.64.2.1 operator!=(())

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

### 7.64.2.2 operator==(())

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

### 7.64.2.3 print()

```
virtual void qevercloud::NotebookRestrictions::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.64.3 Member Data Documentation

### 7.64.3.1 canMoveToContainerRestrictions

```
Optional< CanMoveToContainerRestrictions > qevercloud::NotebookRestrictions::canMoveToContainer↔
Restrictions
```

Specifies if the client can move this notebook to a container and if not, the reason why.

### 7.64.3.2 expungeWhichSharedNotebookRestrictions

```
Optional< SharedNotebookInstanceRestrictions > 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.

#### 7.64.3.3 localData

`EverCloudLocalData qevercloud::NotebookRestrictions::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.64.3.4 noCanMoveNote

`Optional< bool > qevercloud::NotebookRestrictions::noCanMoveNote`

If set, the client cannot move a [Note](#) into or out of the [Notebook](#).

#### 7.64.3.5 noChangeContact

`Optional< bool > qevercloud::NotebookRestrictions::noChangeContact`

NOT DOCUMENTED

#### 7.64.3.6 noCreateNotes

`Optional< bool > qevercloud::NotebookRestrictions::noCreateNotes`

The client may not create new notes in the notebook.

#### 7.64.3.7 noCreateSharedNotebooks

`Optional< bool > qevercloud::NotebookRestrictions::noCreateSharedNotebooks`

The client is unable to create shared notebooks for the notebook.

#### 7.64.3.8 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.

#### 7.64.3.9 noEmailNotes

`Optional< bool > qevercloud::NotebookRestrictions::noEmailNotes`

The client may not e-mail notes by guid via the Evernote service by using the `emailNote` method. Email notes by value by populating the `note` parameter instead.

#### 7.64.3.10 noExpungeNotebook

`Optional< bool > qevercloud::NotebookRestrictions::noExpungeNotebook`

The client may not expunge the [Notebook](#) object itself, for example, via the `expungeNotebook` method.

#### 7.64.3.11 noExpungeNotes

`Optional< bool > qevercloud::NotebookRestrictions::noExpungeNotes`

The client may not expunge notes currently in the notebook.

#### 7.64.3.12 noExpungeTags

`Optional< bool > qevercloud::NotebookRestrictions::noExpungeTags`

The client may not expunge tags in the owner's account.

#### 7.64.3.13 noPublishToBusinessLibrary

`Optional< bool > qevercloud::NotebookRestrictions::noPublishToBusinessLibrary`

The client may not publish the notebook to the business library.

#### 7.64.3.14 noPublishToPublic

`Optional< bool > qevercloud::NotebookRestrictions::noPublishToPublic`

The client may not publish the notebook to the public. For example, business notebooks may not be shared publicly.

#### 7.64.3.15 noReadNotes

`Optional< bool > qevercloud::NotebookRestrictions::noReadNotes`

The client is not able to read notes from the service and the notebook is write-only.

#### 7.64.3.16 noRenameNotebook

`Optional< bool > qevercloud::NotebookRestrictions::noRenameNotebook`

The client may not rename this notebook.

#### 7.64.3.17 noSendMessageToRecipients

`Optional< bool > qevercloud::NotebookRestrictions::noSendMessageToRecipients`

The client may not send messages to the share recipients of the notebook.

#### 7.64.3.18 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.

#### 7.64.3.19 noSetInMyList

`Optional< bool > qevercloud::NotebookRestrictions::noSetInMyList`

clients may not change the [NotebookRecipientSettings.inMyList](#) settings for this notebook.

#### 7.64.3.20 noSetNotebookStack

`Optional< bool > qevercloud::NotebookRestrictions::noSetNotebookStack`

If the client is able to update the [Notebook](#), the [Notebook.stack](#) value may not be set.

#### 7.64.3.21 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.

#### 7.64.3.22 noSetRecipientSettingsStack

`Optional< bool > qevercloud::NotebookRestrictions::noSetRecipientSettingsStack`

NOT DOCUMENTED

#### 7.64.3.23 noSetReminderNotifyEmail

`Optional< bool > qevercloud::NotebookRestrictions::noSetReminderNotifyEmail`

NOT DOCUMENTED

#### 7.64.3.24 noSetReminderNotifyInApp

`Optional< bool > qevercloud::NotebookRestrictions::noSetReminderNotifyInApp`

NOT DOCUMENTED

#### 7.64.3.25 noShareNotes

`Optional< bool > qevercloud::NotebookRestrictions::noShareNotes`

The client may not share notes in the notebook via the `shareNote` or `createOrUpdateSharedNotes` methods.

#### 7.64.3.26 noShareNotesWithBusiness

`Optional< bool > qevercloud::NotebookRestrictions::noShareNotesWithBusiness`

The client may not share notes in the notebook via the `shareNoteWithBusiness` method.

#### 7.64.3.27 noUpdateNotebook

`Optional< bool > qevercloud::NotebookRestrictions::noUpdateNotebook`

The client may not update the [Notebook](#) object itself, for example, via the `updateNotebook` method.

#### 7.64.3.28 noUpdateNotes

`Optional< bool > qevercloud::NotebookRestrictions::noUpdateNotes`

The client may not update notes currently in the notebook.

#### 7.64.3.29 noUpdateTags

`Optional< bool > qevercloud::NotebookRestrictions::noUpdateTags`

The client may not update tags in the owner's account.

#### 7.64.3.30 updateWhichSharedNotebookRestrictions

`Optional< SharedNotebookInstanceRestrictions > qevercloud::NotebookRestrictions::updateWhich↵  
SharedNotebookRestrictions`

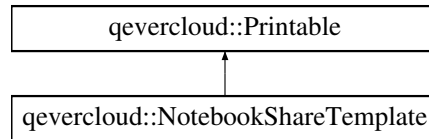
Restrictions on which shared notebook instances can be updated. If the value is not set or null, then the client can update any of the shared notebooks associated with the notebook on which the [NotebookRestrictions](#) are defined. See the enumeration for further details.



## 7.65 qevercloud::NotebookShareTemplate Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NotebookShareTemplate:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NotebookShareTemplate](#) &other) const
- bool [operator!=](#) (const [NotebookShareTemplate](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [Guid](#) > [notebookGuid](#)
- [Optional](#)< [MessageThreadID](#) > [recipientThreadId](#)
- [Optional](#)< [QList](#)< [Contact](#) > > [recipientContacts](#)
- [Optional](#)< [SharedNotebookPrivilegeLevel](#) > [privilege](#)

### Properties

- [Optional](#) [QList](#)

#### 7.65.1 Detailed Description

A structure used to share a notebook with one or more recipients at a given privilege.

#### 7.65.2 Member Function Documentation

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

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

### 7.65.2.2 operator==( )

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

### 7.65.2.3 print()

```
virtual void qevercloud::NotebookShareTemplate::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.65.3 Member Data Documentation

### 7.65.3.1 localData

```
EverCloudLocalData qevercloud::NotebookShareTemplate::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.65.3.2 notebookGuid

```
Optional< Guid > qevercloud::NotebookShareTemplate::notebookGuid
```

The GUID of the notebook.

### 7.65.3.3 privilege

```
Optional< SharedNotebookPrivilegeLevel > qevercloud::NotebookShareTemplate::privilege
```

The privilege level to be granted.

### 7.65.3.4 recipientContacts

```
Optional<QList<Contact> > qevercloud::NotebookShareTemplate::recipientContacts
```

The recipients of the notebook share specified as a list of contacts. This should only be set if the sharing takes place before the thread is created. Use `recipientThreadId` instead when sharing with an existing thread. Either this field or `recipientThreadId` must be set.

## 7.65.3.5 recipientThreadId

```
Optional< MessageThreadId > qevercloud::NotebookShareTemplate::recipientThreadId
```

The recipients of the notebook share specified as a messaging thread ID. If you have an existing messaging thread to share the note with, specify its ID here instead of recipientContacts in order to properly support defunct identities. The sharer must be a participant of the thread. Either this field or recipientContacts must be set.

## 7.65.4 Property Documentation

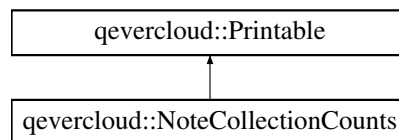
## 7.65.4.1 QList

```
Optional qevercloud::NotebookShareTemplate::QList
```

## 7.66 qevercloud::NoteCollectionCounts Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteCollectionCounts:



## Public Types

- using [TagCounts](#) = QMap< [Guid](#), qint32 >

## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteCollectionCounts](#) &other) const
- bool [operator!=](#) (const [NoteCollectionCounts](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- Optional< QMap< [Guid](#), qint32 > > [notebookCounts](#)
- Optional< QMap< [Guid](#), qint32 > > [tagCounts](#)
- Optional< qint32 > [trashCount](#)

## Properties

- [Optional< TagCounts > notebookCounts](#)
- [Optional< TagCounts > tagCounts](#)

### 7.66.1 Detailed Description

A data structure representing the number of notes for each notebook and tag with a non-zero set of applicable notes.

### 7.66.2 Member Typedef Documentation

#### 7.66.2.1 TagCounts

```
using qevercloud::NoteCollectionCounts::TagCounts = QMap<Guid, qint32>
```

### 7.66.3 Member Function Documentation

#### 7.66.3.1 operator!=(())

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

#### 7.66.3.2 operator==(())

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

#### 7.66.3.3 print()

```
virtual void qevercloud::NoteCollectionCounts::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.66.4 Member Data Documentation

### 7.66.4.1 localData

`EverCloudLocalData` `qevercloud::NoteCollectionCounts::localData`

See the declaration of `EverCloudLocalData` for details

### 7.66.4.2 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.

### 7.66.4.3 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.

### 7.66.4.4 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.)

## 7.66.5 Property Documentation

### 7.66.5.1 notebookCounts

`Optional<TagCounts>` `qevercloud::NoteCollectionCounts::notebookCounts`

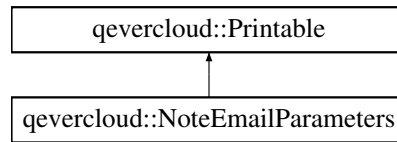
### 7.66.5.2 tagCounts

`Optional<TagCounts>` `qevercloud::NoteCollectionCounts::tagCounts`

## 7.67 qevercloud::NoteEmailParameters Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteEmailParameters:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteEmailParameters](#) &other) const
- bool [operator!=](#) (const [NoteEmailParameters](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QString](#) > [guid](#)
- [Optional](#)< [Note](#) > [note](#)
- [Optional](#)< [QStringList](#) > [toAddresses](#)
- [Optional](#)< [QStringList](#) > [ccAddresses](#)
- [Optional](#)< [QString](#) > [subject](#)
- [Optional](#)< [QString](#) > [message](#)

### 7.67.1 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.

### 7.67.2 Member Function Documentation

#### 7.67.2.1 operator"!=()

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

### 7.67.2.2 operator==( )

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

### 7.67.2.3 print()

```
virtual void qevercloud::NoteEmailParameters::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.67.3 Member Data Documentation

### 7.67.3.1 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.

### 7.67.3.2 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.

### 7.67.3.3 localData

```
EverCloudLocalData qevercloud::NoteEmailParameters::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.67.3.4 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).

### 7.67.3.5 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.

### 7.67.3.6 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.

### 7.67.3.7 toAddresses

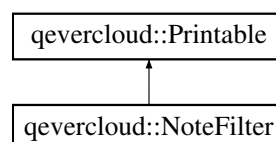
```
Optional< QStringList > qevercloud::NoteEmailParameters::toAddresses
```

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.

## 7.68 qevercloud::NoteFilter Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteFilter:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteFilter](#) &other) const
- bool [operator!=](#) (const [NoteFilter](#) &other) const



## Public Attributes

- [EverCloudLocalData](#) `localData`
- [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`
- [Optional](#)< [bool](#) > `includeAllReadableNotebooks`
- [Optional](#)< [bool](#) > `includeAllReadableWorkspaces`
- [Optional](#)< [QString](#) > `context`
- [Optional](#)< [QString](#) > `rawWords`
- [Optional](#)< [QByteArray](#) > `searchContextBytes`

## Properties

- [Optional](#) [QList](#)

### 7.68.1 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.

### 7.68.2 Member Function Documentation

#### 7.68.2.1 `operator!=( )`

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

#### 7.68.2.2 `operator==( )`

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

### 7.68.2.3 print()

```
virtual void qevercloud::NoteFilter::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.68.3 Member Data Documentation

### 7.68.3.1 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.

### 7.68.3.2 context

```
Optional< QString > qevercloud::NoteFilter::context
```

Specifies the context to consider when determining result ranking. Clients must leave this value unset unless they wish to explicitly specify a known non-default context.

### 7.68.3.3 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.

### 7.68.3.4 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.

### 7.68.3.5 includeAllReadableNotebooks

```
Optional< bool > qevercloud::NoteFilter::includeAllReadableNotebooks
```

If true, then the search will include all business notebooks that are readable by the user. A business authentication token must be supplied for this option to take effect when calling search APIs.

#### 7.68.3.6 includeAllReadableWorkspaces

`Optional< bool > qevercloud::NoteFilter::includeAllReadableWorkspaces`

If true, then the search will include all workspaces that are readable by the user. A business authentication token must be supplied for this option to take effect when calling search APIs.

#### 7.68.3.7 localData

`EverCloudLocalData qevercloud::NoteFilter::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.68.3.8 notebookGuid

`Optional< Guid > qevercloud::NoteFilter::notebookGuid`

If present, the Guid of the notebook that must contain the notes.

#### 7.68.3.9 order

`Optional< qint32 > qevercloud::NoteFilter::order`

The NoteSortOrder value indicating what criterion should be used to sort the results of the filter.

#### 7.68.3.10 rawWords

`Optional< QString > qevercloud::NoteFilter::rawWords`

If present, the raw user query input. Accepts the full search grammar documented in the Evernote API Overview.

#### 7.68.3.11 searchContextBytes

`Optional< QByteArray > qevercloud::NoteFilter::searchContextBytes`

Specifies the correlating information about the current search session, in byte array. If this request is not for the first page of search results, the client should populate this field with the value of searchContextBytes from the [NotesMetadataList](#) of the original search response.

#### 7.68.3.12 tagGuids

`Optional< QList< Guid > > qevercloud::NoteFilter::tagGuids`

If present, the list of tags (by GUID) that must be present on the notes.

### 7.68.3.13 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".

### 7.68.3.14 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.

## 7.68.4 Property Documentation

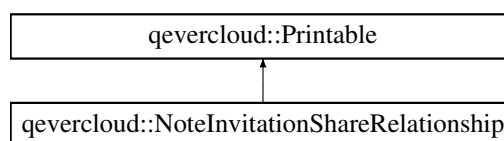
### 7.68.4.1 QList

`Optional qevercloud::NoteFilter::QList`

## 7.69 qevercloud::NoteInvitationShareRelationship Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteInvitationShareRelationship:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NoteInvitationShareRelationship` &other) const
- bool `operator!=` (const `NoteInvitationShareRelationship` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< QString >` `displayName`
- `Optional< IdentityID >` `recipientIdentityId`
- `Optional< SharedNotePrivilegeLevel >` `privilege`
- `Optional< UserID >` `sharerUserId`

### 7.69.1 Detailed Description

Describes an invitation to a person to use their Evernote credentials to gain access to a note belonging to another user.

### 7.69.2 Member Function Documentation

#### 7.69.2.1 operator!=( )

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

#### 7.69.2.2 operator==( )

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

#### 7.69.2.3 print( )

```
virtual void qevercloud::NoteInvitationShareRelationship::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.69.3 Member Data Documentation

#### 7.69.3.1 displayName

```
Optional< QString > qevercloud::NoteInvitationShareRelationship::displayName
```

The string that clients should show to users to represent this invitation.

#### 7.69.3.2 localData

```
EverCloudLocalData qevercloud::NoteInvitationShareRelationship::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.69.3.3 privilege

`Optional< SharedNotePrivilegeLevel > qevercloud::NoteInvitationShareRelationship::privilege`

The privilege level that the recipient will be granted when they accept this invitation. If the user already has a higher privilege to access this note then this will not affect the recipient's privileges.

### 7.69.3.4 recipientIdentityId

`Optional< IdentityID > qevercloud::NoteInvitationShareRelationship::recipientIdentityId`

Identifies the identity of the invitation recipient. Once the identity has been claimed by an Evernote user and they have accessed the note at least once, the invitation will be used up and will no longer be returned by the service to clients. Instead, that recipient will be included in the list of `NoteMemberShareRelationships`.

### 7.69.3.5 sharerUserId

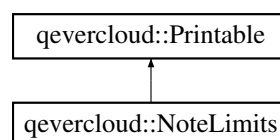
`Optional< UserID > qevercloud::NoteInvitationShareRelationship::sharerUserId`

The user id of the user who most recently shared this note to this recipient. This field is used by the service to convey information to the user, so clients should treat it as read-only.

## 7.70 qevercloud::NoteLimits Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::NoteLimits`:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NoteLimits` &other) const
- bool `operator!=` (const `NoteLimits` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< qint32 >` `noteResourceCountMax`
- `Optional< qint64 >` `uploadLimit`
- `Optional< qint64 >` `resourceSizeMax`
- `Optional< qint64 >` `noteSizeMax`
- `Optional< qint64 >` `uploaded`

### 7.70.1 Detailed Description

Represents the owner's account related limits on a [Note](#). The field uploaded represents the total number of bytes that have been uploaded to this account and is taken from the [SyncState](#) struct. All other fields represent account related limits and are taken from the [AccountLimits](#) struct.

See [SyncState](#) and [AccountLimits](#) struct field definitions for more details.

### 7.70.2 Member Function Documentation

#### 7.70.2.1 operator!=(())

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

#### 7.70.2.2 operator==(())

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

#### 7.70.2.3 print()

```
virtual void qevercloud::NoteLimits::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.70.3 Member Data Documentation

#### 7.70.3.1 localData

[EverCloudLocalData](#) qevercloud::NoteLimits::localData

See the declaration of [EverCloudLocalData](#) for details

**7.70.3.2 noteResourceCountMax**

`Optional< qint32 > qevercloud::NoteLimits::noteResourceCountMax`

NOT DOCUMENTED

**7.70.3.3 noteSizeMax**

`Optional< qint64 > qevercloud::NoteLimits::noteSizeMax`

NOT DOCUMENTED

**7.70.3.4 resourceSizeMax**

`Optional< qint64 > qevercloud::NoteLimits::resourceSizeMax`

NOT DOCUMENTED

**7.70.3.5 uploaded**

`Optional< qint64 > qevercloud::NoteLimits::uploaded`

NOT DOCUMENTED

**7.70.3.6 uploadLimit**

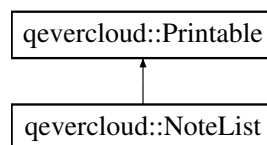
`Optional< qint64 > qevercloud::NoteLimits::uploadLimit`

NOT DOCUMENTED

**7.71 qevercloud::NoteList Struct Reference**

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteList:

**Public Member Functions**

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NoteList` &other) const
- bool `operator!=` (const `NoteList` &other) const



## Public Attributes

- [EverCloudLocalData](#) `localData`
- `qint32` `startIndex` = 0
- `qint32` `totalNotes` = 0
- `QList< Note >` `notes`
- `Optional< QStringList >` `stoppedWords`
- `Optional< QStringList >` `searchedWords`
- `Optional< qint32 >` `updateCount`
- `Optional< QByteArray >` `searchContextBytes`
- `Optional< QString >` `debugInfo`

### 7.71.1 Detailed Description

A small structure for returning a list of notes out of a larger set.

### 7.71.2 Member Function Documentation

#### 7.71.2.1 `operator!=()`

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

#### 7.71.2.2 `operator==()`

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

#### 7.71.2.3 `print()`

```
virtual void qevercloud::NoteList::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.71.3 Member Data Documentation

#### 7.71.3.1 debugInfo

```
Optional< QString > qevercloud::NoteList::debugInfo
```

Depends on the value of `context` in [NoteFilter](#), this field may contain debug information if the service decides to do so.

#### 7.71.3.2 localData

```
EverCloudLocalData qevercloud::NoteList::localData
```

See the declaration of [EverCloudLocalData](#) for details

#### 7.71.3.3 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.

#### 7.71.3.4 searchContextBytes

```
Optional< QByteArray > qevercloud::NoteList::searchContextBytes
```

Specifies the correlating information about the current search session, in byte array.

#### 7.71.3.5 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.

#### 7.71.3.6 startIndex

```
qint32 qevercloud::NoteList::startIndex = 0
```

The starting index within the overall set of notes. This is also the number of notes that are "before" this list in the set.

#### 7.71.3.7 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.

## 7.71.3.8 totalNotes

```
qint32 qevercloud::NoteList::totalNotes = 0
```

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) )

## 7.71.3.9 updateCount

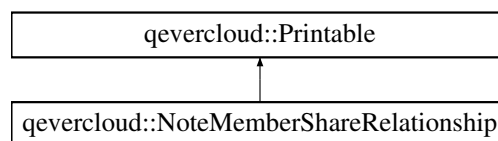
```
Optional< qint32 > qevercloud::NoteList::updateCount
```

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.

## 7.72 qevercloud::NoteMemberShareRelationship Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteMemberShareRelationship:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteMemberShareRelationship](#) &other) const
- bool [operator!=](#) (const [NoteMemberShareRelationship](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QString](#) > [displayName](#)
- [Optional](#)< [UserID](#) > [recipientUserId](#)
- [Optional](#)< [SharedNotePrivilegeLevel](#) > [privilege](#)
- [Optional](#)< [NoteShareRelationshipRestrictions](#) > [restrictions](#)
- [Optional](#)< [UserID](#) > [sharerUserId](#)

## 7.72.1 Detailed Description

Describes the association between a [Note](#) and an Evernote [User](#) who is a member of that note.

## 7.72.2 Member Function Documentation

### 7.72.2.1 operator!=(())

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

### 7.72.2.2 operator==(())

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

### 7.72.2.3 print()

```
virtual void qevercloud::NoteMemberShareRelationship::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.72.3 Member Data Documentation

### 7.72.3.1 displayName

```
Optional< QString > qevercloud::NoteMemberShareRelationship::displayName
```

The string that clients should show to users to represent this member.

### 7.72.3.2 localData

```
EverCloudLocalData qevercloud::NoteMemberShareRelationship::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.72.3.3 privilege

```
Optional< SharedNotePrivilegeLevel > qevercloud::NoteMemberShareRelationship::privilege
```

The privilege at which the member can access the note, which is the best privilege granted to the user across all of their individual shares for this note. This field is used by the service to convey information to the user, so clients should treat it as read-only.

## 7.72.3.4 recipientUserId

`Optional< UserID > qevercloud::NoteMemberShareRelationship::recipientUserId`

The Evernote UserID of the user who is a member to the note.

## 7.72.3.5 restrictions

`Optional< NoteShareRelationshipRestrictions > qevercloud::NoteMemberShareRelationship::restrictions`

The restrictions on which privileges may be individually assigned to the recipient of this share relationship. This field is used by the service to convey information to the user, so clients should treat it as read-only.

## 7.72.3.6 sharerUserId

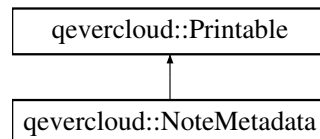
`Optional< UserID > qevercloud::NoteMemberShareRelationship::sharerUserId`

The user id of the user who most recently shared the note with this user. This field is used by the service to convey information to the user, so clients should treat it as read-only.

## 7.73 qevercloud::NoteMetadata Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteMetadata:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NoteMetadata` &other) const
- bool `operator!=` (const `NoteMetadata` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `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`

## Properties

- [Optional QList](#)

### 7.73.1 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:

### 7.73.2 Member Function Documentation

#### 7.73.2.1 `operator!=( )`

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

#### 7.73.2.2 `operator==( )`

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

#### 7.73.2.3 `print( )`

```
virtual void qevercloud::NoteMetadata::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.73.3 Member Data Documentation

#### 7.73.3.1 `attributes`

```
Optional< NoteAttributes > qevercloud::NoteMetadata::attributes
```

NOT DOCUMENTED

#### 7.73.3.2 contentLength

`Optional< qint32 > qevercloud::NoteMetadata::contentLength`

NOT DOCUMENTED

#### 7.73.3.3 created

`Optional< Timestamp > qevercloud::NoteMetadata::created`

NOT DOCUMENTED

#### 7.73.3.4 deleted

`Optional< Timestamp > qevercloud::NoteMetadata::deleted`

NOT DOCUMENTED

#### 7.73.3.5 guid

`Guid qevercloud::NoteMetadata::guid`

NOT DOCUMENTED

#### 7.73.3.6 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](#).

#### 7.73.3.7 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](#).

#### 7.73.3.8 localData

`EverCloudLocalData qevercloud::NoteMetadata::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.73.3.9 notebookGuid

`Optional< QString > qevercloud::NoteMetadata::notebookGuid`

NOT DOCUMENTED

#### 7.73.3.10 tagGuids

`Optional< QList< Guid > > qevercloud::NoteMetadata::tagGuids`

NOT DOCUMENTED

#### 7.73.3.11 title

`Optional< QString > qevercloud::NoteMetadata::title`

NOT DOCUMENTED

#### 7.73.3.12 updated

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

NOT DOCUMENTED

#### 7.73.3.13 updateSequenceNum

`Optional< qint32 > qevercloud::NoteMetadata::updateSequenceNum`

NOT DOCUMENTED

### 7.73.4 Property Documentation

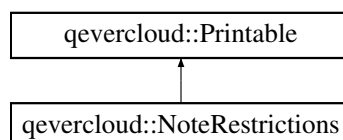
#### 7.73.4.1 QList

`Optional qevercloud::NoteMetadata::QList`

## 7.74 qevercloud::NoteRestrictions Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteRestrictions:





## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteRestrictions](#) &other) const
- bool [operator!=](#) (const [NoteRestrictions](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) localData
- [Optional](#)< bool > noUpdateTitle
- [Optional](#)< bool > noUpdateContent
- [Optional](#)< bool > noEmail
- [Optional](#)< bool > noShare
- [Optional](#)< bool > noSharePublicly

### 7.74.1 Detailed Description

This structure captures information about the operations that cannot be performed on a given note that has been shared with a recipient via a [SharedNote](#). The following operations are **never** allowed based on SharedNotes, and as such are left out of the [NoteRestrictions](#) structure for brevity:

- Expunging a note ([NoteStore.expungeNote](#))
- Moving a note to the trash ([Note.active](#))
- Updating a note's notebook ([Note.notebookGuid](#))
- Updating a note's tags ([Note.tagGuids](#), [Note.tagNames](#))
- Updating a note's attributes ([Note.attributes](#))
- Sharing a note with the business ([NoteStore.shareNoteWithBusiness](#))
- Getting a note's version history ([NoteStore.listNoteVersions](#), [NoteStore.getNoteVersion](#))

When a client has permission to update a note's title or content, it may also update the [Note.updated](#) timestamp.

**This structure reflects only the privileges / restrictions conveyed by the [SharedNote](#).** It does not incorporate privileges conveyed by a potential [SharedNotebook](#) to the same recipient. As such, the actual permissions that the recipient has on the note may differ from the permissions expressed in this structure.

For example, consider a user with read-only access to a shared notebook, and a read-write share of a specific note in the notebook. The note restrictions would contain noUpdateTitle = false, while the notebook restrictions would contain noUpdateNotes = true. In this case, the user is allowed to update the note title based on the note restrictions.

Alternatively, consider a user with read-write access to a shared notebook, and a read-only share of a specific note in that notebook. The note restrictions would contain noUpdateTitle = true, while the notebook restrictions would contain noUpdateNotes = false. In this case, the user would have full edit permissions on the note based on the notebook restrictions.

### 7.74.2 Member Function Documentation

#### 7.74.2.1 operator!=(())

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

#### 7.74.2.2 operator==(())

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

#### 7.74.2.3 print()

```
virtual void qevercloud::NoteRestrictions::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.74.3 Member Data Documentation

#### 7.74.3.1 localData

[EverCloudLocalData](#) qevercloud::NoteRestrictions::localData

See the declaration of [EverCloudLocalData](#) for details

#### 7.74.3.2 noEmail

[Optional](#)< bool > qevercloud::NoteRestrictions::noEmail

The client may not email the note (NoteStore.emailNote).

#### 7.74.3.3 noShare

[Optional](#)< bool > qevercloud::NoteRestrictions::noShare

The client may not share the note with specific recipients (NoteStore.createOrUpdateSharedNotes).

## 7.74.3.4 noSharePublicly

```
Optional< bool > qevercloud::NoteRestrictions::noSharePublicly
```

The client may not make the note public (`NoteStore.shareNote`).

## 7.74.3.5 noUpdateContent

```
Optional< bool > qevercloud::NoteRestrictions::noUpdateContent
```

NOT DOCUMENTED

## 7.74.3.6 noUpdateTitle

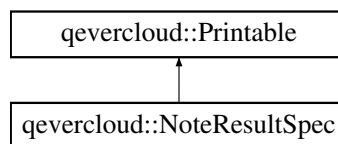
```
Optional< bool > qevercloud::NoteRestrictions::noUpdateTitle
```

The client may not update the note's title (`Note.title`).

## 7.75 qevercloud::NoteResultSpec Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::NoteResultSpec`:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NoteResultSpec` &other) const
- bool `operator!=` (const `NoteResultSpec` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< bool >` `includeContent`
- `Optional< bool >` `includeResourcesData`
- `Optional< bool >` `includeResourcesRecognition`
- `Optional< bool >` `includeResourcesAlternateData`
- `Optional< bool >` `includeSharedNotes`
- `Optional< bool >` `includeNoteAppDataValues`
- `Optional< bool >` `includeResourceAppDataValues`
- `Optional< bool >` `includeAccountLimits`

### 7.75.1 Detailed Description

This structure is provided to the `getNoteWithResultSpec` function to specify the subset of fields that should be included in the [Note](#) that is returned. This allows clients to request the minimum set of information that they require when retrieving a note, reducing the size of the response and improving the response time.

If one of the fields in this spec is not set, then it will be treated as 'false' by the service, so that the default behavior is to include none of the fields below in the [Note](#).

### 7.75.2 Member Function Documentation

#### 7.75.2.1 `operator!=()`

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

#### 7.75.2.2 `operator==()`

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

#### 7.75.2.3 `print()`

```
virtual void qevercloud::NoteResultSpec::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.75.3 Member Data Documentation

#### 7.75.3.1 `includeAccountLimits`

```
Optional< bool > qevercloud::NoteResultSpec::includeAccountLimits
```

If true, the [Note.limits](#) field will be populated with the note owner's account limits.

### 7.75.3.2 includeContent

`Optional< bool > qevercloud::NoteResultSpec::includeContent`

If true, the [Note.content](#) field will be populated with the note's ENML contents.

### 7.75.3.3 includeNoteAppDataValues

`Optional< bool > qevercloud::NoteResultSpec::includeNoteAppDataValues`

If true, the `Note.attributes.applicationData.fullMap` field will be populated.

### 7.75.3.4 includeResourceAppDataValues

`Optional< bool > qevercloud::NoteResultSpec::includeResourceAppDataValues`

If true, the `Note.resource.attributes.applicationData.fullMap` field will be populated.

### 7.75.3.5 includeResourcesAlternateData

`Optional< bool > qevercloud::NoteResultSpec::includeResourcesAlternateData`

If true, any [Resource](#) elements will include the binary contents of their 'alternateData' field's body, if an alternate form is available.

### 7.75.3.6 includeResourcesData

`Optional< bool > qevercloud::NoteResultSpec::includeResourcesData`

If true, any [Resource](#) elements will include the binary contents of their 'data' field's body.

### 7.75.3.7 includeResourcesRecognition

`Optional< bool > qevercloud::NoteResultSpec::includeResourcesRecognition`

If true, any [Resource](#) elements will include the binary contents of their 'recognition' field's body if recognition data is available.

### 7.75.3.8 includeSharedNotes

`Optional< bool > qevercloud::NoteResultSpec::includeSharedNotes`

If true, the [Note.sharedNotes](#) field will be populated with the note's shares.

### 7.75.3.9 localData

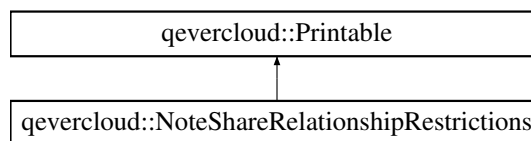
[EverCloudLocalData](#) `qevercloud::NoteResultSpec::localData`

See the declaration of [EverCloudLocalData](#) for details

## 7.76 qevercloud::NoteShareRelationshipRestrictions Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::NoteShareRelationshipRestrictions`:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteShareRelationshipRestrictions](#) &other) const
- bool [operator!=](#) (const [NoteShareRelationshipRestrictions](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) `localData`
- [Optional](#)< bool > `noSetReadNote`
- [Optional](#)< bool > `noSetModifyNote`
- [Optional](#)< bool > `noSetFullAccess`

### 7.76.1 Detailed Description

This structure is used by the service to communicate to clients, via `getNoteShareRelationships`, which privilege levels are assignable to the target of a note share relationship.

### 7.76.2 Member Function Documentation

#### 7.76.2.1 `operator!=()`

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

### 7.76.2.2 operator==( )

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

### 7.76.2.3 print()

```
virtual void qevercloud::NoteShareRelationshipRestrictions::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.76.3 Member Data Documentation

### 7.76.3.1 localData

[EverCloudLocalData](#) qevercloud::NoteShareRelationshipRestrictions::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.76.3.2 noSetFullAccess

[Optional](#)< bool > qevercloud::NoteShareRelationshipRestrictions::noSetFullAccess

This value is true if the user is not allowed to set the privilege level to [SharedNotePrivilegeLevel.FULL\\_ACCESS](#).

### 7.76.3.3 noSetModifyNote

[Optional](#)< bool > qevercloud::NoteShareRelationshipRestrictions::noSetModifyNote

This value is true if the user is not allowed to set the privilege level to [SharedNotePrivilegeLevel.MODIFY\\_NOTE](#).

### 7.76.3.4 noSetReadNote

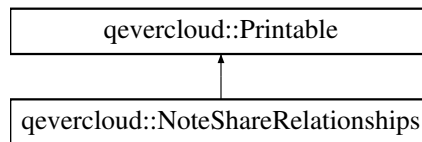
[Optional](#)< bool > qevercloud::NoteShareRelationshipRestrictions::noSetReadNote

This value is true if the user is not allowed to set the privilege level to [SharedNotePrivilegeLevel.READ\\_NOTE](#).

## 7.77 qevercloud::NoteShareRelationships Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteShareRelationships:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteShareRelationships](#) &other) const
- bool [operator!=](#) (const [NoteShareRelationships](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QList](#)< [NoteInvitationShareRelationship](#) > > [invitations](#)
- [Optional](#)< [QList](#)< [NoteMemberShareRelationship](#) > > [memberships](#)
- [Optional](#)< [NoteShareRelationshipRestrictions](#) > [invitationRestrictions](#)

### Properties

- [Optional](#) [QList](#)

#### 7.77.1 Detailed Description

Captures a collection of share relationships for a single note, for example, as returned by the `getNoteShares` method. The share relationships fall into two broad categories: members, and invitations that can be used to become members.

#### 7.77.2 Member Function Documentation

##### 7.77.2.1 `operator!=()`

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



### 7.77.2.2 operator==( )

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

### 7.77.2.3 print()

```
virtual void qevercloud::NoteShareRelationships::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.77.3 Member Data Documentation

### 7.77.3.1 invitationRestrictions

[Optional](#)< [NoteShareRelationshipRestrictions](#) > qevercloud::NoteShareRelationships::invitation↔  
Restrictions

NOT DOCUMENTED

### 7.77.3.2 invitations

[Optional](#)<[QList](#)<[NoteInvitationShareRelationship](#)> > qevercloud::NoteShareRelationships::invitations

A list of open invitations that can be redeemed into memberships to the note.

### 7.77.3.3 localData

[EverCloudLocalData](#) qevercloud::NoteShareRelationships::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.77.3.4 memberships

[Optional](#)<[QList](#)<[NoteMemberShareRelationship](#)> > qevercloud::NoteShareRelationships::memberships

A list of memberships of the noteb. A member is identified by their Evernote UserID and has rights to access the note.

## 7.77.4 Property Documentation

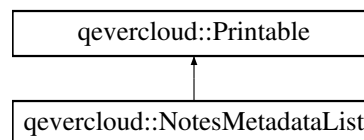
### 7.77.4.1 QList

`Optional` `qevercloud::NoteShareRelationships::QList`

## 7.78 qevercloud::NotesMetadataList Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::NotesMetadataList`:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `NotesMetadataList` &other) const
- bool `operator!=` (const `NotesMetadataList` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- quint32 `startIndex` = 0
- quint32 `totalNotes` = 0
- QList< `NoteMetadata` > `notes`
- `Optional`< QStringList > `stoppedWords`
- `Optional`< QStringList > `searchedWords`
- `Optional`< quint32 > `updateCount`
- `Optional`< QByteArray > `searchContextBytes`
- `Optional`< QString > `debugInfo`

### 7.78.1 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.

### 7.78.2 Member Function Documentation

### 7.78.2.1 operator!=(())

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

### 7.78.2.2 operator==(())

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

### 7.78.2.3 print()

```
virtual void qevercloud::NotesMetadataList::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.78.3 Member Data Documentation

### 7.78.3.1 debugInfo

[Optional](#)< [QString](#) > qevercloud::NotesMetadataList::debugInfo

Depends on the value of context in [NoteFilter](#), this field may contain debug information if the service decides to do so.

### 7.78.3.2 localData

[EverCloudLocalData](#) qevercloud::NotesMetadataList::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.78.3.3 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](#).

#### 7.78.3.4 searchContextBytes

```
Optional< QByteArray > qevercloud::NotesMetadataList::searchContextBytes
```

Specifies the correlating information about the current search session, in byte array.

#### 7.78.3.5 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.

#### 7.78.3.6 startIndex

```
qint32 qevercloud::NotesMetadataList::startIndex = 0
```

The starting index within the overall set of notes. This is also the number of notes that are "before" this list in the set.

#### 7.78.3.7 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.

#### 7.78.3.8 totalNotes

```
qint32 qevercloud::NotesMetadataList::totalNotes = 0
```

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.  $\text{remaining} = \text{totalNotes} - (\text{startIndex} + \text{notes.length})$  )

#### 7.78.3.9 updateCount

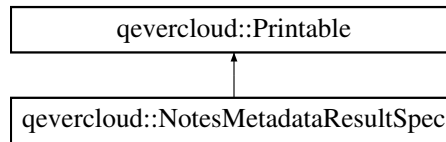
```
Optional< qint32 > qevercloud::NotesMetadataList::updateCount
```

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.

## 7.79 qevercloud::NotesMetadataResultSpec Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NotesMetadataResultSpec:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NotesMetadataResultSpec](#) &other) const
- bool [operator!=](#) (const [NotesMetadataResultSpec](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) localData
- [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](#)

#### 7.79.1 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)

#### 7.79.2 Member Function Documentation

#### 7.79.2.1 operator!=( )

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

#### 7.79.2.2 operator==( )

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

#### 7.79.2.3 print()

```
virtual void qevercloud::NotesMetadataResultSpec::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.79.3 Member Data Documentation

#### 7.79.3.1 includeAttributes

```
Optional< bool > qevercloud::NotesMetadataResultSpec::includeAttributes
```

NOT DOCUMENTED

#### 7.79.3.2 includeContentLength

```
Optional< bool > qevercloud::NotesMetadataResultSpec::includeContentLength
```

NOT DOCUMENTED

#### 7.79.3.3 includeCreated

```
Optional< bool > qevercloud::NotesMetadataResultSpec::includeCreated
```

NOT DOCUMENTED

#### 7.79.3.4 includeDeleted

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeDeleted`

NOT DOCUMENTED

#### 7.79.3.5 includeLargestResourceMime

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeLargestResourceMime`

NOT DOCUMENTED

#### 7.79.3.6 includeLargestResourceSize

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeLargestResourceSize`

NOT DOCUMENTED

#### 7.79.3.7 includeNotebookGuid

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeNotebookGuid`

NOT DOCUMENTED

#### 7.79.3.8 includeTagGuids

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeTagGuids`

NOT DOCUMENTED

#### 7.79.3.9 includeTitle

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeTitle`

NOT DOCUMENTED

#### 7.79.3.10 includeUpdated

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeUpdated`

NOT DOCUMENTED

#### 7.79.3.11 includeUpdateSequenceNum

`Optional< bool > qevercloud::NotesMetadataResultSpec::includeUpdateSequenceNum`

NOT DOCUMENTED

### 7.79.3.12 localData

[EverCloudLocalData](#) `qevercloud::NotesMetadataResultSpec::localData`

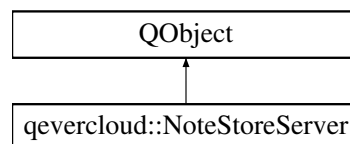
See the declaration of [EverCloudLocalData](#) for details

## 7.80 qevercloud::NoteStoreServer Class Reference

The [NoteStoreServer](#) class represents customizable server for NoteStore requests. It is primarily used for testing of QEverCloud.

```
#include <Servers.h>
```

Inheritance diagram for `qevercloud::NoteStoreServer`:



### Public Slots

- void [onRequest](#) (QByteArray data)
- void [onGetSyncStateRequestReady](#) (SyncState value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetFilteredSyncChunkRequestReady](#) (SyncChunk value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetLinkedNotebookSyncStateRequestReady](#) (SyncState value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetLinkedNotebookSyncChunkRequestReady](#) (SyncChunk value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListNotebooksRequestReady](#) (QList< Notebook > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListAccessibleBusinessNotebooksRequestReady](#) (QList< Notebook > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNotebookRequestReady](#) (Notebook value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetDefaultNotebookRequestReady](#) (Notebook value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCreateNotebookRequestReady](#) (Notebook value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateNotebookRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onExpungeNotebookRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListTagsRequestReady](#) (QList< Tag > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListTagsByNotebookRequestReady](#) (QList< Tag > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetTagRequestReady](#) (Tag value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCreateTagRequestReady](#) (Tag value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateTagRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUntagAllRequestReady](#) ([EverCloudExceptionDataPtr](#) exceptionData)
- void [onExpungeTagRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListSearchesRequestReady](#) (QList< SavedSearch > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetSearchRequestReady](#) (SavedSearch value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCreateSearchRequestReady](#) (SavedSearch value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateSearchRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)



- void [onExpungeSearchRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onFindNoteOffsetRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onFindNotesMetadataRequestReady](#) ([NotesMetadataList](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onFindNoteCountsRequestReady](#) ([NoteCollectionCounts](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteWithResultSpecRequestReady](#) ([Note](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteRequestReady](#) ([Note](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteApplicationDataRequestReady](#) ([LazyMap](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteApplicationDataEntryRequestReady](#) (QString value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onSetNoteApplicationDataEntryRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUnsetNoteApplicationDataEntryRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteContentRequestReady](#) (QString value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteSearchTextRequestReady](#) (QString value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceSearchTextRequestReady](#) (QString value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteTagNamesRequestReady](#) (QStringList value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCreateNoteRequestReady](#) ([Note](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateNoteRequestReady](#) ([Note](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onDeleteNoteRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onExpungeNoteRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCopyNoteRequestReady](#) ([Note](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListNoteVersionsRequestReady](#) (QList< [NoteVersionId](#) > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNoteVersionRequestReady](#) ([Note](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceRequestReady](#) ([Resource](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceApplicationDataRequestReady](#) ([LazyMap](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceApplicationDataEntryRequestReady](#) (QString value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onSetResourceApplicationDataEntryRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUnsetResourceApplicationDataEntryRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateResourceRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceDataRequestReady](#) (QByteArray value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceByHashRequestReady](#) ([Resource](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceRecognitionRequestReady](#) (QByteArray value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceAlternateDataRequestReady](#) (QByteArray value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetResourceAttributesRequestReady](#) ([ResourceAttributes](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetPublicNotebookRequestReady](#) ([Notebook](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onShareNotebookRequestReady](#) ([SharedNotebook](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCreateOrUpdateNotebookSharesRequestReady](#) ([CreateOrUpdateNotebookSharesResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateSharedNotebookRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onSetNotebookRecipientSettingsRequestReady](#) ([Notebook](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListSharedNotebooksRequestReady](#) (QList< [SharedNotebook](#) > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCreateLinkedNotebookRequestReady](#) ([LinkedNotebook](#) value, [EverCloudExceptionDataPtr](#) exceptionData)

- void [onUpdateLinkedNotebookRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListLinkedNotebooksRequestReady](#) (QList< [LinkedNotebook](#) > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onExpungeLinkedNotebookRequestReady](#) (qint32 value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onAuthenticateToSharedNotebookRequestReady](#) ([AuthenticationResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetSharedNotebookByAuthRequestReady](#) ([SharedNotebook](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onEmailNoteRequestReady](#) ([EverCloudExceptionDataPtr](#) exceptionData)
- void [onShareNoteRequestReady](#) (QString value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onStopSharingNoteRequestReady](#) ([EverCloudExceptionDataPtr](#) exceptionData)
- void [onAuthenticateToSharedNoteRequestReady](#) ([AuthenticationResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onFindRelatedRequestReady](#) ([RelatedResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateNotelfUsnMatchesRequestReady](#) ([UpdateNotelfUsnMatchesResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onManageNotebookSharesRequestReady](#) ([ManageNotebookSharesResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetNotebookSharesRequestReady](#) ([ShareRelationships](#) value, [EverCloudExceptionDataPtr](#) exceptionData)

## Signals

- void [getSyncStateRequest](#) ([IRequestContextPtr](#) ctx)
- void [getFilteredSyncChunkRequest](#) (qint32 afterUSN, qint32 maxEntries, [SyncChunkFilter](#) filter, [IRequestContextPtr](#) ctx)
- void [getLinkedNotebookSyncStateRequest](#) ([LinkedNotebook](#) linkedNotebook, [IRequestContextPtr](#) ctx)
- void [getLinkedNotebookSyncChunkRequest](#) ([LinkedNotebook](#) linkedNotebook, qint32 afterUSN, qint32 maxEntries, bool fullSyncOnly, [IRequestContextPtr](#) ctx)
- void [listNotebooksRequest](#) ([IRequestContextPtr](#) ctx)
- void [listAccessibleBusinessNotebooksRequest](#) ([IRequestContextPtr](#) ctx)
- void [getNotebookRequest](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [getDefaultNotebookRequest](#) ([IRequestContextPtr](#) ctx)
- void [createNotebookRequest](#) ([Notebook](#) notebook, [IRequestContextPtr](#) ctx)
- void [updateNotebookRequest](#) ([Notebook](#) notebook, [IRequestContextPtr](#) ctx)
- void [expungeNotebookRequest](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [listTagsRequest](#) ([IRequestContextPtr](#) ctx)
- void [listTagsByNotebookRequest](#) ([Guid](#) notebookGuid, [IRequestContextPtr](#) ctx)
- void [getTagRequest](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [createTagRequest](#) ([Tag](#) tag, [IRequestContextPtr](#) ctx)
- void [updateTagRequest](#) ([Tag](#) tag, [IRequestContextPtr](#) ctx)
- void [untagAllRequest](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [expungeTagRequest](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [listSearchesRequest](#) ([IRequestContextPtr](#) ctx)
- void [getSearchRequest](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [createSearchRequest](#) ([SavedSearch](#) search, [IRequestContextPtr](#) ctx)
- void [updateSearchRequest](#) ([SavedSearch](#) search, [IRequestContextPtr](#) ctx)
- void [expungeSearchRequest](#) ([Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [findNoteOffsetRequest](#) ([NoteFilter](#) filter, [Guid](#) guid, [IRequestContextPtr](#) ctx)
- void [findNotesMetadataRequest](#) ([NoteFilter](#) filter, qint32 offset, qint32 maxNotes, [NotesMetadataResultSpec](#) resultSpec, [IRequestContextPtr](#) ctx)
- void [findNoteCountsRequest](#) ([NoteFilter](#) filter, bool withTrash, [IRequestContextPtr](#) ctx)
- void [getNoteWithResultSpecRequest](#) ([Guid](#) guid, [NoteResultSpec](#) resultSpec, [IRequestContextPtr](#) ctx)

- void [getNoteRequest](#) (Guid guid, bool withContent, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [IRequestContextPtr](#) ctx)
- void [getNoteApplicationDataRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getNoteApplicationDataEntryRequest](#) (Guid guid, QString key, [IRequestContextPtr](#) ctx)
- void [setNoteApplicationDataEntryRequest](#) (Guid guid, QString key, QString value, [IRequestContextPtr](#) ctx)
- void [unsetNoteApplicationDataEntryRequest](#) (Guid guid, QString key, [IRequestContextPtr](#) ctx)
- void [getNoteContentRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getNoteSearchTextRequest](#) (Guid guid, bool noteOnly, bool tokenizeForIndexing, [IRequestContextPtr](#) ctx)
- void [getResourceSearchTextRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getNoteTagNamesRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [createNoteRequest](#) (Note note, [IRequestContextPtr](#) ctx)
- void [updateNoteRequest](#) (Note note, [IRequestContextPtr](#) ctx)
- void [deleteNoteRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [expungeNoteRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [copyNoteRequest](#) (Guid noteGuid, Guid toNotebookGuid, [IRequestContextPtr](#) ctx)
- void [listNoteVersionsRequest](#) (Guid noteGuid, [IRequestContextPtr](#) ctx)
- void [getNoteVersionRequest](#) (Guid noteGuid, qint32 updateSequenceNum, bool withResourcesData, bool withResourcesRecognition, bool withResourcesAlternateData, [IRequestContextPtr](#) ctx)
- void [getResourceRequest](#) (Guid guid, bool withData, bool withRecognition, bool withAttributes, bool withAlternateData, [IRequestContextPtr](#) ctx)
- void [getResourceApplicationDataRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getResourceApplicationDataEntryRequest](#) (Guid guid, QString key, [IRequestContextPtr](#) ctx)
- void [setResourceApplicationDataEntryRequest](#) (Guid guid, QString key, QString value, [IRequestContextPtr](#) ctx)
- void [unsetResourceApplicationDataEntryRequest](#) (Guid guid, QString key, [IRequestContextPtr](#) ctx)
- void [updateResourceRequest](#) (Resource resource, [IRequestContextPtr](#) ctx)
- void [getResourceDataRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getResourceByHashRequest](#) (Guid noteGuid, QByteArray contentHash, bool withData, bool withRecognition, bool withAlternateData, [IRequestContextPtr](#) ctx)
- void [getResourceRecognitionRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getResourceAlternateDataRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getResourceAttributesRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [getPublicNotebookRequest](#) (UserID userId, QString publicUri, [IRequestContextPtr](#) ctx)
- void [shareNotebookRequest](#) (SharedNotebook sharedNotebook, QString message, [IRequestContextPtr](#) ctx)
- void [createOrUpdateNotebookSharesRequest](#) (NotebookShareTemplate shareTemplate, [IRequestContextPtr](#) ctx)
- void [updateSharedNotebookRequest](#) (SharedNotebook sharedNotebook, [IRequestContextPtr](#) ctx)
- void [setNotebookRecipientSettingsRequest](#) (QString notebookGuid, NotebookRecipientSettings recipientSettings, [IRequestContextPtr](#) ctx)
- void [listSharedNotebooksRequest](#) ([IRequestContextPtr](#) ctx)
- void [createLinkedNotebookRequest](#) (LinkedNotebook linkedNotebook, [IRequestContextPtr](#) ctx)
- void [updateLinkedNotebookRequest](#) (LinkedNotebook linkedNotebook, [IRequestContextPtr](#) ctx)
- void [listLinkedNotebooksRequest](#) ([IRequestContextPtr](#) ctx)
- void [expungeLinkedNotebookRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [authenticateToSharedNotebookRequest](#) (QString shareKeyOrGlobalId, [IRequestContextPtr](#) ctx)
- void [getSharedNotebookByAuthRequest](#) ([IRequestContextPtr](#) ctx)
- void [emailNoteRequest](#) (NoteEmailParameters parameters, [IRequestContextPtr](#) ctx)
- void [shareNoteRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [stopSharingNoteRequest](#) (Guid guid, [IRequestContextPtr](#) ctx)
- void [authenticateToSharedNoteRequest](#) (QString guid, QString noteKey, [IRequestContextPtr](#) ctx)
- void [findRelatedRequest](#) (RelatedQuery query, RelatedResultSpec resultSpec, [IRequestContextPtr](#) ctx)
- void [updateNoteIfUsnMatchesRequest](#) (Note note, [IRequestContextPtr](#) ctx)
- void [manageNotebookSharesRequest](#) (ManageNotebookSharesParameters parameters, [IRequestContextPtr](#) ctx)

- void [getNotebookSharesRequest](#) (QString notebookGuid, [IRequestContextPtr](#) ctx)
- void [getSyncStateRequestReady](#) (QByteArray data)
- void [getFilteredSyncChunkRequestReady](#) (QByteArray data)
- void [getLinkedNotebookSyncStateRequestReady](#) (QByteArray data)
- void [getLinkedNotebookSyncChunkRequestReady](#) (QByteArray data)
- void [listNotebooksRequestReady](#) (QByteArray data)
- void [listAccessibleBusinessNotebooksRequestReady](#) (QByteArray data)
- void [getNotebookRequestReady](#) (QByteArray data)
- void [getDefaultNotebookRequestReady](#) (QByteArray data)
- void [createNotebookRequestReady](#) (QByteArray data)
- void [updateNotebookRequestReady](#) (QByteArray data)
- void [expungeNotebookRequestReady](#) (QByteArray data)
- void [listTagsRequestReady](#) (QByteArray data)
- void [listTagsByNotebookRequestReady](#) (QByteArray data)
- void [getTagRequestReady](#) (QByteArray data)
- void [createTagRequestReady](#) (QByteArray data)
- void [updateTagRequestReady](#) (QByteArray data)
- void [untagAllRequestReady](#) (QByteArray data)
- void [expungeTagRequestReady](#) (QByteArray data)
- void [listSearchesRequestReady](#) (QByteArray data)
- void [getSearchRequestReady](#) (QByteArray data)
- void [createSearchRequestReady](#) (QByteArray data)
- void [updateSearchRequestReady](#) (QByteArray data)
- void [expungeSearchRequestReady](#) (QByteArray data)
- void [findNoteOffsetRequestReady](#) (QByteArray data)
- void [findNotesMetadataRequestReady](#) (QByteArray data)
- void [findNoteCountsRequestReady](#) (QByteArray data)
- void [getNoteWithResultSpecRequestReady](#) (QByteArray data)
- void [getNoteRequestReady](#) (QByteArray data)
- void [getNoteApplicationDataRequestReady](#) (QByteArray data)
- void [getNoteApplicationDataEntryRequestReady](#) (QByteArray data)
- void [setNoteApplicationDataEntryRequestReady](#) (QByteArray data)
- void [unsetNoteApplicationDataEntryRequestReady](#) (QByteArray data)
- void [getNoteContentRequestReady](#) (QByteArray data)
- void [getNoteSearchTextRequestReady](#) (QByteArray data)
- void [getResourceSearchTextRequestReady](#) (QByteArray data)
- void [getNoteTagNamesRequestReady](#) (QByteArray data)
- void [createNoteRequestReady](#) (QByteArray data)
- void [updateNoteRequestReady](#) (QByteArray data)
- void [deleteNoteRequestReady](#) (QByteArray data)
- void [expungeNoteRequestReady](#) (QByteArray data)
- void [copyNoteRequestReady](#) (QByteArray data)
- void [listNoteVersionsRequestReady](#) (QByteArray data)
- void [getNoteVersionRequestReady](#) (QByteArray data)
- void [getResourceRequestReady](#) (QByteArray data)
- void [getResourceApplicationDataRequestReady](#) (QByteArray data)
- void [getResourceApplicationDataEntryRequestReady](#) (QByteArray data)
- void [setResourceApplicationDataEntryRequestReady](#) (QByteArray data)
- void [unsetResourceApplicationDataEntryRequestReady](#) (QByteArray data)
- void [updateResourceRequestReady](#) (QByteArray data)
- void [getResourceDataRequestReady](#) (QByteArray data)
- void [getResourceByHashRequestReady](#) (QByteArray data)
- void [getResourceRecognitionRequestReady](#) (QByteArray data)
- void [getResourceAlternateDataRequestReady](#) (QByteArray data)
- void [getResourceAttributesRequestReady](#) (QByteArray data)

- void [getPublicNotebookRequestReady](#) (QByteArray data)
- void [shareNotebookRequestReady](#) (QByteArray data)
- void [createOrUpdateNotebookSharesRequestReady](#) (QByteArray data)
- void [updateSharedNotebookRequestReady](#) (QByteArray data)
- void [setNotebookRecipientSettingsRequestReady](#) (QByteArray data)
- void [listSharedNotebooksRequestReady](#) (QByteArray data)
- void [createLinkedNotebookRequestReady](#) (QByteArray data)
- void [updateLinkedNotebookRequestReady](#) (QByteArray data)
- void [listLinkedNotebooksRequestReady](#) (QByteArray data)
- void [expungeLinkedNotebookRequestReady](#) (QByteArray data)
- void [authenticateToSharedNotebookRequestReady](#) (QByteArray data)
- void [getSharedNotebookByAuthRequestReady](#) (QByteArray data)
- void [emailNoteRequestReady](#) (QByteArray data)
- void [shareNoteRequestReady](#) (QByteArray data)
- void [stopSharingNoteRequestReady](#) (QByteArray data)
- void [authenticateToSharedNoteRequestReady](#) (QByteArray data)
- void [findRelatedRequestReady](#) (QByteArray data)
- void [updateNoteIfUsnMatchesRequestReady](#) (QByteArray data)
- void [manageNotebookSharesRequestReady](#) (QByteArray data)
- void [getNotebookSharesRequestReady](#) (QByteArray data)

## Public Member Functions

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

### 7.80.1 Detailed Description

The [NoteStoreServer](#) class represents customizable server for NoteStore requests. It is primarily used for testing of QEverCloud.

### 7.80.2 Constructor & Destructor Documentation

#### 7.80.2.1 NoteStoreServer()

```
qevercloud::NoteStoreServer::NoteStoreServer (
    QObject * parent = nullptr ) [explicit]
```

### 7.80.3 Member Function Documentation

#### 7.80.3.1 authenticateToSharedNotebookRequest

```
void qevercloud::NoteStoreServer::authenticateToSharedNotebookRequest (
    QString shareKeyOrGlobalId,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.2 authenticateToSharedNotebookRequestReady

```
void qevercloud::NoteStoreServer::authenticateToSharedNotebookRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.3 authenticateToSharedNoteRequest

```
void qevercloud::NoteStoreServer::authenticateToSharedNoteRequest (
    QString guid,
    QString noteKey,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.4 authenticateToSharedNoteRequestReady

```
void qevercloud::NoteStoreServer::authenticateToSharedNoteRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.5 copyNoteRequest

```
void qevercloud::NoteStoreServer::copyNoteRequest (
    Guid noteGuid,
    Guid toNotebookGuid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.6 copyNoteRequestReady

```
void qevercloud::NoteStoreServer::copyNoteRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.7 createLinkedNotebookRequest

```
void qevercloud::NoteStoreServer::createLinkedNotebookRequest (
    LinkedNotebook linkedNotebook,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.8 createLinkedNotebookRequestReady

```
void qevercloud::NoteStoreServer::createLinkedNotebookRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.9 createNotebookRequest

```
void qevercloud::NoteStoreServer::createNotebookRequest (
    Notebook notebook,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.10 createNotebookRequestReady

```
void qevercloud::NoteStoreServer::createNotebookRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.11 createNoteRequest

```
void qevercloud::NoteStoreServer::createNoteRequest (
    Note note,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.12 createNoteRequestReady

```
void qevercloud::NoteStoreServer::createNoteRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.13 createOrUpdateNotebookSharesRequest

```
void qevercloud::NoteStoreServer::createOrUpdateNotebookSharesRequest (
    NotebookShareTemplate shareTemplate,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.14 createOrUpdateNotebookSharesRequestReady

```
void qevercloud::NoteStoreServer::createOrUpdateNotebookSharesRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.15 createSearchRequest

```
void qevercloud::NoteStoreServer::createSearchRequest (
    SavedSearch search,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.16 createSearchRequestReady

```
void qevercloud::NoteStoreServer::createSearchRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.17 createTagRequest

```
void qevercloud::NoteStoreServer::createTagRequest (
    Tag tag,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.18 createTagRequestReady

```
void qevercloud::NoteStoreServer::createTagRequestReady (
    QByteArray data ) [signal]
```



#### 7.80.3.19 deleteNoteRequest

```
void qevercloud::NoteStoreServer::deleteNoteRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.20 deleteNoteRequestReady

```
void qevercloud::NoteStoreServer::deleteNoteRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.21 emailNoteRequest

```
void qevercloud::NoteStoreServer::emailNoteRequest (
    NoteEmailParameters parameters,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.22 emailNoteRequestReady

```
void qevercloud::NoteStoreServer::emailNoteRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.23 expungeLinkedNotebookRequest

```
void qevercloud::NoteStoreServer::expungeLinkedNotebookRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.24 expungeLinkedNotebookRequestReady

```
void qevercloud::NoteStoreServer::expungeLinkedNotebookRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.25 expungeNotebookRequest

```
void qevercloud::NoteStoreServer::expungeNotebookRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.26 expungeNotebookRequestReady

```
void qevercloud::NoteStoreServer::expungeNotebookRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.27 expungeNoteRequest

```
void qevercloud::NoteStoreServer::expungeNoteRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.28 expungeNoteRequestReady

```
void qevercloud::NoteStoreServer::expungeNoteRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.29 expungeSearchRequest

```
void qevercloud::NoteStoreServer::expungeSearchRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.30 expungeSearchRequestReady

```
void qevercloud::NoteStoreServer::expungeSearchRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.31 expungeTagRequest

```
void qevercloud::NoteStoreServer::expungeTagRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.32 expungeTagRequestReady

```
void qevercloud::NoteStoreServer::expungeTagRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.33 findNoteCountsRequest

```
void qevercloud::NoteStoreServer::findNoteCountsRequest (
    NoteFilter filter,
    bool withTrash,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.34 findNoteCountsRequestReady

```
void qevercloud::NoteStoreServer::findNoteCountsRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.35 findNoteOffsetRequest

```
void qevercloud::NoteStoreServer::findNoteOffsetRequest (
    NoteFilter filter,
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.36 findNoteOffsetRequestReady

```
void qevercloud::NoteStoreServer::findNoteOffsetRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.37 findNotesMetadataRequest

```
void qevercloud::NoteStoreServer::findNotesMetadataRequest (
    NoteFilter filter,
    qint32 offset,
    qint32 maxNotes,
    NotesMetadataResultSpec resultSpec,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.38 findNotesMetadataRequestReady

```
void qevercloud::NoteStoreServer::findNotesMetadataRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.39 findRelatedRequest

```
void qevercloud::NoteStoreServer::findRelatedRequest (
    RelatedQuery query,
    RelatedResultSpec resultSpec,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.40 findRelatedRequestReady

```
void qevercloud::NoteStoreServer::findRelatedRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.41 getDefaultNotebookRequest

```
void qevercloud::NoteStoreServer::getDefaultNotebookRequest (
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.42 getDefaultNotebookRequestReady

```
void qevercloud::NoteStoreServer::getDefaultNotebookRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.43 getFilteredSyncChunkRequest**

```
void qevercloud::NoteStoreServer::getFilteredSyncChunkRequest (
    qint32 afterUSN,
    qint32 maxEntries,
    SyncChunkFilter filter,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.44 getFilteredSyncChunkRequestReady**

```
void qevercloud::NoteStoreServer::getFilteredSyncChunkRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.45 getLinkedNotebookSyncChunkRequest**

```
void qevercloud::NoteStoreServer::getLinkedNotebookSyncChunkRequest (
    LinkedNotebook linkedNotebook,
    qint32 afterUSN,
    qint32 maxEntries,
    bool fullSyncOnly,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.46 getLinkedNotebookSyncChunkRequestReady**

```
void qevercloud::NoteStoreServer::getLinkedNotebookSyncChunkRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.47 getLinkedNotebookSyncStateRequest**

```
void qevercloud::NoteStoreServer::getLinkedNotebookSyncStateRequest (
    LinkedNotebook linkedNotebook,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.48 getLinkedNotebookSyncStateRequestReady**

```
void qevercloud::NoteStoreServer::getLinkedNotebookSyncStateRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.49 getNoteApplicationDataEntryRequest

```
void qevercloud::NoteStoreServer::getNoteApplicationDataEntryRequest (
    Guid guid,
    QString key,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.50 getNoteApplicationDataEntryRequestReady

```
void qevercloud::NoteStoreServer::getNoteApplicationDataEntryRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.51 getNoteApplicationDataRequest

```
void qevercloud::NoteStoreServer::getNoteApplicationDataRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.52 getNoteApplicationDataRequestReady

```
void qevercloud::NoteStoreServer::getNoteApplicationDataRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.53 getNotebookRequest

```
void qevercloud::NoteStoreServer::getNotebookRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.54 getNotebookRequestReady

```
void qevercloud::NoteStoreServer::getNotebookRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.55 getNotebookSharesRequest**

```
void qevercloud::NoteStoreServer::getNotebookSharesRequest (
    QString notebookGuid,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.56 getNotebookSharesRequestReady**

```
void qevercloud::NoteStoreServer::getNotebookSharesRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.57 getNoteContentRequest**

```
void qevercloud::NoteStoreServer::getNoteContentRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.58 getNoteContentRequestReady**

```
void qevercloud::NoteStoreServer::getNoteContentRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.59 getNoteRequest**

```
void qevercloud::NoteStoreServer::getNoteRequest (
    Guid guid,
    bool withContent,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.60 getNoteRequestReady**

```
void qevercloud::NoteStoreServer::getNoteRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.61 `getNoteSearchTextRequest`

```
void qevercloud::NoteStoreServer::getNoteSearchTextRequest (
    Guid guid,
    bool noteOnly,
    bool tokenizeForIndexing,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.62 `getNoteSearchTextRequestReady`

```
void qevercloud::NoteStoreServer::getNoteSearchTextRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.63 `getNoteTagNamesRequest`

```
void qevercloud::NoteStoreServer::getNoteTagNamesRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.64 `getNoteTagNamesRequestReady`

```
void qevercloud::NoteStoreServer::getNoteTagNamesRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.65 `getNoteVersionRequest`

```
void qevercloud::NoteStoreServer::getNoteVersionRequest (
    Guid noteGuid,
    quint32 updateSequenceNum,
    bool withResourcesData,
    bool withResourcesRecognition,
    bool withResourcesAlternateData,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.66 `getNoteVersionRequestReady`

```
void qevercloud::NoteStoreServer::getNoteVersionRequestReady (
    QByteArray data ) [signal]
```



**7.80.3.67 getNoteWithResultSpecRequest**

```
void qevercloud::NoteStoreServer::getNoteWithResultSpecRequest (
    Guid guid,
    NoteResultSpec resultSpec,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.68 getNoteWithResultSpecRequestReady**

```
void qevercloud::NoteStoreServer::getNoteWithResultSpecRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.69 getPublicNotebookRequest**

```
void qevercloud::NoteStoreServer::getPublicNotebookRequest (
    UserID userId,
    QString publicUri,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.70 getPublicNotebookRequestReady**

```
void qevercloud::NoteStoreServer::getPublicNotebookRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.71 getResourceAlternateDataRequest**

```
void qevercloud::NoteStoreServer::getResourceAlternateDataRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.72 getResourceAlternateDataRequestReady**

```
void qevercloud::NoteStoreServer::getResourceAlternateDataRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.73 getResourceApplicationDataEntryRequest

```
void qevercloud::NoteStoreServer::getResourceApplicationDataEntryRequest (
    Guid guid,
    QString key,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.74 getResourceApplicationDataEntryRequestReady

```
void qevercloud::NoteStoreServer::getResourceApplicationDataEntryRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.75 getResourceApplicationDataRequest

```
void qevercloud::NoteStoreServer::getResourceApplicationDataRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.76 getResourceApplicationDataRequestReady

```
void qevercloud::NoteStoreServer::getResourceApplicationDataRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.77 getResourceAttributesRequest

```
void qevercloud::NoteStoreServer::getResourceAttributesRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.78 getResourceAttributesRequestReady

```
void qevercloud::NoteStoreServer::getResourceAttributesRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.79 getResourceByHashRequest

```
void qevercloud::NoteStoreServer::getResourceByHashRequest (
    Guid noteGuid,
    QByteArray contentHash,
    bool withData,
    bool withRecognition,
    bool withAlternateData,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.80 getResourceByHashRequestReady

```
void qevercloud::NoteStoreServer::getResourceByHashRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.81 getResourceDataRequest

```
void qevercloud::NoteStoreServer::getResourceDataRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.82 getResourceDataRequestReady

```
void qevercloud::NoteStoreServer::getResourceDataRequestReady (
    QByteArray data ) [signal]
```

### 7.80.3.83 getResourceRecognitionRequest

```
void qevercloud::NoteStoreServer::getResourceRecognitionRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

### 7.80.3.84 getResourceRecognitionRequestReady

```
void qevercloud::NoteStoreServer::getResourceRecognitionRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.85 getResourceRequest

```
void qevercloud::NoteStoreServer::getResourceRequest (
    Guid guid,
    bool withData,
    bool withRecognition,
    bool withAttributes,
    bool withAlternateData,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.86 getResourceRequestReady

```
void qevercloud::NoteStoreServer::getResourceRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.87 getResourceSearchTextRequest

```
void qevercloud::NoteStoreServer::getResourceSearchTextRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.88 getResourceSearchTextRequestReady

```
void qevercloud::NoteStoreServer::getResourceSearchTextRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.89 getSearchRequest

```
void qevercloud::NoteStoreServer::getSearchRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.90 getSearchRequestReady

```
void qevercloud::NoteStoreServer::getSearchRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.91 getSharedNotebookByAuthRequest**

```
void qevercloud::NoteStoreServer::getSharedNotebookByAuthRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.92 getSharedNotebookByAuthRequestReady**

```
void qevercloud::NoteStoreServer::getSharedNotebookByAuthRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.93 getSyncStateRequest**

```
void qevercloud::NoteStoreServer::getSyncStateRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.94 getSyncStateRequestReady**

```
void qevercloud::NoteStoreServer::getSyncStateRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.95 getTagRequest**

```
void qevercloud::NoteStoreServer::getTagRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.96 getTagRequestReady**

```
void qevercloud::NoteStoreServer::getTagRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.97 listAccessibleBusinessNotebooksRequest**

```
void qevercloud::NoteStoreServer::listAccessibleBusinessNotebooksRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.98 listAccessibleBusinessNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::listAccessibleBusinessNotebooksRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.99 listLinkedNotebooksRequest**

```
void qevercloud::NoteStoreServer::listLinkedNotebooksRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.100 listLinkedNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::listLinkedNotebooksRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.101 listNotebooksRequest**

```
void qevercloud::NoteStoreServer::listNotebooksRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.102 listNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::listNotebooksRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.103 listNoteVersionsRequest**

```
void qevercloud::NoteStoreServer::listNoteVersionsRequest (
    Guid noteGuid,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.104 listNoteVersionsRequestReady**

```
void qevercloud::NoteStoreServer::listNoteVersionsRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.105 listSearchesRequest**

```
void qevercloud::NoteStoreServer::listSearchesRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.106 listSearchesRequestReady**

```
void qevercloud::NoteStoreServer::listSearchesRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.107 listSharedNotebooksRequest**

```
void qevercloud::NoteStoreServer::listSharedNotebooksRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.108 listSharedNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::listSharedNotebooksRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.109 listTagsByNotebookRequest**

```
void qevercloud::NoteStoreServer::listTagsByNotebookRequest (
    Guid notebookGuid,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.110 listTagsByNotebookRequestReady**

```
void qevercloud::NoteStoreServer::listTagsByNotebookRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.111 listTagsRequest**

```
void qevercloud::NoteStoreServer::listTagsRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.112 listTagsRequestReady**

```
void qevercloud::NoteStoreServer::listTagsRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.113 manageNotebookSharesRequest**

```
void qevercloud::NoteStoreServer::manageNotebookSharesRequest (
    ManageNotebookSharesParameters parameters,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.114 manageNotebookSharesRequestReady**

```
void qevercloud::NoteStoreServer::manageNotebookSharesRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.115 onAuthenticateToSharedNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onAuthenticateToSharedNotebookRequestReady (
    AuthenticationResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.116 onAuthenticateToSharedNoteRequestReady**

```
void qevercloud::NoteStoreServer::onAuthenticateToSharedNoteRequestReady (
    AuthenticationResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.117 onCopyNoteRequestReady**

```
void qevercloud::NoteStoreServer::onCopyNoteRequestReady (
    Note value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```



**7.80.3.118 onCreateLinkedNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onCreateLinkedNotebookRequestReady (
    LinkedNotebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.119 onCreateNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onCreateNotebookRequestReady (
    Notebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.120 onCreateNoteRequestReady**

```
void qevercloud::NoteStoreServer::onCreateNoteRequestReady (
    Note value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.121 onCreateOrUpdateNotebookSharesRequestReady**

```
void qevercloud::NoteStoreServer::onCreateOrUpdateNotebookSharesRequestReady (
    CreateOrUpdateNotebookSharesResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.122 onCreateSearchRequestReady**

```
void qevercloud::NoteStoreServer::onCreateSearchRequestReady (
    SavedSearch value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.123 onCreateTagRequestReady**

```
void qevercloud::NoteStoreServer::onCreateTagRequestReady (
    Tag value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.124 onDeleteNoteRequestReady**

```
void qevercloud::NoteStoreServer::onDeleteNoteRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.125 onEmailNoteRequestReady**

```
void qevercloud::NoteStoreServer::onEmailNoteRequestReady (
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.126 onExpungeLinkedNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onExpungeLinkedNotebookRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.127 onExpungeNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onExpungeNotebookRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.128 onExpungeNoteRequestReady**

```
void qevercloud::NoteStoreServer::onExpungeNoteRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.129 onExpungeSearchRequestReady**

```
void qevercloud::NoteStoreServer::onExpungeSearchRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.130 onExpungeTagRequestReady**

```
void qevercloud::NoteStoreServer::onExpungeTagRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.131 onFindNoteCountsRequestReady**

```
void qevercloud::NoteStoreServer::onFindNoteCountsRequestReady (
    NoteCollectionCounts value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.132 onFindNoteOffsetRequestReady**

```
void qevercloud::NoteStoreServer::onFindNoteOffsetRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.133 onFindNotesMetadataRequestReady**

```
void qevercloud::NoteStoreServer::onFindNotesMetadataRequestReady (
    NotesMetadataList value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.134 onFindRelatedRequestReady**

```
void qevercloud::NoteStoreServer::onFindRelatedRequestReady (
    RelatedResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.135 onGetDefaultNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onGetDefaultNotebookRequestReady (
    Notebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.136 onGetFilteredSyncChunkRequestReady**

```
void qevercloud::NoteStoreServer::onGetFilteredSyncChunkRequestReady (
    SyncChunk value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.137 onGetLinkedNotebookSyncChunkRequestReady**

```
void qevercloud::NoteStoreServer::onGetLinkedNotebookSyncChunkRequestReady (
    SyncChunk value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.138 onGetLinkedNotebookSyncStateRequestReady**

```
void qevercloud::NoteStoreServer::onGetLinkedNotebookSyncStateRequestReady (
    SyncState value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.139 onGetNoteApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteApplicationDataEntryRequestReady (
    QString value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.140 onGetNoteApplicationDataRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteApplicationDataRequestReady (
    LazyMap value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.141 onGetNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onGetNotebookRequestReady (
    Notebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.142 onGetNotebookSharesRequestReady**

```
void qevercloud::NoteStoreServer::onGetNotebookSharesRequestReady (
    ShareRelationships value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.143 onGetNoteContentRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteContentRequestReady (
    QString value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.144 onGetNoteRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteRequestReady (
    Note value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.145 onGetNoteSearchTextRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteSearchTextRequestReady (
    QString value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.146 onGetNoteTagNamesRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteTagNamesRequestReady (
    QStringList value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.147 onGetNoteVersionRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteVersionRequestReady (
    Note value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.148 onGetNoteWithResultSpecRequestReady**

```
void qevercloud::NoteStoreServer::onGetNoteWithResultSpecRequestReady (
    Note value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.149 onGetPublicNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onGetPublicNotebookRequestReady (
    Notebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.150 onGetResourceAlternateDataRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceAlternateDataRequestReady (
    QByteArray value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.151 onGetResourceApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceApplicationDataEntryRequestReady (
    QString value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.152 onGetResourceApplicationDataRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceApplicationDataRequestReady (
    LazyMap value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.153 onGetResourceAttributesRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceAttributesRequestReady (
    ResourceAttributes value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.154 onGetResourceByHashRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceByHashRequestReady (
    Resource value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.155 onGetResourceDataRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceDataRequestReady (
    QByteArray value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.156 onGetResourceRecognitionRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceRecognitionRequestReady (
    QByteArray value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.157 onGetResourceRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceRequestReady (
    Resource value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.158 onGetResourceSearchTextRequestReady**

```
void qevercloud::NoteStoreServer::onGetResourceSearchTextRequestReady (
    QString value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.159 onGetSearchRequestReady**

```
void qevercloud::NoteStoreServer::onGetSearchRequestReady (
    SavedSearch value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.160 onGetSharedNotebookByAuthRequestReady**

```
void qevercloud::NoteStoreServer::onGetSharedNotebookByAuthRequestReady (
    SharedNotebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.161 onGetSyncStateRequestReady**

```
void qevercloud::NoteStoreServer::onGetSyncStateRequestReady (
    SyncState value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.162 onGetTagRequestReady**

```
void qevercloud::NoteStoreServer::onGetTagRequestReady (
    Tag value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.163 onListAccessibleBusinessNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::onListAccessibleBusinessNotebooksRequestReady (
    QList< Notebook > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.164 onListLinkedNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::onListLinkedNotebooksRequestReady (
    QList< LinkedNotebook > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.165 onListNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::onListNotebooksRequestReady (
    QList< Notebook > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```



**7.80.3.166 onListNoteVersionsRequestReady**

```
void qevercloud::NoteStoreServer::onListNoteVersionsRequestReady (
    QList< NoteVersionId > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.167 onListSearchesRequestReady**

```
void qevercloud::NoteStoreServer::onListSearchesRequestReady (
    QList< SavedSearch > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.168 onListSharedNotebooksRequestReady**

```
void qevercloud::NoteStoreServer::onListSharedNotebooksRequestReady (
    QList< SharedNotebook > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.169 onListTagsByNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onListTagsByNotebookRequestReady (
    QList< Tag > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.170 onListTagsRequestReady**

```
void qevercloud::NoteStoreServer::onListTagsRequestReady (
    QList< Tag > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.171 onManageNotebookSharesRequestReady**

```
void qevercloud::NoteStoreServer::onManageNotebookSharesRequestReady (
    ManageNotebookSharesResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.172 onRequest**

```
void qevercloud::NoteStoreServer::onRequest (
    QByteArray data ) [slot]
```

**7.80.3.173 onSetNoteApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::onSetNoteApplicationDataEntryRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.174 onSetNotebookRecipientSettingsRequestReady**

```
void qevercloud::NoteStoreServer::onSetNotebookRecipientSettingsRequestReady (
    Notebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.175 onSetResourceApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::onSetResourceApplicationDataEntryRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.176 onShareNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onShareNotebookRequestReady (
    SharedNotebook value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.177 onShareNoteRequestReady**

```
void qevercloud::NoteStoreServer::onShareNoteRequestReady (
    QString value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.178 onStopSharingNoteRequestReady**

```
void qevercloud::NoteStoreServer::onStopSharingNoteRequestReady (
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.179 onUnsetNoteApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::onUnsetNoteApplicationDataEntryRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.180 onUnsetResourceApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::onUnsetResourceApplicationDataEntryRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.181 onUntagAllRequestReady**

```
void qevercloud::NoteStoreServer::onUntagAllRequestReady (
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.182 onUpdateLinkedNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateLinkedNotebookRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.183 onUpdateNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateNotebookRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.184 onUpdateNoteIfUsnMatchesRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateNoteIfUsnMatchesRequestReady (
    UpdateNoteIfUsnMatchesResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.185 onUpdateNoteRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateNoteRequestReady (
    Note value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.186 onUpdateResourceRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateResourceRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.187 onUpdateSearchRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateSearchRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.188 onUpdateSharedNotebookRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateSharedNotebookRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.189 onUpdateTagRequestReady**

```
void qevercloud::NoteStoreServer::onUpdateTagRequestReady (
    qint32 value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.80.3.190 setNoteApplicationDataEntryRequest**

```
void qevercloud::NoteStoreServer::setNoteApplicationDataEntryRequest (
    Guid guid,
    QString key,
    QString value,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.191 setNoteApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::setNoteApplicationDataEntryRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.192 setNotebookRecipientSettingsRequest**

```
void qevercloud::NoteStoreServer::setNotebookRecipientSettingsRequest (
    QString notebookGuid,
    NotebookRecipientSettings recipientSettings,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.193 setNotebookRecipientSettingsRequestReady**

```
void qevercloud::NoteStoreServer::setNotebookRecipientSettingsRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.194 setResourceApplicationDataEntryRequest**

```
void qevercloud::NoteStoreServer::setResourceApplicationDataEntryRequest (
    Guid guid,
    QString key,
    QString value,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.195 setResourceApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::setResourceApplicationDataEntryRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.196 shareNotebookRequest

```
void qevercloud::NoteStoreServer::shareNotebookRequest (
    SharedNotebook sharedNotebook,
    QString message,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.197 shareNotebookRequestReady

```
void qevercloud::NoteStoreServer::shareNotebookRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.198 shareNoteRequest

```
void qevercloud::NoteStoreServer::shareNoteRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.199 shareNoteRequestReady

```
void qevercloud::NoteStoreServer::shareNoteRequestReady (
    QByteArray data ) [signal]
```

#### 7.80.3.200 stopSharingNoteRequest

```
void qevercloud::NoteStoreServer::stopSharingNoteRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

#### 7.80.3.201 stopSharingNoteRequestReady

```
void qevercloud::NoteStoreServer::stopSharingNoteRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.202 unsetNoteApplicationDataEntryRequest**

```
void qevercloud::NoteStoreServer::unsetNoteApplicationDataEntryRequest (
    Guid guid,
    QString key,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.203 unsetNoteApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::unsetNoteApplicationDataEntryRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.204 unsetResourceApplicationDataEntryRequest**

```
void qevercloud::NoteStoreServer::unsetResourceApplicationDataEntryRequest (
    Guid guid,
    QString key,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.205 unsetResourceApplicationDataEntryRequestReady**

```
void qevercloud::NoteStoreServer::unsetResourceApplicationDataEntryRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.206 untagAllRequest**

```
void qevercloud::NoteStoreServer::untagAllRequest (
    Guid guid,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.207 untagAllRequestReady**

```
void qevercloud::NoteStoreServer::untagAllRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.208 updateLinkedNotebookRequest**

```
void qevercloud::NoteStoreServer::updateLinkedNotebookRequest (
    LinkedNotebook linkedNotebook,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.209 updateLinkedNotebookRequestReady**

```
void qevercloud::NoteStoreServer::updateLinkedNotebookRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.210 updateNotebookRequest**

```
void qevercloud::NoteStoreServer::updateNotebookRequest (
    Notebook notebook,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.211 updateNotebookRequestReady**

```
void qevercloud::NoteStoreServer::updateNotebookRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.212 updateNoteIfUsnMatchesRequest**

```
void qevercloud::NoteStoreServer::updateNoteIfUsnMatchesRequest (
    Note note,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.213 updateNoteIfUsnMatchesRequestReady**

```
void qevercloud::NoteStoreServer::updateNoteIfUsnMatchesRequestReady (
    QByteArray data ) [signal]
```



**7.80.3.214 updateNoteRequest**

```
void qevercloud::NoteStoreServer::updateNoteRequest (
    Note note,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.215 updateNoteRequestReady**

```
void qevercloud::NoteStoreServer::updateNoteRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.216 updateResourceRequest**

```
void qevercloud::NoteStoreServer::updateResourceRequest (
    Resource resource,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.217 updateResourceRequestReady**

```
void qevercloud::NoteStoreServer::updateResourceRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.218 updateSearchRequest**

```
void qevercloud::NoteStoreServer::updateSearchRequest (
    SavedSearch search,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.219 updateSearchRequestReady**

```
void qevercloud::NoteStoreServer::updateSearchRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.220 updateSharedNotebookRequest**

```
void qevercloud::NoteStoreServer::updateSharedNotebookRequest (
    SharedNotebook sharedNotebook,
    IRequestContextPtr ctx ) [signal]
```

**7.80.3.221 updateSharedNotebookRequestReady**

```
void qevercloud::NoteStoreServer::updateSharedNotebookRequestReady (
    QByteArray data ) [signal]
```

**7.80.3.222 updateTagRequest**

```
void qevercloud::NoteStoreServer::updateTagRequest (
    Tag tag,
    IRequestContextPtr ctx ) [signal]
```

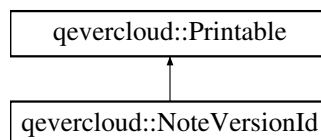
**7.80.3.223 updateTagRequestReady**

```
void qevercloud::NoteStoreServer::updateTagRequestReady (
    QByteArray data ) [signal]
```

**7.81 qevercloud::NoteVersionId Struct Reference**

```
#include <Types.h>
```

Inheritance diagram for qevercloud::NoteVersionId:

**Public Member Functions**

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [NoteVersionId](#) &other) const
- bool [operator!=](#) (const [NoteVersionId](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- `qint32` `updateSequenceNum` = 0
- `Timestamp` `updated` = 0
- `Timestamp` `saved` = 0
- `QString` `title`
- `Optional`< `UserID` > `lastEditorId`

### 7.81.1 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.

### 7.81.2 Member Function Documentation

#### 7.81.2.1 `operator!=()`

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

#### 7.81.2.2 `operator==()`

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

#### 7.81.2.3 `print()`

```
virtual void qevercloud::NoteVersionId::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.81.3 Member Data Documentation

#### 7.81.3.1 lastEditorId

```
Optional< UserID > qevercloud::NoteVersionId::lastEditorId
```

The ID of the user who made the change to this version of the note. This will be unset if the note version was edited by the owner of the account.

#### 7.81.3.2 localData

```
EverCloudLocalData qevercloud::NoteVersionId::localData
```

See the declaration of [EverCloudLocalData](#) for details

#### 7.81.3.3 saved

```
Timestamp qevercloud::NoteVersionId::saved = 0
```

A timestamp that holds the date and time when this version of the note was backed up by Evernote's servers.

#### 7.81.3.4 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.)

#### 7.81.3.5 updated

```
Timestamp qevercloud::NoteVersionId::updated = 0
```

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.)

#### 7.81.3.6 updateSequenceNum

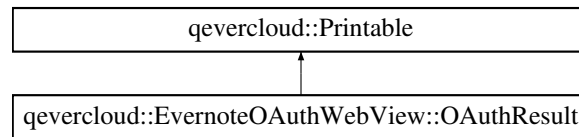
```
qint32 qevercloud::NoteVersionId::updateSequenceNum = 0
```

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.

## 7.82 qevercloud::EvernoteOAuthWebView::OAuthResult Struct Reference

```
#include <OAuth.h>
```

Inheritance diagram for qevercloud::EvernoteOAuthWebView::OAuthResult:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override

### Public Attributes

- QString [noteStoreUrl](#)
- [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!*
- QList< QNetworkCookie > [cookies](#)

### 7.82.1 Detailed Description

Holds data that is returned by Evernote on a successful authentication

### 7.82.2 Member Function Documentation

#### 7.82.2.1 [print\(\)](#)

```
virtual void qevercloud::EvernoteOAuthWebView::OAuthResult::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.82.3 Member Data Documentation

#### 7.82.3.1 authenticationToken

```
QString qevercloud::EvernoteOAuthWebView::OAuthResult::authenticationToken
```

This is what this all was for!

Cookies set by Evernote during OAuth procedure. In April 2020 these cookies silently started to be required for UserStore API calls. Probably it was a bug on Evernote side which hopefully would be fixed at some point but nevertheless cookies set during OAuth procedure are now available as a part of OAuth result and can be used in subsequent calls to Evernote service. These cookies can be set when creating an instance of [IRequestContext](#). Then this context can be used in Q↔EverCloud calls. Cookies from context would propagate to HTTP requests performed by QEverCloud. See this thread on Evernote discussions for more details: <https://discussion.evernote.com/topic/124257-calls-to-userstore-from-evernote-api-stopped-working>

#### 7.82.3.2 cookies

```
QList<QNetworkCookie> qevercloud::EvernoteOAuthWebView::OAuthResult::cookies
```

#### 7.82.3.3 expires

```
Timestamp qevercloud::EvernoteOAuthWebView::OAuthResult::expires
```

authenticationToken time of expiration.

#### 7.82.3.4 noteStoreUrl

```
QString qevercloud::EvernoteOAuthWebView::OAuthResult::noteStoreUrl
```

note store url for the user; no need to question UserStore::getNoteStoreUrl for it.

#### 7.82.3.5 shardId

```
QString qevercloud::EvernoteOAuthWebView::OAuthResult::shardId
```

usually is not used

7.82.3.6 `userId`

`UserID` `qevercloud::EvernoteOAuthWebView::OAuthResult::userId`

same as [PublicUserInfo::userId](#)

7.82.3.7 `webApiUrlPrefix`

`QString` `qevercloud::EvernoteOAuthWebView::OAuthResult::webApiUrlPrefix`

see [PublicUserInfo::webApiUrlPrefix](#)

7.83 `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
- `bool operator== (const Optional< T > &other)` const
- `bool operator!= (const Optional< T > &other)` const
- `bool operator== (const T &other)` const
- `bool operator!= (const T &other)` const
- [Optional](#) ([Optional](#) &&other)
- [Optional](#) & `operator=` ([Optional](#) &&other)
- [Optional](#) (T &&other)
- [Optional](#) & `operator=` (T &&other)

## Friends

- `template<typename X >`  
class `Optional`
- `void swap (Optional &first, Optional &second)`

### 7.83.1 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.

### 7.83.2 Constructor & Destructor Documentation

#### 7.83.2.1 `Optional()` [1/7]

```
template<typename T>
qevercloud::Optional< T >::Optional ( ) [inline]
```

Default constructor. Default `Optional` is not set.

#### 7.83.2.2 `Optional()` [2/7]

```
template<typename T>
qevercloud::Optional< T >::Optional (
    const Optional< T > & o ) [inline]
```

Copy constructor.



#### 7.83.2.3 Optional() [3/7]

```
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.

#### 7.83.2.4 Optional() [4/7]

```
template<typename T>
qevercloud::Optional< T >::Optional (
    const T & value ) [inline]
```

Initialization with a value of the type T. [Note](#): it's implicit.

#### 7.83.2.5 Optional() [5/7]

```
template<typename T>
template<typename X >
qevercloud::Optional< T >::Optional (
    const X & value ) [inline]
```

Template initialization with a value of any compatible type.

#### 7.83.2.6 Optional() [6/7]

```
template<typename T>
qevercloud::Optional< T >::Optional (
    Optional< T > && other ) [inline]
```

#### 7.83.2.7 Optional() [7/7]

```
template<typename T>
qevercloud::Optional< T >::Optional (
    T && other ) [inline]
```

### 7.83.3 Member Function Documentation

### 7.83.3.1 clear()

```
template<typename T>
void qevercloud::Optional< T >::clear ( ) [inline]
```

Clears an [Optional](#).

```
Optional<int> o(1);
o.clear();
cout << o; // exception
```

### 7.83.3.2 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() != ture

// 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

### 7.83.3.3 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.

#### 7.83.3.4 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.

#### 7.83.3.5 operator const T &()

```
template<typename T>
qevercloud::Optional< T >::operator const T & ( ) const [inline]
```

Implicit conversion of Optional<T> to T.

const version.

#### 7.83.3.6 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.

#### 7.83.3.7 operator!=( ) [1/2]

```
template<typename T>
bool qevercloud::Optional< T >::operator!= (
    const Optional< T > & other ) const [inline]
```

#### 7.83.3.8 operator!=( ) [2/2]

```
template<typename T>
bool qevercloud::Optional< T >::operator!= (
    const T & other ) const [inline]
```

7.83.3.9 `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 concersions 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.

7.83.3.10 `operator->()` [2/2]

```
template<typename T>
const T* qevercloud::Optional< T >::operator-> ( ) const [inline]
```

const version.

7.83.3.11 `operator=()` [1/6]

```
template<typename T>
Optional& qevercloud::Optional< T >::operator= (
    const Optional< T > & o ) [inline]
```

Assignment.

7.83.3.12 `operator=()` [2/6]

```
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.

**7.83.3.13 operator=()** [3/6]

```
template<typename T>
Optional& qevercloud::Optional< T >::operator= (
    const T & value ) [inline]
```

Assignment with a value of the type T.

**7.83.3.14 operator=()** [4/6]

```
template<typename T>
template<typename X >
Optional& qevercloud::Optional< T >::operator= (
    const X & value ) [inline]
```

Template assignment with a value of any compatible type.

**7.83.3.15 operator=()** [5/6]

```
template<typename T>
Optional& qevercloud::Optional< T >::operator= (
    Optional< T > && other ) [inline]
```

**7.83.3.16 operator=()** [6/6]

```
template<typename T>
Optional& qevercloud::Optional< T >::operator= (
    T && other ) [inline]
```

**7.83.3.17 operator==( )** [1/2]

```
template<typename T>
bool qevercloud::Optional< T >::operator==(
    const Optional< T > & other ) const [inline]
```

**7.83.3.18 operator==( )** [2/2]

```
template<typename T>
bool qevercloud::Optional< T >::operator==(
    const T & other ) const [inline]
```

**7.83.3.19 ref()** [1/2]

```
template<typename T>
const T& qevercloud::Optional< T >::ref ( ) const [inline]
```

Returns a reference to the holded value.

const version.

**7.83.3.20 ref()** [2/2]

```
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
```

**7.83.3.21 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 <code>Optional</code> is not set.
---------------------	----------------------------------------------------------

**Returns**

[Optional](#) value if set and defaultValue otherwise.

**7.83.4 Friends And Related Function Documentation****7.83.4.1 Optional**

```
template<typename T>
template<typename X >
friend class Optional [friend]
```

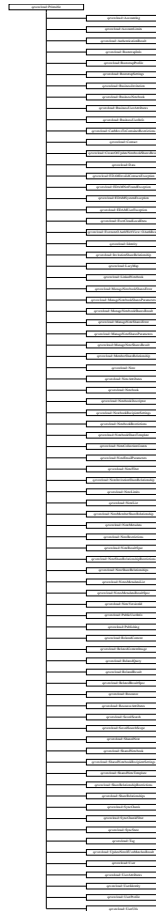
**7.83.4.2 swap**

```
template<typename T>
void swap (
    Optional< T > & first,
    Optional< T > & second ) [friend]
```

**7.84 qevercloud::Printable Class Reference**

```
#include <Printable.h>
```

Inheritance diagram for qevercloud::Printable:



## Public Member Functions

- [Printable](#) ()=default
- virtual [~Printable](#) ()=default
- virtual void [print](#) (QTextStream &strm) const =0
- virtual QString [toString](#) () const

## Friends

- [QEVERTCLOUD\\_EXPORT](#) QTextStream & [operator<<](#) (QTextStream &strm, const [Printable](#) &printable)
- [QEVERTCLOUD\\_EXPORT](#) QDebug & [operator<<](#) (QDebug &dbg, const [Printable](#) &printable)

## 7.84.1 Constructor & Destructor Documentation

### 7.84.1.1 Printable()

```
qevercloud::Printable::Printable ( ) [default]
```



## 7.84.1.2 ~Printable()

```
virtual qevercloud::Printable::~~Printable ( ) [virtual], [default]
```

## 7.84.2 Member Function Documentation

## 7.84.2.1 print()

```
virtual void qevercloud::Printable::print (
    QTextStream & strm ) const [pure virtual]
```

Implemented in [qevercloud::ManageNoteSharesResult](#), [qevercloud::ManageNoteSharesError](#), [qevercloud::CreateOrUpdateNotebookSharesResult](#), [qevercloud::NotebookShareTemplate](#), [qevercloud::SharedNoteTemplate](#), [qevercloud::ManageNotebookSharesResult](#), [qevercloud::ManageNotebookSharesError](#), [qevercloud::ManageNotebookSharesParameters](#), [qevercloud::ShareRelationships](#), [qevercloud::InvitationShareRelationship](#), [qevercloud::UpdateNoteIfUsnMatchesResult](#), [qevercloud::RelatedResult](#), [qevercloud::NoteEmailParameters](#), [qevercloud::NotesMetadataList](#), [qevercloud::NoteMetadata](#), [qevercloud::NoteList](#), [qevercloud::SyncChunk](#), [qevercloud::EDAMInvalidContactsException](#), [qevercloud::EDAMNotFoundException](#), [qevercloud::EDAMSystemException](#), [qevercloud::EDAMUserException](#), [qevercloud::BootstrapInfo](#), [qevercloud::BootstrapProfile](#), [qevercloud::BootstrapSettings](#), [qevercloud::AuthenticationResult](#), [qevercloud::UserUrls](#), [qevercloud::PublicUserInfo](#), [qevercloud::UserIdentity](#), [qevercloud::BusinessInvitation](#), [qevercloud::RelatedContent](#), [qevercloud::RelatedContentImage](#), [qevercloud::UserProfile](#), [qevercloud::NotebookDescriptor](#), [qevercloud::LinkedNotebook](#), [qevercloud::Notebook](#), [qevercloud::NotebookRestrictions](#), [qevercloud::CanMoveToContainerRestrictions](#), [qevercloud::SharedNotebook](#), [qevercloud::NotebookRecipientSettings](#), [qevercloud::SharedNotebookRecipientSettings](#), [qevercloud::SavedSearch](#), [qevercloud::SavedSearchScope](#), [qevercloud::BusinessNotebook](#), [qevercloud::Publishing](#), [qevercloud::Note](#), [qevercloud::NoteLimits](#), [qevercloud::NoteRestrictions](#), [qevercloud::SharedNote](#), [qevercloud::NoteAttributes](#), [qevercloud::Resource](#), [qevercloud::ResourceAttributes](#), [qevercloud::LazyMap](#), [qevercloud::Tag](#), [qevercloud::Identity](#), [qevercloud::Contact](#), [qevercloud::User](#), [qevercloud::AccountLimits](#), [qevercloud::BusinessUserInfo](#), [qevercloud::Accounting](#), [qevercloud::BusinessUserAttributes](#), [qevercloud::UserAttributes](#), [qevercloud::Data](#), [qevercloud::ManageNoteSharesParameters](#), [qevercloud::NoteShareRelationships](#), [qevercloud::NoteInvitationShareRelationship](#), [qevercloud::NoteMemberShareRelationship](#), [qevercloud::NoteShareRelationshipRestrictions](#), [qevercloud::MemberShareRelationship](#), [qevercloud::ShareRelationshipRestrictions](#), [qevercloud::RelatedResultSpec](#), [qevercloud::RelatedQuery](#), [qevercloud::NoteVersionId](#), [qevercloud::NoteResultSpec](#), [qevercloud::NoteCollectionCounts](#), [qevercloud::NotesMetadataResultSpec](#), [qevercloud::NoteFilter](#), [qevercloud::SyncChunkFilter](#), [qevercloud::SyncState](#), [qevercloud::EvernoteOAuthWebView::OAuthResult](#), and [qevercloud::EverCloudLocalData](#).

## 7.84.2.2 toString()

```
virtual QString qevercloud::Printable::toString ( ) const [virtual]
```

## 7.84.3 Friends And Related Function Documentation

### 7.84.3.1 operator<< [1/2]

```
QEVERCLOUD_EXPORT QTextStream& operator<< (
    QTextStream & strm,
    const Printable & printable ) [friend]
```

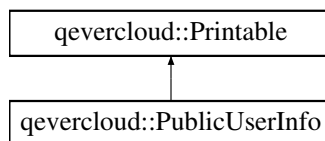
### 7.84.3.2 operator<< [2/2]

```
QEVERCLOUD_EXPORT QDebug& operator<< (
    QDebug & dbg,
    const Printable & printable ) [friend]
```

## 7.85 qevercloud::PublicUserInfo Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::PublicUserInfo:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [PublicUserInfo](#) &other) const
- bool [operator!=](#) (const [PublicUserInfo](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [UserID](#) [userId](#) = 0
- [Optional](#)< [ServiceLevel](#) > [serviceLevel](#)
- [Optional](#)< QString > [username](#)
- [Optional](#)< QString > [noteStoreUrl](#)
- [Optional](#)< QString > [webApiUrlPrefix](#)

### 7.85.1 Detailed Description

This structure is used to provide publicly-available user information about a particular account.

## 7.85.2 Member Function Documentation

### 7.85.2.1 operator!=(())

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

### 7.85.2.2 operator==(())

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

### 7.85.2.3 print()

```
virtual void qevercloud::PublicUserInfo::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.85.3 Member Data Documentation

### 7.85.3.1 localData

[EverCloudLocalData](#) qevercloud::PublicUserInfo::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.85.3.2 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.

### 7.85.3.3 serviceLevel

```
Optional< ServiceLevel > qevercloud::PublicUserInfo::serviceLevel
```

The service level of the account.

### 7.85.3.4 userId

```
UserID qevercloud::PublicUserInfo::userId = 0
```

The unique numeric user identifier for the user account.

### 7.85.3.5 username

```
Optional< QString > qevercloud::PublicUserInfo::username
```

NOT DOCUMENTED

### 7.85.3.6 webApiUrlPrefix

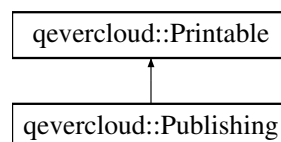
```
Optional< QString > qevercloud::PublicUserInfo::webApiUrlPrefix
```

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.

## 7.86 qevercloud::Publishing Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::Publishing:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [Publishing](#) &other) const
- bool [operator!=](#) (const [Publishing](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- [Optional](#)< [QString](#) > `uri`
- [Optional](#)< [NoteSortOrder](#) > `order`
- [Optional](#)< [bool](#) > `ascending`
- [Optional](#)< [QString](#) > `publicDescription`

### 7.86.1 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](#).

### 7.86.2 Member Function Documentation

#### 7.86.2.1 `operator!=(())`

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

#### 7.86.2.2 `operator==(())`

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

#### 7.86.2.3 `print()`

```
virtual void qevercloud::Publishing::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.86.3 Member Data Documentation

#### 7.86.3.1 `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).

### 7.86.3.2 `localData`

`EverCloudLocalData` `qevercloud::Publishing::localData`

See the declaration of `EverCloudLocalData` for details

### 7.86.3.3 `order`

`Optional< NoteSortOrder >` `qevercloud::Publishing::order`

When the notes are publicly displayed, they will be sorted based on the requested criteria.

### 7.86.3.4 `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`

### 7.86.3.5 `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`

## 7.87 `qevercloud::QAssociativeContainerConstReferenceWrapper< Container >` Class Template Reference

```
#include <Helpers.h>
```

### Classes

- struct `iterator`

### Public Member Functions

- `QAssociativeContainerConstReferenceWrapper` (const Container &container)
- `iterator begin` () const
- `iterator end` () const

## 7.87.1 Constructor & Destructor Documentation

### 7.87.1.1 QAssociativeContainerConstReferenceWrapper()

```
template<typename Container >
qevercloud::QAssociativeContainerConstReferenceWrapper< Container >::QAssociativeContainer←
ConstReferenceWrapper (
    const Container & container ) [inline]
```

## 7.87.2 Member Function Documentation

### 7.87.2.1 begin()

```
template<typename Container >
iterator qevercloud::QAssociativeContainerConstReferenceWrapper< Container >::begin ( ) const
[inline]
```

### 7.87.2.2 end()

```
template<typename Container >
iterator qevercloud::QAssociativeContainerConstReferenceWrapper< Container >::end ( ) const
[inline]
```

## 7.88 qevercloud::QAssociativeContainerReferenceWrapper< Container > Class Template Reference

```
#include <Helpers.h>
```

### Classes

- struct [iterator](#)

### Public Member Functions

- [QAssociativeContainerReferenceWrapper](#) (Container &container)
- [iterator begin](#) ()
- [iterator end](#) ()

## 7.88.1 Constructor & Destructor Documentation

### 7.88.1.1 QAssociativeContainerReferenceWrapper()

```
template<typename Container >
qevercloud::QAssociativeContainerReferenceWrapper< Container >::QAssociativeContainerReferenceWrapper (
    Container & container ) [inline]
```

## 7.88.2 Member Function Documentation

### 7.88.2.1 begin()

```
template<typename Container >
iterator qevercloud::QAssociativeContainerReferenceWrapper< Container >::begin ( ) [inline]
```

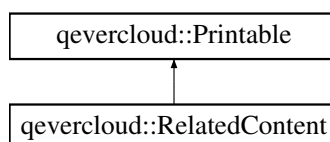
### 7.88.2.2 end()

```
template<typename Container >
iterator qevercloud::QAssociativeContainerReferenceWrapper< Container >::end ( ) [inline]
```

## 7.89 qevercloud::RelatedContent Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::RelatedContent:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `RelatedContent` &other) const
- bool `operator!=` (const `RelatedContent` &other) const



## Public Attributes

- [EverCloudLocalData](#) `localData`
- `Optional< QString >` `contentId`
- `Optional< QString >` `title`
- `Optional< QString >` `url`
- `Optional< QString >` `sourceId`
- `Optional< QString >` `sourceUrl`
- `Optional< QString >` `sourceFaviconUrl`
- `Optional< QString >` `sourceName`
- `Optional< Timestamp >` `date`
- `Optional< QString >` `teaser`
- `Optional< QList< RelatedContentImage > >` `thumbnails`
- `Optional< RelatedContentType >` `contentType`
- `Optional< RelatedContentAccess >` `accessType`
- `Optional< QString >` `visibleUrl`
- `Optional< QString >` `clipUrl`
- `Optional< Contact >` `contact`
- `Optional< QStringList >` `authors`

## Properties

- `Optional QList`

### 7.89.1 Detailed Description

A structure identifying one snippet of related content (some information that is not part of an Evernote account but might still be relevant to the user).

### 7.89.2 Member Function Documentation

#### 7.89.2.1 `operator!=(())`

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

#### 7.89.2.2 `operator==(())`

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

### 7.89.2.3 print()

```
virtual void qevercloud::RelatedContent::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.89.3 Member Data Documentation

### 7.89.3.1 accessType

```
Optional< RelatedContentAccess > qevercloud::RelatedContent::accessType
```

An indication of how this content can be accessed. This type influences the semantics of the `url` parameter.

### 7.89.3.2 authors

```
Optional< QStringList > qevercloud::RelatedContent::authors
```

For News articles only. A list of names of the article authors, if available.

### 7.89.3.3 clipUrl

```
Optional< QString > qevercloud::RelatedContent::clipUrl
```

If set, the client should use this URL for clipping purposes, instead of the URL that was used to retrieve the content. The `clipUrl` may directly point to an `.enex` file, for example.

### 7.89.3.4 contact

```
Optional< Contact > qevercloud::RelatedContent::contact
```

If set, the client may use this [Contact](#) for messaging purposes. This will typically only be set for user profiles.

### 7.89.3.5 contentId

```
Optional< QString > qevercloud::RelatedContent::contentId
```

An identifier that uniquely identifies the content.

### 7.89.3.6 contentType

```
Optional< RelatedContentType > qevercloud::RelatedContent::contentType
```

The type of this related content.

#### 7.89.3.7 date

`Optional< Timestamp > qevercloud::RelatedContent::date`

A timestamp telling the user about the recency of the content.

#### 7.89.3.8 localData

`EverCloudLocalData qevercloud::RelatedContent::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.89.3.9 sourceFaviconUrl

`Optional< QString > qevercloud::RelatedContent::sourceFaviconUrl`

The favicon URL of the source which the content belongs to.

#### 7.89.3.10 sourceId

`Optional< QString > qevercloud::RelatedContent::sourceId`

An identifier that uniquely identifies the source.

#### 7.89.3.11 sourceName

`Optional< QString > qevercloud::RelatedContent::sourceName`

A human-readable name of the source that provided this content.

#### 7.89.3.12 sourceUrl

`Optional< QString > qevercloud::RelatedContent::sourceUrl`

A URL the client can access to know more about the source.

#### 7.89.3.13 teaser

`Optional< QString > qevercloud::RelatedContent::teaser`

A teaser text to show to the user; usually the first few sentences of the content, excluding the title.

#### 7.89.3.14 thumbnails

`Optional< QList< RelatedContentImage > > qevercloud::RelatedContent::thumbnails`

A list of thumbnails the client can show in the snippet.

#### 7.89.3.15 title

```
Optional< QString > qevercloud::RelatedContent::title
```

The main title to show.

#### 7.89.3.16 url

```
Optional< QString > qevercloud::RelatedContent::url
```

The URL the client can use to retrieve the content.

#### 7.89.3.17 visibleUrl

```
Optional< QString > qevercloud::RelatedContent::visibleUrl
```

If set, the client should show this URL to the user, instead of the URL that was used to retrieve the content. This URL should be used when opening the content in an external browser window, or when sharing with another person.

### 7.89.4 Property Documentation

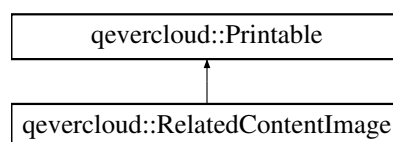
#### 7.89.4.1 QList

```
Optional qevercloud::RelatedContent::QList
```

## 7.90 qevercloud::RelatedContentImage Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::RelatedContentImage:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [RelatedContentImage](#) &other) const
- bool [operator!=](#) (const [RelatedContentImage](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- [Optional](#)< [QString](#) > `url`
- [Optional](#)< [qint32](#) > `width`
- [Optional](#)< [qint32](#) > `height`
- [Optional](#)< [double](#) > `pixelRatio`
- [Optional](#)< [qint32](#) > `fileSize`

### 7.90.1 Detailed Description

An external image that can be shown with a related content snippet, usually either a JPEG or PNG image. It is up to the client which image(s) are shown, depending on available screen real estate, resolution and aspect ratio.

### 7.90.2 Member Function Documentation

#### 7.90.2.1 `operator!=( )`

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

#### 7.90.2.2 `operator==( )`

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

#### 7.90.2.3 `print( )`

```
virtual void qevercloud::RelatedContentImage::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.90.3 Member Data Documentation

### 7.90.3.1 fileSize

`Optional< qint32 > qevercloud::RelatedContentImage::fileSize`

the size of the image file, in bytes

### 7.90.3.2 height

`Optional< qint32 > qevercloud::RelatedContentImage::height`

The height of the image, in pixels.

### 7.90.3.3 localData

`EverCloudLocalData qevercloud::RelatedContentImage::localData`

See the declaration of [EverCloudLocalData](#) for details

### 7.90.3.4 pixelRatio

`Optional< double > qevercloud::RelatedContentImage::pixelRatio`

the pixel ratio (usually either 1.0, 1.5 or 2.0)

### 7.90.3.5 url

`Optional< QString > qevercloud::RelatedContentImage::url`

The external URL of the image

### 7.90.3.6 width

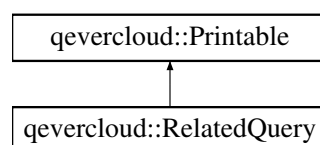
`Optional< qint32 > qevercloud::RelatedContentImage::width`

The width of the image, in pixels.

## 7.91 qevercloud::RelatedQuery Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::RelatedQuery`:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [RelatedQuery](#) &other) const
- bool [operator!=](#) (const [RelatedQuery](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< QString > [noteGuid](#)
- [Optional](#)< QString > [plainText](#)
- [Optional](#)< [NoteFilter](#) > [filter](#)
- [Optional](#)< QString > [referenceUri](#)
- [Optional](#)< QString > [context](#)
- [Optional](#)< QString > [cacheKey](#)

### 7.91.1 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.

### 7.91.2 Member Function Documentation

#### 7.91.2.1 [operator!=\(\)](#)

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

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

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

#### 7.91.2.3 [print\(\)](#)

```
virtual void qevercloud::RelatedQuery::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.91.3 Member Data Documentation

#### 7.91.3.1 cacheKey

`Optional< QString > qevercloud::RelatedQuery::cacheKey`

If set and non-empty, this is an indicator for the server whether it is actually necessary to perform a new findRelated call at all. Cache Keys are opaque strings which are returned by the server as part of "RelatedResult" in response to a "NoteStore.findRelated" query. Cache Keys are inherently query specific.

If set to an empty string, this indicates that the server should generate a cache key in the response as part of "RelatedResult".

If not set, the server will not attempt to generate a cache key at all.

#### 7.91.3.2 context

`Optional< QString > qevercloud::RelatedQuery::context`

Specifies the context to consider when determining related results. Clients must leave this value unset unless they wish to explicitly specify a known non-default context.

#### 7.91.3.3 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*.

#### 7.91.3.4 localData

`EverCloudLocalData qevercloud::RelatedQuery::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.91.3.5 noteGuid

`Optional< QString > qevercloud::RelatedQuery::noteGuid`

The GUID of an existing note in your account for which related entities will be found.

#### 7.91.3.6 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.



## 7.91.3.7 referenceUri

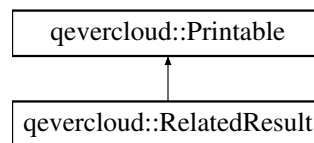
```
Optional< QString > qevercloud::RelatedQuery::referenceUri
```

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.

## 7.92 qevercloud::RelatedResult Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::RelatedResult:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [RelatedResult](#) &other) const
- bool [operator!=](#) (const [RelatedResult](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional< QList< Note > >](#) [notes](#)
- [Optional< QList< Notebook > >](#) [notebooks](#)
- [Optional< QList< Tag > >](#) [tags](#)
- [Optional< QList< NotebookDescriptor > >](#) [containingNotebooks](#)
- [Optional< QString >](#) [debugInfo](#)
- [Optional< QList< UserProfile > >](#) [experts](#)
- [Optional< QList< RelatedContent > >](#) [relatedContent](#)
- [Optional< QString >](#) [cacheKey](#)
- [Optional< qint32 >](#) [cacheExpires](#)

## Properties

- [Optional QList](#)

## 7.92.1 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.

## 7.92.2 Member Function Documentation

### 7.92.2.1 operator!=(())

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

### 7.92.2.2 operator==(())

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

### 7.92.2.3 print()

```
virtual void qevercloud::RelatedResult::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.92.3 Member Data Documentation

### 7.92.3.1 cacheExpires

```
Optional< qint32 > qevercloud::RelatedResult::cacheExpires
```

If set, clients should reuse this response for any situations where the same input parameters are applicable for up to this many seconds after receiving this result.

After this time has passed, the client may request a new result from the service, but it should supply the stored cacheKey to the service when checking for an update.

## 7.92.3.2 cacheKey

`Optional< QString > qevercloud::RelatedResult::cacheKey`

If set and non-empty, this cache key may be used in subsequent "NoteStore.findRelated" calls (via "RelatedQuery") to re-use previous responses that were cached on the client-side, instead of actually performing another search.

If set to an empty string, this indicates that the server could not determine a specific key for this response, but the client should nevertheless remove any previously cached result for this request.

If unset/null, it is up to the client whether to re-use cached results or to use the server's response.

If set to the exact non-empty cache key that was specified in "RelatedQuery.cacheKey", this indicates that the server decided that cached results could be reused.

Depending on the cache key specified in the query, the "RelatedResult" may only be partially filled. For each set field, the client should replace the corresponding part in the previously cached result with the new partial result.

For example, for a specific query that has both "RelatedResultSpec.maxNotes" and "RelatedResultSpec.max↔RelatedContent" set to positive values, the server may decide that the previously requested and cached *Related Content* are unchanged, but new results for *Related Notes* are available. The response will have a new cache key and have "RelatedResult.notes" set, but have "RelatedResult.relatedContent" unset (not just empty, but really unset).

In this situation, the client should replace any cached notes with the newly returned "RelatedResult.notes", but it can re-use the previously cached entries for "RelatedResult.relatedContent". List fields that are set, but empty indicate that no results could be found; the cache should be updated correspondingly.

## 7.92.3.3 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.

## 7.92.3.4 debugInfo

`Optional< QString > qevercloud::RelatedResult::debugInfo`

NOT DOCUMENTED

## 7.92.3.5 experts

`Optional<QList<UserProfile> > qevercloud::RelatedResult::experts`

If experts have been requested to be included, this will return a list of users within your business who have knowledge about the specified query.

### 7.92.3.6 `localData`

`EverCloudLocalData` `qevercloud::RelatedResult::localData`

See the declaration of `EverCloudLocalData` for details

### 7.92.3.7 `notebooks`

`Optional<QList<Notebook>>` `> qevercloud::RelatedResult::notebooks`

If notebooks have been requested to be included, this will be the list of notebooks.

### 7.92.3.8 `notes`

`Optional<QList<Note>>` `> qevercloud::RelatedResult::notes`

If notes have been requested to be included, this will be the list of notes.

### 7.92.3.9 `relatedContent`

`Optional<QList<RelatedContent>>` `> qevercloud::RelatedResult::relatedContent`

If related content has been requested to be included, this will be the list of related content snippets.

### 7.92.3.10 `tags`

`Optional<QList<Tag>>` `> qevercloud::RelatedResult::tags`

If tags have been requested to be included, this will be the list of tags.

## 7.92.4 Property Documentation

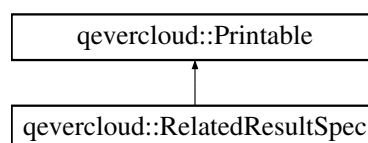
### 7.92.4.1 `QList`

`Optional` `qevercloud::RelatedResult::QList`

## 7.93 `qevercloud::RelatedResultSpec` Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::RelatedResultSpec`:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [RelatedResultSpec](#) &other) const
- bool [operator!=](#) (const [RelatedResultSpec](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< qint32 > [maxNotes](#)
- [Optional](#)< qint32 > [maxNotebooks](#)
- [Optional](#)< qint32 > [maxTags](#)
- [Optional](#)< bool > [writableNotebooksOnly](#)
- [Optional](#)< bool > [includeContainingNotebooks](#)
- [Optional](#)< bool > [includeDebugInfo](#)
- [Optional](#)< qint32 > [maxExperts](#)
- [Optional](#)< qint32 > [maxRelatedContent](#)
- [Optional](#)< [QSet](#)< [RelatedContentType](#) > > [relatedContentTypes](#)

## Properties

- [Optional](#) [QSet](#)

### 7.93.1 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](#).

### 7.93.2 Member Function Documentation

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

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

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

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

### 7.93.2.3 print()

```
virtual void qevercloud::RelatedResultSpec::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.93.3 Member Data Documentation

### 7.93.3.1 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.

### 7.93.3.2 includeDebugInfo

```
Optional< bool > qevercloud::RelatedResultSpec::includeDebugInfo
```

If set to `true`, indicate that debug information should be returned in the 'debugInfo' field of [RelatedResult](#). **Note** that the call may be slower if this flag is set.

### 7.93.3.3 localData

```
EverCloudLocalData qevercloud::RelatedResultSpec::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.93.3.4 maxExperts

```
Optional< qint32 > qevercloud::RelatedResultSpec::maxExperts
```

This can only be used when making a `findRelated` call against a business. Find users within your business who have knowledge about the specified query. No more than this many users will be returned. Any value greater than `EDAM_RELATED_MAX_EXPERTS` will be silently capped.

### 7.93.3.5 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.

#### 7.93.3.6 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.

#### 7.93.3.7 maxRelatedContent

`Optional< qint32 > qevercloud::RelatedResultSpec::maxRelatedContent`

Return snippets of related content that is related to the query, but no more than this many. Any value greater than EDAM\_RELATED\_MAX\_RELATED\_CONTENT will be silently capped. If you do not set this field, then no related content will be returned.

#### 7.93.3.8 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.

#### 7.93.3.9 relatedContentTypes

`Optional< QSet< RelatedContentType > > qevercloud::RelatedResultSpec::relatedContentTypes`

Specifies the types of Related Content that should be returned.

#### 7.93.3.10 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.

### 7.93.4 Property Documentation

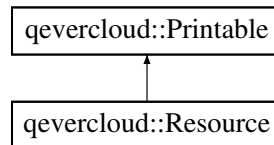
#### 7.93.4.1 QSet

`Optional qevercloud::RelatedResultSpec::QSet`

## 7.94 qevercloud::Resource Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::Resource:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [Resource](#) &other) const
- bool [operator!=](#) (const [Resource](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [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](#)

### 7.94.1 Detailed Description

Every media file that is embedded or attached to a note is represented through a [Resource](#) entry.

### 7.94.2 Member Function Documentation

#### 7.94.2.1 [operator!=\(\)](#)

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



### 7.94.2.2 operator==( )

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

### 7.94.2.3 print()

```
virtual void qevercloud::Resource::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.94.3 Member Data Documentation

### 7.94.3.1 active

```
Optional< bool > qevercloud::Resource::active
```

If the resource is active or not.

### 7.94.3.2 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.

### 7.94.3.3 attributes

```
Optional< ResourceAttributes > qevercloud::Resource::attributes
```

A list of the attributes for this resource.

### 7.94.3.4 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.

#### 7.94.3.5 duration

`Optional< qint16 > qevercloud::Resource::duration`

DEPRECATED: ignored.

#### 7.94.3.6 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

#### 7.94.3.7 height

`Optional< qint16 > qevercloud::Resource::height`

If set, this contains the display height of this resource, in pixels.

#### 7.94.3.8 localData

`EverCloudLocalData qevercloud::Resource::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.94.3.9 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

#### 7.94.3.10 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

#### 7.94.3.11 recognition

`Optional< Data > qevercloud::Resource::recognition`

If set, this will hold the encoded data that provides information on search and recognition within this resource.

## 7.94.3.12 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.

## 7.94.3.13 width

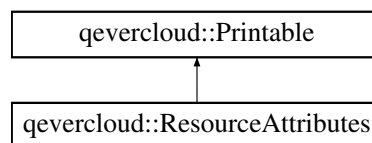
`Optional< qint16 > qevercloud::Resource::width`

If set, this contains the display width of this resource, in pixels.

## 7.95 qevercloud::ResourceAttributes Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::ResourceAttributes:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const ResourceAttributes &other) const
- bool `operator!=` (const ResourceAttributes &other) const

## Public Attributes

- EverCloudLocalData `localData`
- 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`

### 7.95.1 Detailed Description

Structure holding the optional attributes of a [Resource](#)

### 7.95.2 Member Function Documentation

#### 7.95.2.1 `operator!=( )`

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

#### 7.95.2.2 `operator==( )`

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

#### 7.95.2.3 `print()`

```
virtual void qevercloud::ResourceAttributes::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.95.3 Member Data Documentation

#### 7.95.3.1 `altitude`

[Optional](#)< double > qevercloud::ResourceAttributes::altitude

the altitude where the resource was captured

### 7.95.3.2 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

### 7.95.3.3 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).

### 7.95.3.4 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

### 7.95.3.5 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

### 7.95.3.6 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.

### 7.95.3.7 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.

#### 7.95.3.8 latitude

`Optional< double > qevercloud::ResourceAttributes::latitude`

the latitude where the resource was captured

#### 7.95.3.9 localData

`EverCloudLocalData qevercloud::ResourceAttributes::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.95.3.10 longitude

`Optional< double > qevercloud::ResourceAttributes::longitude`

the longitude where the resource was captured

#### 7.95.3.11 recoType

`Optional< QString > qevercloud::ResourceAttributes::recoType`

DEPRECATED - this field is no longer set by the service, so should be ignored.

#### 7.95.3.12 sourceURL

`Optional< QString > qevercloud::ResourceAttributes::sourceURL`

the original location where the resource was hosted

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

#### 7.95.3.13 timestamp

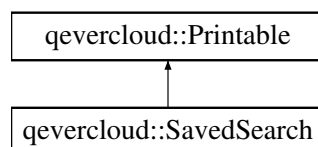
`Optional< Timestamp > qevercloud::ResourceAttributes::timestamp`

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)

## 7.96 qevercloud::SavedSearch Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::SavedSearch:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [SavedSearch](#) &other) const
- bool [operator!=](#) (const [SavedSearch](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [Guid](#) > [guid](#)
- [Optional](#)< [QString](#) > [name](#)
- [Optional](#)< [QString](#) > [query](#)
- [Optional](#)< [QueryFormat](#) > [format](#)
- [Optional](#)< [qint32](#) > [updateSequenceNum](#)
- [Optional](#)< [SavedSearchScope](#) > [scope](#)

### 7.96.1 Detailed Description

A named search associated with the account that can be quickly re-used.

### 7.96.2 Member Function Documentation

#### 7.96.2.1 [operator!=\(\)](#)

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

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

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

#### 7.96.2.3 [print\(\)](#)

```
virtual void qevercloud::SavedSearch::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.96.3 Member Data Documentation

#### 7.96.3.1 format

`Optional< QueryFormat > qevercloud::SavedSearch::format`

The format of the query string, to determine how to parse and process it.

#### 7.96.3.2 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

#### 7.96.3.3 localData

`EverCloudLocalData qevercloud::SavedSearch::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.96.3.4 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

#### 7.96.3.5 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

#### 7.96.3.6 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.



## 7.96.3.7 updateSequenceNum

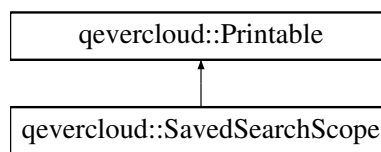
`Optional< qint32 > qevercloud::SavedSearch::updateSequenceNum`

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.

## 7.97 qevercloud::SavedSearchScope Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::SavedSearchScope:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `SavedSearchScope` &other) const
- bool `operator!=` (const `SavedSearchScope` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< bool >` `includeAccount`
- `Optional< bool >` `includePersonalLinkedNotebooks`
- `Optional< bool >` `includeBusinessLinkedNotebooks`

## 7.97.1 Detailed Description

A structure defining the scope of a `SavedSearch`.

## 7.97.2 Member Function Documentation

7.97.2.1 `operator!=()`

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

### 7.97.2.2 operator==( )

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

### 7.97.2.3 print()

```
virtual void qevercloud::SavedSearchScope::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.97.3 Member Data Documentation

### 7.97.3.1 includeAccount

```
Optional< bool > qevercloud::SavedSearchScope::includeAccount
```

The search should include notes from the account that contains the [SavedSearch](#).

### 7.97.3.2 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.

### 7.97.3.3 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.

### 7.97.3.4 localData

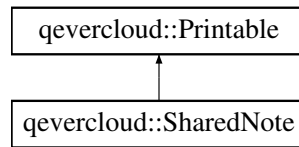
```
EverCloudLocalData qevercloud::SavedSearchScope::localData
```

See the declaration of [EverCloudLocalData](#) for details

## 7.98 qevercloud::SharedNote Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::SharedNote:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [SharedNote](#) &other) const
- bool [operator!=](#) (const [SharedNote](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [UserID](#) > [sharerUserID](#)
- [Optional](#)< [Identity](#) > [recipientIdentity](#)
- [Optional](#)< [SharedNotePrivilegeLevel](#) > [privilege](#)
- [Optional](#)< [Timestamp](#) > [serviceCreated](#)
- [Optional](#)< [Timestamp](#) > [serviceUpdated](#)
- [Optional](#)< [Timestamp](#) > [serviceAssigned](#)

### 7.98.1 Detailed Description

Represents a relationship between a note and a single share invitation recipient. The recipient is identified via an [Identity](#), and has a given privilege that specifies what actions they may take on the note.

### 7.98.2 Member Function Documentation

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

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

### 7.98.2.2 operator==( )

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

### 7.98.2.3 print()

```
virtual void qevercloud::SharedNote::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.98.3 Member Data Documentation

### 7.98.3.1 localData

```
EverCloudLocalData qevercloud::SharedNote::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.98.3.2 privilege

```
Optional< SharedNotePrivilegeLevel > qevercloud::SharedNote::privilege
```

The privilege level that the share grants to the recipient.

### 7.98.3.3 recipientIdentity

```
Optional< Identity > qevercloud::SharedNote::recipientIdentity
```

The identity of the recipient of the share. For a given note, there may be only one [SharedNote](#) per recipient identity. Only recipientIdentity.id is guaranteed to be set. Other fields on the [Identity](#) may or may not be set based on the requesting user's relationship with the recipient.

### 7.98.3.4 serviceAssigned

```
Optional< Timestamp > qevercloud::SharedNote::serviceAssigned
```

The time at which the share was assigned to a specific recipient user ID.

## 7.98.3.5 serviceCreated

```
Optional< Timestamp > qevercloud::SharedNote::serviceCreated
```

The time at which the share was created.

## 7.98.3.6 serviceUpdated

```
Optional< Timestamp > qevercloud::SharedNote::serviceUpdated
```

The time at which the share was last updated.

## 7.98.3.7 sharerUserID

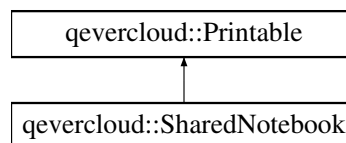
```
Optional< UserID > qevercloud::SharedNote::sharerUserID
```

The user ID of the user who shared the note with the recipient.

## 7.99 qevercloud::SharedNotebook Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::SharedNotebook:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `SharedNotebook` &other) const
- bool `operator!=` (const `SharedNotebook` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< qint64 >` `id`
- `Optional< UserID >` `userId`
- `Optional< Guid >` `notebookGuid`
- `Optional< QString >` `email`
- `Optional< IdentityID >` `recipientIdentityId`
- `Optional< bool >` `notebookModifiable`
- `Optional< Timestamp >` `serviceCreated`
- `Optional< Timestamp >` `serviceUpdated`
- `Optional< QString >` `globalId`
- `Optional< QString >` `username`
- `Optional< SharedNotebookPrivilegeLevel >` `privilege`
- `Optional< SharedNotebookRecipientSettings >` `recipientSettings`
- `Optional< UserID >` `sharerUserId`
- `Optional< QString >` `recipientUsername`
- `Optional< UserID >` `recipientUserId`
- `Optional< Timestamp >` `serviceAssigned`

### 7.99.1 Detailed Description

Shared notebooks represent a relationship between a notebook and a single share invitation recipient.

### 7.99.2 Member Function Documentation

#### 7.99.2.1 operator!=(())

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

#### 7.99.2.2 operator==(())

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

#### 7.99.2.3 print()

```
virtual void qevercloud::SharedNotebook::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.99.3 Member Data Documentation

#### 7.99.3.1 email

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

A string containing a display name for the recipient of the share. This may be an email address, a phone number, a full name, or some other descriptive string This field is read-only to clients. It will be filled in by the service when returning shared notebooks.

#### 7.99.3.2 globalId

```
Optional< QString > qevercloud::SharedNotebook::globalId
```

An immutable, opaque string that acts as a globally unique identifier for this shared notebook record. You can use this field to match linked notebook and shared notebook records as well as to create new [LinkedNotebook](#) records. This field replaces the deprecated shareKey field.

### 7.99.3.3 id

`Optional< qint64 > qevercloud::SharedNotebook::id`

The primary identifier of the share, which is not globally unique.

### 7.99.3.4 localData

`EverCloudLocalData qevercloud::SharedNotebook::localData`

See the declaration of [EverCloudLocalData](#) for details

### 7.99.3.5 notebookGuid

`Optional< Guid > qevercloud::SharedNotebook::notebookGuid`

The GUID of the notebook that has been shared.

### 7.99.3.6 notebookModifiable

`Optional< bool > qevercloud::SharedNotebook::notebookModifiable`

DEPRECATED

### 7.99.3.7 privilege

`Optional< SharedNotebookPrivilegeLevel > qevercloud::SharedNotebook::privilege`

The privilege level granted to the notebook, activity stream, and invitations. See the corresponding enumeration for details.

### 7.99.3.8 recipientIdentityId

`Optional< IdentityID > qevercloud::SharedNotebook::recipientIdentityId`

The IdentityID of the share recipient. If present, only the user who has claimed that identity may access this share.

### 7.99.3.9 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.

#### 7.99.3.10 recipientUserId

```
Optional< UserID > qevercloud::SharedNotebook::recipientUserId
```

The id of the user who can access this share. This is the id for the user with the username in recipientUsername. This value is read-only and set by the service. Value set by clients will be ignored. This field may be unset for unjoined notebooks and is always set if serviceAssigned is set. Clients should prefer this field over recipientUsername unless they need to use usernames directly.

#### 7.99.3.11 recipientUsername

```
Optional< QString > qevercloud::SharedNotebook::recipientUsername
```

The username of the user who can access this share. This is the username for the user with the id in recipientUserId. This value can be set by clients when calling shareNotebook(...), and that will result in the created [SharedNotebook](#) being assigned to a user. This value is always set if serviceAssigned is set.

#### 7.99.3.12 serviceAssigned

```
Optional< Timestamp > qevercloud::SharedNotebook::serviceAssigned
```

The date this [SharedNotebook](#) was assigned (i.e. has been associated with an Evernote user whose user ID is set in recipientUserId). Unset if the [SharedNotebook](#) is not assigned. This field is a read-only value that is set by the service.

#### 7.99.3.13 serviceCreated

```
Optional< Timestamp > qevercloud::SharedNotebook::serviceCreated
```

The date that the owner first created the share with the specific email address.

#### 7.99.3.14 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 (i.e. when the username is bound to that shared notebook), and also when the [SharedNotebook](#) privilege is updated as part of a shareNotebook(...) call, as well as on any calls to updateSharedNotebook(...).

#### 7.99.3.15 sharerUserId

```
Optional< UserID > qevercloud::SharedNotebook::sharerUserId
```

The user id of the user who shared a notebook via this shared notebook instance. This may not be the same as userId, since a user with full access to a notebook may have created a new share for that notebook. For Business, this represents the user who shared the business notebook. This field is currently unset for a [SharedNotebook](#) created by joining a notebook that has been published to the business.



7.99.3.16 `userId`

```
Optional< UserID > qevercloud::SharedNotebook::userId
```

The user id of the owner of the notebook.

7.99.3.17 `username`

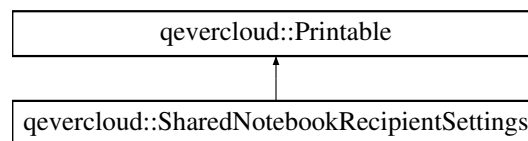
```
Optional< QString > qevercloud::SharedNotebook::username
```

DEPRECATED. The username of the user who can access this share. This value is read-only to clients. It will be filled in by the service when returning shared notebooks.

7.100 `qevercloud::SharedNotebookRecipientSettings` Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::SharedNotebookRecipientSettings`:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `SharedNotebookRecipientSettings` &other) const
- bool `operator!=` (const `SharedNotebookRecipientSettings` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< bool >` `reminderNotifyEmail`
- `Optional< bool >` `reminderNotifyInApp`

## 7.100.1 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.

## 7.100.2 Member Function Documentation

### 7.100.2.1 operator!=(())

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

### 7.100.2.2 operator==(())

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

### 7.100.2.3 print()

```
virtual void qevercloud::SharedNotebookRecipientSettings::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.100.3 Member Data Documentation

### 7.100.3.1 localData

```
EverCloudLocalData qevercloud::SharedNotebookRecipientSettings::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.100.3.2 reminderNotifyEmail

```
Optional< bool > qevercloud::SharedNotebookRecipientSettings::reminderNotifyEmail
```

Indicates that the user wishes to receive daily e-mail notifications for reminders associated with the 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.

## 7.100.3.3 reminderNotifyInApp

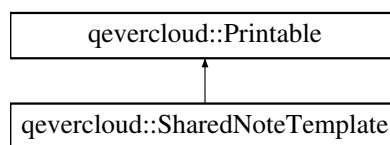
`Optional< bool > qevercloud::SharedNotebookRecipientSettings::reminderNotifyInApp`

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.

## 7.101 qevercloud::SharedNoteTemplate Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::SharedNoteTemplate:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const [SharedNoteTemplate](#) &other) const
- bool `operator!=` (const [SharedNoteTemplate](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- `Optional< Guid >` `noteGuid`
- `Optional< MessageThreadID >` `recipientThreadId`
- `Optional< QList< Contact > >` `recipientContacts`
- `Optional< SharedNotePrivilegeLevel >` `privilege`

## Properties

- `Optional` `QList`

## 7.101.1 Detailed Description

A structure used to share a note with one or more recipients at a given privilege.

## 7.101.2 Member Function Documentation

#### 7.101.2.1 operator!=( )

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

#### 7.101.2.2 operator==( )

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

#### 7.101.2.3 print( )

```
virtual void qevercloud::SharedNoteTemplate::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.101.3 Member Data Documentation

#### 7.101.3.1 localData

[EverCloudLocalData](#) qevercloud::SharedNoteTemplate::localData

See the declaration of [EverCloudLocalData](#) for details

#### 7.101.3.2 noteGuid

[Optional](#)< [Guid](#) > qevercloud::SharedNoteTemplate::noteGuid

The GUID of the note.

#### 7.101.3.3 privilege

[Optional](#)< [SharedNotePrivilegeLevel](#) > qevercloud::SharedNoteTemplate::privilege

The privilege level to be granted.

## 7.101.3.4 recipientContacts

`Optional<QList<Contact> > qevercloud::SharedNoteTemplate::recipientContacts`

The recipients of the note share specified as a list of contacts. This should only be set if the sharing takes place before the thread is created. Use `recipientThreadId` instead when sharing with an existing thread. Either this field or `recipientThreadId` must be set.

## 7.101.3.5 recipientThreadId

`Optional< MessageThreadID > qevercloud::SharedNoteTemplate::recipientThreadId`

The recipients of the note share specified as a messaging thread ID. If you have an existing messaging thread to share the note with, specify its ID here instead of `recipientContacts` in order to properly support defunct identities. The sharer must be a participant of the thread. Either this field or `recipientContacts` must be set.

## 7.101.4 Property Documentation

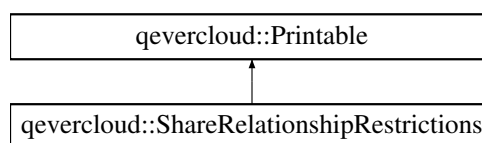
## 7.101.4.1 QList

`Optional qevercloud::SharedNoteTemplate::QList`

## 7.102 qevercloud::ShareRelationshipRestrictions Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::ShareRelationshipRestrictions`:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `ShareRelationshipRestrictions` &other) const
- bool `operator!=` (const `ShareRelationshipRestrictions` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< bool >` `noSetReadOnly`
- `Optional< bool >` `noSetReadPlusActivity`
- `Optional< bool >` `noSetModify`
- `Optional< bool >` `noSetFullAccess`

### 7.102.1 Detailed Description

NO DOC COMMENT ID FOUND

### 7.102.2 Member Function Documentation

#### 7.102.2.1 operator!=(())

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

#### 7.102.2.2 operator==(())

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

#### 7.102.2.3 print()

```
virtual void qevercloud::ShareRelationshipRestrictions::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.102.3 Member Data Documentation

#### 7.102.3.1 localData

[EverCloudLocalData](#) qevercloud::ShareRelationshipRestrictions::localData

See the declaration of [EverCloudLocalData](#) for details

#### 7.102.3.2 noSetFullAccess

[Optional](#)< bool > qevercloud::ShareRelationshipRestrictions::noSetFullAccess

NOT DOCUMENTED

## 7.102.3.3 noSetModify

`Optional< bool > qevercloud::ShareRelationshipRestrictions::noSetModify`

NOT DOCUMENTED

## 7.102.3.4 noSetReadOnly

`Optional< bool > qevercloud::ShareRelationshipRestrictions::noSetReadOnly`

NOT DOCUMENTED

## 7.102.3.5 noSetReadPlusActivity

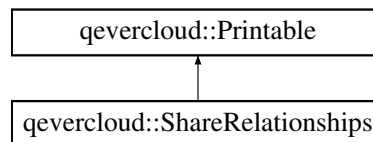
`Optional< bool > qevercloud::ShareRelationshipRestrictions::noSetReadPlusActivity`

NOT DOCUMENTED

## 7.103 qevercloud::ShareRelationships Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::ShareRelationships:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `ShareRelationships` &other) const
- bool `operator!=` (const `ShareRelationships` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< QList< InvitationShareRelationship > >` `invitations`
- `Optional< QList< MemberShareRelationship > >` `memberships`
- `Optional< ShareRelationshipRestrictions >` `invitationRestrictions`

## Properties

- `Optional` `QList`

### 7.103.1 Detailed Description

Captures a collection of share relationships for a notebook, for example, as returned by the `getNotebookShares` method. The share relationships fall into two broad categories: members, and invitations that can be used to become members.

### 7.103.2 Member Function Documentation

#### 7.103.2.1 `operator!=( )`

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

#### 7.103.2.2 `operator==( )`

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

#### 7.103.2.3 `print( )`

```
virtual void qevercloud::ShareRelationships::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.103.3 Member Data Documentation

#### 7.103.3.1 `invitationRestrictions`

```
Optional< ShareRelationshipRestrictions > qevercloud::ShareRelationships::invitationRestrictions
```

The restrictions on what privileges may be granted to invitees to this notebook. These restrictions may be specific to the calling user or to the notebook itself. They represent the union of all possible invite cases, so it is possible that once the recipient of the invitation has been identified by the service, such as by a business auto-join, the actual assigned privilege may change.



## 7.103.3.2 invitations

```
Optional<QList<InvitationShareRelationship> > qevercloud::ShareRelationships::invitations
```

A list of open invitations that can be redeemed into memberships to the notebook.

## 7.103.3.3 localData

```
EverCloudLocalData qevercloud::ShareRelationships::localData
```

See the declaration of [EverCloudLocalData](#) for details

## 7.103.3.4 memberships

```
Optional<QList<MemberShareRelationship> > qevercloud::ShareRelationships::memberships
```

A list of memberships of the notebook. A member is identified by their Evernote UserID and has rights to access the notebook.

## 7.103.4 Property Documentation

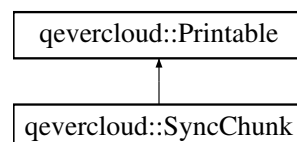
## 7.103.4.1 QList

```
Optional qevercloud::ShareRelationships::QList
```

## 7.104 qevercloud::SyncChunk Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::SyncChunk:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [SyncChunk](#) &other) const
- bool [operator!=](#) (const [SyncChunk](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- [Timestamp](#) `currentTime` = 0
- [Optional](#)< [qint32](#) > `chunkHighUSN`
- [qint32](#) `updateCount` = 0
- [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`

## Properties

- [Optional](#) [QList](#)

### 7.104.1 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).

### 7.104.2 Member Function Documentation

#### 7.104.2.1 `operator!=(())`

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

#### 7.104.2.2 `operator==(())`

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

### 7.104.2.3 print()

```
virtual void qevercloud::SyncChunk::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.104.3 Member Data Documentation

### 7.104.3.1 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.

### 7.104.3.2 currentTime

```
Timestamp qevercloud::SyncChunk::currentTime = 0
```

The server's current date and time.

### 7.104.3.3 expungedLinkedNotebooks

```
Optional<QList<Guid> > qevercloud::SyncChunk::expungedLinkedNotebooks
```

If present, the GUIDs of all of the LinkedNotebooks that were permanently expunged in this chunk.

### 7.104.3.4 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.

### 7.104.3.5 expungedNotes

```
Optional<QList<Guid> > qevercloud::SyncChunk::expungedNotes
```

If present, the GUIDs of all of the notes that were permanently expunged in this chunk.

#### 7.104.3.6 expungedSearches

`Optional<QList<Guid> > qevercloud::SyncChunk::expungedSearches`

If present, the GUIDs of all of the saved searches that were permanently expunged in this chunk.

#### 7.104.3.7 expungedTags

`Optional<QList<Guid> > qevercloud::SyncChunk::expungedTags`

If present, the GUIDs of all of the tags that were permanently expunged in this chunk.

#### 7.104.3.8 linkedNotebooks

`Optional<QList<LinkedNotebook> > qevercloud::SyncChunk::linkedNotebooks`

If present, this is a list of non-expunged LinkedNotebooks that have a USN in this chunk.

#### 7.104.3.9 localData

`EverCloudLocalData qevercloud::SyncChunk::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.104.3.10 notebooks

`Optional<QList<Notebook> > qevercloud::SyncChunk::notebooks`

If present, this is a list of non-expunged notebooks that have a USN in this chunk.

#### 7.104.3.11 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.

#### 7.104.3.12 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.

## 7.104.3.13 searches

`Optional<QList<SavedSearch> > qevercloud::SyncChunk::searches`

If present, this is a list of non-expunged searches that have a USN in this chunk.

## 7.104.3.14 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.

## 7.104.3.15 updateCount

`qint32 qevercloud::SyncChunk::updateCount = 0`

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.

## 7.104.4 Property Documentation

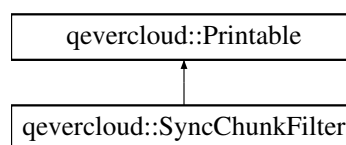
## 7.104.4.1 QList

`Optional qevercloud::SyncChunk::QList`

## 7.105 qevercloud::SyncChunkFilter Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::SyncChunkFilter`:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const [SyncChunkFilter](#) &other) const
- bool `operator!=` (const [SyncChunkFilter](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- `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< bool >` `includeSharedNotes`
- `Optional< bool >` `omitSharedNotebooks`
- `Optional< QString >` `requireNoteContentClass`
- `Optional< QSet< QString > >` `notebookGuids`

## Properties

- `Optional QSet`

### 7.105.1 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.

### 7.105.2 Member Function Documentation

#### 7.105.2.1 `operator!=(())`

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

#### 7.105.2.2 `operator==(())`

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

### 7.105.2.3 print()

```
virtual void qevercloud::SyncChunkFilter::print (
    QTextStream & strm ) const    [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.105.3 Member Data Documentation

### 7.105.3.1 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.

### 7.105.3.2 includeLinkedNotebooks

```
Optional< bool > qevercloud::SyncChunkFilter::includeLinkedNotebooks
```

If true, then the server will include the SyncChunks.linkedNotebooks field.

### 7.105.3.3 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.

### 7.105.3.4 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 'includeNotes' is false, then this will have no effect.

### 7.105.3.5 includeNotebooks

```
Optional< bool > qevercloud::SyncChunkFilter::includeNotebooks
```

If true, then the server will include the SyncChunks.notebooks field

#### 7.105.3.6 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.

#### 7.105.3.7 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.

#### 7.105.3.8 includeNotes

```
Optional< bool > qevercloud::SyncChunkFilter::includeNotes
```

If true, then the server will include the SyncChunks.notes field

#### 7.105.3.9 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.

#### 7.105.3.10 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.

#### 7.105.3.11 includeSearches

```
Optional< bool > qevercloud::SyncChunkFilter::includeSearches
```

If true, then the server will include the SyncChunks.searches field

#### 7.105.3.12 includeSharedNotes

```
Optional< bool > qevercloud::SyncChunkFilter::includeSharedNotes
```

If true, then the service will include the sharedNotes field on all notes that are in [SyncChunk.notes](#). If 'includeNotes' is false, then this will have no effect.



#### 7.105.3.13 includeTags

`Optional< bool > qevercloud::SyncChunkFilter::includeTags`

If true, then the server will include the SyncChunks.tags field

#### 7.105.3.14 localData

`EverCloudLocalData qevercloud::SyncChunkFilter::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.105.3.15 notebookGuids

`Optional<QSet<QString> > qevercloud::SyncChunkFilter::notebookGuids`

If set, then restrict the returned notebooks, notes, and resources to those associated with one of the notebooks whose GUID is provided in this list. If not set, then no filtering on notebook GUID will be performed. If you set this field, you may not also set includeExpunged else an [EDAMUserException](#) with an error code of DATA\_CONFLICT will be thrown. You only need to set this field if you want to restrict the returned entities more than what your authentication token allows you to access. For example, there is no need to set this field for single notebook tokens such as for shared notebooks. You can use this field to synchronize a newly discovered business notebook while incrementally synchronizing a business account, in which case you will only need to consider setting includeNotes, includeNotebooks, includeNoteAttributes, includeNoteResources, and maybe some of the "FullMap" fields.

#### 7.105.3.16 omitSharedNotebooks

`Optional< bool > qevercloud::SyncChunkFilter::omitSharedNotebooks`

NOT DOCUMENTED

#### 7.105.3.17 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.

### 7.105.4 Property Documentation

#### 7.105.4.1 QSet

`Optional qevercloud::SyncChunkFilter::QSet`

## 7.106 qevercloud::IDurableService::SyncRequest Struct Reference

```
#include <DurableService.h>
```

### Public Member Functions

- [SyncRequest](#) (const char \*name, QString description, [SyncServiceCall](#) &&call)

### Public Attributes

- const char \* [m\\_name](#)
- QString [m\\_description](#)
- [SyncServiceCall](#) [m\\_call](#)

### 7.106.1 Constructor & Destructor Documentation

#### 7.106.1.1 SyncRequest()

```
qevercloud::IDurableService::SyncRequest::SyncRequest (
    const char * name,
    QString description,
    SyncServiceCall && call ) [inline]
```

### 7.106.2 Member Data Documentation

#### 7.106.2.1 m\_call

[SyncServiceCall](#) qevercloud::IDurableService::SyncRequest::m\_call

#### 7.106.2.2 m\_description

QString qevercloud::IDurableService::SyncRequest::m\_description

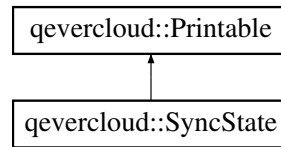
#### 7.106.2.3 m\_name

const char\* qevercloud::IDurableService::SyncRequest::m\_name

## 7.107 qevercloud::SyncState Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::SyncState:



### Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [SyncState](#) &other) const
- bool [operator!=](#) (const [SyncState](#) &other) const

### Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Timestamp](#) [currentTime](#) = 0
- [Timestamp](#) [fullSyncBefore](#) = 0
- qint32 [updateCount](#) = 0
- [Optional](#)< qint64 > [uploaded](#)
- [Optional](#)< [Timestamp](#) > [userLastUpdated](#)
- [Optional](#)< [MessageEventID](#) > [userMaxMessageEventId](#)

### 7.107.1 Detailed Description

This structure encapsulates the information about the state of the user's account for the purpose of "state based" synchronization.

### 7.107.2 Member Function Documentation

#### 7.107.2.1 [operator!=\(\)](#)

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

### 7.107.2.2 operator==()

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

### 7.107.2.3 print()

```
virtual void qevercloud::SyncState::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.107.3 Member Data Documentation

### 7.107.3.1 currentTime

```
Timestamp qevercloud::SyncState::currentTime = 0
```

The server's current date and time.

### 7.107.3.2 fullSyncBefore

```
Timestamp qevercloud::SyncState::fullSyncBefore = 0
```

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.

### 7.107.3.3 localData

```
EverCloudLocalData qevercloud::SyncState::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.107.3.4 updateCount

```
qint32 qevercloud::SyncState::updateCount = 0
```

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.

## 7.107.3.5 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.

## 7.107.3.6 userLastUpdated

```
Optional< Timestamp > qevercloud::SyncState::userLastUpdated
```

The last time when a user's account level information was changed. This value is the latest time when a modification was made to any of the following: accounting information (billing, quota, premium status, etc.), user attributes and business user information (business name, business user attributes, etc.) if the user is in a business. Clients who need to maintain account information about a `User` should watch this field for updates rather than polling `UserStore.getUser` for updates. Here is the basic flow that clients should follow:

1. Call `NoteStore.getSyncState` to retrieve the `SyncState` object
2. Compare `SyncState.userLastUpdated` to previously stored value: if (`SyncState.userLastUpdated` > `previousValue`) call `UserStore.getUser` to get the latest `User` object; else do nothing;
3. Update `previousValue` = `SyncState.userLastUpdated`

## 7.107.3.7 userMaxMessageEventId

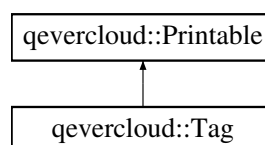
```
Optional< MessageEventID > qevercloud::SyncState::userMaxMessageEventId
```

The greatest `MessageEventID` for this user's account. Clients that do a full sync should store this value locally and compare their local copy to the value returned by `getSyncState` to determine if they need to sync with `MessageStore`. This value will be omitted if the user has never sent or received a message.

## 7.108 qevercloud::Tag Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::Tag`:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [Tag](#) &other) const
- bool [operator!=](#) (const [Tag](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [Guid](#) > [guid](#)
- [Optional](#)< [QString](#) > [name](#)
- [Optional](#)< [Guid](#) > [parentGuid](#)
- [Optional](#)< [qint32](#) > [updateSequenceNum](#)

### 7.108.1 Detailed Description

A tag within a user's account is a unique name which may be organized a simple hierarchy.

### 7.108.2 Member Function Documentation

#### 7.108.2.1 [operator!=\(\)](#)

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

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

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

#### 7.108.2.3 [print\(\)](#)

```
virtual void qevercloud::Tag::print (
    QTextStream & strm ) const    [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.108.3 Member Data Documentation

## 7.108.3.1 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

## 7.108.3.2 localData

```
EverCloudLocalData qevercloud::Tag::localData
```

See the declaration of [EverCloudLocalData](#) for details

## 7.108.3.3 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

## 7.108.3.4 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

## 7.108.3.5 updateSequenceNum

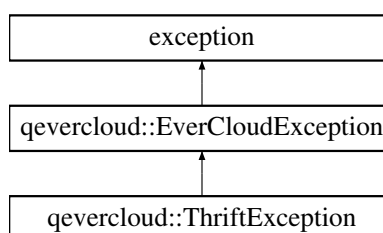
```
Optional< qint32 > qevercloud::Tag::updateSequenceNum
```

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.

## 7.109 qevercloud::ThriftException Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::ThriftException:



## Public Types

- enum [Type](#) {  
[Type::UNKNOWN](#) = 0, [Type::UNKNOWN\\_METHOD](#) = 1, [Type::INVALID\\_MESSAGE\\_TYPE](#) = 2, [Type::WRONG\\_METHOD\\_NAME](#) = 3,  
[Type::BAD\\_SEQUENCE\\_ID](#) = 4, [Type::MISSING\\_RESULT](#) = 5, [Type::INTERNAL\\_ERROR](#) = 6, [Type::PROTOCOL\\_ERROR](#) = 7,  
[Type::INVALID\\_DATA](#) = 8 }

## Public Member Functions

- [ThriftException](#) ()
- [ThriftException](#) ([Type](#) type)
- [ThriftException](#) ([Type](#) type, QString message)
- virtual [~ThriftException](#) () noexcept override
- bool [operator==](#) (const [ThriftException](#) &other) const
- bool [operator!=](#) (const [ThriftException](#) &other) const
- [Type](#) type () const
- const char \* [what](#) () const noexcept override
- virtual [EverCloudExceptionDataPtr](#) [exceptionData](#) () const override

## Protected Attributes

- [Type](#) m\_type

## Friends

- [QEVERCLOUD\\_EXPORT](#) QTextStream & [operator<<](#) (QTextStream &strm, const [Type](#) type)

### 7.109.1 Detailed Description

Errors of the Thrift protocol level. It could be wrongly formatted parameters or return values for example.

### 7.109.2 Member Enumeration Documentation

#### 7.109.2.1 Type

```
enum qevercloud::ThriftException::Type [strong]
```

#### Enumerator

UNKNOWN	
UNKNOWN_METHOD	
INVALID_MESSAGE_TYPE	
WRONG_METHOD_NAME	
BAD_SEQUENCE_ID	
MISSING_RESULT	
INTERNAL_ERROR	
PROTOCOL_ERROR	
INVALID_DATA	



### 7.109.3 Constructor & Destructor Documentation

#### 7.109.3.1 ThriftException() [1/3]

```
qevercloud::ThriftException::ThriftException ( )
```

#### 7.109.3.2 ThriftException() [2/3]

```
qevercloud::ThriftException::ThriftException (
    Type type )
```

#### 7.109.3.3 ThriftException() [3/3]

```
qevercloud::ThriftException::ThriftException (
    Type type,
    QString message )
```

#### 7.109.3.4 ~ThriftException()

```
virtual qevercloud::ThriftException::~~ThriftException ( ) [override], [virtual], [noexcept]
```

### 7.109.4 Member Function Documentation

#### 7.109.4.1 exceptionData()

```
virtual EverCloudExceptionDataPtr qevercloud::ThriftException::exceptionData ( ) const [override],
[virtual]
```

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

#### 7.109.4.2 operator!=( )

```
bool qevercloud::ThriftException::operator!= (
    const ThriftException & other ) const
```

#### 7.109.4.3 operator==( )

```
bool qevercloud::ThriftException::operator== (
    const ThriftException & other ) const
```

#### 7.109.4.4 type( )

```
Type qevercloud::ThriftException::type ( ) const
```

#### 7.109.4.5 what( )

```
const char* qevercloud::ThriftException::what ( ) const [override], [virtual], [noexcept]
```

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

### 7.109.5 Friends And Related Function Documentation

#### 7.109.5.1 operator<<( )

```
QEVERCLOUD\_EXPORT QTextStream& operator<< (
    QTextStream & strm,
    const Type type ) [friend]
```

### 7.109.6 Member Data Documentation

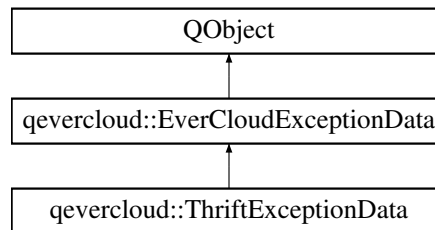
#### 7.109.6.1 m\_type

```
Type qevercloud::ThriftException::m_type [protected]
```

## 7.110 qevercloud::ThriftExceptionData Class Reference

```
#include <Exceptions.h>
```

Inheritance diagram for qevercloud::ThriftExceptionData:



### Public Member Functions

- [ThriftExceptionData](#) (QString error, [ThriftException::Type](#) type)
- virtual void [throwException](#) () const override

### Protected Attributes

- [ThriftException::Type](#) m\_type

### Additional Inherited Members

#### 7.110.1 Detailed Description

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

#### 7.110.2 Constructor & Destructor Documentation

##### 7.110.2.1 ThriftExceptionData()

```
qevercloud::ThriftExceptionData::ThriftExceptionData (
    QString error,
    ThriftException::Type type ) [explicit]
```

#### 7.110.3 Member Function Documentation

### 7.110.3.1 `throwException()`

```
virtual void qevercloud::ThriftExceptionData::throwException ( ) const [override], [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](#).

## 7.110.4 Member Data Documentation

### 7.110.4.1 `m_type`

```
ThriftException::Type qevercloud::ThriftExceptionData::m_type [protected]
```

## 7.111 `qevercloud::Thumbnail` Class Reference

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

```
#include <Thumbnail.h>
```

### Public Types

- enum [ImageType](#) { [ImageType::PNG](#), [ImageType::JPEG](#), [ImageType::GIF](#), [ImageType::BMP](#) }

### Public Member Functions

- [Thumbnail](#) ()  
*Default constructor.*
- [Thumbnail](#) (QString host, QString shardId, QString authenticationToken, int size=300, [ImageType](#) image↔Type=[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](#) imageType)
- QByteArray [download](#) ([Guid](#) guid, const bool isPublic=false, const bool isResourceGuid=false, const qint64 timeoutMsec=30000)  
*Downloads the thumbnail for a resource or a note.*
- [AsyncResult](#) \* [downloadAsync](#) ([Guid](#) guid, const bool isPublic=false, const bool isResourceGuid=false, const qint64 timeoutMsec=30000)
- std::pair< QNetworkRequest, QByteArray > [createPostRequest](#) ([qevercloud::Guid](#) guid, bool isPublic=false, bool isResourceGuid=false)  
*Prepares a POST request for a thumbnail download.*

## Friends

- [QEVERCLOUD\\_EXPORT](#) QTextStream & [operator<<](#) (QTextStream &strm, const [ImageType](#) imageType)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [operator<<](#) (QDebug &dbg, const [ImageType](#) imageType)

### 7.111.1 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.

### 7.111.2 Member Enumeration Documentation

#### 7.111.2.1 ImageType

```
enum qevercloud::Thumbnail::ImageType [strong]
```

Specifies image type of the returned thumbnail.

Can be PNG, JPEG, GIF or BMP.

Enumerator

PNG	
JPEG	
GIF	
BMP	

### 7.111.3 Constructor & Destructor Documentation

#### 7.111.3.1 Thumbnail() [1/2]

```
qevercloud::Thumbnail::Thumbnail ( )
```

Default constructor.

host, shardId, authenticationToken have to be specified before calling [download](#) or [createPostRequest](#)

### 7.111.3.2 Thumbnail() [2/2]

```
gevercloud::Thumbnail::Thumbnail (
    QString host,
    QString shardId,
    QString authenticationToken,
    int size = 300,
    ImageType imageType = ImageType::PNG )
```

Constructs [Thumbnail](#).

#### Parameters

<i>host</i>	www.evernote.com or sandbox.evernote.com
<i>shardId</i>	You can get the value from UserStore 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 ImageType. By default PNG is used.

### 7.111.3.3 ~Thumbnail()

```
virtual gevercloud::Thumbnail::~~Thumbnail ( ) [virtual]
```

## 7.111.4 Member Function Documentation

### 7.111.4.1 createPostRequest()

```
std::pair<QNetworkRequest, QByteArray> gevercloud::Thumbnail::createPostRequest (
    gevercloud::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.

**7.111.4.2 download()**

```
QByteArray qevercloud::Thumbnail::download (
    Guid guid,
    const bool isPublic = false,
    const bool isResourceGuid = false,
    const qint64 timeoutMsec = 30000 )
```

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.
<i>timeoutMsec</i>	Timeout for download request in milliseconds

**Returns**

downloaded data.

**7.111.4.3 downloadAsync()**

```
AsyncResult* qevercloud::Thumbnail::downloadAsync (
    Guid guid,
    const bool isPublic = false,
    const bool isResourceGuid = false,
    const qint64 timeoutMsec = 30000 )
```

Asynchronous version of [download](#) function

**7.111.4.4 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.
----------------------------	-----------------------------------------------------------------------------------------------------------------------------------------

#### 7.111.4.5 setHost()

```
Thumbnail& qevercloud::Thumbnail::setHost (
    QString host )
```

##### Parameters

<i>host</i>	www.evernote.com or sandbox.evernote.com
-------------	------------------------------------------

#### 7.111.4.6 setImageType()

```
Thumbnail& qevercloud::Thumbnail::setImageType (
    ImageType imageType )
```

##### Parameters

<i>imageType</i>	Thumbnail image type. See ImageType. By default PNG is used.
------------------	--------------------------------------------------------------

#### 7.111.4.7 setShardId()

```
Thumbnail& qevercloud::Thumbnail::setShardId (
    QString shardId )
```

##### Parameters

<i>shardId</i>	You can get the value from UserStore service or as a result of an authentication.
----------------	-----------------------------------------------------------------------------------

#### 7.111.4.8 setSize()

```
Thumbnail& qevercloud::Thumbnail::setSize (
    int size )
```

##### Parameters

<i>size</i>	The size of the thumbnail. Evernote supports values from from 1 to 300. By default 300 is used.
-------------	-------------------------------------------------------------------------------------------------



### 7.111.5 Friends And Related Function Documentation

#### 7.111.5.1 operator<< [1/2]

```
QEVERCLOUD_EXPORT QTextStream& operator<< (
    QTextStream & strm,
    const ImageType imageType ) [friend]
```

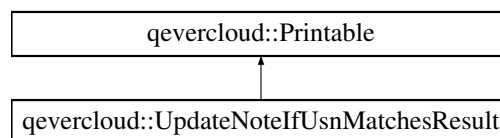
#### 7.111.5.2 operator<< [2/2]

```
QEVERCLOUD_EXPORT QDebug& operator<< (
    QDebug & dbg,
    const ImageType imageType ) [friend]
```

## 7.112 qevercloud::UpdateNotelfUsnMatchesResult Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::UpdateNotelfUsnMatchesResult:



### Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `UpdateNotelfUsnMatchesResult` &other) const
- bool `operator!=` (const `UpdateNotelfUsnMatchesResult` &other) const

### Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< Note >` `note`
- `Optional< bool >` `updated`

### 7.112.1 Detailed Description

The result of a call to `updateNotelfUsnMatches`, which optionally updates a note based on the current value of the note's update sequence number on the service.

## 7.112.2 Member Function Documentation

### 7.112.2.1 operator!=(())

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

### 7.112.2.2 operator==(())

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

### 7.112.2.3 print()

```
virtual void qevercloud::UpdateNoteIfUsnMatchesResult::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.112.3 Member Data Documentation

### 7.112.3.1 localData

```
EverCloudLocalData qevercloud::UpdateNoteIfUsnMatchesResult::localData
```

See the declaration of [EverCloudLocalData](#) for details

### 7.112.3.2 note

```
Optional< Note > qevercloud::UpdateNoteIfUsnMatchesResult::note
```

Either the current state of the note if `updated` is false or the result of updating the note as would be done via the `updateNote` method. If the note was not updated, you will receive a [Note](#) that does not include note content, resources data, resources recognition data, or resources alternate data. You can check for updates to these large objects by checking the [Data.bodyHash](#) values and downloading accordingly.

## 7.112.3.3 updated

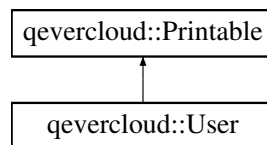
`Optional< bool > qevercloud::UpdateNoteIfUsnMatchesResult::updated`

Whether or not the note was updated by the operation.

## 7.113 qevercloud::User Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::User:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `User` &other) const
- bool `operator!=` (const `User` &other) const

## Public Attributes

- `EverCloudLocalData` `localData`
- `Optional< UserID >` `id`
- `Optional< QString >` `username`
- `Optional< QString >` `email`
- `Optional< QString >` `name`
- `Optional< QString >` `timezone`
- `Optional< PrivilegeLevel >` `privilege`
- `Optional< ServiceLevel >` `serviceLevel`
- `Optional< Timestamp >` `created`
- `Optional< Timestamp >` `updated`
- `Optional< Timestamp >` `deleted`
- `Optional< bool >` `active`
- `Optional< QString >` `shardId`
- `Optional< UserAttributes >` `attributes`
- `Optional< Accounting >` `accounting`
- `Optional< BusinessUserInfo >` `businessUserInfo`
- `Optional< QString >` `photoUrl`
- `Optional< Timestamp >` `photoLastUpdated`
- `Optional< AccountLimits >` `accountLimits`

## 7.113.1 Detailed Description

This represents the information about a single user account.

## 7.113.2 Member Function Documentation

### 7.113.2.1 operator!=(())

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

### 7.113.2.2 operator==(())

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

### 7.113.2.3 print()

```
virtual void qevercloud::User::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.113.3 Member Data Documentation

### 7.113.3.1 accounting

```
Optional< Accounting > qevercloud::User::accounting
```

Bookkeeping information for the user's subscription.

### 7.113.3.2 accountLimits

```
Optional< AccountLimits > qevercloud::User::accountLimits
```

Account limits applicable for this user.

### 7.113.3.3 active

```
Optional< bool > qevercloud::User::active
```

If the user account is available for login and synchronization, this flag will be set to true.

#### 7.113.3.4 attributes

`Optional< UserAttributes > qevercloud::User::attributes`

If present, this will contain a list of the attributes for this user account.

#### 7.113.3.5 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.

#### 7.113.3.6 created

`Optional< Timestamp > qevercloud::User::created`

The date and time when this user account was created in the service.

#### 7.113.3.7 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.

#### 7.113.3.8 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\_LEN\_MIN - EDAM\_EMAIL\_LEN\_MAX

Regex: EDAM\_EMAIL\_REGEX

#### 7.113.3.9 id

`Optional< UserID > qevercloud::User::id`

The unique numeric identifier for the account, which will not change for the lifetime of the account.

#### 7.113.3.10 localData

`EverCloudLocalData qevercloud::User::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.113.3.11 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

#### 7.113.3.12 photoLastUpdated

`Optional< Timestamp > qevercloud::User::photoLastUpdated`

The time at which the photo at 'photoUrl' was last updated by this [User](#). This field will be null if the [User](#) never set a profile photo. This field is filled in by the service and is read-only to clients.

#### 7.113.3.13 photoUrl

`Optional< QString > qevercloud::User::photoUrl`

The URL of the photo that represents this [User](#). This field is filled in by the service and is read-only to clients. If photoLastUpdated is not set, this url will point to a placeholder user photo generated by the service.

#### 7.113.3.14 privilege

`Optional< PrivilegeLevel > qevercloud::User::privilege`

NOT DOCUMENTED

#### 7.113.3.15 serviceLevel

`Optional< ServiceLevel > qevercloud::User::serviceLevel`

The level of service the user currently receives. This will always be populated for users retrieved from the Evernote service.

#### 7.113.3.16 shardId

`Optional< QString > qevercloud::User::shardId`

DEPRECATED - Client applications should have no need to use this field.

7.113.3.17 `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

7.113.3.18 `updated`

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

The date and time when this user account was last modified in the service.

7.113.3.19 `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 '-'

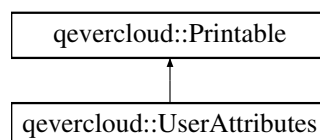
Length: EDAM\_USER\_USERNAME\_LEN\_MIN - EDAM\_USER\_USERNAME\_LEN\_MAX

Regex: EDAM\_USER\_USERNAME\_REGEX

7.114 `qevercloud::UserAttributes` Struct Reference

```
#include <Types.h>
```

Inheritance diagram for `qevercloud::UserAttributes`:



## Public Member Functions

- virtual void `print` (QTextStream &strm) const override
- bool `operator==` (const `UserAttributes` &other) const
- bool `operator!=` (const `UserAttributes` &other) const

## Public Attributes

- [EverCloudLocalData](#) `localData`
- [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](#) > `referrerCode`
- [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](#) > `useEmailAutoFiling`
- [Optional](#)< [ReminderEmailConfig](#) > `reminderEmailConfig`
- [Optional](#)< [Timestamp](#) > `emailAddressLastConfirmed`
- [Optional](#)< [Timestamp](#) > `passwordUpdated`
- [Optional](#)< [bool](#) > `salesforcePushEnabled`
- [Optional](#)< [bool](#) > `shouldLogClientEvent`
- [Optional](#)< [bool](#) > `optOutMachineLearning`

### 7.114.1 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.

### 7.114.2 Member Function Documentation



#### 7.114.2.1 operator!=(())

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

#### 7.114.2.2 operator==(())

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

#### 7.114.2.3 print()

```
virtual void qevercloud::UserAttributes::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.114.3 Member Data Documentation

#### 7.114.3.1 businessAddress

[Optional](#)< [QString](#) > qevercloud::UserAttributes::businessAddress

A string recording the business address of a Sponsored Account user who has requested invoicing.

#### 7.114.3.2 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.

#### 7.114.3.3 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

#### 7.114.3.4 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.

#### 7.114.3.5 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.

#### 7.114.3.6 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.

#### 7.114.3.7 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

#### 7.114.3.8 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.

#### 7.114.3.9 educationalDiscount

`Optional< bool > qevercloud::UserAttributes::educationalDiscount`

NOT DOCUMENTED

#### 7.114.3.10 emailAddressLastConfirmed

`Optional< Timestamp > qevercloud::UserAttributes::emailAddressLastConfirmed`

If set, this contains the time at which the user last confirmed that the configured email address for this account is correct and up-to-date. If this is unset that indicates that the user's email address is unverified.

#### 7.114.3.11 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.

#### 7.114.3.12 groupName

`Optional< QString > qevercloud::UserAttributes::groupName`

A name identifier used to identify a particular set of branding and light customization.

#### 7.114.3.13 hideSponsorBilling

`Optional< bool > qevercloud::UserAttributes::hideSponsorBilling`

A flag indicating whether to hide the billing information on a sponsored account owner's settings page

#### 7.114.3.14 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

#### 7.114.3.15 localData

`EverCloudLocalData qevercloud::UserAttributes::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.114.3.16 maxReferrals

`Optional< qint32 > qevercloud::UserAttributes::maxReferrals`

The number of referrals that the user is permitted to make.

#### 7.114.3.17 optOutMachineLearning

`Optional< bool > qevercloud::UserAttributes::optOutMachineLearning`

If set to True, no Machine Learning nor human review will be done to this user's note contents.

**7.114.3.18 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.

**7.114.3.19 passwordUpdated**

`Optional< Timestamp > qevercloud::UserAttributes::passwordUpdated`

If set, this contains the time at which the user's password last changed. This will be unset for users created before the addition of this field who have not changed their passwords since the addition of this field.

**7.114.3.20 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.

**7.114.3.21 preferredCountry**

`Optional< QString > qevercloud::UserAttributes::preferredCountry`

Preferred country code based on ISO 3166-1-alpha-2 indicating the users preferred country

**7.114.3.22 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).

**7.114.3.23 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

#### 7.114.3.24 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.

#### 7.114.3.25 refererCode

`Optional< QString > qevercloud::UserAttributes::referrerCode`

A code indicating where the user was sent from. AKA promotion code

#### 7.114.3.26 referralCount

`Optional< qint32 > qevercloud::UserAttributes::referralCount`

The number of referrals sent from this account.

#### 7.114.3.27 referralProof

`Optional< QString > qevercloud::UserAttributes::referralProof`

NOT DOCUMENTED

#### 7.114.3.28 reminderEmailConfig

`Optional< ReminderEmailConfig > 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.

#### 7.114.3.29 salesforcePushEnabled

`Optional< bool > qevercloud::UserAttributes::salesforcePushEnabled`

NOT DOCUMENTED

#### 7.114.3.30 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.

**7.114.3.31 setEmailDate**

```
Optional< Timestamp > qevercloud::UserAttributes::setEmailDate
```

The most recent date when the user sent outbound emails from the service. Used with setEmailCount to limit the number of emails that can be sent per day.

**7.114.3.32 shouldLogClientEvent**

```
Optional< bool > qevercloud::UserAttributes::shouldLogClientEvent
```

If set to True, the server will record LogRequest send from clients of this user as ClientEventLog.

**7.114.3.33 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.

**7.114.3.34 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.

**7.114.3.35 useEmailAutoFiling**

```
Optional< bool > qevercloud::UserAttributes::useEmailAutoFiling
```

A flag indicating whether the user chooses to allow Evernote to automatically file and tag emailed notes

**7.114.3.36 viewedPromotions**

```
Optional< QStringList > qevercloud::UserAttributes::viewedPromotions
```

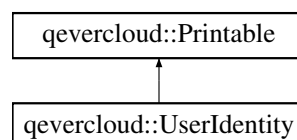
a list of promotions the user has seen. This list may occasionally be modified by the system when promotions are no longer available.

Length: EDAM\_ATTRIBUTE\_LEN\_MIN - EDAM\_ATTRIBUTE\_LEN\_MAX

**7.115 qevercloud::UserIdentity Struct Reference**

```
#include <Types.h>
```

Inheritance diagram for qevercloud::UserIdentity:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [UserIdentity](#) &other) const
- bool [operator!=](#) (const [UserIdentity](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) localData
- [Optional](#)< [UserIdentityType](#) > type
- [Optional](#)< QString > [stringIdentifier](#)
- [Optional](#)< quint64 > [longIdentifier](#)

### 7.115.1 Detailed Description

A structure that holds user identifying information such as an email address, Evernote user ID, or an identifier from a 3rd party service. An instance consists of a type and a value, where the value will be stored in one of the value fields depending upon the data type required for the identity type.

When used with shared notebook invitations, a [UserIdentity](#) identifies a particular person who may not (yet) have an Evernote UserID [UserIdentity](#) but who has (almost) unique access to the service endpoint described by the [UserIdentity](#). For example, an e-mail [UserIdentity](#) can identify the person who receives e-mail at the given address, and who can therefore read the share key that has a cryptographic signature from the Evernote service. With the share key, this person can supply their Evernote UserID via an authentication token to join the notebook ([authenticateToSharedNotebook](#)), at which time we have associated the e-mail [UserIdentity](#) with an Evernote UserID [UserIdentity](#). [Note](#) that using shared notebook records, the relationship between Evernote UserIDs and e-mail addresses is many to many.

[Note](#) that the identifier may not directly identify a particular Evernote UserID [UserIdentity](#) without further verification. For example, an e-mail [UserIdentity](#) may be associated with an invitation to join a notebook (via a shared notebook record), but until a user uses a share key, that was sent to that e-mail address, to join the notebook, we do not know an Evernote UserID [UserIdentity](#) ID to match the e-mail address.

### 7.115.2 Member Function Documentation

#### 7.115.2.1 [operator!=\(\)](#)

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

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

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

### 7.115.2.3 print()

```
virtual void qevercloud::UserIdentity::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.115.3 Member Data Documentation

### 7.115.3.1 localData

[EverCloudLocalData](#) qevercloud::UserIdentity::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.115.3.2 longIdentifier

[Optional](#)< qint64 > qevercloud::UserIdentity::longIdentifier

NOT DOCUMENTED

### 7.115.3.3 stringIdentifier

[Optional](#)< QString > qevercloud::UserIdentity::stringIdentifier

NOT DOCUMENTED

### 7.115.3.4 type

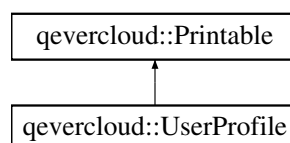
[Optional](#)< [UserIdentityType](#) > qevercloud::UserIdentity::type

NOT DOCUMENTED

## 7.116 qevercloud::UserProfile Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::UserProfile:





## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [UserProfile](#) &other) const
- bool [operator!=](#) (const [UserProfile](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [UserID](#) > [id](#)
- [Optional](#)< [QString](#) > [name](#)
- [Optional](#)< [QString](#) > [email](#)
- [Optional](#)< [QString](#) > [username](#)
- [Optional](#)< [BusinessUserAttributes](#) > [attributes](#)
- [Optional](#)< [Timestamp](#) > [joined](#)
- [Optional](#)< [Timestamp](#) > [photoLastUpdated](#)
- [Optional](#)< [QString](#) > [photoUrl](#)
- [Optional](#)< [BusinessUserRole](#) > [role](#)
- [Optional](#)< [BusinessUserStatus](#) > [status](#)

### 7.116.1 Detailed Description

This structure represents profile information for a user in a business.

### 7.116.2 Member Function Documentation

#### 7.116.2.1 [operator!=\(\)](#)

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

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

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

#### 7.116.2.3 [print\(\)](#)

```
virtual void qevercloud::UserProfile::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

### 7.116.3 Member Data Documentation

#### 7.116.3.1 attributes

`Optional< BusinessUserAttributes > qevercloud::UserProfile::attributes`

The user's business specific attributes.

#### 7.116.3.2 email

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

The user's business email address. If the user has not registered their business email address, this field will be empty.

#### 7.116.3.3 id

`Optional< UserID > qevercloud::UserProfile::id`

The numeric identifier that uniquely identifies a user.

#### 7.116.3.4 joined

`Optional< Timestamp > qevercloud::UserProfile::joined`

The time when the user joined the business

#### 7.116.3.5 localData

`EverCloudLocalData qevercloud::UserProfile::localData`

See the declaration of [EverCloudLocalData](#) for details

#### 7.116.3.6 name

`Optional< QString > qevercloud::UserProfile::name`

The full name of the user.

#### 7.116.3.7 photoLastUpdated

`Optional< Timestamp > qevercloud::UserProfile::photoLastUpdated`

The time when the user's profile photo was most recently updated

## 7.116.3.8 photoUrl

```
Optional< QString > qevercloud::UserProfile::photoUrl
```

A URL identifying a copy of the user's current profile photo

## 7.116.3.9 role

```
Optional< BusinessUserRole > qevercloud::UserProfile::role
```

The BusinessUserRole for the user

## 7.116.3.10 status

```
Optional< BusinessUserStatus > qevercloud::UserProfile::status
```

The BusinessUserStatus for the user

## 7.116.3.11 username

```
Optional< QString > qevercloud::UserProfile::username
```

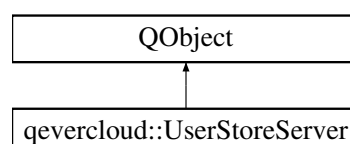
The user's Evernote username.

## 7.117 qevercloud::UserStoreServer Class Reference

The [UserStoreServer](#) class represents customizable server for UserStore requests. It is primarily used for testing of QEverCloud.

```
#include <Servers.h>
```

Inheritance diagram for qevercloud::UserStoreServer:



## Public Slots

- void [onRequest](#) (QByteArray data)
- void [onCheckVersionRequestReady](#) (bool value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetBootstrapInfoRequestReady](#) ([BootstrapInfo](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onAuthenticateLongSessionRequestReady](#) ([AuthenticationResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onCompleteTwoFactorAuthenticationRequestReady](#) ([AuthenticationResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onRevokeLongSessionRequestReady](#) ([EverCloudExceptionDataPtr](#) exceptionData)
- void [onAuthenticateToBusinessRequestReady](#) ([AuthenticationResult](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetUserRequestReady](#) ([User](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetPublicUserInfoRequestReady](#) ([PublicUserInfo](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetUserUrlsRequestReady](#) ([UserUrls](#) value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onInviteToBusinessRequestReady](#) ([EverCloudExceptionDataPtr](#) exceptionData)
- void [onRemoveFromBusinessRequestReady](#) ([EverCloudExceptionDataPtr](#) exceptionData)
- void [onUpdateBusinessUserIdentifierRequestReady](#) ([EverCloudExceptionDataPtr](#) exceptionData)
- void [onListBusinessUsersRequestReady](#) (QList< [UserProfile](#) > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onListBusinessInvitationsRequestReady](#) (QList< [BusinessInvitation](#) > value, [EverCloudExceptionDataPtr](#) exceptionData)
- void [onGetAccountLimitsRequestReady](#) ([AccountLimits](#) value, [EverCloudExceptionDataPtr](#) exceptionData)

## Signals

- void [checkVersionRequest](#) (QString clientName, qint16 edamVersionMajor, qint16 edamVersionMinor, [IRequestContextPtr](#) ctx)
- void [getBootstrapInfoRequest](#) (QString locale, [IRequestContextPtr](#) ctx)
- void [authenticateLongSessionRequest](#) (QString username, QString password, QString consumerKey, QString consumerSecret, QString deviceIdIdentifier, QString deviceDescription, bool supportsTwoFactor, [IRequestContextPtr](#) ctx)
- void [completeTwoFactorAuthenticationRequest](#) (QString oneTimeCode, QString deviceIdIdentifier, QString deviceDescription, [IRequestContextPtr](#) ctx)
- void [revokeLongSessionRequest](#) ([IRequestContextPtr](#) ctx)
- void [authenticateToBusinessRequest](#) ([IRequestContextPtr](#) ctx)
- void [getUserRequest](#) ([IRequestContextPtr](#) ctx)
- void [getPublicUserInfoRequest](#) (QString username, [IRequestContextPtr](#) ctx)
- void [getUserUrlsRequest](#) ([IRequestContextPtr](#) ctx)
- void [inviteToBusinessRequest](#) (QString emailAddress, [IRequestContextPtr](#) ctx)
- void [removeFromBusinessRequest](#) (QString emailAddress, [IRequestContextPtr](#) ctx)
- void [updateBusinessUserIdentifierRequest](#) (QString oldEmailAddress, QString newEmailAddress, [IRequestContextPtr](#) ctx)
- void [listBusinessUsersRequest](#) ([IRequestContextPtr](#) ctx)
- void [listBusinessInvitationsRequest](#) (bool includeRequestedInvitations, [IRequestContextPtr](#) ctx)
- void [getAccountLimitsRequest](#) ([ServiceLevel](#) serviceLevel, [IRequestContextPtr](#) ctx)
- void [checkVersionRequestReady](#) (QByteArray data)
- void [getBootstrapInfoRequestReady](#) (QByteArray data)
- void [authenticateLongSessionRequestReady](#) (QByteArray data)
- void [completeTwoFactorAuthenticationRequestReady](#) (QByteArray data)
- void [revokeLongSessionRequestReady](#) (QByteArray data)
- void [authenticateToBusinessRequestReady](#) (QByteArray data)
- void [getUserRequestReady](#) (QByteArray data)
- void [getPublicUserInfoRequestReady](#) (QByteArray data)

- void [getUserUrlsRequestReady](#) (QByteArray data)
- void [inviteToBusinessRequestReady](#) (QByteArray data)
- void [removeFromBusinessRequestReady](#) (QByteArray data)
- void [updateBusinessUserIdentifierRequestReady](#) (QByteArray data)
- void [listBusinessUsersRequestReady](#) (QByteArray data)
- void [listBusinessInvitationsRequestReady](#) (QByteArray data)
- void [getAccountLimitsRequestReady](#) (QByteArray data)

## Public Member Functions

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

### 7.117.1 Detailed Description

The [UserStoreServer](#) class represents customizable server for UserStore requests. It is primarily used for testing of QEverCloud.

### 7.117.2 Constructor & Destructor Documentation

#### 7.117.2.1 UserStoreServer()

```
qevercloud::UserStoreServer::UserStoreServer (  
    QObject * parent = nullptr ) [explicit]
```

### 7.117.3 Member Function Documentation

#### 7.117.3.1 authenticateLongSessionRequest

```
void qevercloud::UserStoreServer::authenticateLongSessionRequest (  
    QString username,  
    QString password,  
    QString consumerKey,  
    QString consumerSecret,  
    QString deviceIdentifier,  
    QString deviceDescription,  
    bool supportsTwoFactor,  
    IRequestContextPtr ctx ) [signal]
```

#### 7.117.3.2 authenticateLongSessionRequestReady

```
void qevercloud::UserStoreServer::authenticateLongSessionRequestReady (
    QByteArray data ) [signal]
```

#### 7.117.3.3 authenticateToBusinessRequest

```
void qevercloud::UserStoreServer::authenticateToBusinessRequest (
    IRequestContextPtr ctx ) [signal]
```

#### 7.117.3.4 authenticateToBusinessRequestReady

```
void qevercloud::UserStoreServer::authenticateToBusinessRequestReady (
    QByteArray data ) [signal]
```

#### 7.117.3.5 checkVersionRequest

```
void qevercloud::UserStoreServer::checkVersionRequest (
    QString clientName,
    quint16 edamVersionMajor,
    quint16 edamVersionMinor,
    IRequestContextPtr ctx ) [signal]
```

#### 7.117.3.6 checkVersionRequestReady

```
void qevercloud::UserStoreServer::checkVersionRequestReady (
    QByteArray data ) [signal]
```

#### 7.117.3.7 completeTwoFactorAuthenticationRequest

```
void qevercloud::UserStoreServer::completeTwoFactorAuthenticationRequest (
    QString oneTimeCode,
    QString deviceIdentifier,
    QString deviceDescription,
    IRequestContextPtr ctx ) [signal]
```

#### 7.117.3.8 completeTwoFactorAuthenticationRequestReady

```
void qevercloud::UserStoreServer::completeTwoFactorAuthenticationRequestReady (
    QByteArray data ) [signal]
```

#### 7.117.3.9 getAccountLimitsRequest

```
void qevercloud::UserStoreServer::getAccountLimitsRequest (
    ServiceLevel serviceLevel,
    IRequestContextPtr ctx ) [signal]
```

#### 7.117.3.10 getAccountLimitsRequestReady

```
void qevercloud::UserStoreServer::getAccountLimitsRequestReady (
    QByteArray data ) [signal]
```

#### 7.117.3.11 getBootstrapInfoRequest

```
void qevercloud::UserStoreServer::getBootstrapInfoRequest (
    QString locale,
    IRequestContextPtr ctx ) [signal]
```

#### 7.117.3.12 getBootstrapInfoRequestReady

```
void qevercloud::UserStoreServer::getBootstrapInfoRequestReady (
    QByteArray data ) [signal]
```

#### 7.117.3.13 getPublicUserInfoRequest

```
void qevercloud::UserStoreServer::getPublicUserInfoRequest (
    QString username,
    IRequestContextPtr ctx ) [signal]
```

**7.117.3.14 getPublicUserInfoRequestReady**

```
void qevercloud::UserStoreServer::getPublicUserInfoRequestReady (
    QByteArray data ) [signal]
```

**7.117.3.15 getUserRequest**

```
void qevercloud::UserStoreServer::getUserRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.117.3.16 getUserRequestReady**

```
void qevercloud::UserStoreServer::getUserRequestReady (
    QByteArray data ) [signal]
```

**7.117.3.17 getUserUrlsRequest**

```
void qevercloud::UserStoreServer::getUserUrlsRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.117.3.18 getUserUrlsRequestReady**

```
void qevercloud::UserStoreServer::getUserUrlsRequestReady (
    QByteArray data ) [signal]
```

**7.117.3.19 inviteToBusinessRequest**

```
void qevercloud::UserStoreServer::inviteToBusinessRequest (
    QString emailAddress,
    IRequestContextPtr ctx ) [signal]
```

**7.117.3.20 inviteToBusinessRequestReady**

```
void qevercloud::UserStoreServer::inviteToBusinessRequestReady (
    QByteArray data ) [signal]
```



### 7.117.3.21 listBusinessInvitationsRequest

```
void qevercloud::UserStoreServer::listBusinessInvitationsRequest (
    bool includeRequestedInvitations,
    IRequestContextPtr ctx ) [signal]
```

### 7.117.3.22 listBusinessInvitationsRequestReady

```
void qevercloud::UserStoreServer::listBusinessInvitationsRequestReady (
    QByteArray data ) [signal]
```

### 7.117.3.23 listBusinessUsersRequest

```
void qevercloud::UserStoreServer::listBusinessUsersRequest (
    IRequestContextPtr ctx ) [signal]
```

### 7.117.3.24 listBusinessUsersRequestReady

```
void qevercloud::UserStoreServer::listBusinessUsersRequestReady (
    QByteArray data ) [signal]
```

### 7.117.3.25 onAuthenticateLongSessionRequestReady

```
void qevercloud::UserStoreServer::onAuthenticateLongSessionRequestReady (
    AuthenticationResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

### 7.117.3.26 onAuthenticateToBusinessRequestReady

```
void qevercloud::UserStoreServer::onAuthenticateToBusinessRequestReady (
    AuthenticationResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.27 onCheckVersionRequestReady**

```
void qevercloud::UserStoreServer::onCheckVersionRequestReady (
    bool value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.28 onCompleteTwoFactorAuthenticationRequestReady**

```
void qevercloud::UserStoreServer::onCompleteTwoFactorAuthenticationRequestReady (
    AuthenticationResult value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.29 onGetAccountLimitsRequestReady**

```
void qevercloud::UserStoreServer::onGetAccountLimitsRequestReady (
    AccountLimits value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.30 onGetBootstrapInfoRequestReady**

```
void qevercloud::UserStoreServer::onGetBootstrapInfoRequestReady (
    BootstrapInfo value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.31 onGetPublicUserInfoRequestReady**

```
void qevercloud::UserStoreServer::onGetPublicUserInfoRequestReady (
    PublicUserInfo value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.32 onGetUserRequestReady**

```
void qevercloud::UserStoreServer::onGetUserRequestReady (
    User value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.33 onGetUserUrlsRequestReady**

```
void qevercloud::UserStoreServer::onGetUserUrlsRequestReady (
    UserUrls value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.34 onInviteToBusinessRequestReady**

```
void qevercloud::UserStoreServer::onInviteToBusinessRequestReady (
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.35 onListBusinessInvitationsRequestReady**

```
void qevercloud::UserStoreServer::onListBusinessInvitationsRequestReady (
    QList< BusinessInvitation > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.36 onListBusinessUsersRequestReady**

```
void qevercloud::UserStoreServer::onListBusinessUsersRequestReady (
    QList< UserProfile > value,
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.37 onRemoveFromBusinessRequestReady**

```
void qevercloud::UserStoreServer::onRemoveFromBusinessRequestReady (
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.38 onRequest**

```
void qevercloud::UserStoreServer::onRequest (
    QByteArray data ) [slot]
```

**7.117.3.39 onRevokeLongSessionRequestReady**

```
void qevercloud::UserStoreServer::onRevokeLongSessionRequestReady (
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.40 onUpdateBusinessUserIdentifierRequestReady**

```
void qevercloud::UserStoreServer::onUpdateBusinessUserIdentifierRequestReady (
    EverCloudExceptionDataPtr exceptionData ) [slot]
```

**7.117.3.41 removeFromBusinessRequest**

```
void qevercloud::UserStoreServer::removeFromBusinessRequest (
    QString emailAddress,
    IRequestContextPtr ctx ) [signal]
```

**7.117.3.42 removeFromBusinessRequestReady**

```
void qevercloud::UserStoreServer::removeFromBusinessRequestReady (
    QByteArray data ) [signal]
```

**7.117.3.43 revokeLongSessionRequest**

```
void qevercloud::UserStoreServer::revokeLongSessionRequest (
    IRequestContextPtr ctx ) [signal]
```

**7.117.3.44 revokeLongSessionRequestReady**

```
void qevercloud::UserStoreServer::revokeLongSessionRequestReady (
    QByteArray data ) [signal]
```

**7.117.3.45 updateBusinessUserIdentifierRequest**

```
void qevercloud::UserStoreServer::updateBusinessUserIdentifierRequest (
    QString oldEmailAddress,
    QString newEmailAddress,
    IRequestContextPtr ctx ) [signal]
```

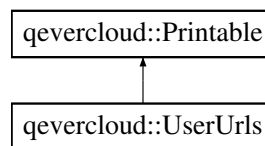
## 7.117.3.46 updateBusinessUserIdentifierRequestReady

```
void qevercloud::UserStoreServer::updateBusinessUserIdentifierRequestReady (
    QByteArray data ) [signal]
```

## 7.118 qevercloud::UserUrls Struct Reference

```
#include <Types.h>
```

Inheritance diagram for qevercloud::UserUrls:



## Public Member Functions

- virtual void [print](#) (QTextStream &strm) const override
- bool [operator==](#) (const [UserUrls](#) &other) const
- bool [operator!=](#) (const [UserUrls](#) &other) const

## Public Attributes

- [EverCloudLocalData](#) [localData](#)
- [Optional](#)< [QString](#) > [noteStoreUrl](#)
- [Optional](#)< [QString](#) > [webApiUrlPrefix](#)
- [Optional](#)< [QString](#) > [userStoreUrl](#)
- [Optional](#)< [QString](#) > [utilityUrl](#)
- [Optional](#)< [QString](#) > [messageStoreUrl](#)
- [Optional](#)< [QString](#) > [userWebSocketUrl](#)

## 7.118.1 Member Function Documentation

## 7.118.1.1 operator!=(())

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

### 7.118.1.2 operator==( )

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

### 7.118.1.3 print()

```
virtual void qevercloud::UserUrls::print (
    QTextStream & strm ) const [override], [virtual]
```

Implements [qevercloud::Printable](#).

## 7.118.2 Member Data Documentation

### 7.118.2.1 localData

[EverCloudLocalData](#) qevercloud::UserUrls::localData

See the declaration of [EverCloudLocalData](#) for details

### 7.118.2.2 messageStoreUrl

[Optional](#)< [QString](#) > qevercloud::UserUrls::messageStoreUrl

This field will contain the full URL that clients should use to make MessageStore requests to the server. I.e. this is the URL that should be used to create the Thrift HTTP client transport to send messages to the MessageStore service for the account.

### 7.118.2.3 noteStoreUrl

[Optional](#)< [QString](#) > qevercloud::UserUrls::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.

### 7.118.2.4 userStoreUrl

[Optional](#)< [QString](#) > qevercloud::UserUrls::userStoreUrl

This field will contain the full URL that clients should use to make UserStore requests after successfully authenticating. I.e. this is the URL that should be used to create the Thrift HTTP client transport to send messages to the UserStore service for this account.

#### 7.118.2.5 userWebSocketUrl

`Optional< QString > qevercloud::UserUrls::userWebSocketUrl`

This field will contain the full URL that clients should use when opening a persistent web socket to receive notification of events for the authenticated user.

#### 7.118.2.6 utilityUrl

`Optional< QString > qevercloud::UserUrls::utilityUrl`

This field will contain the full URL that clients should use to make Utility 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 Utility service for the account.

#### 7.118.2.7 webApiUrlPrefix

`Optional< QString > qevercloud::UserUrls::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.





## Chapter 8

# File Documentation

### 8.1 AsyncResult.h File Reference

```
#include "EverCloudException.h"
#include "Export.h"
#include "Helpers.h"
#include "RequestContext.h"
#include <QNetworkRequest>
#include <QObject>
#include <QUuid>
```

#### Classes

- class [qevercloud::AsyncResult](#)  
*Returned by asynchronous versions of functions.*

#### Namespaces

- [qevercloud](#)

### 8.2 Constants.h File Reference

```
#include "../Export.h"
```

#### Namespaces

- [qevercloud](#)

## Variables

- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_ATTRIBUTE_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_ATTRIBUTE_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_ATTRIBUTE_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_ATTRIBUTE_LIST_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_ATTRIBUTE_MAP_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_GUID_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_GUID_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_GUID_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_EMAIL_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_EMAIL_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_EMAIL_LOCAL_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_EMAIL_DOMAIN_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_EMAIL_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_VAT_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_TIMEZONE_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_TIMEZONE_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_TIMEZONE_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MIME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MIME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_GIF`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_JPEG`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_PNG`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_TIFF`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_BMP`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_WAV`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_MP3`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_AMR`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_AAC`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_M4A`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_MP4_VIDEO`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_INK`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_PDF`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MIME_TYPE_DEFAULT`
- `QEVERCLOUD_EXPORT` `const QSet< QString > qevercloud::EDAM_MIME_TYPES`
- `QEVERCLOUD_EXPORT` `const QSet< QString > qevercloud::EDAM_INDEXABLE_RESOURCE_MIME←  
_TYPES`
- `QEVERCLOUD_EXPORT` `const QSet< QString > qevercloud::EDAM_INDEXABLE_PLAINTEXT_MIME←  
_TYPES`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SEARCH_QUERY_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SEARCH_QUERY_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SEARCH_QUERY_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_HASH_LEN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_USERNAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_USERNAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_USER_USERNAME_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_NAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_NAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_USER_NAME_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_TAG_NAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_TAG_NAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_TAG_NAME_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_TITLE_LEN_MIN`

- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_TITLE_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_NOTE_TITLE_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_CONTENT_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_CONTENT_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_APPLICATIONDATA_NAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_APPLICATIONDATA_NAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_APPLICATIONDATA_VALUE_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_APPLICATIONDATA_VALUE_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_APPLICATIONDATA_ENTRY_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_APPLICATIONDATA_NAME_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_APPLICATIONDATA_VALUE_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTEBOOK_NAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTEBOOK_NAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_NOTEBOOK_NAME_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTEBOOK_STACK_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTEBOOK_STACK_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_NOTEBOOK_STACK_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_WORKSPACE_NAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_WORKSPACE_NAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_WORKSPACE_DESCRIPTION_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_WORKSPACE_NAME_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PUBLISHING_URI_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PUBLISHING_URI_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PUBLISHING_URI_REGEX`
- `QEVERCLOUD_EXPORT` `const QSet< QString > qevercloud::EDAM_PUBLISHING_URI_PROHIBITED`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PUBLISHING_DESCRIPTION_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PUBLISHING_DESCRIPTION_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PUBLISHING_DESCRIPTION_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SAVED_SEARCH_NAME_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SAVED_SEARCH_NAME_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SAVED_SEARCH_NAME_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_PASSWORD_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_PASSWORD_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_USER_PASSWORD_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_URI_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_BUSINESS_MARKETING_CODE_REGEX_PATTERN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_TAGS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_RESOURCES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_TAGS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_TAGS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_SAVED_SEARCHES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_NOTES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_NOTES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_NOTEBOOKS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_WORKSPACES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_NOTEBOOKS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_WORKSPACES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_RECENT_MAILED_ADDRESSES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_MAIL_LIMIT_DAILY_FREE`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_MAIL_LIMIT_DAILY_PREMIUM`
- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_FREE`
- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_PREMIUM`
- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS_FIRST_MONTH`

- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS_NEXT_MONTH`
- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_PLUS`
- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_SURVEY_THRESHOLD`
- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS`
- `QEVERCLOUD_EXPORT` `const qint64 qevercloud::EDAM_USER_UPLOAD_LIMIT_BUSINESS_PER_USER`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_SIZE_MAX_FREE`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_SIZE_MAX_PREMIUM`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RESOURCE_SIZE_MAX_FREE`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RESOURCE_SIZE_MAX_PREMIUM`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_LINKED_NOTEBOOK_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_LINKED_NOTEBOOK_MAX_PREMIUM`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTEBOOK_BUSINESS_SHARED_NOTEBOOK_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTEBOOK_PERSONAL_SHARED_NOTEBOOK_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_BUSINESS_SHARED_NOTE_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_PERSONAL_SHARED_NOTE_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_CONTENT_CLASS_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_CONTENT_CLASS_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_NOTE_CONTENT_CLASS_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_HELLO_APP_CONTENT_CLASS_PREFIX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_FOOD_APP_CONTENT_CLASS_PREFIX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_HELLO_ENCOUNTER`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_HELLO_PROFILE`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_FOOD_MEAL`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_SKITCH_PREFIX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_SKITCH`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_SKITCH_PDF`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_PENULTIMATE_PREFIX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_CONTENT_CLASS_PENULTIMATE_NOTEBOOK`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_APPLICATION_POSTIT`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_APPLICATION_MOLESKINE`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_APPLICATION_EN_SCANSNAP`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_APPLICATION_EWC`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_APPLICATION_ANDROID_SHARE_EXTENSION`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_APPLICATION_IOS_SHARE_EXTENSION`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_APPLICATION_WEB_CLIPPER`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_SOURCE_OUTLOOK_CLIPPER`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_UNTITLED`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_LOW`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_MEDIUM`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_TITLE_QUALITY_HIGH`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RELATED_PLAINTEXT_LEN_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RELATED_PLAINTEXT_LEN_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RELATED_MAX_NOTES`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RELATED_MAX_NOTEBOOKS`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RELATED_MAX_TAGS`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RELATED_MAX_EXPERTS`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_RELATED_MAX_RELATED_CONTENT`

- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LENGTH_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_BUSINESS_NOTEBOOK_DESCRIPTION_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_BUSINESS_PHONE_NUMBER_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PREFERENCE_NAME_LENGTH_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PREFERENCE_NAME_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PREFERENCE_VALUE_LENGTH_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PREFERENCE_VALUE_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MAX_PREFERENCES`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MAX_VALUES_PER_PREFERENCE`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PREFERENCE_ONLY_ONE_VALUE_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PREFERENCE_NAME_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PREFERENCE_VALUE_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PREFERENCE_ONLY_ONE_VALUE_REGEX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PREFERENCE_SHORTCUTS`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PREFERENCE_BUSINESS_DEFAULT_NOTEBOOK`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PREFERENCE_BUSINESS_QUICKNOTE`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PREFERENCE_SHORTCUTS_MAX_VALUES`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_DEVICE_ID_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_DEVICE_ID_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_DEVICE_DESCRIPTION_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_DEVICE_DESCRIPTION_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SEARCH_SUGGESTIONS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SEARCH_SUGGESTIONS_PREFIX_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SEARCH_SUGGESTIONS_PREFIX_LENGTH_MIN`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_FIND_CONTACT_DEFAULT_MAX_RESULTS`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_FIND_CONTACT_MAX_RESULTS`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_NOTE_LOCK_VIEWERS_NOTES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_GET_ORDERS_MAX_RESULTS`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MESSAGE_BODY_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MESSAGE_BODY_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MESSAGE_RECIPIENTS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MESSAGE_ATTACHMENTS_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MESSAGE_ATTACHMENT_TITLE_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MESSAGE_ATTACHMENT_TITLE_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_MESSAGE_ATTACHMENT_SNIPPET_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_MESSAGE_ATTACHMENT_SNIPPET_REGEX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_USER_PROFILE_PHOTO_MAX_BYTES`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_PROMOTION_ID_LENGTH_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::EDAM_PROMOTION_ID_REGEX`
- `QEVERCLOUD_EXPORT` `const qint16 qevercloud::EDAM_APP_RATING_MIN`
- `QEVERCLOUD_EXPORT` `const qint16 qevercloud::EDAM_APP_RATING_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_SNIPPETS_NOTES_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_CONNECTED_IDENTITY_REQUEST_MAX`
- `QEVERCLOUD_EXPORT` `const qint32 qevercloud::EDAM_OPEN_ID_ACCESS_TOKEN_MAX`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::CLASSIFICATION_RECIPE_USER_NON_RECIPE`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::CLASSIFICATION_RECIPE_USER_RECIPE`
- `QEVERCLOUD_EXPORT` `const QString qevercloud::CLASSIFICATION_RECIPE_SERVICE_RECIPE`

- `QEVERCLOUD_EXPORT` const QString `qevercloud::EDAM_NOTE_SOURCE_WEB_CLIP`
- `QEVERCLOUD_EXPORT` const QString `qevercloud::EDAM_NOTE_SOURCE_WEB_CLIP_SIMPLIFIED`
- `QEVERCLOUD_EXPORT` const QString `qevercloud::EDAM_NOTE_SOURCE_MAIL_CLIP`
- `QEVERCLOUD_EXPORT` const QString `qevercloud::EDAM_NOTE_SOURCE_MAIL_SMTP_GATEWAY`
- `QEVERCLOUD_EXPORT` const qint16 `qevercloud::EDAM_VERSION_MAJOR`
- `QEVERCLOUD_EXPORT` const qint16 `qevercloud::EDAM_VERSION_MINOR`

### 8.3 DurableService.h File Reference

```
#include "AsyncResult.h"
#include "Export.h"
#include "RequestContext.h"
#include <QDateTime>
#include <QVariant>
#include <functional>
#include <memory>
#include <utility>
```

#### Classes

- struct `qevercloud::IRetryPolicy`
- class `qevercloud::IDurableService`
- struct `qevercloud::IDurableService::SyncRequest`
- struct `qevercloud::IDurableService::AsyncRequest`

#### Namespaces

- `qevercloud`

#### Typedefs

- using `qevercloud::IRetryPolicyPtr` = `std::shared_ptr< IRetryPolicy >`
- using `qevercloud::IDurableServicePtr` = `std::shared_ptr< IDurableService >`

#### Functions

- `QEVERCLOUD_EXPORT` `IRetryPolicyPtr qevercloud::newRetryPolicy ()`
- `QEVERCLOUD_EXPORT` `IRetryPolicyPtr qevercloud::nullRetryPolicy ()`
- `QEVERCLOUD_EXPORT` `IDurableServicePtr qevercloud::newDurableService (IRetryPolicyPtr={}, I←  
RequestContextPtr={})`

### 8.4 EDAMErrorCode.h File Reference

```
#include "../Export.h"
#include "../Helpers.h"
#include <QDebug>
#include <QMetaType>
#include <QTextStream>
```



## Namespaces

- [qevercloud](#)

## Enumerations

- `enum qevercloud::EDAMErrorCode {`  
`qevercloud::EDAMErrorCode::UNKNOWN = 1, qevercloud::EDAMErrorCode::BAD_DATA_FORMAT = 2,`  
`qevercloud::EDAMErrorCode::PERMISSION_DENIED = 3, qevercloud::EDAMErrorCode::INTERNAL_ER←`  
`ROR = 4,`  
`qevercloud::EDAMErrorCode::DATA_REQUIRED = 5, qevercloud::EDAMErrorCode::LIMIT_REACHED = 6,`  
`qevercloud::EDAMErrorCode::QUOTA_REACHED = 7, qevercloud::EDAMErrorCode::INVALID_AUTH = 8,`  
`qevercloud::EDAMErrorCode::AUTH_EXPIRED = 9, qevercloud::EDAMErrorCode::DATA_CONFLICT = 10,`  
`qevercloud::EDAMErrorCode::ENML_VALIDATION = 11, qevercloud::EDAMErrorCode::SHARD_UNAVAI←`  
`TABLE = 12,`  
`qevercloud::EDAMErrorCode::LEN_TOO_SHORT = 13, qevercloud::EDAMErrorCode::LEN_TOO_LONG =`  
`14, qevercloud::EDAMErrorCode::TOO_FEW = 15, qevercloud::EDAMErrorCode::TOO_MANY = 16,`  
`qevercloud::EDAMErrorCode::UNSUPPORTED_OPERATION = 17, qevercloud::EDAMErrorCode::TAKE←`  
`N_DOWN = 18, qevercloud::EDAMErrorCode::RATE_LIMIT_REACHED = 19, qevercloud::EDAMError←`  
`Code::BUSINESS_SECURITY_LOGIN_REQUIRED = 20,`  
`qevercloud::EDAMErrorCode::DEVICE_LIMIT_REACHED = 21, qevercloud::EDAMErrorCode::OPENID_←`  
`ALREADY_TAKEN = 22, qevercloud::EDAMErrorCode::INVALID_OPENID_TOKEN = 23, qevercloud::ED←`  
`AMErrorCode::USER_NOT_ASSOCIATED = 24,`  
`qevercloud::EDAMErrorCode::USER_NOT_REGISTERED = 25, qevercloud::EDAMErrorCode::USER_AL←`  
`READY_ASSOCIATED = 26, qevercloud::EDAMErrorCode::ACCOUNT_CLEAR = 27, qevercloud::EDAM←`  
`ErrorCode::SSO_AUTHENTICATION_REQUIRED = 28 }`
- `enum qevercloud::EDAMInvalidContactReason { qevercloud::EDAMInvalidContactReason::BAD_ADDRE←`  
`SS, qevercloud::EDAMInvalidContactReason::DUPLICATE_CONTACT, qevercloud::EDAMInvalidContact←`  
`Reason::NO_CONNECTION }`
- `enum qevercloud::ShareRelationshipPrivilegeLevel { qevercloud::ShareRelationshipPrivilegeLevel::RE←`  
`AD_NOTEBOOK = 0, qevercloud::ShareRelationshipPrivilegeLevel::READ_NOTEBOOK_PLUS_ACTI←`  
`VITY = 10, qevercloud::ShareRelationshipPrivilegeLevel::MODIFY_NOTEBOOK_PLUS_ACTIVITY = 20,`  
`qevercloud::ShareRelationshipPrivilegeLevel::FULL_ACCESS = 30 }`
- `enum qevercloud::PrivilegeLevel {`  
`qevercloud::PrivilegeLevel::NORMAL = 1, qevercloud::PrivilegeLevel::PREMIUM = 3, qevercloud::Privilege←`  
`Level::VIP = 5, qevercloud::PrivilegeLevel::MANAGER = 7,`  
`qevercloud::PrivilegeLevel::SUPPORT = 8, qevercloud::PrivilegeLevel::ADMIN = 9 }`
- `enum qevercloud::ServiceLevel { qevercloud::ServiceLevel::BASIC = 1, qevercloud::ServiceLevel::PLUS = 2,`  
`qevercloud::ServiceLevel::PREMIUM = 3, qevercloud::ServiceLevel::BUSINESS = 4 }`
- `enum qevercloud::QueryFormat { qevercloud::QueryFormat::USER = 1, qevercloud::QueryFormat::SEXP =`  
`2 }`
- `enum qevercloud::NoteSortOrder {`  
`qevercloud::NoteSortOrder::CREATED = 1, qevercloud::NoteSortOrder::UPDATED = 2, qevercloud::Note←`  
`SortOrder::RELEVANCE = 3, qevercloud::NoteSortOrder::UPDATE_SEQUENCE_NUMBER = 4,`  
`qevercloud::NoteSortOrder::TITLE = 5 }`
- `enum qevercloud::PremiumOrderStatus {`  
`qevercloud::PremiumOrderStatus::NONE = 0, qevercloud::PremiumOrderStatus::PENDING = 1, qevercloud←`  
`::PremiumOrderStatus::ACTIVE = 2, qevercloud::PremiumOrderStatus::FAILED = 3,`  
`qevercloud::PremiumOrderStatus::CANCELLATION_PENDING = 4, qevercloud::PremiumOrderStatus::C←`  
`ANCELED = 5 }`
- `enum qevercloud::SharedNotebookPrivilegeLevel {`  
`qevercloud::SharedNotebookPrivilegeLevel::READ_NOTEBOOK = 0, qevercloud::SharedNotebook←`  
`PrivilegeLevel::MODIFY_NOTEBOOK_PLUS_ACTIVITY = 1, qevercloud::SharedNotebookPrivilegeLevel←`  
`::READ_NOTEBOOK_PLUS_ACTIVITY = 2, qevercloud::SharedNotebookPrivilegeLevel::GROUP = 3,`  
`qevercloud::SharedNotebookPrivilegeLevel::FULL_ACCESS = 4, qevercloud::SharedNotebookPrivilege←`  
`Level::BUSINESS_FULL_ACCESS = 5 }`

- enum `qevercloud::SharedNotePrivilegeLevel` { `qevercloud::SharedNotePrivilegeLevel::READ_NOTE` = 0, `qevercloud::SharedNotePrivilegeLevel::MODIFY_NOTE` = 1, `qevercloud::SharedNotePrivilegeLevel::FULL_ACCESS` = 2 }
- enum `qevercloud::SponsoredGroupRole` { `qevercloud::SponsoredGroupRole::GROUP_MEMBER` = 1, `qevercloud::SponsoredGroupRole::GROUP_ADMIN` = 2, `qevercloud::SponsoredGroupRole::GROUP_OWNER` = 3 }
- enum `qevercloud::BusinessUserRole` { `qevercloud::BusinessUserRole::ADMIN` = 1, `qevercloud::BusinessUserRole::NORMAL` = 2 }
- enum `qevercloud::BusinessUserStatus` { `qevercloud::BusinessUserStatus::ACTIVE` = 1, `qevercloud::BusinessUserStatus::DEACTIVATED` = 2 }
- enum `qevercloud::SharedNotebookInstanceRestrictions` { `qevercloud::SharedNotebookInstanceRestrictions::ASSIGNED` = 1, `qevercloud::SharedNotebookInstanceRestrictions::NO_SHARED_NOTEBOOKS` = 2 }
- enum `qevercloud::ReminderEmailConfig` { `qevercloud::ReminderEmailConfig::DO_NOT_SEND` = 1, `qevercloud::ReminderEmailConfig::SEND_DAILY_EMAIL` = 2 }
- enum `qevercloud::BusinessInvitationStatus` { `qevercloud::BusinessInvitationStatus::APPROVED` = 0, `qevercloud::BusinessInvitationStatus::REQUESTED` = 1, `qevercloud::BusinessInvitationStatus::REDEEMED` = 2 }
- enum `qevercloud::ContactType` { `qevercloud::ContactType::EVERNOTE` = 1, `qevercloud::ContactType::SMS` = 2, `qevercloud::ContactType::FACEBOOK` = 3, `qevercloud::ContactType::EMAIL` = 4, `qevercloud::ContactType::TWITTER` = 5, `qevercloud::ContactType::LINKEDIN` = 6 }
- enum `qevercloud::EntityType` { `qevercloud::EntityType::NOTE` = 1, `qevercloud::EntityType::NOTEBOOK` = 2, `qevercloud::EntityType::WORKSPACE` = 3 }
- enum `qevercloud::RecipientStatus` { `qevercloud::RecipientStatus::NOT_IN_MY_LIST` = 1, `qevercloud::RecipientStatus::IN_MY_LIST` = 2, `qevercloud::RecipientStatus::IN_MY_LIST_AND_DEFAULT_NOTEBOOK_OK` = 3 }
- enum `qevercloud::CanMoveToContainerStatus` { `qevercloud::CanMoveToContainerStatus::CAN_BE_MOVED` = 1, `qevercloud::CanMoveToContainerStatus::INSUFFICIENT_ENTITY_PRIVILEGE` = 2, `qevercloud::CanMoveToContainerStatus::INSUFFICIENT_CONTAINER_PRIVILEGE` = 3 }
- enum `qevercloud::RelatedContentType` { `qevercloud::RelatedContentType::NEWS_ARTICLE` = 1, `qevercloud::RelatedContentType::PROFILE_PERSON` = 2, `qevercloud::RelatedContentType::PROFILE_ORGANIZATION` = 3, `qevercloud::RelatedContentType::REFERENCE_MATERIAL` = 4 }
- enum `qevercloud::RelatedContentAccess` { `qevercloud::RelatedContentAccess::NOT_ACCESSIBLE` = 0, `qevercloud::RelatedContentAccess::DIRECT_LINK_ACCESS_OK` = 1, `qevercloud::RelatedContentAccess::DIRECT_LINK_LOGIN_REQUIRED` = 2, `qevercloud::RelatedContentAccess::DIRECT_LINK_EMBEDDED_VIEW` = 3 }
- enum `qevercloud::UserIdentityType` { `qevercloud::UserIdentityType::EVERNOTE_USERID` = 1, `qevercloud::UserIdentityType::EMAIL` = 2, `qevercloud::UserIdentityType::IDENTITYID` = 3 }

## Functions

- uint `qevercloud::qHash` (EDAMErrorCode value)
- `QEVERCLOUD_EXPORT` `QTextStream & qevercloud::operator<<` (`QTextStream &out`, const EDAMErrorCode value)
- `QEVERCLOUD_EXPORT` `QDebug & qevercloud::operator<<` (`QDebug &out`, const EDAMErrorCode value)
- uint `qevercloud::qHash` (EDAMInvalidContactReason value)
- `QEVERCLOUD_EXPORT` `QTextStream & qevercloud::operator<<` (`QTextStream &out`, const EDAMInvalidContactReason value)
- `QEVERCLOUD_EXPORT` `QDebug & qevercloud::operator<<` (`QDebug &out`, const EDAMInvalidContactReason value)
- uint `qevercloud::qHash` (ShareRelationshipPrivilegeLevel value)
- `QEVERCLOUD_EXPORT` `QTextStream & qevercloud::operator<<` (`QTextStream &out`, const ShareRelationshipPrivilegeLevel value)
- `QEVERCLOUD_EXPORT` `QDebug & qevercloud::operator<<` (`QDebug &out`, const ShareRelationshipPrivilegeLevel value)



- uint [qevercloud::qHash](#) (PrivilegeLevel value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const PrivilegeLevel value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const PrivilegeLevel value)
- uint [qevercloud::qHash](#) (ServiceLevel value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const ServiceLevel value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const ServiceLevel value)
- uint [qevercloud::qHash](#) (QueryFormat value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const QueryFormat value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const QueryFormat value)
- uint [qevercloud::qHash](#) (NoteSortOrder value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const NoteSortOrder value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const NoteSortOrder value)
- uint [qevercloud::qHash](#) (PremiumOrderStatus value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const PremiumOrderStatus value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const PremiumOrderStatus value)
- uint [qevercloud::qHash](#) (SharedNotebookPrivilegeLevel value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const SharedNotebookPrivilegeLevel value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const SharedNotebookPrivilegeLevel value)
- uint [qevercloud::qHash](#) (SharedNotePrivilegeLevel value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const SharedNotePrivilegeLevel value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const SharedNotePrivilegeLevel value)
- uint [qevercloud::qHash](#) (SponsoredGroupRole value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const SponsoredGroupRole value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const SponsoredGroupRole value)
- uint [qevercloud::qHash](#) (BusinessUserRole value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const BusinessUserRole value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const BusinessUserRole value)
- uint [qevercloud::qHash](#) (BusinessUserStatus value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const BusinessUserStatus value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const BusinessUserStatus value)
- uint [qevercloud::qHash](#) (SharedNotebookInstanceRestrictions value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const SharedNotebookInstanceRestrictions value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const SharedNotebookInstanceRestrictions value)
- uint [qevercloud::qHash](#) (ReminderEmailConfig value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const ReminderEmailConfig value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const ReminderEmailConfig value)

- uint [qevercloud::qHash](#) (BusinessInvitationStatus value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const BusinessInvitationStatus value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const BusinessInvitationStatus value)
- uint [qevercloud::qHash](#) (ContactType value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const ContactType value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const ContactType value)
- uint [qevercloud::qHash](#) (EntityType value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const EntityType value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const EntityType value)
- uint [qevercloud::qHash](#) (RecipientStatus value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const RecipientStatus value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const RecipientStatus value)
- uint [qevercloud::qHash](#) (CanMoveToContainerStatus value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const CanMoveToContainerStatus value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const CanMoveToContainerStatus value)
- uint [qevercloud::qHash](#) (RelatedContentType value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const RelatedContentType value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const RelatedContentType value)
- uint [qevercloud::qHash](#) (RelatedContentAccess value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const RelatedContentAccess value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const RelatedContentAccess value)
- uint [qevercloud::qHash](#) (UserIdentityType value)
- [QEVERCLOUD\\_EXPORT](#) QTextStream & [qevercloud::operator<<](#) (QTextStream &out, const UserIdentityType value)
- [QEVERCLOUD\\_EXPORT](#) QDebug & [qevercloud::operator<<](#) (QDebug &out, const UserIdentityType value)

## 8.5 EventLoopFinisher.h File Reference

```
#include "Export.h"
#include "Helpers.h"
#include <QEventLoop>
#include <QObject>
```

### Classes

- class [qevercloud::EventLoopFinisher](#)

### Namespaces

- [qevercloud](#)

## 8.6 EverCloudException.h File Reference

```
#include "Export.h"
#include "Helpers.h"
#include <QObject>
#include <QString>
#include <exception>
#include <memory>
```

### Classes

- class [qevercloud::EverCloudException](#)
- class [qevercloud::EverCloudExceptionData](#)  
*[EverCloudException](#) counterpart for asynchronous API.*
- class [qevercloud::EvernoteException](#)
- class [qevercloud::EvernoteExceptionData](#)

### Namespaces

- [qevercloud](#)

### Typedefs

- using [qevercloud::EverCloudExceptionDataPtr](#) = `std::shared_ptr< EverCloudExceptionData >`

### Variables

- class [QEVERCLOUD\\_EXPORT qevercloud::EverCloudExceptionData](#)

## 8.7 Exceptions.h File Reference

```
#include "EverCloudException.h"
#include "Export.h"
#include "Optional.h"
#include "generated/EDAMErrorCode.h"
#include "generated/Types.h"
#include <QNetworkReply>
#include <QObject>
#include <QString>
```

## Classes

- class [qevercloud::NetworkException](#)  
*The [NetworkException](#) class represents QNetworkReply level errors.*
- class [qevercloud::NetworkExceptionData](#)
- class [qevercloud::ThriftException](#)
- class [qevercloud::ThriftExceptionData](#)
- class [qevercloud::EDAMUserExceptionData](#)
- class [qevercloud::EDAMSystemExceptionData](#)
- class [qevercloud::EDAMNotFoundExceptionData](#)
- class [qevercloud::EDAMInvalidContactsExceptionData](#)
- class [qevercloud::EDAMSystemExceptionRateLimitReached](#)
- class [qevercloud::EDAMSystemExceptionRateLimitReachedData](#)
- class [qevercloud::EDAMSystemExceptionAuthExpired](#)
- class [qevercloud::EDAMSystemExceptionAuthExpiredData](#)

## Namespaces

- [qevercloud](#)

## 8.8 Export.h File Reference

```
#include <QtCore/QtGlobal>
```

## Macros

- `#define QEVERCLOUD\_EXPORT Q_DECL_IMPORT`

### 8.8.1 Macro Definition Documentation

#### 8.8.1.1 QEVERCLOUD\_EXPORT

```
#define QEVERCLOUD_EXPORT Q_DECL_IMPORT
```

Original work: Copyright (c) 2014 Sergey Skoblikov Modified work: Copyright (c) 2015-2019 Dmitry Ivanov

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

## 8.9 Globals.h File Reference

```
#include "Export.h"  
#include <QNetworkProxy>
```

## Namespaces

- [qevercloud](#)

## Functions

- [QEVCLOUD\\_EXPORT](#) [QNetworkProxy](#) [qevercloud::evernoteNetworkProxy](#) ()
- [QEVCLOUD\\_EXPORT](#) void [qevercloud::setEvernoteNetworkProxy](#) ([QNetworkProxy](#) proxy)
- [QEVCLOUD\\_EXPORT](#) void [qevercloud::resetEvernoteNetworkProxy](#) ()
- [QEVCLOUD\\_EXPORT](#) int [qevercloud::libraryVersion](#) ()
- [QEVCLOUD\\_EXPORT](#) void [qevercloud::initializeQEverCloud](#) ()

## 8.10 Helpers.h File Reference

```
#include <QtGlobal>
#include <QObject>
#include "VersionInfo.h"
```

## Classes

- class [qevercloud::QAssociativeContainerReferenceWrapper](#)< Container >
- struct [qevercloud::QAssociativeContainerReferenceWrapper](#)< Container >::iterator
- class [qevercloud::QAssociativeContainerConstReferenceWrapper](#)< Container >
- struct [qevercloud::QAssociativeContainerConstReferenceWrapper](#)< Container >::iterator

## Namespaces

- [qevercloud](#)

## Functions

- template<class Container >  
[QAssociativeContainerReferenceWrapper](#)< Container > [qevercloud::toRange](#) (Container &container)
- template<class Container >  
[QAssociativeContainerConstReferenceWrapper](#)< Container > [qevercloud::toRange](#) (const Container &container)

## 8.11 InkNotelImageDownloader.h File Reference

```
#include "AsyncResult.h"
#include "Export.h"
#include "generated/Types.h"
#include <QByteArray>
#include <QString>
#include <QNetworkAccessManager>
```

## Classes

- class [qevercloud::InkNoteImageDownloader](#)

the [InkNoteImageDownloader](#) class is for downloading the images of ink notes which can be created with the official Evernote client on Windows (only with it, at least at the time of this writing).

## Namespaces

- [qevercloud](#)

## 8.12 Log.h File Reference

```
#include "Export.h"
#include "Helpers.h"
#include <QDateTime>
#include <QDebug>
#include <QObject>
#include <QTextStream>
#include <memory>
```

## Classes

- class [qevercloud::ILogger](#)

## Namespaces

- [qevercloud](#)

## Macros

- #define [\\_\\_QEVERCLOUD\\_LOG\\_BASE](#)(component, level, message)
- #define [QEC\\_TRACE](#)(component, message) [\\_\\_QEVERCLOUD\\_LOG\\_BASE](#)(component, LogLevel::Trace, message) \
- #define [QEC\\_DEBUG](#)(component, message) [\\_\\_QEVERCLOUD\\_LOG\\_BASE](#)(component, LogLevel::Debug, message) \
- #define [QEC\\_INFO](#)(component, message) [\\_\\_QEVERCLOUD\\_LOG\\_BASE](#)(component, LogLevel::Info, message) \
- #define [QEC\\_WARNING](#)(component, message) [\\_\\_QEVERCLOUD\\_LOG\\_BASE](#)(component, LogLevel::Warn, message) \
- #define [QEC\\_ERROR](#)(component, message) [\\_\\_QEVERCLOUD\\_LOG\\_BASE](#)(component, LogLevel::Error, message) \

## Typedefs

- using [qevercloud::ILoggerPtr](#) = std::shared\_ptr< ILogger >

## Enumerations

- enum `qevercloud::LogLevel` {  
`qevercloud::LogLevel::Trace` = 0, `qevercloud::LogLevel::Debug`, `qevercloud::LogLevel::Info`, `qevercloud::LogLevel::Warn`,  
`qevercloud::LogLevel::Error` }

## Functions

- `QEVERCLOUD_EXPORT` `QTextStream & qevercloud::operator<<` (`QTextStream &out`, `const LogLevel level`)
- `QEVERCLOUD_EXPORT` `QDebug & qevercloud::operator<<` (`QDebug &out`, `const LogLevel level`)
- `QEVERCLOUD_EXPORT` `ILoggerPtr qevercloud::logger` ()
- `QEVERCLOUD_EXPORT` `void qevercloud::setLogger` (`ILoggerPtr logger`)
- `QEVERCLOUD_EXPORT` `ILoggerPtr qevercloud::nullLogger` ()
- `QEVERCLOUD_EXPORT` `ILoggerPtr qevercloud::newStdErrLogger` (`LogLevel level=LogLevel::Warn`)

## 8.12.1 Macro Definition Documentation

### 8.12.1.1 \_\_QEVERCLOUD\_LOG\_BASE

```
#define __QEVERCLOUD_LOG_BASE(  
    component,  
    level,  
    message )
```

#### Value:

```
{  
    auto __qevercloudLogger = ::qevercloud::logger();  
    if (__qevercloudLogger->shouldLog(level, component))  
    {  
        QString msg;  
        QDebug dbg(&msg);  
        dbg.nospace();  
        dbg.noquote();  
        dbg << message;  
        __qevercloudLogger->log(  
            level,  
            component,  
            __FILE__,  
            __LINE__,  
            QDateTime::currentMSecsSinceEpoch(),  
            msg);  
    }  
}
```

### 8.12.1.2 QEC\_DEBUG

```
#define QEC_DEBUG(  
    component,  
    message ) __QEVERCLOUD_LOG_BASE(component, LogLevel::Debug, message) \
```

### 8.12.1.3 QEC\_ERROR

```
#define QEC_ERROR(
    component,
    message ) __QEVERCLOUD_LOG_BASE(component, LogLevel::Error, message) \
```

### 8.12.1.4 QEC\_INFO

```
#define QEC_INFO(
    component,
    message ) __QEVERCLOUD_LOG_BASE(component, LogLevel::Info, message) \
```

### 8.12.1.5 QEC\_TRACE

```
#define QEC_TRACE(
    component,
    message ) __QEVERCLOUD_LOG_BASE(component, LogLevel::Trace, message) \
```

### 8.12.1.6 QEC\_WARNING

```
#define QEC_WARNING(
    component,
    message ) __QEVERCLOUD_LOG_BASE(component, LogLevel::Warn, message) \
```

## 8.13 OAuth.h File Reference

```
#include "Export.h"
#include "Helpers.h"
#include "Printable.h"
#include "generated/Types.h"
#include <QDialog>
#include <QList>
#include <QNetworkCookie>
#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.*

## 8.14 Optional.h File Reference

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

## Classes

- class [qevercloud::Optional< T >](#)

## Namespaces

- [qevercloud](#)

## 8.15 Printable.h File Reference

```
#include "Export.h"  
#include <QTextStream>  
#include <QDebug>
```

## Classes

- class [qevercloud::Printable](#)

## Namespaces

- [qevercloud](#)

## 8.16 QEverCloud.h File Reference

```
#include "AsyncResult.h"
#include "DurableService.h"
#include "EventLoopFinisher.h"
#include "EverCloudException.h"
#include "Exceptions.h"
#include "Export.h"
#include "Globals.h"
#include "Helpers.h"
#include "InkNoteImageDownloader.h"
#include "Log.h"
#include "Optional.h"
#include "Printable.h"
#include "RequestContext.h"
#include "Thumbnail.h"
#include "VersionInfo.h"
#include "generated/EDAMErrorCode.h"
#include "generated/Constants.h"
#include "generated/Services.h"
#include "generated/Types.h"
```

## 8.17 QEverCloudOAuth.h File Reference

```
#include "OAuth.h"
#include "QEverCloud.h"
```

## 8.18 README.md File Reference

## 8.19 RequestContext.h File Reference

```
#include "Export.h"
#include <QDebug>
#include <QList>
#include <QNetworkCookie>
#include <QTextStream>
#include <QUuid>
#include <memory>
```

### Classes

- class [qevercloud::IRequestContext](#)

### Namespaces

- [qevercloud](#)

## Typedefs

- using [qevercloud::IRequestContextPtr](#) = std::shared\_ptr< IRequestContext >

## Functions

- [QEVERCLOUD\\_EXPORT](#) IRequestContextPtr [qevercloud::newRequestContext](#) (QString authenticationToken={}, qint64 requestTimeout=DEFAULT\_REQUEST\_TIMEOUT\_MSEC, bool increaseRequestTimeoutExponentially=DEFAULT\_REQUEST\_TIMEOUT\_EXPONENTIAL\_INCREASE, qint64 maxRequestTimeout=DEFAULT\_MAX\_REQUEST\_TIMEOUT\_MSEC, quint32 maxRequestRetryCount=DEFAULT\_MAX\_REQUEST\_RETRY\_COUNT, QList< QNetworkCookie > cookies={})

## 8.20 Servers.h File Reference

```
#include "../Export.h"
#include "../Optional.h"
#include "../RequestContext.h"
#include "Constants.h"
#include "Types.h"
#include <QObject>
#include <functional>
```

## Classes

- class [qevercloud::NoteStoreServer](#)  
*The [NoteStoreServer](#) class represents customizable server for NoteStore requests. It is primarily used for testing of QEverCloud.*
- class [qevercloud::UserStoreServer](#)  
*The [UserStoreServer](#) class represents customizable server for UserStore requests. It is primarily used for testing of QEverCloud.*

## Namespaces

- [qevercloud](#)

## 8.21 Services.h File Reference

```
#include "../Export.h"
#include "../AsyncResult.h"
#include "../DurableService.h"
#include "../Optional.h"
#include "../RequestContext.h"
#include "Constants.h"
#include "Types.h"
#include <QObject>
```

## Classes

- class [qevercloud::INoteStore](#)
- class [qevercloud::IUserStore](#)

## Namespaces

- [qevercloud](#)

## Typedefs

- using [qevercloud::INoteStorePtr](#) = std::shared\_ptr< INoteStore >
- using [qevercloud::IUserStorePtr](#) = std::shared\_ptr< IUserStore >

## Functions

- [QEVERCLOUD\\_EXPORT](#) INoteStore \* [qevercloud::newNoteStore](#) (QString noteStoreUrl={}, IRequestContextPtr ctx={}, QObject \*parent=nullptr, IRetryPolicyPtr retryPolicy={})
- [QEVERCLOUD\\_EXPORT](#) IUserStore \* [qevercloud::newUserStore](#) (QString userStoreUrl={}, IRequestContextPtr ctx={}, QObject \*parent=nullptr, IRetryPolicyPtr retryPolicy={})

## 8.22 Thumbnail.h File Reference

```
#include "AsyncResult.h"
#include "Export.h"
#include "generated/Types.h"
#include <QByteArray>
#include <QNetworkAccessManager>
#include <QString>
#include <utility>
```

## Classes

- class [qevercloud::Thumbnail](#)  
*The class is for downloading thumbnails for notes and resources from Evernote servers.*

## Namespaces

- [qevercloud](#)

## 8.23 Types.h File Reference

```
#include "../Export.h"
#include "../Optional.h"
#include "../Printable.h"
#include "EDAMErrorCode.h"
#include <QByteArray>
#include <QDateTime>
#include <QHash>
#include <QList>
#include <QMap>
#include <QMetaType>
#include <QSet>
#include <QStringList>
#include <QVariant>
```

### Classes

- class [qevercloud::EverCloudLocalData](#)

*The [EverCloudLocalData](#) class contains several data elements which are not synchronized with Evernote service but which are nevertheless useful in applications using QEverCloud to implement feature rich full sync Evernote clients. Values of this class' types are contained within QEverCloud types corresponding to actual Evernote API types.*

- struct [qevercloud::SyncState](#)
- struct [qevercloud::SyncChunkFilter](#)
- struct [qevercloud::NoteFilter](#)
- struct [qevercloud::NotesMetadataResultSpec](#)
- struct [qevercloud::NoteCollectionCounts](#)
- struct [qevercloud::NoteResultSpec](#)
- struct [qevercloud::NoteVersionId](#)
- struct [qevercloud::RelatedQuery](#)
- struct [qevercloud::RelatedResultSpec](#)
- struct [qevercloud::ShareRelationshipRestrictions](#)
- struct [qevercloud::MemberShareRelationship](#)
- struct [qevercloud::NoteShareRelationshipRestrictions](#)
- struct [qevercloud::NoteMemberShareRelationship](#)
- struct [qevercloud::NoteInvitationShareRelationship](#)
- struct [qevercloud::NoteShareRelationships](#)
- struct [qevercloud::ManageNoteSharesParameters](#)
- struct [qevercloud::Data](#)
- struct [qevercloud::UserAttributes](#)
- struct [qevercloud::BusinessUserAttributes](#)
- struct [qevercloud::Accounting](#)
- struct [qevercloud::BusinessUserInfo](#)
- struct [qevercloud::AccountLimits](#)
- struct [qevercloud::User](#)
- struct [qevercloud::Contact](#)
- struct [qevercloud::Identity](#)
- struct [qevercloud::Tag](#)
- struct [qevercloud::LazyMap](#)
- struct [qevercloud::ResourceAttributes](#)
- struct [qevercloud::Resource](#)
- struct [qevercloud::NoteAttributes](#)
- struct [qevercloud::SharedNote](#)

- struct [qevercloud::NoteRestrictions](#)
- struct [qevercloud::NoteLimits](#)
- struct [qevercloud::Note](#)
- struct [qevercloud::Publishing](#)
- struct [qevercloud::BusinessNotebook](#)
- struct [qevercloud::SavedSearchScope](#)
- struct [qevercloud::SavedSearch](#)
- struct [qevercloud::SharedNotebookRecipientSettings](#)
- struct [qevercloud::NotebookRecipientSettings](#)
- struct [qevercloud::SharedNotebook](#)
- struct [qevercloud::CanMoveToContainerRestrictions](#)
- struct [qevercloud::NotebookRestrictions](#)
- struct [qevercloud::Notebook](#)
- struct [qevercloud::LinkedNotebook](#)
- struct [qevercloud::NotebookDescriptor](#)
- struct [qevercloud::UserProfile](#)
- struct [qevercloud::RelatedContentImage](#)
- struct [qevercloud::RelatedContent](#)
- struct [qevercloud::BusinessInvitation](#)
- struct [qevercloud::UserIdentity](#)
- struct [qevercloud::PublicUserInfo](#)
- struct [qevercloud::UserUrls](#)
- struct [qevercloud::AuthenticationResult](#)
- struct [qevercloud::BootstrapSettings](#)
- struct [qevercloud::BootstrapProfile](#)
- struct [qevercloud::BootstrapInfo](#)
- class [qevercloud::EDAMUserException](#)
- class [qevercloud::EDAMSystemException](#)
- class [qevercloud::EDAMNotFoundException](#)
- class [qevercloud::EDAMInvalidContactsException](#)
- struct [qevercloud::SyncChunk](#)
- struct [qevercloud::NoteList](#)
- struct [qevercloud::NoteMetadata](#)
- struct [qevercloud::NotesMetadataList](#)
- struct [qevercloud::NoteEmailParameters](#)
- struct [qevercloud::RelatedResult](#)
- struct [qevercloud::UpdateNoteIfUsnMatchesResult](#)
- struct [qevercloud::InvitationShareRelationship](#)
- struct [qevercloud::ShareRelationships](#)
- struct [qevercloud::ManageNotebookSharesParameters](#)
- struct [qevercloud::ManageNotebookSharesError](#)
- struct [qevercloud::ManageNotebookSharesResult](#)
- struct [qevercloud::SharedNoteTemplate](#)
- struct [qevercloud::NotebookShareTemplate](#)
- struct [qevercloud::CreateOrUpdateNotebookSharesResult](#)
- struct [qevercloud::ManageNoteSharesError](#)
- struct [qevercloud::ManageNoteSharesResult](#)

## Namespaces

- [qevercloud](#)

## Typedefs

- using [qevercloud::InvalidationSequenceNumber](#) = qint64
- using [qevercloud::IdentityID](#) = qint64
- using [qevercloud::UserID](#) = qint32
- using [qevercloud::Guid](#) = QString
- using [qevercloud::Timestamp](#) = qint64
- using [qevercloud::MessageEventID](#) = qint64
- using [qevercloud::MessageThreadID](#) = qint64





# Index

- \_\_QEVERCLOUD\_LOG\_BASE
  - Log.h, 529
- ~AsyncResult
  - qevercloud::AsyncResult, 100
- ~EDAMInvalidContactsException
  - qevercloud::EDAMInvalidContactsException, 129
- ~EDAMNotFoundException
  - qevercloud::EDAMNotFoundException, 134
- ~EDAMSystemException
  - qevercloud::EDAMSystemException, 138
- ~EDAMUserException
  - qevercloud::EDAMUserException, 147
- ~EventLoopFinisher
  - qevercloud::EventLoopFinisher, 151
- ~EverCloudException
  - qevercloud::EverCloudException, 152
- ~EverCloudLocalData
  - qevercloud::EverCloudLocalData, 157
- ~EvernoteOAuthDialog
  - qevercloud::EvernoteOAuthDialog, 164
- ~IRequestContext
  - qevercloud::IRequestContext, 244
- ~InkNoteImageDownloader
  - qevercloud::InkNoteImageDownloader, 174
- ~NetworkException
  - qevercloud::NetworkException, 285
- ~Printable
  - qevercloud::Printable, 410
- ~ThriftException
  - qevercloud::ThriftException, 475
- ~Thumbnail
  - qevercloud::Thumbnail, 480
- accessType
  - qevercloud::RelatedContent, 420
- accountEmailDomain
  - qevercloud::BootstrapSettings, 109
- accountLimits
  - qevercloud::User, 486
- accounting
  - qevercloud::User, 486
- active
  - qevercloud::Note, 289
  - qevercloud::Resource, 435
  - qevercloud::User, 486
- alternateData
  - qevercloud::Resource, 435
- altitude
  - qevercloud::NoteAttributes, 294
  - qevercloud::ResourceAttributes, 438
- applicationData
  - qevercloud::NoteAttributes, 295
  - qevercloud::ResourceAttributes, 438
- asIs
  - qevercloud::AsyncResult, 100
- ascending
  - qevercloud::NoteFilter, 324
  - qevercloud::Publishing, 415
- AsyncRequest
  - qevercloud::IDurableService::AsyncRequest, 97
- AsyncResult
  - qevercloud::AsyncResult, 99, 100
- AsyncResult.h, 515
- AsyncServiceCall
  - qevercloud::IDurableService, 171
- attachment
  - qevercloud::ResourceAttributes, 439
- attributes
  - qevercloud::Note, 289
  - qevercloud::NoteMetadata, 336
  - qevercloud::Resource, 435
  - qevercloud::User, 486
  - qevercloud::UserProfile, 500
- authenticate
  - qevercloud::EvernoteOAuthWebView, 166
- authenticateLongSession
  - qevercloud::IUserStore, 250
- authenticateLongSessionAsync
  - qevercloud::IUserStore, 252
- authenticateLongSessionRequest
  - qevercloud::UserStoreServer, 503
- authenticateLongSessionRequestReady
  - qevercloud::UserStoreServer, 503
- authenticateToBusiness
  - qevercloud::IUserStore, 252
- authenticateToBusinessAsync
  - qevercloud::IUserStore, 253
- authenticateToBusinessRequest
  - qevercloud::UserStoreServer, 504
- authenticateToBusinessRequestReady
  - qevercloud::UserStoreServer, 504
- authenticateToSharedNote
  - qevercloud::INoteStore, 181
- authenticateToSharedNoteAsync
  - qevercloud::INoteStore, 182
- authenticateToSharedNoteRequest
  - qevercloud::NoteStoreServer, 360
- authenticateToSharedNoteRequestReady
  - qevercloud::NoteStoreServer, 360

- authenticateToSharedNotebook
  - qevercloud::INoteStore, 182
- authenticateToSharedNotebookAsync
  - qevercloud::INoteStore, 183
- authenticateToSharedNotebookRequest
  - qevercloud::NoteStoreServer, 359
- authenticateToSharedNotebookRequestReady
  - qevercloud::NoteStoreServer, 360
- authenticationFailed
  - qevercloud::EvernoteOAuthWebView, 167
- authenticationFinished
  - qevercloud::EvernoteOAuthWebView, 167
- authenticationSucceeded
  - qevercloud::EvernoteOAuthWebView, 167
- authenticationToken
  - qevercloud::AuthenticationResult, 103
  - qevercloud::EvernoteOAuthWebView::OAuthResult, 400
  - qevercloud::IRequestContext, 244
- author
  - qevercloud::NoteAttributes, 295
- authors
  - qevercloud::RelatedContent, 420
- availablePoints
  - qevercloud::Accounting, 90
- begin
  - qevercloud::QAssociativeContainerConstReferenceWrapper, 417
  - qevercloud::QAssociativeContainerReferenceWrapper, 418
- bestPrivilege
  - qevercloud::MemberShareRelationship, 283
- blocked
  - qevercloud::Identity, 169
- body
  - qevercloud::Data, 127
- bodyHash
  - qevercloud::Data, 127
- businessAddress
  - qevercloud::UserAttributes, 491
- businessId
  - qevercloud::Accounting, 91
  - qevercloud::BusinessInvitation, 112
  - qevercloud::BusinessUserInfo, 119
  - qevercloud::LinkedNotebook, 266
- BusinessInvitationStatus
  - qevercloud, 32
- businessName
  - qevercloud::Accounting, 91
  - qevercloud::BusinessUserInfo, 119
- businessNotebook
  - qevercloud::Notebook, 301
- businessRole
  - qevercloud::Accounting, 91
- businessUserInfo
  - qevercloud::User, 487
- BusinessUserRole
  - qevercloud, 32
- BusinessUserStatus
  - qevercloud, 33
- CLASSIFICATION\_RECIPE\_SERVICE\_RECIPE
  - qevercloud, 58
- CLASSIFICATION\_RECIPE\_USER\_NON\_RECIPE
  - qevercloud, 58
- CLASSIFICATION\_RECIPE\_USER\_RECIPE
  - qevercloud, 58
- cacheExpires
  - qevercloud::RelatedResult, 428
- cacheKey
  - qevercloud::RelatedQuery, 426
  - qevercloud::RelatedResult, 428
- cameraMake
  - qevercloud::ResourceAttributes, 439
- cameraModel
  - qevercloud::ResourceAttributes, 439
- canMoveToContainer
  - qevercloud::CanMoveToContainerRestrictions, 121
- canMoveToContainerRestrictions
  - qevercloud::NotebookRestrictions, 310
- CanMoveToContainerStatus
  - qevercloud, 33
- ccAddresses
  - qevercloud::NoteEmailParameters, 321
- checkVersion
  - qevercloud::IUserStore, 253
- checkVersionAsync
  - qevercloud::IUserStore, 254
- checkVersionRequest
  - qevercloud::UserStoreServer, 504
- checkVersionRequestReady
  - qevercloud::UserStoreServer, 504
- chunkHighUSN
  - qevercloud::SyncChunk, 461
- Classifications
  - qevercloud::NoteAttributes, 294
- classifications
  - qevercloud::NoteAttributes, 295, 299
- clear
  - qevercloud::Optional, 403
- clientWillIndex
  - qevercloud::ResourceAttributes, 439
- clipFullPage
  - qevercloud::UserAttributes, 491
- clipUrl
  - qevercloud::RelatedContent, 420
- clone
  - qevercloud::IRequestContext, 244
- comments
  - qevercloud::UserAttributes, 491
- companyStartDate
  - qevercloud::BusinessUserAttributes, 117
- completeTwoFactorAuthentication
  - qevercloud::IUserStore, 254
- completeTwoFactorAuthenticationAsync
  - qevercloud::IUserStore, 255
- completeTwoFactorAuthenticationRequest

- qevercloud::UserStoreServer, 504
- completeTwoFactorAuthenticationRequestReady
  - qevercloud::UserStoreServer, 504
- conflictSourceNoteGuid
  - qevercloud::NoteAttributes, 295
- Constants.h, 515
- contact
  - qevercloud::Identity, 169
  - qevercloud::Notebook, 301
  - qevercloud::RelatedContent, 420
- contactName
  - qevercloud::NotebookDescriptor, 305
- ContactType
  - qevercloud, 33
- contacts
  - qevercloud::EDAMInvalidContactsException, 130
- containingNotebooks
  - qevercloud::RelatedResult, 429
- content
  - qevercloud::Note, 289
- contentClass
  - qevercloud::NoteAttributes, 295
- contentHash
  - qevercloud::Note, 290
- contentId
  - qevercloud::RelatedContent, 420
- contentLength
  - qevercloud::Note, 290
  - qevercloud::NoteMetadata, 336
- contentType
  - qevercloud::RelatedContent, 420
- context
  - qevercloud::NoteFilter, 324
  - qevercloud::RelatedQuery, 426
- cookies
  - qevercloud::EvernoteOAuthWebView::OAuth←  
Result, 400
  - qevercloud::IRequestContext, 244
- copyNote
  - qevercloud::INoteStore, 183
- copyNoteAsync
  - qevercloud::INoteStore, 184
- copyNoteRequest
  - qevercloud::NoteStoreServer, 360
- copyNoteRequestReady
  - qevercloud::NoteStoreServer, 360
- createLinkedNotebook
  - qevercloud::INoteStore, 184
- createLinkedNotebookAsync
  - qevercloud::INoteStore, 185
- createLinkedNotebookRequest
  - qevercloud::NoteStoreServer, 360
- createLinkedNotebookRequestReady
  - qevercloud::NoteStoreServer, 361
- createNote
  - qevercloud::INoteStore, 185
- createNoteAsync
  - qevercloud::INoteStore, 186
- createNoteRequest
  - qevercloud::NoteStoreServer, 361
- createNoteRequestReady
  - qevercloud::NoteStoreServer, 361
- createNotebook
  - qevercloud::INoteStore, 187
- createNotebookAsync
  - qevercloud::INoteStore, 188
- createNotebookRequest
  - qevercloud::NoteStoreServer, 361
- createNotebookRequestReady
  - qevercloud::NoteStoreServer, 361
- createOrUpdateNotebookShares
  - qevercloud::INoteStore, 188
- createOrUpdateNotebookSharesAsync
  - qevercloud::INoteStore, 189
- createOrUpdateNotebookSharesRequest
  - qevercloud::NoteStoreServer, 361
- createOrUpdateNotebookSharesRequestReady
  - qevercloud::NoteStoreServer, 362
- createPostRequest
  - qevercloud::Thumbnail, 480
- createSearch
  - qevercloud::INoteStore, 189
- createSearchAsync
  - qevercloud::INoteStore, 190
- createSearchRequest
  - qevercloud::NoteStoreServer, 362
- createSearchRequestReady
  - qevercloud::NoteStoreServer, 362
- createTag
  - qevercloud::INoteStore, 190
- createTagAsync
  - qevercloud::INoteStore, 191
- createTagRequest
  - qevercloud::NoteStoreServer, 362
- createTagRequestReady
  - qevercloud::NoteStoreServer, 362
- created
  - qevercloud::BusinessInvitation, 112
  - qevercloud::Note, 290
  - qevercloud::NoteMetadata, 337
  - qevercloud::User, 487
- creatorId
  - qevercloud::NoteAttributes, 296
- currency
  - qevercloud::Accounting, 91
- currentTime
  - qevercloud::AuthenticationResult, 103
  - qevercloud::SyncChunk, 461
  - qevercloud::SyncState, 470
- dailyEmailLimit
  - qevercloud::UserAttributes, 491
- data
  - qevercloud::Resource, 435
- date
  - qevercloud::RelatedContent, 420
- dateAgreedToTermsOfService

- qevercloud::UserAttributes, 492
- deactivated
  - qevercloud::Identity, 169
- Debug
  - qevercloud, 37
- debugInfo
  - qevercloud::NoteList, 331
  - qevercloud::NotesMetadataList, 349
  - qevercloud::RelatedResult, 429
- defaultLatitude
  - qevercloud::UserAttributes, 492
- defaultLocationName
  - qevercloud::UserAttributes, 492
- defaultLongitude
  - qevercloud::UserAttributes, 492
- defaultNotebook
  - qevercloud::Notebook, 301
- deleteNote
  - qevercloud::INoteStore, 191
- deleteNoteAsync
  - qevercloud::INoteStore, 192
- deleteNoteRequest
  - qevercloud::NoteStoreServer, 362
- deleteNoteRequestReady
  - qevercloud::NoteStoreServer, 363
- deleted
  - qevercloud::Note, 290
  - qevercloud::NoteMetadata, 337
  - qevercloud::User, 487
- department
  - qevercloud::BusinessUserAttributes, 117
- Dict
  - qevercloud::EverCloudLocalData, 157
- dict
  - qevercloud::EverCloudLocalData, 158, 159
- dirty
  - qevercloud::EverCloudLocalData, 158
- displayName
  - qevercloud::InvitationShareRelationship, 242
  - qevercloud::MemberShareRelationship, 283
  - qevercloud::NoteInvitationShareRelationship, 327
  - qevercloud::NoteMemberShareRelationship, 334
- download
  - qevercloud::InkNoteImageDownloader, 175
  - qevercloud::Thumbnail, 481
- downloadAsync
  - qevercloud::Thumbnail, 481
- DurableService
  - qevercloud::AsyncResult, 101
- DurableService.h, 520
- duration
  - qevercloud::Resource, 435
- EDAM\_APP\_RATING\_MAX
  - qevercloud, 59
- EDAM\_APP\_RATING\_MIN
  - qevercloud, 59
- EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX
  - qevercloud, 59
- EDAM\_APPLICATIONDATA\_NAME\_LEN\_MAX
  - qevercloud, 59
- EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN
  - qevercloud, 59
- EDAM\_APPLICATIONDATA\_NAME\_REGEX
  - qevercloud, 59
- EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MAX
  - qevercloud, 59
- EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MIN
  - qevercloud, 60
- EDAM\_APPLICATIONDATA\_VALUE\_REGEX
  - qevercloud, 60
- EDAM\_ATTRIBUTE\_LEN\_MAX
  - qevercloud, 60
- EDAM\_ATTRIBUTE\_LEN\_MIN
  - qevercloud, 60
- EDAM\_ATTRIBUTE\_LIST\_MAX
  - qevercloud, 60
- EDAM\_ATTRIBUTE\_MAP\_MAX
  - qevercloud, 60
- EDAM\_ATTRIBUTE\_REGEX
  - qevercloud, 60
- EDAM\_BUSINESS\_MARKETING\_CODE\_REGEX\_P↔
  - ATTEN
  - qevercloud, 61
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_L↔
  - EN\_MAX
  - qevercloud, 61
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_L↔
  - EN\_MIN
  - qevercloud, 61
- EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTION\_R↔
  - EGEX
  - qevercloud, 61
- EDAM\_BUSINESS\_NOTEBOOKS\_MAX
  - qevercloud, 61
- EDAM\_BUSINESS\_NOTES\_MAX
  - qevercloud, 61
- EDAM\_BUSINESS\_PHONE\_NUMBER\_LEN\_MAX
  - qevercloud, 61
- EDAM\_BUSINESS\_TAGS\_MAX
  - qevercloud, 62
- EDAM\_BUSINESS\_URI\_LEN\_MAX
  - qevercloud, 62
- EDAM\_BUSINESS\_WORKSPACES\_MAX
  - qevercloud, 62
- EDAM\_CONNECTED\_IDENTITY\_REQUEST\_MAX
  - qevercloud, 62
- EDAM\_CONTENT\_CLASS\_FOOD\_MEAL
  - qevercloud, 62
- EDAM\_CONTENT\_CLASS\_HELLO\_ENCOUNTER
  - qevercloud, 62
- EDAM\_CONTENT\_CLASS\_HELLO\_PROFILE
  - qevercloud, 62
- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_NOTEBOOK
  - OOK
  - qevercloud, 63
- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_PREFIX

- qevercloud, [63](#)
- EDAM\_CONTENT\_CLASS\_SKETCH\_PDF
  - qevercloud, [63](#)
- EDAM\_CONTENT\_CLASS\_SKETCH\_PREFIX
  - qevercloud, [63](#)
- EDAM\_CONTENT\_CLASS\_SKETCH
  - qevercloud, [63](#)
- EDAM\_DEVICE\_DESCRIPTION\_LEN\_MAX
  - qevercloud, [63](#)
- EDAM\_DEVICE\_DESCRIPTION\_REGEX
  - qevercloud, [63](#)
- EDAM\_DEVICE\_ID\_LEN\_MAX
  - qevercloud, [64](#)
- EDAM\_DEVICE\_ID\_REGEX
  - qevercloud, [64](#)
- EDAM\_EMAIL\_DOMAIN\_REGEX
  - qevercloud, [64](#)
- EDAM\_EMAIL\_LEN\_MAX
  - qevercloud, [64](#)
- EDAM\_EMAIL\_LEN\_MIN
  - qevercloud, [64](#)
- EDAM\_EMAIL\_LOCAL\_REGEX
  - qevercloud, [64](#)
- EDAM\_EMAIL\_REGEX
  - qevercloud, [64](#)
- EDAM\_FIND\_CONTACT\_DEFAULT\_MAX\_RESULTS
  - qevercloud, [64](#)
- EDAM\_FIND\_CONTACT\_MAX\_RESULTS
  - qevercloud, [65](#)
- EDAM\_FOOD\_APP\_CONTENT\_CLASS\_PREFIX
  - qevercloud, [65](#)
- EDAM\_GET\_ORDERS\_MAX\_RESULTS
  - qevercloud, [65](#)
- EDAM\_GUID\_LEN\_MAX
  - qevercloud, [65](#)
- EDAM\_GUID\_LEN\_MIN
  - qevercloud, [65](#)
- EDAM\_GUID\_REGEX
  - qevercloud, [65](#)
- EDAM\_HASH\_LEN
  - qevercloud, [65](#)
- EDAM\_HELLO\_APP\_CONTENT\_CLASS\_PREFIX
  - qevercloud, [66](#)
- EDAM\_INDEXABLE\_PLAINTEXT\_MIME\_TYPES
  - qevercloud, [66](#)
- EDAM\_INDEXABLE\_RESOURCE\_MIME\_TYPES
  - qevercloud, [66](#)
- EDAM\_MAX\_PREFERENCES
  - qevercloud, [66](#)
- EDAM\_MAX\_VALUES\_PER\_PREFERENCE
  - qevercloud, [66](#)
- EDAM\_MESSAGE\_ATTACHMENT\_SNIPPET\_LEN\_MAX
  - qevercloud, [66](#)
- EDAM\_MESSAGE\_ATTACHMENT\_SNIPPET\_REGEX
  - qevercloud, [66](#)
- EDAM\_MESSAGE\_ATTACHMENT\_TITLE\_LEN\_MAX
  - qevercloud, [67](#)
- EDAM\_MESSAGE\_ATTACHMENT\_TITLE\_REGEX
  - qevercloud, [67](#)
- EDAM\_MESSAGE\_ATTACHMENTS\_MAX
  - qevercloud, [67](#)
- EDAM\_MESSAGE\_BODY\_LEN\_MAX
  - qevercloud, [67](#)
- EDAM\_MESSAGE\_BODY\_REGEX
  - qevercloud, [67](#)
- EDAM\_MESSAGE\_RECIPIENTS\_MAX
  - qevercloud, [67](#)
- EDAM\_MIME\_LEN\_MAX
  - qevercloud, [67](#)
- EDAM\_MIME\_LEN\_MIN
  - qevercloud, [67](#)
- EDAM\_MIME\_REGEX
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_AAC
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_AMR
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_BMP
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_DEFAULT
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_GIF
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_INK
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_JPEG
  - qevercloud, [68](#)
- EDAM\_MIME\_TYPE\_M4A
  - qevercloud, [69](#)
- EDAM\_MIME\_TYPE\_MP3
  - qevercloud, [69](#)
- EDAM\_MIME\_TYPE\_MP4\_VIDEO
  - qevercloud, [69](#)
- EDAM\_MIME\_TYPE\_PDF
  - qevercloud, [69](#)
- EDAM\_MIME\_TYPE\_PNG
  - qevercloud, [69](#)
- EDAM\_MIME\_TYPE\_TIFF
  - qevercloud, [69](#)
- EDAM\_MIME\_TYPE\_WAV
  - qevercloud, [69](#)
- EDAM\_MIME\_TYPES
  - qevercloud, [69](#)
- EDAM\_NOTE\_BUSINESS\_SHARED\_NOTE\_MAX
  - qevercloud, [70](#)
- EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MAX
  - qevercloud, [70](#)
- EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MIN
  - qevercloud, [70](#)
- EDAM\_NOTE\_CONTENT\_CLASS\_REGEX
  - qevercloud, [70](#)
- EDAM\_NOTE\_CONTENT\_LEN\_MAX
  - qevercloud, [70](#)
- EDAM\_NOTE\_CONTENT\_LEN\_MIN
  - qevercloud, [70](#)

- EDAM\_NOTE\_LOCK\_VIEWERS\_NOTES\_MAX  
qevercloud, [70](#)
- EDAM\_NOTE\_PERSONAL\_SHARED\_NOTE\_MAX  
qevercloud, [71](#)
- EDAM\_NOTE\_RESOURCES\_MAX  
qevercloud, [71](#)
- EDAM\_NOTE\_SIZE\_MAX\_FREE  
qevercloud, [71](#)
- EDAM\_NOTE\_SIZE\_MAX\_PREMIUM  
qevercloud, [71](#)
- EDAM\_NOTE\_SOURCE\_MAIL\_CLIP  
qevercloud, [71](#)
- EDAM\_NOTE\_SOURCE\_MAIL\_SMTP\_GATEWAY  
qevercloud, [71](#)
- EDAM\_NOTE\_SOURCE\_WEB\_CLIP\_SIMPLIFIED  
qevercloud, [72](#)
- EDAM\_NOTE\_SOURCE\_WEB\_CLIP  
qevercloud, [71](#)
- EDAM\_NOTE\_TAGS\_MAX  
qevercloud, [72](#)
- EDAM\_NOTE\_TITLE\_LEN\_MAX  
qevercloud, [72](#)
- EDAM\_NOTE\_TITLE\_LEN\_MIN  
qevercloud, [72](#)
- EDAM\_NOTE\_TITLE\_QUALITY\_HIGH  
qevercloud, [72](#)
- EDAM\_NOTE\_TITLE\_QUALITY\_LOW  
qevercloud, [72](#)
- EDAM\_NOTE\_TITLE\_QUALITY\_MEDIUM  
qevercloud, [72](#)
- EDAM\_NOTE\_TITLE\_QUALITY\_UNTITLED  
qevercloud, [73](#)
- EDAM\_NOTE\_TITLE\_REGEX  
qevercloud, [73](#)
- EDAM\_NOTEBOOK\_BUSINESS\_SHARED\_NOTEBOOK\_MAX  
qevercloud, [73](#)
- EDAM\_NOTEBOOK\_NAME\_LEN\_MAX  
qevercloud, [73](#)
- EDAM\_NOTEBOOK\_NAME\_LEN\_MIN  
qevercloud, [73](#)
- EDAM\_NOTEBOOK\_NAME\_REGEX  
qevercloud, [73](#)
- EDAM\_NOTEBOOK\_PERSONAL\_SHARED\_NOTEBOOK\_MAX  
qevercloud, [73](#)
- EDAM\_NOTEBOOK\_STACK\_LEN\_MAX  
qevercloud, [74](#)
- EDAM\_NOTEBOOK\_STACK\_LEN\_MIN  
qevercloud, [74](#)
- EDAM\_NOTEBOOK\_STACK\_REGEX  
qevercloud, [74](#)
- EDAM\_OPEN\_ID\_ACCESS\_TOKEN\_MAX  
qevercloud, [74](#)
- EDAM\_PREFERENCE\_BUSINESS\_DEFAULT\_NOTEBOOK  
qevercloud, [74](#)
- EDAM\_PREFERENCE\_BUSINESS\_QUICKNOTE  
qevercloud, [74](#)
- EDAM\_PREFERENCE\_NAME\_LEN\_MAX  
qevercloud, [75](#)
- EDAM\_PREFERENCE\_NAME\_LEN\_MIN  
qevercloud, [75](#)
- EDAM\_PREFERENCE\_NAME\_REGEX  
qevercloud, [75](#)
- EDAM\_PREFERENCE\_ONLY\_ONE\_VALUE\_LEN\_MAX  
qevercloud, [75](#)
- EDAM\_PREFERENCE\_ONLY\_ONE\_VALUE\_REGEX  
qevercloud, [75](#)
- EDAM\_PREFERENCE\_SHORTCUTS\_MAX\_VALUES  
qevercloud, [76](#)
- EDAM\_PREFERENCE\_SHORTCUTS  
qevercloud, [75](#)
- EDAM\_PREFERENCE\_VALUE\_LEN\_MAX  
qevercloud, [76](#)
- EDAM\_PREFERENCE\_VALUE\_LEN\_MIN  
qevercloud, [76](#)
- EDAM\_PREFERENCE\_VALUE\_REGEX  
qevercloud, [76](#)
- EDAM\_PROMOTION\_ID\_LEN\_MAX  
qevercloud, [76](#)
- EDAM\_PROMOTION\_ID\_REGEX  
qevercloud, [76](#)
- EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MAX  
qevercloud, [76](#)
- EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MIN  
qevercloud, [76](#)
- EDAM\_PUBLISHING\_DESCRIPTION\_REGEX  
qevercloud, [77](#)
- EDAM\_PUBLISHING\_URI\_LEN\_MAX  
qevercloud, [77](#)
- EDAM\_PUBLISHING\_URI\_LEN\_MIN  
qevercloud, [77](#)
- EDAM\_PUBLISHING\_URI\_PROHIBITED  
qevercloud, [77](#)
- EDAM\_PUBLISHING\_URI\_REGEX  
qevercloud, [77](#)
- EDAM\_RELATED\_MAX\_EXPERTS  
qevercloud, [77](#)
- EDAM\_RELATED\_MAX\_NOTEBOOKS  
qevercloud, [77](#)
- EDAM\_RELATED\_MAX\_NOTES  
qevercloud, [78](#)
- EDAM\_RELATED\_MAX\_RELATED\_CONTENT  
qevercloud, [78](#)
- EDAM\_RELATED\_MAX\_TAGS  
qevercloud, [78](#)
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MAX  
qevercloud, [78](#)
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MIN  
qevercloud, [78](#)
- EDAM\_RESOURCE\_SIZE\_MAX\_FREE  
qevercloud, [78](#)
- EDAM\_RESOURCE\_SIZE\_MAX\_PREMIUM  
qevercloud, [78](#)



- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MAX
  - qevercloud, [78](#)
- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MIN
  - qevercloud, [79](#)
- EDAM\_SAVED\_SEARCH\_NAME\_REGEX
  - qevercloud, [79](#)
- EDAM\_SEARCH\_QUERY\_LEN\_MAX
  - qevercloud, [79](#)
- EDAM\_SEARCH\_QUERY\_LEN\_MIN
  - qevercloud, [79](#)
- EDAM\_SEARCH\_QUERY\_REGEX
  - qevercloud, [79](#)
- EDAM\_SEARCH\_SUGGESTIONS\_MAX
  - qevercloud, [79](#)
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MAX
  - qevercloud, [79](#)
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MIN
  - qevercloud, [80](#)
- EDAM\_SNIPPETS\_NOTES\_MAX
  - qevercloud, [80](#)
- EDAM\_SOURCE\_APPLICATION\_ANDROID\_SHARE\_EXTENSION
  - qevercloud, [80](#)
- EDAM\_SOURCE\_APPLICATION\_EN\_SCANSNAP
  - qevercloud, [80](#)
- EDAM\_SOURCE\_APPLICATION\_EWC
  - qevercloud, [80](#)
- EDAM\_SOURCE\_APPLICATION\_IOS\_SHARE\_EXTENSION
  - qevercloud, [80](#)
- EDAM\_SOURCE\_APPLICATION\_MOLESKINE
  - qevercloud, [80](#)
- EDAM\_SOURCE\_APPLICATION\_POSTIT
  - qevercloud, [80](#)
- EDAM\_SOURCE\_APPLICATION\_WEB\_CLIPPER
  - qevercloud, [81](#)
- EDAM\_SOURCE\_OUTLOOK\_CLIPPER
  - qevercloud, [81](#)
- EDAM\_TAG\_NAME\_LEN\_MAX
  - qevercloud, [81](#)
- EDAM\_TAG\_NAME\_LEN\_MIN
  - qevercloud, [81](#)
- EDAM\_TAG\_NAME\_REGEX
  - qevercloud, [81](#)
- EDAM\_TIMEZONE\_LEN\_MAX
  - qevercloud, [81](#)
- EDAM\_TIMEZONE\_LEN\_MIN
  - qevercloud, [81](#)
- EDAM\_TIMEZONE\_REGEX
  - qevercloud, [82](#)
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX\_PREMIUM
  - qevercloud, [82](#)
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX
  - qevercloud, [82](#)
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_FREE
  - qevercloud, [82](#)
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_PREMIUM
  - qevercloud, [82](#)
- EDAM\_USER\_NAME\_LEN\_MAX
  - qevercloud, [82](#)
- EDAM\_USER\_NAME\_LEN\_MIN
  - qevercloud, [83](#)
- EDAM\_USER\_NAME\_REGEX
  - qevercloud, [83](#)
- EDAM\_USER\_NOTEBOOKS\_MAX
  - qevercloud, [83](#)
- EDAM\_USER\_NOTES\_MAX
  - qevercloud, [83](#)
- EDAM\_USER\_PASSWORD\_LEN\_MAX
  - qevercloud, [83](#)
- EDAM\_USER\_PASSWORD\_LEN\_MIN
  - qevercloud, [83](#)
- EDAM\_USER\_PASSWORD\_REGEX
  - qevercloud, [83](#)
- EDAM\_USER\_PROFILE\_PHOTO\_MAX\_BYTES
  - qevercloud, [83](#)
- EDAM\_USER\_RECENT\_MAILED\_ADDRESSES\_MAX
  - qevercloud, [84](#)
- EDAM\_USER\_SAVED\_SEARCHES\_MAX
  - qevercloud, [84](#)
- EDAM\_USER\_TAGS\_MAX
  - qevercloud, [84](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_FIRST\_MONTH
  - qevercloud, [84](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_NEXT\_MONTH
  - qevercloud, [84](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_PER\_USER
  - qevercloud, [84](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS
  - qevercloud, [84](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_FREE
  - qevercloud, [85](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_PLUS
  - qevercloud, [85](#)
- EDAM\_USER\_UPLOAD\_LIMIT\_PREMIUM
  - qevercloud, [85](#)
- EDAM\_USER\_UPLOAD\_SURVEY\_THRESHOLD
  - qevercloud, [85](#)
- EDAM\_USER\_USERNAME\_LEN\_MAX
  - qevercloud, [85](#)
- EDAM\_USER\_USERNAME\_LEN\_MIN
  - qevercloud, [85](#)
- EDAM\_USER\_USERNAME\_REGEX
  - qevercloud, [85](#)
- EDAM\_USER\_WORKSPACES\_MAX
  - qevercloud, [86](#)
- EDAM\_VAT\_REGEX
  - qevercloud, [86](#)
- EDAM\_VERSION\_MAJOR
  - qevercloud, [86](#)
- EDAM\_VERSION\_MINOR
  - qevercloud, [86](#)

- EDAM\_WORKSPACE\_DESCRIPTION\_LEN\_MAX
  - qevercloud, [86](#)
- EDAM\_WORKSPACE\_NAME\_LEN\_MAX
  - qevercloud, [86](#)
- EDAM\_WORKSPACE\_NAME\_LEN\_MIN
  - qevercloud, [86](#)
- EDAM\_WORKSPACE\_NAME\_REGEX
  - qevercloud, [87](#)
- EDAMErrorCode
  - qevercloud, [34](#)
- EDAMErrorCode.h, [520](#)
- EDAMInvalidContactReason
  - qevercloud, [36](#)
- EDAMInvalidContactsException
  - qevercloud::EDAMInvalidContactsException, [129](#)
- EDAMInvalidContactsExceptionData
  - qevercloud::EDAMInvalidContactsExceptionData, [132](#)
- EDAMNotFoundException
  - qevercloud::EDAMNotFoundException, [134](#)
- EDAMNotFoundExceptionData
  - qevercloud::EDAMNotFoundExceptionData, [136](#)
- EDAMSystemException
  - qevercloud::EDAMSystemException, [138](#)
- EDAMSystemExceptionAuthExpiredData
  - qevercloud::EDAMSystemExceptionAuthExpiredData, [141](#)
- EDAMSystemExceptionData
  - qevercloud::EDAMSystemExceptionData, [143](#)
- EDAMSystemExceptionRateLimitReachedData
  - qevercloud::EDAMSystemExceptionRateLimitReachedData, [145](#)
- EDAMUserException
  - qevercloud::EDAMUserException, [147](#)
- EDAMUserExceptionData
  - qevercloud::EDAMUserExceptionData, [149](#)
- educationalDiscount
  - qevercloud::UserAttributes, [492](#)
- email
  - qevercloud::BusinessInvitation, [112](#)
  - qevercloud::BusinessUserInfo, [119](#)
  - qevercloud::SharedNotebook, [448](#)
  - qevercloud::User, [487](#)
  - qevercloud::UserProfile, [500](#)
- emailAddressLastConfirmed
  - qevercloud::UserAttributes, [492](#)
- emailNote
  - qevercloud::INoteStore, [192](#)
- emailNoteAsync
  - qevercloud::INoteStore, [193](#)
- emailNoteRequest
  - qevercloud::NoteStoreServer, [363](#)
- emailNoteRequestReady
  - qevercloud::NoteStoreServer, [363](#)
- emailOptOutDate
  - qevercloud::UserAttributes, [492](#)
- emphasized
  - qevercloud::NoteFilter, [324](#)
- enableFacebookSharing
  - qevercloud::BootstrapSettings, [109](#)
- enableGiftSubscriptions
  - qevercloud::BootstrapSettings, [109](#)
- enableGoogle
  - qevercloud::BootstrapSettings, [109](#)
- enableLinkedInSharing
  - qevercloud::BootstrapSettings, [109](#)
- enablePublicNotebooks
  - qevercloud::BootstrapSettings, [110](#)
- enableSharedNotebooks
  - qevercloud::BootstrapSettings, [110](#)
- enableSingleNoteSharing
  - qevercloud::BootstrapSettings, [110](#)
- enableSponsoredAccounts
  - qevercloud::BootstrapSettings, [110](#)
- enableSupportTickets
  - qevercloud::BootstrapSettings, [110](#)
- enableTwitterSharing
  - qevercloud::BootstrapSettings, [110](#)
- end
  - qevercloud::QAssociativeContainerConstReferenceWrapper, [417](#)
  - qevercloud::QAssociativeContainerReferenceWrapper, [418](#)
- EntityType
  - qevercloud, [36](#)
- Error
  - qevercloud, [37](#)
- errorCode
  - qevercloud::EDAMSystemException, [139](#)
  - qevercloud::EDAMUserException, [148](#)
- errorMessage
  - qevercloud::EverCloudExceptionData, [155](#)
- errors
  - qevercloud::ManageNoteSharesResult, [281](#)
  - qevercloud::ManageNotebookSharesResult, [274](#)
- eventId
  - qevercloud::Identity, [170](#)
- EventLoopFinisher
  - qevercloud::EventLoopFinisher, [151](#)
- EventLoopFinisher.h, [524](#)
- EverCloudException
  - qevercloud::EverCloudException, [152](#)
- EverCloudException.h, [525](#)
- EverCloudExceptionData
  - qevercloud, [87](#)
  - qevercloud::EverCloudExceptionData, [155](#)
- EverCloudExceptionDataPtr
  - qevercloud, [30](#)
- EverCloudLocalData
  - qevercloud::EverCloudLocalData, [157](#)
- EvernoteException
  - qevercloud::EvernoteException, [160](#)
- EvernoteExceptionData
  - qevercloud::EvernoteExceptionData, [161](#)
- evernoteNetworkProxy
  - qevercloud, [43](#)



- EvernoteOAuthDialog
  - qevercloud::EvernoteOAuthDialog, [163](#)
- EvernoteOAuthWebView
  - qevercloud::EvernoteOAuthWebView, [166](#)
- exceptionData
  - qevercloud::EDAMInvalidContactsException, [129](#)
  - qevercloud::EDAMNotFoundException, [134](#)
  - qevercloud::EDAMSystemException, [138](#)
  - qevercloud::EDAMSystemExceptionAuthExpired, [141](#)
  - qevercloud::EDAMSystemExceptionRateLimit←Reached, [144](#)
  - qevercloud::EDAMUserException, [147](#)
  - qevercloud::EverCloudException, [153](#)
  - qevercloud::EvernoteException, [160](#)
  - qevercloud::NetworkException, [285](#)
  - qevercloud::ThriftException, [475](#)
- Exceptions.h, [525](#)
- exec
  - qevercloud::EvernoteOAuthDialog, [164](#)
- executeAsyncRequest
  - qevercloud::IDurableService, [172](#)
- executeSyncRequest
  - qevercloud::IDurableService, [172](#)
- experts
  - qevercloud::RelatedResult, [429](#)
- expiration
  - qevercloud::AuthenticationResult, [103](#)
- expires
  - qevercloud::EvernoteOAuthWebView::OAuth←Result, [400](#)
- Export.h, [526](#)
  - QEVERCLOUD\_EXPORT, [526](#)
- expungeLinkedNotebook
  - qevercloud::INoteStore, [193](#)
- expungeLinkedNotebookAsync
  - qevercloud::INoteStore, [193](#)
- expungeLinkedNotebookRequest
  - qevercloud::NoteStoreServer, [363](#)
- expungeLinkedNotebookRequestReady
  - qevercloud::NoteStoreServer, [363](#)
- expungeNote
  - qevercloud::INoteStore, [194](#)
- expungeNoteAsync
  - qevercloud::INoteStore, [194](#)
- expungeNoteRequest
  - qevercloud::NoteStoreServer, [364](#)
- expungeNoteRequestReady
  - qevercloud::NoteStoreServer, [364](#)
- expungeNotebook
  - qevercloud::INoteStore, [194](#)
- expungeNotebookAsync
  - qevercloud::INoteStore, [195](#)
- expungeNotebookRequest
  - qevercloud::NoteStoreServer, [363](#)
- expungeNotebookRequestReady
  - qevercloud::NoteStoreServer, [364](#)
- expungeSearch
  - qevercloud::INoteStore, [195](#)
- expungeSearchAsync
  - qevercloud::INoteStore, [196](#)
- expungeSearchRequest
  - qevercloud::NoteStoreServer, [364](#)
- expungeSearchRequestReady
  - qevercloud::NoteStoreServer, [364](#)
- expungeTag
  - qevercloud::INoteStore, [196](#)
- expungeTagAsync
  - qevercloud::INoteStore, [197](#)
- expungeTagRequest
  - qevercloud::NoteStoreServer, [364](#)
- expungeTagRequestReady
  - qevercloud::NoteStoreServer, [365](#)
- expungeWhichSharedNotebookRestrictions
  - qevercloud::NotebookRestrictions, [310](#)
- expungedLinkedNotebooks
  - qevercloud::SyncChunk, [461](#)
- expungedNotebooks
  - qevercloud::SyncChunk, [461](#)
- expungedNotes
  - qevercloud::SyncChunk, [461](#)
- expungedSearches
  - qevercloud::SyncChunk, [461](#)
- expungedTags
  - qevercloud::SyncChunk, [462](#)
- favorited
  - qevercloud::EverCloudLocalData, [158](#)
- fileName
  - qevercloud::ResourceAttributes, [439](#)
- fileSize
  - qevercloud::RelatedContentImage, [423](#)
- filter
  - qevercloud::RelatedQuery, [426](#)
- findNoteCounts
  - qevercloud::INoteStore, [197](#)
- findNoteCountsAsync
  - qevercloud::INoteStore, [198](#)
- findNoteCountsRequest
  - qevercloud::NoteStoreServer, [365](#)
- findNoteCountsRequestReady
  - qevercloud::NoteStoreServer, [365](#)
- findNoteOffset
  - qevercloud::INoteStore, [198](#)
- findNoteOffsetAsync
  - qevercloud::INoteStore, [199](#)
- findNoteOffsetRequest
  - qevercloud::NoteStoreServer, [365](#)
- findNoteOffsetRequestReady
  - qevercloud::NoteStoreServer, [365](#)
- findNotesMetadata
  - qevercloud::INoteStore, [199](#)
- findNotesMetadataAsync
  - qevercloud::INoteStore, [200](#)
- findNotesMetadataRequest
  - qevercloud::NoteStoreServer, [365](#)
- findNotesMetadataRequestReady

- qevercloud::NoteStoreServer, 366
- findRelated
  - qevercloud::INoteStore, 200
- findRelatedAsync
  - qevercloud::INoteStore, 201
- findRelatedRequest
  - qevercloud::NoteStoreServer, 366
- findRelatedRequestReady
  - qevercloud::NoteStoreServer, 366
- finished
  - qevercloud::AsyncResult, 100
- format
  - qevercloud::SavedSearch, 442
- fromWorkChat
  - qevercloud::BusinessInvitation, 113
- FullMap
  - qevercloud::LazyMap, 264
- fullMap
  - qevercloud::LazyMap, 264, 265
- fullSyncBefore
  - qevercloud::SyncState, 470
- getAccountLimits
  - qevercloud::IUserStore, 255
- getAccountLimitsAsync
  - qevercloud::IUserStore, 256
- getAccountLimitsRequest
  - qevercloud::UserStoreServer, 505
- getAccountLimitsRequestReady
  - qevercloud::UserStoreServer, 505
- getBootstrapInfo
  - qevercloud::IUserStore, 256
- getBootstrapInfoAsync
  - qevercloud::IUserStore, 256
- getBootstrapInfoRequest
  - qevercloud::UserStoreServer, 505
- getBootstrapInfoRequestReady
  - qevercloud::UserStoreServer, 505
- getDefaultNotebook
  - qevercloud::INoteStore, 201
- getDefaultNotebookAsync
  - qevercloud::INoteStore, 202
- getDefaultNotebookRequest
  - qevercloud::NoteStoreServer, 366
- getDefaultNotebookRequestReady
  - qevercloud::NoteStoreServer, 366
- getFilteredSyncChunk
  - qevercloud::INoteStore, 202
- getFilteredSyncChunkAsync
  - qevercloud::INoteStore, 202
- getFilteredSyncChunkRequest
  - qevercloud::NoteStoreServer, 366
- getFilteredSyncChunkRequestReady
  - qevercloud::NoteStoreServer, 367
- getLinkedNotebookSyncChunk
  - qevercloud::INoteStore, 203
- getLinkedNotebookSyncChunkAsync
  - qevercloud::INoteStore, 204
- getLinkedNotebookSyncChunkRequest
  - qevercloud::NoteStoreServer, 367
- getLinkedNotebookSyncChunkRequestReady
  - qevercloud::NoteStoreServer, 367
- getLinkedNotebookSyncState
  - qevercloud::INoteStore, 204
- getLinkedNotebookSyncStateAsync
  - qevercloud::INoteStore, 205
- getLinkedNotebookSyncStateRequest
  - qevercloud::NoteStoreServer, 367
- getLinkedNotebookSyncStateRequestReady
  - qevercloud::NoteStoreServer, 367
- getNote
  - qevercloud::INoteStore, 205
- getNoteApplicationData
  - qevercloud::INoteStore, 205
- getNoteApplicationDataAsync
  - qevercloud::INoteStore, 205
- getNoteApplicationDataEntry
  - qevercloud::INoteStore, 206
- getNoteApplicationDataEntryAsync
  - qevercloud::INoteStore, 206
- getNoteApplicationDataEntryRequest
  - qevercloud::NoteStoreServer, 367
- getNoteApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, 368
- getNoteApplicationDataRequest
  - qevercloud::NoteStoreServer, 368
- getNoteApplicationDataRequestReady
  - qevercloud::NoteStoreServer, 368
- getNoteAsync
  - qevercloud::INoteStore, 206
- getNoteContent
  - qevercloud::INoteStore, 208
- getNoteContentAsync
  - qevercloud::INoteStore, 209
- getNoteContentRequest
  - qevercloud::NoteStoreServer, 369
- getNoteContentRequestReady
  - qevercloud::NoteStoreServer, 369
- getNoteRequest
  - qevercloud::NoteStoreServer, 369
- getNoteRequestReady
  - qevercloud::NoteStoreServer, 369
- getNoteSearchText
  - qevercloud::INoteStore, 209
- getNoteSearchTextAsync
  - qevercloud::INoteStore, 210
- getNoteSearchTextRequest
  - qevercloud::NoteStoreServer, 369
- getNoteSearchTextRequestReady
  - qevercloud::NoteStoreServer, 370
- getNoteTagNames
  - qevercloud::INoteStore, 210
- getNoteTagNamesAsync
  - qevercloud::INoteStore, 210
- getNoteTagNamesRequest
  - qevercloud::NoteStoreServer, 370
- getNoteTagNamesRequestReady
  - qevercloud::NoteStoreServer, 370

- qevercloud::NoteStoreServer, 370
- getNoteVersion
  - qevercloud::INoteStore, 211
- getNoteVersionAsync
  - qevercloud::INoteStore, 211
- getNoteVersionRequest
  - qevercloud::NoteStoreServer, 370
- getNoteVersionRequestReady
  - qevercloud::NoteStoreServer, 370
- getNoteWithResultSpec
  - qevercloud::INoteStore, 212
- getNoteWithResultSpecAsync
  - qevercloud::INoteStore, 212
- getNoteWithResultSpecRequest
  - qevercloud::NoteStoreServer, 370
- getNoteWithResultSpecRequestReady
  - qevercloud::NoteStoreServer, 371
- getNotebook
  - qevercloud::INoteStore, 206
- getNotebookAsync
  - qevercloud::INoteStore, 208
- getNotebookRequest
  - qevercloud::NoteStoreServer, 368
- getNotebookRequestReady
  - qevercloud::NoteStoreServer, 368
- getNotebookShares
  - qevercloud::INoteStore, 208
- getNotebookSharesAsync
  - qevercloud::INoteStore, 208
- getNotebookSharesRequest
  - qevercloud::NoteStoreServer, 368
- getNotebookSharesRequestReady
  - qevercloud::NoteStoreServer, 369
- getPublicNotebook
  - qevercloud::INoteStore, 212
- getPublicNotebookAsync
  - qevercloud::INoteStore, 213
- getPublicNotebookRequest
  - qevercloud::NoteStoreServer, 371
- getPublicNotebookRequestReady
  - qevercloud::NoteStoreServer, 371
- getPublicUserInfo
  - qevercloud::IUserStore, 256
- getPublicUserInfoAsync
  - qevercloud::IUserStore, 257
- getPublicUserInfoRequest
  - qevercloud::UserStoreServer, 505
- getPublicUserInfoRequestReady
  - qevercloud::UserStoreServer, 505
- getResource
  - qevercloud::INoteStore, 213
- getResourceAlternateData
  - qevercloud::INoteStore, 214
- getResourceAlternateDataAsync
  - qevercloud::INoteStore, 214
- getResourceAlternateDataRequest
  - qevercloud::NoteStoreServer, 371
- getResourceAlternateDataRequestReady
  - qevercloud::NoteStoreServer, 371
- getResourceApplicationData
  - qevercloud::INoteStore, 215
- getResourceApplicationDataAsync
  - qevercloud::INoteStore, 215
- getResourceApplicationDataEntry
  - qevercloud::INoteStore, 215
- getResourceApplicationDataEntryAsync
  - qevercloud::INoteStore, 215
- getResourceApplicationDataEntryRequest
  - qevercloud::NoteStoreServer, 371
- getResourceApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, 372
- getResourceApplicationDataRequest
  - qevercloud::NoteStoreServer, 372
- getResourceApplicationDataRequestReady
  - qevercloud::NoteStoreServer, 372
- getResourceAsync
  - qevercloud::INoteStore, 216
- getResourceAttributes
  - qevercloud::INoteStore, 216
- getResourceAttributesAsync
  - qevercloud::INoteStore, 216
- getResourceAttributesRequest
  - qevercloud::NoteStoreServer, 372
- getResourceAttributesRequestReady
  - qevercloud::NoteStoreServer, 372
- getResourceByHash
  - qevercloud::INoteStore, 216
- getResourceByHashAsync
  - qevercloud::INoteStore, 217
- getResourceByHashRequest
  - qevercloud::NoteStoreServer, 372
- getResourceByHashRequestReady
  - qevercloud::NoteStoreServer, 373
- getResourceData
  - qevercloud::INoteStore, 217
- getResourceDataAsync
  - qevercloud::INoteStore, 218
- getResourceDataRequest
  - qevercloud::NoteStoreServer, 373
- getResourceDataRequestReady
  - qevercloud::NoteStoreServer, 373
- getResourceRecognition
  - qevercloud::INoteStore, 218
- getResourceRecognitionAsync
  - qevercloud::INoteStore, 219
- getResourceRecognitionRequest
  - qevercloud::NoteStoreServer, 373
- getResourceRecognitionRequestReady
  - qevercloud::NoteStoreServer, 373
- getResourceRequest
  - qevercloud::NoteStoreServer, 373
- getResourceRequestReady
  - qevercloud::NoteStoreServer, 374
- getResourceSearchText
  - qevercloud::INoteStore, 219
- getResourceSearchTextAsync

- qevercloud::INoteStore, 220
- getResourceSearchTextRequest
  - qevercloud::NoteStoreServer, 374
- getResourceSearchTextRequestReady
  - qevercloud::NoteStoreServer, 374
- getSearch
  - qevercloud::INoteStore, 220
- getSearchAsync
  - qevercloud::INoteStore, 220
- getSearchRequest
  - qevercloud::NoteStoreServer, 374
- getSearchRequestReady
  - qevercloud::NoteStoreServer, 374
- getSharedNotebookByAuth
  - qevercloud::INoteStore, 220
- getSharedNotebookByAuthAsync
  - qevercloud::INoteStore, 221
- getSharedNotebookByAuthRequest
  - qevercloud::NoteStoreServer, 374
- getSharedNotebookByAuthRequestReady
  - qevercloud::NoteStoreServer, 375
- getSyncState
  - qevercloud::INoteStore, 221
- getSyncStateAsync
  - qevercloud::INoteStore, 221
- getSyncStateRequest
  - qevercloud::NoteStoreServer, 375
- getSyncStateRequestReady
  - qevercloud::NoteStoreServer, 375
- getTag
  - qevercloud::INoteStore, 221
- getTagAsync
  - qevercloud::INoteStore, 222
- getTagRequest
  - qevercloud::NoteStoreServer, 375
- getTagRequestReady
  - qevercloud::NoteStoreServer, 375
- getUser
  - qevercloud::IUserStore, 257
- getUserAsync
  - qevercloud::IUserStore, 257
- getUserRequest
  - qevercloud::UserStoreServer, 506
- getUserRequestReady
  - qevercloud::UserStoreServer, 506
- getUserUrls
  - qevercloud::IUserStore, 257
- getUserUrlsAsync
  - qevercloud::IUserStore, 257
- getUserUrlsRequest
  - qevercloud::UserStoreServer, 506
- getUserUrlsRequestReady
  - qevercloud::UserStoreServer, 506
- globalId
  - qevercloud::SharedNotebook, 448
- Globals.h, 526
- groupName
  - qevercloud::UserAttributes, 493
- Guid
  - qevercloud, 30
- guid
  - qevercloud::LinkedNotebook, 267
  - qevercloud::Note, 290
  - qevercloud::NoteEmailParameters, 321
  - qevercloud::NoteMetadata, 337
  - qevercloud::Notebook, 301
  - qevercloud::NotebookDescriptor, 305
  - qevercloud::Resource, 436
  - qevercloud::SavedSearch, 442
  - qevercloud::Tag, 472
- hasSharedNotebook
  - qevercloud::NotebookDescriptor, 305
- height
  - qevercloud::RelatedContentImage, 424
  - qevercloud::Resource, 436
- Helpers.h, 527
- hideSponsorBilling
  - qevercloud::UserAttributes, 493
- IDurableServicePtr
  - qevercloud, 30
- ILoggerPtr
  - qevercloud, 30
- INoteStore
  - qevercloud::INoteStore, 181
- INoteStorePtr
  - qevercloud, 30
- IRequestContextPtr
  - qevercloud, 31
- IRetryPolicyPtr
  - qevercloud, 31
- IUserStore
  - qevercloud::IUserStore, 250
- IUserStorePtr
  - qevercloud, 31
- id
  - qevercloud::Contact, 123
  - qevercloud::EverCloudLocalData, 158
  - qevercloud::Identity, 170
  - qevercloud::SharedNotebook, 448
  - qevercloud::User, 487
  - qevercloud::UserProfile, 500
- identifier
  - qevercloud::EDAMNotFoundException, 135
- IdentityID
  - qevercloud, 30
- identityID
  - qevercloud::ManageNoteSharesError, 276
- ImageType
  - qevercloud::Thumbnail, 479
- inMyList
  - qevercloud::NotebookRecipientSettings, 307
- inactive
  - qevercloud::NoteFilter, 324
- includeAccount
  - qevercloud::SavedSearchScope, 444

- includeAccountLimits
  - qevercloud::NoteResultSpec, [342](#)
- includeAllReadableNotebooks
  - qevercloud::NoteFilter, [324](#)
- includeAllReadableWorkspaces
  - qevercloud::NoteFilter, [324](#)
- includeAttributes
  - qevercloud::NotesMetadataResultSpec, [352](#)
- includeBusinessLinkedNotebooks
  - qevercloud::SavedSearchScope, [444](#)
- includeContainingNotebooks
  - qevercloud::RelatedResultSpec, [432](#)
- includeContent
  - qevercloud::NoteResultSpec, [342](#)
- includeContentLength
  - qevercloud::NotesMetadataResultSpec, [352](#)
- includeCreated
  - qevercloud::NotesMetadataResultSpec, [352](#)
- includeDebugInfo
  - qevercloud::RelatedResultSpec, [432](#)
- includeDeleted
  - qevercloud::NotesMetadataResultSpec, [352](#)
- includeExpunged
  - qevercloud::SyncChunkFilter, [465](#)
- includeLargestResourceMime
  - qevercloud::NotesMetadataResultSpec, [353](#)
- includeLargestResourceSize
  - qevercloud::NotesMetadataResultSpec, [353](#)
- includeLinkedNotebooks
  - qevercloud::SyncChunkFilter, [465](#)
- includeNoteAppDataValues
  - qevercloud::NoteResultSpec, [343](#)
- includeNoteApplicationDataFullMap
  - qevercloud::SyncChunkFilter, [465](#)
- includeNoteAttributes
  - qevercloud::SyncChunkFilter, [465](#)
- includeNoteResourceApplicationDataFullMap
  - qevercloud::SyncChunkFilter, [465](#)
- includeNoteResources
  - qevercloud::SyncChunkFilter, [466](#)
- includeNotebookGuid
  - qevercloud::NotesMetadataResultSpec, [353](#)
- includeNotebooks
  - qevercloud::SyncChunkFilter, [465](#)
- includeNotes
  - qevercloud::SyncChunkFilter, [466](#)
- includePersonalLinkedNotebooks
  - qevercloud::SavedSearchScope, [444](#)
- includeResourceAppDataValues
  - qevercloud::NoteResultSpec, [343](#)
- includeResourceApplicationDataFullMap
  - qevercloud::SyncChunkFilter, [466](#)
- includeResources
  - qevercloud::SyncChunkFilter, [466](#)
- includeResourcesAlternateData
  - qevercloud::NoteResultSpec, [343](#)
- includeResourcesData
  - qevercloud::NoteResultSpec, [343](#)
- includeResourcesRecognition
  - qevercloud::NoteResultSpec, [343](#)
- includeSearches
  - qevercloud::SyncChunkFilter, [466](#)
- includeSharedNotes
  - qevercloud::NoteResultSpec, [343](#)
  - qevercloud::SyncChunkFilter, [466](#)
- includeTagGuids
  - qevercloud::NotesMetadataResultSpec, [353](#)
- includeTags
  - qevercloud::SyncChunkFilter, [466](#)
- includeTitle
  - qevercloud::NotesMetadataResultSpec, [353](#)
- includeUpdateSequenceNum
  - qevercloud::NotesMetadataResultSpec, [353](#)
- includeUpdated
  - qevercloud::NotesMetadataResultSpec, [353](#)
- incomingEmailAddress
  - qevercloud::UserAttributes, [493](#)
- increaseRequestTimeoutExponentially
  - qevercloud::IRequestContext, [244](#)
- individualPrivilege
  - qevercloud::MemberShareRelationship, [283](#)
- Info
  - qevercloud, [37](#)
- init
  - qevercloud::Optional, [404](#)
- initializeQEverCloud
  - qevercloud, [44](#)
- InkNoteImageDownloader
  - qevercloud::InkNoteImageDownloader, [174](#)
- InkNoteImageDownloader.h, [527](#)
- InvalidationSequenceNumber
  - qevercloud, [30](#)
- invitationRestrictions
  - qevercloud::NoteShareRelationships, [347](#)
  - qevercloud::ShareRelationships, [458](#)
- invitations
  - qevercloud::NoteShareRelationships, [347](#)
  - qevercloud::ShareRelationships, [458](#)
- invitationsToCreateOrUpdate
  - qevercloud::ManageNotebookSharesParameters, [272](#)
- invitationsToUnshare
  - qevercloud::ManageNoteSharesParameters, [278](#)
- invitationsToUpdate
  - qevercloud::ManageNoteSharesParameters, [278](#)
- inviteMessage
  - qevercloud::ManageNotebookSharesParameters, [272](#)
- inviteToBusiness
  - qevercloud::IUserStore, [257](#)
- inviteToBusinessAsync
  - qevercloud::IUserStore, [258](#)
- inviteToBusinessRequest
  - qevercloud::UserStoreServer, [506](#)
- inviteToBusinessRequestReady
  - qevercloud::UserStoreServer, [506](#)



- isEqual
  - qevercloud::Optional, 404
- isSet
  - qevercloud::Optional, 404
- isSucceeded
  - qevercloud::EvernoteOAuthDialog, 164
  - qevercloud::EvernoteOAuthWebView, 167
- iterator
  - qevercloud::QAssociativeContainerConstReference↔Wrapper::iterator, 248
  - qevercloud::QAssociativeContainerReference↔Wrapper::iterator, 246
- joined
  - qevercloud::UserProfile, 500
- joinedUserCount
  - qevercloud::NotebookDescriptor, 305
- key
  - qevercloud::EDAMNotFoundException, 135
- keysOnly
  - qevercloud::LazyMap, 264
- largestResourceMime
  - qevercloud::NoteMetadata, 337
- largestResourceSize
  - qevercloud::NoteMetadata, 337
- lastEditedBy
  - qevercloud::NoteAttributes, 296
- lastEditorId
  - qevercloud::NoteAttributes, 296
  - qevercloud::NoteVersionId, 397
- lastFailedCharge
  - qevercloud::Accounting, 91
- lastFailedChargeReason
  - qevercloud::Accounting, 91
- lastRequestedCharge
  - qevercloud::Accounting, 91
- lastSuccessfulCharge
  - qevercloud::Accounting, 91
- latitude
  - qevercloud::NoteAttributes, 296
  - qevercloud::ResourceAttributes, 439
- level
  - qevercloud::ILogger, 172
- libraryVersion
  - qevercloud, 44
- limits
  - qevercloud::Note, 290
- linkedInProfileUrl
  - qevercloud::BusinessUserAttributes, 117
- linkedNotebooks
  - qevercloud::SyncChunk, 462
- listAccessibleBusinessNotebooks
  - qevercloud::INoteStore, 222
- listAccessibleBusinessNotebooksAsync
  - qevercloud::INoteStore, 222
- listAccessibleBusinessNotebooksRequest
  - qevercloud::NoteStoreServer, 375
- listAccessibleBusinessNotebooksRequestReady
  - qevercloud::NoteStoreServer, 375
- listBusinessInvitations
  - qevercloud::IUserStore, 258
- listBusinessInvitationsAsync
  - qevercloud::IUserStore, 259
- listBusinessInvitationsRequest
  - qevercloud::UserStoreServer, 506
- listBusinessInvitationsRequestReady
  - qevercloud::UserStoreServer, 507
- listBusinessUsers
  - qevercloud::IUserStore, 259
- listBusinessUsersAsync
  - qevercloud::IUserStore, 260
- listBusinessUsersRequest
  - qevercloud::UserStoreServer, 507
- listBusinessUsersRequestReady
  - qevercloud::UserStoreServer, 507
- listLinkedNotebooks
  - qevercloud::INoteStore, 223
- listLinkedNotebooksAsync
  - qevercloud::INoteStore, 223
- listLinkedNotebooksRequest
  - qevercloud::NoteStoreServer, 376
- listLinkedNotebooksRequestReady
  - qevercloud::NoteStoreServer, 376
- listNoteVersions
  - qevercloud::INoteStore, 223
- listNoteVersionsAsync
  - qevercloud::INoteStore, 224
- listNoteVersionsRequest
  - qevercloud::NoteStoreServer, 376
- listNoteVersionsRequestReady
  - qevercloud::NoteStoreServer, 376
- listNotebooks
  - qevercloud::INoteStore, 223
- listNotebooksAsync
  - qevercloud::INoteStore, 223
- listNotebooksRequest
  - qevercloud::NoteStoreServer, 376
- listNotebooksRequestReady
  - qevercloud::NoteStoreServer, 376
- listSearches
  - qevercloud::INoteStore, 224
- listSearchesAsync
  - qevercloud::INoteStore, 224
- listSearchesRequest
  - qevercloud::NoteStoreServer, 376
- listSearchesRequestReady
  - qevercloud::NoteStoreServer, 377
- listSharedNotebooks
  - qevercloud::INoteStore, 224
- listSharedNotebooksAsync
  - qevercloud::INoteStore, 224
- listSharedNotebooksRequest
  - qevercloud::NoteStoreServer, 377
- listSharedNotebooksRequestReady
  - qevercloud::NoteStoreServer, 377

- listTags
  - qevercloud::INoteStore, 225
- listTagsAsync
  - qevercloud::INoteStore, 225
- listTagsByNotebook
  - qevercloud::INoteStore, 225
- listTagsByNotebookAsync
  - qevercloud::INoteStore, 225
- listTagsByNotebookRequest
  - qevercloud::NoteStoreServer, 377
- listTagsByNotebookRequestReady
  - qevercloud::NoteStoreServer, 377
- listTagsRequest
  - qevercloud::NoteStoreServer, 377
- listTagsRequestReady
  - qevercloud::NoteStoreServer, 377
- local
  - qevercloud::EverCloudLocalData, 159
- localData
  - qevercloud::AccountLimits, 95
  - qevercloud::Accounting, 92
  - qevercloud::AuthenticationResult, 103
  - qevercloud::BootstrapInfo, 105
  - qevercloud::BootstrapProfile, 107
  - qevercloud::BootstrapSettings, 110
  - qevercloud::BusinessInvitation, 113
  - qevercloud::BusinessNotebook, 115
  - qevercloud::BusinessUserAttributes, 117
  - qevercloud::BusinessUserInfo, 119
  - qevercloud::CanMoveToContainerRestrictions, 121
  - qevercloud::Contact, 123
  - qevercloud::CreateOrUpdateNotebookShares←  
Result, 125
  - qevercloud::Data, 127
  - qevercloud::Identity, 170
  - qevercloud::InvitationShareRelationship, 243
  - qevercloud::LazyMap, 265
  - qevercloud::LinkedNotebook, 267
  - qevercloud::ManageNoteSharesError, 276
  - qevercloud::ManageNoteSharesParameters, 279
  - qevercloud::ManageNoteSharesResult, 281
  - qevercloud::ManageNotebookSharesError, 270
  - qevercloud::ManageNotebookSharesParameters,  
272
  - qevercloud::ManageNotebookSharesResult, 274
  - qevercloud::MemberShareRelationship, 283
  - qevercloud::Note, 291
  - qevercloud::NoteAttributes, 296
  - qevercloud::NoteCollectionCounts, 319
  - qevercloud::NoteEmailParameters, 321
  - qevercloud::NoteFilter, 325
  - qevercloud::NoteInvitationShareRelationship, 327
  - qevercloud::NoteLimits, 329
  - qevercloud::NoteList, 332
  - qevercloud::NoteMemberShareRelationship, 334
  - qevercloud::NoteMetadata, 337
  - qevercloud::NoteRestrictions, 340
  - qevercloud::NoteResultSpec, 343
  - qevercloud::NoteShareRelationshipRestrictions,  
345
  - qevercloud::NoteShareRelationships, 347
  - qevercloud::NoteVersionId, 398
  - qevercloud::Notebook, 301
  - qevercloud::NotebookDescriptor, 306
  - qevercloud::NotebookRecipientSettings, 307
  - qevercloud::NotebookRestrictions, 310
  - qevercloud::NotebookShareTemplate, 316
  - qevercloud::NotesMetadataList, 349
  - qevercloud::NotesMetadataResultSpec, 353
  - qevercloud::PublicUserInfo, 413
  - qevercloud::Publishing, 415
  - qevercloud::RelatedContent, 421
  - qevercloud::RelatedContentImage, 424
  - qevercloud::RelatedQuery, 426
  - qevercloud::RelatedResult, 429
  - qevercloud::RelatedResultSpec, 432
  - qevercloud::Resource, 436
  - qevercloud::ResourceAttributes, 440
  - qevercloud::SavedSearch, 442
  - qevercloud::SavedSearchScope, 444
  - qevercloud::ShareRelationshipRestrictions, 456
  - qevercloud::ShareRelationships, 459
  - qevercloud::SharedNote, 446
  - qevercloud::SharedNoteTemplate, 454
  - qevercloud::SharedNotebook, 449
  - qevercloud::SharedNotebookRecipientSettings,  
452
  - qevercloud::SyncChunk, 462
  - qevercloud::SyncChunkFilter, 467
  - qevercloud::SyncState, 470
  - qevercloud::Tag, 473
  - qevercloud::UpdateNoteIfUsnMatchesResult, 484
  - qevercloud::User, 487
  - qevercloud::UserAttributes, 493
  - qevercloud::UserIdentity, 498
  - qevercloud::UserProfile, 500
  - qevercloud::UserUrls, 512
- location
  - qevercloud::BusinessUserAttributes, 117
- log
  - qevercloud::ILogger, 172
- Log.h, 528
  - \_\_QEVERCLOUD\_LOG\_BASE, 529
  - QEC\_DEBUG, 529
  - QEC\_ERROR, 529
  - QEC\_INFO, 530
  - QEC\_TRACE, 530
  - QEC\_WARNING, 530
- LogLevel
  - qevercloud, 37
- logger
  - qevercloud, 44
- longIdentifier
  - qevercloud::UserIdentity, 498
- longitude
  - qevercloud::NoteAttributes, 297

- qevercloud::ResourceAttributes, 440
- m\_call
  - qevercloud::IDurableService::AsyncRequest, 97
  - qevercloud::IDurableService::SyncRequest, 468
- m\_contacts
  - qevercloud::EDAMInvalidContactsExceptionData, 132
- m\_description
  - qevercloud::IDurableService::AsyncRequest, 98
  - qevercloud::IDurableService::SyncRequest, 468
- m\_error
  - qevercloud::EverCloudException, 153
- m\_errorCode
  - qevercloud::EDAMSystemExceptionData, 143
  - qevercloud::EDAMUserExceptionData, 150
- m\_identifier
  - qevercloud::EDAMNotFoundExceptionData, 136
- m\_iterator
  - qevercloud::QAssociativeContainerConstReference←Wrapper::iterator, 248
  - qevercloud::QAssociativeContainerReference←Wrapper::iterator, 247
- m\_key
  - qevercloud::EDAMNotFoundExceptionData, 137
- m\_message
  - qevercloud::EDAMSystemExceptionData, 143
- m\_name
  - qevercloud::IDurableService::AsyncRequest, 98
  - qevercloud::IDurableService::SyncRequest, 468
- m\_parameter
  - qevercloud::EDAMInvalidContactsExceptionData, 132
  - qevercloud::EDAMUserExceptionData, 150
- m\_rateLimitDuration
  - qevercloud::EDAMSystemExceptionData, 144
- m\_reasons
  - qevercloud::EDAMInvalidContactsExceptionData, 132
- m\_type
  - qevercloud::NetworkException, 286
  - qevercloud::NetworkExceptionData, 287
  - qevercloud::ThriftException, 476
  - qevercloud::ThriftExceptionData, 478
- manageNotebookShares
  - qevercloud::INoteStore, 225
- manageNotebookSharesAsync
  - qevercloud::INoteStore, 226
- manageNotebookSharesRequest
  - qevercloud::NoteStoreServer, 378
- manageNotebookSharesRequestReady
  - qevercloud::NoteStoreServer, 378
- marketingUrl
  - qevercloud::BootstrapSettings, 110
- matchingShares
  - qevercloud::CreateOrUpdateNotebookShares←Result, 125
- maxExperts
  - qevercloud::RelatedResultSpec, 432
- maxNotebooks
  - qevercloud::RelatedResultSpec, 432
- maxNotes
  - qevercloud::RelatedResultSpec, 432
- maxReferrals
  - qevercloud::UserAttributes, 493
- maxRelatedContent
  - qevercloud::RelatedResultSpec, 433
- maxRequestRetryCount
  - qevercloud::IRequestContext, 244
- maxRequestTimeout
  - qevercloud::IRequestContext, 245
- maxTags
  - qevercloud::RelatedResultSpec, 433
- memberships
  - qevercloud::NoteShareRelationships, 347
  - qevercloud::ShareRelationships, 459
- membershipsToUnshare
  - qevercloud::ManageNoteSharesParameters, 279
- membershipsToUpdate
  - qevercloud::ManageNoteSharesParameters, 279
  - qevercloud::ManageNotebookSharesParameters, 272
- message
  - qevercloud::EDAMSystemException, 140
  - qevercloud::NoteEmailParameters, 321
- MessageEventID
  - qevercloud, 31
- messageStoreUrl
  - qevercloud::UserUrls, 512
- MessageThreadID
  - qevercloud, 31
- messagingPermit
  - qevercloud::Contact, 123
- messagingPermitExpires
  - qevercloud::Contact, 123
- mime
  - qevercloud::Resource, 436
- mobilePhone
  - qevercloud::BusinessUserAttributes, 117
- mostRecentReminder
  - qevercloud::BusinessInvitation, 113
- name
  - qevercloud::BootstrapProfile, 107
  - qevercloud::Contact, 123
  - qevercloud::Notebook, 302
  - qevercloud::SavedSearch, 442
  - qevercloud::Tag, 473
  - qevercloud::User, 487
  - qevercloud::UserProfile, 500
- NetworkException
  - qevercloud::NetworkException, 285
- NetworkExceptionData
  - qevercloud::NetworkExceptionData, 287
- newDurableService
  - qevercloud, 44
- newNoteStore
  - qevercloud, 44



- newRequestContext
  - qevercloud, [44](#)
- newRetryPolicy
  - qevercloud, [45](#)
- newStdErrLogger
  - qevercloud, [45](#)
- newUserStore
  - qevercloud, [45](#)
- nextChargeDate
  - qevercloud::Accounting, [92](#)
- nextPaymentDue
  - qevercloud::Accounting, [92](#)
- noCanMoveNote
  - qevercloud::NotebookRestrictions, [311](#)
- noChangeContact
  - qevercloud::NotebookRestrictions, [311](#)
- noCreateNotes
  - qevercloud::NotebookRestrictions, [311](#)
- noCreateSharedNotebooks
  - qevercloud::NotebookRestrictions, [311](#)
- noCreateTags
  - qevercloud::NotebookRestrictions, [311](#)
- noEmail
  - qevercloud::NoteRestrictions, [340](#)
- noEmailNotes
  - qevercloud::NotebookRestrictions, [311](#)
- noExpungeNotebook
  - qevercloud::NotebookRestrictions, [311](#)
- noExpungeNotes
  - qevercloud::NotebookRestrictions, [312](#)
- noExpungeTags
  - qevercloud::NotebookRestrictions, [312](#)
- noPublishToBusinessLibrary
  - qevercloud::NotebookRestrictions, [312](#)
- noPublishToPublic
  - qevercloud::NotebookRestrictions, [312](#)
- noReadNotes
  - qevercloud::NotebookRestrictions, [312](#)
- noRenameNotebook
  - qevercloud::NotebookRestrictions, [312](#)
- noSendMessageToRecipients
  - qevercloud::NotebookRestrictions, [312](#)
- noSetDefaultNotebook
  - qevercloud::NotebookRestrictions, [313](#)
- noSetFullAccess
  - qevercloud::NoteShareRelationshipRestrictions, [345](#)
  - qevercloud::ShareRelationshipRestrictions, [456](#)
- noSetInMyList
  - qevercloud::NotebookRestrictions, [313](#)
- noSetModify
  - qevercloud::ShareRelationshipRestrictions, [456](#)
- noSetModifyNote
  - qevercloud::NoteShareRelationshipRestrictions, [345](#)
- noSetNotebookStack
  - qevercloud::NotebookRestrictions, [313](#)
- noSetParentTag
  - qevercloud::NotebookRestrictions, [313](#)
- noSetReadNote
  - qevercloud::NoteShareRelationshipRestrictions, [345](#)
- noSetReadOnly
  - qevercloud::ShareRelationshipRestrictions, [457](#)
- noSetReadPlusActivity
  - qevercloud::ShareRelationshipRestrictions, [457](#)
- noSetRecipientSettingsStack
  - qevercloud::NotebookRestrictions, [313](#)
- noSetReminderNotifyEmail
  - qevercloud::NotebookRestrictions, [313](#)
- noSetReminderNotifyInApp
  - qevercloud::NotebookRestrictions, [313](#)
- noShare
  - qevercloud::NoteRestrictions, [340](#)
- noShareNotes
  - qevercloud::NotebookRestrictions, [314](#)
- noShareNotesWithBusiness
  - qevercloud::NotebookRestrictions, [314](#)
- noSharePublicly
  - qevercloud::NoteRestrictions, [340](#)
- noUpdateContent
  - qevercloud::NoteRestrictions, [341](#)
- noUpdateNotebook
  - qevercloud::NotebookRestrictions, [314](#)
- noUpdateNotes
  - qevercloud::NotebookRestrictions, [314](#)
- noUpdateTags
  - qevercloud::NotebookRestrictions, [314](#)
- noUpdateTitle
  - qevercloud::NoteRestrictions, [341](#)
- notFoundException
  - qevercloud::ManageNoteSharesError, [276](#)
  - qevercloud::ManageNotebookSharesError, [270](#)
- note
  - qevercloud::NoteEmailParameters, [321](#)
  - qevercloud::UpdateNoteIfUsnMatchesResult, [484](#)
- noteGuid
  - qevercloud::ManageNoteSharesParameters, [279](#)
  - qevercloud::RelatedQuery, [426](#)
  - qevercloud::Resource, [436](#)
  - qevercloud::SharedNoteTemplate, [454](#)
- noteResourceCountMax
  - qevercloud::AccountLimits, [95](#)
  - qevercloud::NoteLimits, [329](#)
- noteSizeMax
  - qevercloud::AccountLimits, [95](#)
  - qevercloud::NoteLimits, [330](#)
- NoteSortOrder
  - qevercloud, [37](#)
- NoteStoreServer
  - qevercloud::NoteStoreServer, [359](#)
- noteStoreUrl
  - qevercloud::AuthenticationResult, [103](#)
  - qevercloud::EvernoteOAuthWebView::OAuthResult, [400](#)
  - qevercloud::INoteStore, [226](#)

- qevercloud::LinkedNotebook, 267
  - qevercloud::PublicUserInfo, 413
  - qevercloud::UserUrls, 512
- noteTagCountMax
  - qevercloud::AccountLimits, 95
- noteTitleQuality
  - qevercloud::NoteAttributes, 297
- notebookCounts
  - qevercloud::NoteCollectionCounts, 319
- notebookDescription
  - qevercloud::BusinessNotebook, 115
- notebookDisplayName
  - qevercloud::NotebookDescriptor, 306
- notebookGuid
  - qevercloud::ManageNotebookSharesParameters, 272
  - qevercloud::Note, 291
  - qevercloud::NoteFilter, 325
  - qevercloud::NoteMetadata, 337
  - qevercloud::NotebookShareTemplate, 316
  - qevercloud::SharedNotebook, 449
- notebookGuids
  - qevercloud::SyncChunkFilter, 467
- notebookModifiable
  - qevercloud::SharedNotebook, 449
- notebooks
  - qevercloud::RelatedResult, 430
  - qevercloud::SyncChunk, 462
- notes
  - qevercloud::NoteList, 332
  - qevercloud::NotesMetadataList, 349
  - qevercloud::RelatedResult, 430
  - qevercloud::SyncChunk, 462
- nullLogger
  - qevercloud, 45
- nullRetryPolicy
  - qevercloud, 45
- OAuth.h, 530
- OAuthResult
  - qevercloud::EvernoteOAuthDialog, 163
- oauthError
  - qevercloud::EvernoteOAuthDialog, 164
  - qevercloud::EvernoteOAuthWebView, 167
- oauthResult
  - qevercloud::EvernoteOAuthDialog, 164
  - qevercloud::EvernoteOAuthWebView, 167
- omitSharedNotebooks
  - qevercloud::SyncChunkFilter, 467
- onAuthenticateLongSessionRequestReady
  - qevercloud::UserStoreServer, 507
- onAuthenticateToBusinessRequestReady
  - qevercloud::UserStoreServer, 507
- onAuthenticateToSharedNoteRequestReady
  - qevercloud::NoteStoreServer, 378
- onAuthenticateToSharedNotebookRequestReady
  - qevercloud::NoteStoreServer, 378
- onCheckVersionRequestReady
  - qevercloud::UserStoreServer, 507
- onCompleteTwoFactorAuthenticationRequestReady
  - qevercloud::UserStoreServer, 508
- onCopyNoteRequestReady
  - qevercloud::NoteStoreServer, 378
- onCreateLinkedNotebookRequestReady
  - qevercloud::NoteStoreServer, 378
- onCreateNoteRequestReady
  - qevercloud::NoteStoreServer, 379
- onCreateNotebookRequestReady
  - qevercloud::NoteStoreServer, 379
- onCreateOrUpdateNotebookSharesRequestReady
  - qevercloud::NoteStoreServer, 379
- onCreateSearchRequestReady
  - qevercloud::NoteStoreServer, 379
- onCreateTagRequestReady
  - qevercloud::NoteStoreServer, 379
- onDeleteNoteRequestReady
  - qevercloud::NoteStoreServer, 379
- onEmailNoteRequestReady
  - qevercloud::NoteStoreServer, 380
- onExpungeLinkedNotebookRequestReady
  - qevercloud::NoteStoreServer, 380
- onExpungeNoteRequestReady
  - qevercloud::NoteStoreServer, 380
- onExpungeNotebookRequestReady
  - qevercloud::NoteStoreServer, 380
- onExpungeSearchRequestReady
  - qevercloud::NoteStoreServer, 380
- onExpungeTagRequestReady
  - qevercloud::NoteStoreServer, 380
- onFindNoteCountsRequestReady
  - qevercloud::NoteStoreServer, 381
- onFindNoteOffsetRequestReady
  - qevercloud::NoteStoreServer, 381
- onFindNotesMetadataRequestReady
  - qevercloud::NoteStoreServer, 381
- onFindRelatedRequestReady
  - qevercloud::NoteStoreServer, 381
- onGetAccountLimitsRequestReady
  - qevercloud::UserStoreServer, 508
- onGetBootstrapInfoRequestReady
  - qevercloud::UserStoreServer, 508
- onGetDefaultNotebookRequestReady
  - qevercloud::NoteStoreServer, 381
- onGetFilteredSyncChunkRequestReady
  - qevercloud::NoteStoreServer, 381
- onGetLinkedNotebookSyncChunkRequestReady
  - qevercloud::NoteStoreServer, 382
- onGetLinkedNotebookSyncStateRequestReady
  - qevercloud::NoteStoreServer, 382
- onGetNoteApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, 382
- onGetNoteApplicationDataRequestReady
  - qevercloud::NoteStoreServer, 382
- onGetNoteContentRequestReady
  - qevercloud::NoteStoreServer, 383
- onGetNoteRequestReady
  - qevercloud::NoteStoreServer, 383

- onGetNoteSearchTextRequestReady
  - qevercloud::NoteStoreServer, [383](#)
- onGetNoteTagNamesRequestReady
  - qevercloud::NoteStoreServer, [383](#)
- onGetNoteVersionRequestReady
  - qevercloud::NoteStoreServer, [383](#)
- onGetNoteWithResultSpecRequestReady
  - qevercloud::NoteStoreServer, [383](#)
- onGetNotebookRequestReady
  - qevercloud::NoteStoreServer, [382](#)
- onGetNotebookSharesRequestReady
  - qevercloud::NoteStoreServer, [382](#)
- onGetPublicNotebookRequestReady
  - qevercloud::NoteStoreServer, [384](#)
- onGetPublicUserInfoRequestReady
  - qevercloud::UserStoreServer, [508](#)
- onGetResourceAlternateDataRequestReady
  - qevercloud::NoteStoreServer, [384](#)
- onGetResourceApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, [384](#)
- onGetResourceApplicationDataRequestReady
  - qevercloud::NoteStoreServer, [384](#)
- onGetResourceAttributesRequestReady
  - qevercloud::NoteStoreServer, [384](#)
- onGetResourceByHashRequestReady
  - qevercloud::NoteStoreServer, [384](#)
- onGetResourceDataRequestReady
  - qevercloud::NoteStoreServer, [385](#)
- onGetResourceRecognitionRequestReady
  - qevercloud::NoteStoreServer, [385](#)
- onGetResourceRequestReady
  - qevercloud::NoteStoreServer, [385](#)
- onGetResourceSearchTextRequestReady
  - qevercloud::NoteStoreServer, [385](#)
- onGetSearchRequestReady
  - qevercloud::NoteStoreServer, [385](#)
- onGetSharedNotebookByAuthRequestReady
  - qevercloud::NoteStoreServer, [385](#)
- onGetSyncStateRequestReady
  - qevercloud::NoteStoreServer, [386](#)
- onGetTagRequestReady
  - qevercloud::NoteStoreServer, [386](#)
- onGetUserRequestReady
  - qevercloud::UserStoreServer, [508](#)
- onGetUserUrlsRequestReady
  - qevercloud::UserStoreServer, [508](#)
- onInviteToBusinessRequestReady
  - qevercloud::UserStoreServer, [509](#)
- onListAccessibleBusinessNotebooksRequestReady
  - qevercloud::NoteStoreServer, [386](#)
- onListBusinessInvitationsRequestReady
  - qevercloud::UserStoreServer, [509](#)
- onListBusinessUsersRequestReady
  - qevercloud::UserStoreServer, [509](#)
- onListLinkedNotebooksRequestReady
  - qevercloud::NoteStoreServer, [386](#)
- onListNoteVersionsRequestReady
  - qevercloud::NoteStoreServer, [386](#)
- onListNotebooksRequestReady
  - qevercloud::NoteStoreServer, [386](#)
- onListSearchesRequestReady
  - qevercloud::NoteStoreServer, [387](#)
- onListSharedNotebooksRequestReady
  - qevercloud::NoteStoreServer, [387](#)
- onListTagsByNotebookRequestReady
  - qevercloud::NoteStoreServer, [387](#)
- onListTagsRequestReady
  - qevercloud::NoteStoreServer, [387](#)
- onManageNotebookSharesRequestReady
  - qevercloud::NoteStoreServer, [387](#)
- onRemoveFromBusinessRequestReady
  - qevercloud::UserStoreServer, [509](#)
- onRequest
  - qevercloud::NoteStoreServer, [387](#)
  - qevercloud::UserStoreServer, [509](#)
- onRevokeLongSessionRequestReady
  - qevercloud::UserStoreServer, [509](#)
- onSetNoteApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, [388](#)
- onSetNotebookRecipientSettingsRequestReady
  - qevercloud::NoteStoreServer, [388](#)
- onSetResourceApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, [388](#)
- onShareNoteRequestReady
  - qevercloud::NoteStoreServer, [388](#)
- onShareNotebookRequestReady
  - qevercloud::NoteStoreServer, [388](#)
- onStopSharingNoteRequestReady
  - qevercloud::NoteStoreServer, [388](#)
- onUnsetNoteApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, [389](#)
- onUnsetResourceApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, [389](#)
- onUntagAllRequestReady
  - qevercloud::NoteStoreServer, [389](#)
- onUpdateBusinessUserIdentifierRequestReady
  - qevercloud::UserStoreServer, [510](#)
- onUpdateLinkedNotebookRequestReady
  - qevercloud::NoteStoreServer, [389](#)
- onUpdateNoteIfUsnMatchesRequestReady
  - qevercloud::NoteStoreServer, [389](#)
- onUpdateNoteRequestReady
  - qevercloud::NoteStoreServer, [390](#)
- onUpdateNotebookRequestReady
  - qevercloud::NoteStoreServer, [389](#)
- onUpdateResourceRequestReady
  - qevercloud::NoteStoreServer, [390](#)
- onUpdateSearchRequestReady
  - qevercloud::NoteStoreServer, [390](#)
- onUpdateSharedNotebookRequestReady
  - qevercloud::NoteStoreServer, [390](#)
- onUpdateTagRequestReady
  - qevercloud::NoteStoreServer, [390](#)
- open
  - qevercloud::EvernoteOAuthDialog, [165](#)
- operator const T &

- qevercloud::Optional, 405
- operator T &
  - qevercloud::Optional, 405
- operator!=
  - qevercloud::AccountLimits, 94
  - qevercloud::Accounting, 90
  - qevercloud::AuthenticationResult, 102
  - qevercloud::BootstrapInfo, 105
  - qevercloud::BootstrapProfile, 106
  - qevercloud::BootstrapSettings, 108
  - qevercloud::BusinessInvitation, 112
  - qevercloud::BusinessNotebook, 114
  - qevercloud::BusinessUserAttributes, 116
  - qevercloud::BusinessUserInfo, 118
  - qevercloud::CanMoveToContainerRestrictions, 120
  - qevercloud::Contact, 122
  - qevercloud::CreateOrUpdateNotebookShares↔
    - Result, 125
  - qevercloud::Data, 127
  - qevercloud::EDAMInvalidContactsException, 130
  - qevercloud::EDAMNotFoundException, 134
  - qevercloud::EDAMSystemException, 139
  - qevercloud::EDAMUserException, 148
  - qevercloud::EverCloudLocalData, 157
  - qevercloud::Identity, 169
  - qevercloud::InvitationShareRelationship, 242
  - qevercloud::LazyMap, 264
  - qevercloud::LinkedNotebook, 266
  - qevercloud::ManageNoteSharesError, 276
  - qevercloud::ManageNoteSharesParameters, 278
  - qevercloud::ManageNoteSharesResult, 280
  - qevercloud::ManageNotebookSharesError, 269
  - qevercloud::ManageNotebookSharesParameters, 271
  - qevercloud::ManageNotebookSharesResult, 274
  - qevercloud::MemberShareRelationship, 282
  - qevercloud::NetworkException, 285
  - qevercloud::Note, 289
  - qevercloud::NoteAttributes, 294
  - qevercloud::NoteCollectionCounts, 318
  - qevercloud::NoteEmailParameters, 320
  - qevercloud::NoteFilter, 323
  - qevercloud::NoteInvitationShareRelationship, 327
  - qevercloud::NoteLimits, 329
  - qevercloud::NoteList, 331
  - qevercloud::NoteMemberShareRelationship, 334
  - qevercloud::NoteMetadata, 336
  - qevercloud::NoteRestrictions, 339
  - qevercloud::NoteResultSpec, 342
  - qevercloud::NoteShareRelationshipRestrictions, 344
  - qevercloud::NoteShareRelationships, 346
  - qevercloud::NoteVersionId, 397
  - qevercloud::Notebook, 300
  - qevercloud::NotebookDescriptor, 304
  - qevercloud::NotebookRecipientSettings, 307
  - qevercloud::NotebookRestrictions, 310
  - qevercloud::NotebookShareTemplate, 315
  - qevercloud::NotesMetadataList, 348
  - qevercloud::NotesMetadataResultSpec, 351
  - qevercloud::Optional, 405
  - qevercloud::PublicUserInfo, 413
  - qevercloud::Publishing, 415
  - qevercloud::QAssociativeContainerConstReference↔
    - Wrapper::iterator, 248
  - qevercloud::QAssociativeContainerReference↔
    - Wrapper::iterator, 246
  - qevercloud::RelatedContent, 419
  - qevercloud::RelatedContentImage, 423
  - qevercloud::RelatedQuery, 425
  - qevercloud::RelatedResult, 428
  - qevercloud::RelatedResultSpec, 431
  - qevercloud::Resource, 434
  - qevercloud::ResourceAttributes, 438
  - qevercloud::SavedSearch, 441
  - qevercloud::SavedSearchScope, 443
  - qevercloud::ShareRelationshipRestrictions, 456
  - qevercloud::ShareRelationships, 458
  - qevercloud::SharedNote, 445
  - qevercloud::SharedNoteTemplate, 453
  - qevercloud::SharedNotebook, 448
  - qevercloud::SharedNotebookRecipientSettings, 452
  - qevercloud::SyncChunk, 460
  - qevercloud::SyncChunkFilter, 464
  - qevercloud::SyncState, 469
  - qevercloud::Tag, 472
  - qevercloud::ThriftException, 475
  - qevercloud::UpdateNoteIfUsnMatchesResult, 484
  - qevercloud::User, 486
  - qevercloud::UserAttributes, 490
  - qevercloud::UserIdentity, 497
  - qevercloud::UserProfile, 499
  - qevercloud::UserUrls, 511
- operator<<
  - qevercloud, 45–53
  - qevercloud::IRequestContext, 245
  - qevercloud::Printable, 411, 412
  - qevercloud::ThriftException, 476
  - qevercloud::Thumbnail, 483
- operator\*
  - qevercloud::QAssociativeContainerConstReference↔
    - Wrapper::iterator, 248
  - qevercloud::QAssociativeContainerReference↔
    - Wrapper::iterator, 247
- operator++
  - qevercloud::QAssociativeContainerConstReference↔
    - Wrapper::iterator, 248
  - qevercloud::QAssociativeContainerReference↔
    - Wrapper::iterator, 247
- operator->
  - qevercloud::Optional, 405, 406
- operator=
  - qevercloud::Optional, 406, 407
- operator==
  - qevercloud::AccountLimits, 95

- qevercloud::Accounting, 90
- qevercloud::AuthenticationResult, 102
- qevercloud::BootstrapInfo, 105
- qevercloud::BootstrapProfile, 107
- qevercloud::BootstrapSettings, 108
- qevercloud::BusinessInvitation, 112
- qevercloud::BusinessNotebook, 114
- qevercloud::BusinessUserAttributes, 116
- qevercloud::BusinessUserInfo, 118
- qevercloud::CanMoveToContainerRestrictions, 121
- qevercloud::Contact, 122
- qevercloud::CreateOrUpdateNotebookShares←  
Result, 125
- qevercloud::Data, 127
- qevercloud::EDAMInvalidContactsException, 130
- qevercloud::EDAMNotFoundException, 134
- qevercloud::EDAMSystemException, 139
- qevercloud::EDAMUserException, 148
- qevercloud::EverCloudLocalData, 157
- qevercloud::Identity, 169
- qevercloud::InvitationShareRelationship, 242
- qevercloud::LazyMap, 264
- qevercloud::LinkedNotebook, 266
- qevercloud::ManageNoteSharesError, 276
- qevercloud::ManageNoteSharesParameters, 278
- qevercloud::ManageNoteSharesResult, 280
- qevercloud::ManageNotebookSharesError, 269
- qevercloud::ManageNotebookSharesParameters,  
271
- qevercloud::ManageNotebookSharesResult, 274
- qevercloud::MemberShareRelationship, 282
- qevercloud::NetworkException, 286
- qevercloud::Note, 289
- qevercloud::NoteAttributes, 294
- qevercloud::NoteCollectionCounts, 318
- qevercloud::NoteEmailParameters, 320
- qevercloud::NoteFilter, 323
- qevercloud::NoteInvitationShareRelationship, 327
- qevercloud::NoteLimits, 329
- qevercloud::NoteList, 331
- qevercloud::NoteMemberShareRelationship, 334
- qevercloud::NoteMetadata, 336
- qevercloud::NoteRestrictions, 340
- qevercloud::NoteResultSpec, 342
- qevercloud::NoteShareRelationshipRestrictions,  
344
- qevercloud::NoteShareRelationships, 346
- qevercloud::NoteVersionId, 397
- qevercloud::Notebook, 300
- qevercloud::NotebookDescriptor, 305
- qevercloud::NotebookRecipientSettings, 307
- qevercloud::NotebookRestrictions, 310
- qevercloud::NotebookShareTemplate, 315
- qevercloud::NotesMetadataList, 349
- qevercloud::NotesMetadataResultSpec, 352
- qevercloud::Optional, 407
- qevercloud::PublicUserInfo, 413
- qevercloud::Publishing, 415
- qevercloud::RelatedContent, 419
- qevercloud::RelatedContentImage, 423
- qevercloud::RelatedQuery, 425
- qevercloud::RelatedResult, 428
- qevercloud::RelatedResultSpec, 431
- qevercloud::Resource, 434
- qevercloud::ResourceAttributes, 438
- qevercloud::SavedSearch, 441
- qevercloud::SavedSearchScope, 443
- qevercloud::ShareRelationshipRestrictions, 456
- qevercloud::ShareRelationships, 458
- qevercloud::SharedNote, 445
- qevercloud::SharedNoteTemplate, 454
- qevercloud::SharedNotebook, 448
- qevercloud::SharedNotebookRecipientSettings,  
452
- qevercloud::SyncChunk, 460
- qevercloud::SyncChunkFilter, 464
- qevercloud::SyncState, 469
- qevercloud::Tag, 472
- qevercloud::ThriftException, 476
- qevercloud::UpdateNoteIfUsnMatchesResult, 484
- qevercloud::User, 486
- qevercloud::UserAttributes, 491
- qevercloud::UserIdentity, 497
- qevercloud::UserProfile, 499
- qevercloud::UserUrls, 511
- optOutMachineLearning
  - qevercloud::UserAttributes, 493
- Optional
  - qevercloud::Optional, 402, 403, 409
- Optional.h, 531
- order
  - qevercloud::NoteFilter, 325
  - qevercloud::Publishing, 416
- parameter
  - qevercloud::EDAMInvalidContactsException, 130
  - qevercloud::EDAMUserException, 148
- parentGuid
  - qevercloud::Tag, 473
- partnerEmailOptInDate
  - qevercloud::UserAttributes, 493
- passwordUpdated
  - qevercloud::UserAttributes, 494
- photoLastUpdated
  - qevercloud::Contact, 123
  - qevercloud::User, 488
  - qevercloud::UserProfile, 500
- photoUrl
  - qevercloud::Contact, 123
  - qevercloud::User, 488
  - qevercloud::UserProfile, 500
- pixelRatio
  - qevercloud::RelatedContentImage, 424
- placeName
  - qevercloud::NoteAttributes, 297
- plainText
  - qevercloud::RelatedQuery, 426



- preactivation
  - qevercloud::UserAttributes, 494
- preferredCountry
  - qevercloud::UserAttributes, 494
- preferredLanguage
  - qevercloud::UserAttributes, 494
- premiumCommerceService
  - qevercloud::Accounting, 92
- premiumLockUntil
  - qevercloud::Accounting, 92
- premiumOrderNumber
  - qevercloud::Accounting, 92
- PremiumOrderStatus
  - qevercloud, 37
- premiumServiceSKU
  - qevercloud::Accounting, 92
- premiumServiceStart
  - qevercloud::Accounting, 92
- premiumServiceStatus
  - qevercloud::Accounting, 93
- premiumSubscriptionNumber
  - qevercloud::Accounting, 93
- print
  - qevercloud::AccountLimits, 95
  - qevercloud::Accounting, 90
  - qevercloud::AuthenticationResult, 102
  - qevercloud::BootstrapInfo, 105
  - qevercloud::BootstrapProfile, 107
  - qevercloud::BootstrapSettings, 109
  - qevercloud::BusinessInvitation, 112
  - qevercloud::BusinessNotebook, 114
  - qevercloud::BusinessUserAttributes, 116
  - qevercloud::BusinessUserInfo, 119
  - qevercloud::CanMoveToContainerRestrictions, 121
  - qevercloud::Contact, 122
  - qevercloud::CreateOrUpdateNotebookShares←  
Result, 125
  - qevercloud::Data, 127
  - qevercloud::EDAMInvalidContactsException, 130
  - qevercloud::EDAMNotFoundException, 135
  - qevercloud::EDAMSystemException, 139
  - qevercloud::EDAMUserException, 148
  - qevercloud::EverCloudLocalData, 158
  - qevercloud::EvernoteOAuthWebView::OAuth←  
Result, 399
  - qevercloud::Identity, 169
  - qevercloud::InvitationShareRelationship, 242
  - qevercloud::LazyMap, 264
  - qevercloud::LinkedNotebook, 266
  - qevercloud::ManageNoteSharesError, 276
  - qevercloud::ManageNoteSharesParameters, 278
  - qevercloud::ManageNoteSharesResult, 281
  - qevercloud::ManageNotebookSharesError, 270
  - qevercloud::ManageNotebookSharesParameters,  
272
  - qevercloud::ManageNotebookSharesResult, 274
  - qevercloud::MemberShareRelationship, 282
  - qevercloud::Note, 289
  - qevercloud::NoteAttributes, 294
  - qevercloud::NoteCollectionCounts, 318
  - qevercloud::NoteEmailParameters, 321
  - qevercloud::NoteFilter, 323
  - qevercloud::NoteInvitationShareRelationship, 327
  - qevercloud::NoteLimits, 329
  - qevercloud::NoteList, 331
  - qevercloud::NoteMemberShareRelationship, 334
  - qevercloud::NoteMetadata, 336
  - qevercloud::NoteRestrictions, 340
  - qevercloud::NoteResultSpec, 342
  - qevercloud::NoteShareRelationshipRestrictions,  
345
  - qevercloud::NoteShareRelationships, 347
  - qevercloud::NoteVersionId, 397
  - qevercloud::Notebook, 300
  - qevercloud::NotebookDescriptor, 305
  - qevercloud::NotebookRecipientSettings, 307
  - qevercloud::NotebookRestrictions, 310
  - qevercloud::NotebookShareTemplate, 316
  - qevercloud::NotesMetadataList, 349
  - qevercloud::NotesMetadataResultSpec, 352
  - qevercloud::Printable, 411
  - qevercloud::PublicUserInfo, 413
  - qevercloud::Publishing, 415
  - qevercloud::RelatedContent, 419
  - qevercloud::RelatedContentImage, 423
  - qevercloud::RelatedQuery, 425
  - qevercloud::RelatedResult, 428
  - qevercloud::RelatedResultSpec, 431
  - qevercloud::Resource, 435
  - qevercloud::ResourceAttributes, 438
  - qevercloud::SavedSearch, 441
  - qevercloud::SavedSearchScope, 444
  - qevercloud::ShareRelationshipRestrictions, 456
  - qevercloud::ShareRelationships, 458
  - qevercloud::SharedNote, 446
  - qevercloud::SharedNoteTemplate, 454
  - qevercloud::SharedNotebook, 448
  - qevercloud::SharedNotebookRecipientSettings,  
452
  - qevercloud::SyncChunk, 460
  - qevercloud::SyncChunkFilter, 464
  - qevercloud::SyncState, 470
  - qevercloud::Tag, 472
  - qevercloud::UpdateNoteIfUsnMatchesResult, 484
  - qevercloud::User, 486
  - qevercloud::UserAttributes, 491
  - qevercloud::UserIdentity, 497
  - qevercloud::UserProfile, 499
  - qevercloud::UserUrls, 512
- Printable
  - qevercloud::Printable, 410
- Printable.h, 531
- privilege
  - qevercloud::BusinessNotebook, 115
  - qevercloud::InvitationShareRelationship, 243
  - qevercloud::NoteInvitationShareRelationship, 327

- qevercloud::NoteMemberShareRelationship, 334
- qevercloud::NotebookShareTemplate, 316
- qevercloud::SharedNote, 446
- qevercloud::SharedNoteTemplate, 454
- qevercloud::SharedNotebook, 449
- qevercloud::User, 488
- PrivilegeLevel
  - qevercloud, 38
- profiles
  - qevercloud::BootstrapInfo, 106
- publicDescription
  - qevercloud::Publishing, 416
- publicUserInfo
  - qevercloud::AuthenticationResult, 103
- published
  - qevercloud::Notebook, 302
- publishing
  - qevercloud::Notebook, 302
- QAssociativeContainerConstReferenceWrapper
  - qevercloud::QAssociativeContainerConstReference↔  
Wrapper, 417
- QAssociativeContainerReferenceWrapper
  - qevercloud::QAssociativeContainerReference↔  
Wrapper, 418
- QEC\_DEBUG
  - Log.h, 529
- QEC\_ERROR
  - Log.h, 529
- QEC\_INFO
  - Log.h, 530
- QEC\_TRACE
  - Log.h, 530
- QEC\_WARNING
  - Log.h, 530
- QEVERCLOUD\_EXPORT
  - Export.h, 526
- QEverCloud.h, 532
- QEverCloudOAuth.h, 532
- qHash
  - qevercloud, 53–57
- QList
  - qevercloud::CreateOrUpdateNotebookShares↔  
Result, 126
  - qevercloud::EDAMInvalidContactsException, 131
  - qevercloud::ManageNoteSharesParameters, 279
  - qevercloud::ManageNoteSharesResult, 281
  - qevercloud::ManageNotebookSharesParameters,  
273
  - qevercloud::ManageNotebookSharesResult, 275
  - qevercloud::Note, 292
  - qevercloud::NoteFilter, 326
  - qevercloud::NoteMetadata, 338
  - qevercloud::NoteShareRelationships, 348
  - qevercloud::Notebook, 304
  - qevercloud::NotebookShareTemplate, 317
  - qevercloud::RelatedContent, 422
  - qevercloud::RelatedResult, 430
  - qevercloud::ShareRelationships, 459
  - qevercloud::SharedNoteTemplate, 455
  - qevercloud::SyncChunk, 463
- QSet
  - qevercloud::LazyMap, 265
  - qevercloud::RelatedResultSpec, 433
  - qevercloud::SyncChunkFilter, 467
- qevercloud, 19
  - BusinessInvitationStatus, 32
  - BusinessUserRole, 32
  - BusinessUserStatus, 33
  - CLASSIFICATION\_RECIPE\_SERVICE\_RECIPE,  
58
  - CLASSIFICATION\_RECIPE\_USER\_NON\_REC↔  
IPE, 58
  - CLASSIFICATION\_RECIPE\_USER\_RECIPE, 58
  - CanMoveToContainerStatus, 33
  - ContactType, 33
  - Debug, 37
  - EDAM\_APP\_RATING\_MAX, 59
  - EDAM\_APP\_RATING\_MIN, 59
  - EDAM\_APPLICATIONDATA\_ENTRY\_LEN\_MAX,  
59
  - EDAM\_APPLICATIONDATA\_NAME\_LEN\_MAX,  
59
  - EDAM\_APPLICATIONDATA\_NAME\_LEN\_MIN,  
59
  - EDAM\_APPLICATIONDATA\_NAME\_REGEX, 59
  - EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MAX,  
59
  - EDAM\_APPLICATIONDATA\_VALUE\_LEN\_MIN,  
60
  - EDAM\_APPLICATIONDATA\_VALUE\_REGEX, 60
  - EDAM\_ATTRIBUTE\_LEN\_MAX, 60
  - EDAM\_ATTRIBUTE\_LEN\_MIN, 60
  - EDAM\_ATTRIBUTE\_LIST\_MAX, 60
  - EDAM\_ATTRIBUTE\_MAP\_MAX, 60
  - EDAM\_ATTRIBUTE\_REGEX, 60
  - EDAM\_BUSINESS\_MARKETING\_CODE\_REG↔  
EX\_PATTERN, 61
  - EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTOR↔  
N\_LEN\_MAX, 61
  - EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTOR↔  
N\_LEN\_MIN, 61
  - EDAM\_BUSINESS\_NOTEBOOK\_DESCRIPTOR↔  
N\_REGEX, 61
  - EDAM\_BUSINESS\_NOTEBOOKS\_MAX, 61
  - EDAM\_BUSINESS\_NOTES\_MAX, 61
  - EDAM\_BUSINESS\_PHONE\_NUMBER\_LEN\_M↔  
AX, 61
  - EDAM\_BUSINESS\_TAGS\_MAX, 62
  - EDAM\_BUSINESS\_URI\_LEN\_MAX, 62
  - EDAM\_BUSINESS\_WORKSPACES\_MAX, 62
  - EDAM\_CONNECTED\_IDENTITY\_REQUEST\_↔  
MAX, 62
  - EDAM\_CONTENT\_CLASS\_FOOD\_MEAL, 62
  - EDAM\_CONTENT\_CLASS\_HELLO\_ENCOUN↔  
ER, 62
  - EDAM\_CONTENT\_CLASS\_HELLO\_PROFILE, 62

- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_NOTEBOOK, 63
- EDAM\_CONTENT\_CLASS\_PENULTIMATE\_PREFIX, 63
- EDAM\_CONTENT\_CLASS\_SKITCH\_PDF, 63
- EDAM\_CONTENT\_CLASS\_SKITCH\_PREFIX, 63
- EDAM\_CONTENT\_CLASS\_SKITCH, 63
- EDAM\_DEVICE\_DESCRIPTION\_LEN\_MAX, 63
- EDAM\_DEVICE\_DESCRIPTION\_REGEX, 63
- EDAM\_DEVICE\_ID\_LEN\_MAX, 64
- EDAM\_DEVICE\_ID\_REGEX, 64
- EDAM\_EMAIL\_DOMAIN\_REGEX, 64
- EDAM\_EMAIL\_LEN\_MAX, 64
- EDAM\_EMAIL\_LEN\_MIN, 64
- EDAM\_EMAIL\_LOCAL\_REGEX, 64
- EDAM\_EMAIL\_REGEX, 64
- EDAM\_FIND\_CONTACT\_DEFAULT\_MAX\_RESULTS, 64
- EDAM\_FIND\_CONTACT\_MAX\_RESULTS, 65
- EDAM\_FOOD\_APP\_CONTENT\_CLASS\_PREFIX, 65
- EDAM\_GET\_ORDERS\_MAX\_RESULTS, 65
- EDAM\_GUID\_LEN\_MAX, 65
- EDAM\_GUID\_LEN\_MIN, 65
- EDAM\_GUID\_REGEX, 65
- EDAM\_HASH\_LEN, 65
- EDAM\_HELLO\_APP\_CONTENT\_CLASS\_PREFIX, 66
- EDAM\_INDEXABLE\_PLAINTEXT\_MIME\_TYPES, 66
- EDAM\_INDEXABLE\_RESOURCE\_MIME\_TYPES, 66
- EDAM\_MAX\_PREFERENCES, 66
- EDAM\_MAX\_VALUES\_PER\_PREFERENCE, 66
- EDAM\_MESSAGE\_ATTACHMENT\_SNIPPET\_LEN\_MAX, 66
- EDAM\_MESSAGE\_ATTACHMENT\_SNIPPET\_REGEX, 66
- EDAM\_MESSAGE\_ATTACHMENT\_TITLE\_LEN\_MAX, 67
- EDAM\_MESSAGE\_ATTACHMENT\_TITLE\_REGEX, 67
- EDAM\_MESSAGE\_ATTACHMENTS\_MAX, 67
- EDAM\_MESSAGE\_BODY\_LEN\_MAX, 67
- EDAM\_MESSAGE\_BODY\_REGEX, 67
- EDAM\_MESSAGE\_RECIPIENTS\_MAX, 67
- EDAM\_MIME\_LEN\_MAX, 67
- EDAM\_MIME\_LEN\_MIN, 67
- EDAM\_MIME\_REGEX, 68
- EDAM\_MIME\_TYPE\_AAC, 68
- EDAM\_MIME\_TYPE\_AMR, 68
- EDAM\_MIME\_TYPE\_BMP, 68
- EDAM\_MIME\_TYPE\_DEFAULT, 68
- EDAM\_MIME\_TYPE\_GIF, 68
- EDAM\_MIME\_TYPE\_INK, 68
- EDAM\_MIME\_TYPE\_JPEG, 68
- EDAM\_MIME\_TYPE\_M4A, 69
- EDAM\_MIME\_TYPE\_MP3, 69
- EDAM\_MIME\_TYPE\_MP4\_VIDEO, 69
- EDAM\_MIME\_TYPE\_PDF, 69
- EDAM\_MIME\_TYPE\_PNG, 69
- EDAM\_MIME\_TYPE\_TIFF, 69
- EDAM\_MIME\_TYPE\_WAV, 69
- EDAM\_MIME\_TYPES, 69
- EDAM\_NOTE\_BUSINESS\_SHARED\_NOTE\_MAX, 70
- EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MAX, 70
- EDAM\_NOTE\_CONTENT\_CLASS\_LEN\_MIN, 70
- EDAM\_NOTE\_CONTENT\_CLASS\_REGEX, 70
- EDAM\_NOTE\_CONTENT\_LEN\_MAX, 70
- EDAM\_NOTE\_CONTENT\_LEN\_MIN, 70
- EDAM\_NOTE\_LOCK\_VIEWERS\_NOTES\_MAX, 70
- EDAM\_NOTE\_PERSONAL\_SHARED\_NOTE\_MAX, 71
- EDAM\_NOTE\_RESOURCES\_MAX, 71
- EDAM\_NOTE\_SIZE\_MAX\_FREE, 71
- EDAM\_NOTE\_SIZE\_MAX\_PREMIUM, 71
- EDAM\_NOTE\_SOURCE\_MAIL\_CLIP, 71
- EDAM\_NOTE\_SOURCE\_MAIL\_SMTP\_GATEWAY, 71
- EDAM\_NOTE\_SOURCE\_WEB\_CLIP\_SIMPLIFIED, 72
- EDAM\_NOTE\_SOURCE\_WEB\_CLIP, 72
- EDAM\_NOTE\_TAGS\_MAX, 72
- EDAM\_NOTE\_TITLE\_LEN\_MAX, 72
- EDAM\_NOTE\_TITLE\_LEN\_MIN, 72
- EDAM\_NOTE\_TITLE\_QUALITY\_HIGH, 72
- EDAM\_NOTE\_TITLE\_QUALITY\_LOW, 72
- EDAM\_NOTE\_TITLE\_QUALITY\_MEDIUM, 72
- EDAM\_NOTE\_TITLE\_QUALITY\_UNTITLED, 73
- EDAM\_NOTE\_TITLE\_REGEX, 73
- EDAM\_NOTEBOOK\_BUSINESS\_SHARED\_NOTEBOOK\_MAX, 73
- EDAM\_NOTEBOOK\_NAME\_LEN\_MAX, 73
- EDAM\_NOTEBOOK\_NAME\_LEN\_MIN, 73
- EDAM\_NOTEBOOK\_NAME\_REGEX, 73
- EDAM\_NOTEBOOK\_PERSONAL\_SHARED\_NOTEBOOK\_MAX, 73
- EDAM\_NOTEBOOK\_STACK\_LEN\_MAX, 74
- EDAM\_NOTEBOOK\_STACK\_LEN\_MIN, 74
- EDAM\_NOTEBOOK\_STACK\_REGEX, 74
- EDAM\_OPEN\_ID\_ACCESS\_TOKEN\_MAX, 74
- EDAM\_PREFERENCE\_BUSINESS\_DEFAULT\_NOTEBOOK, 74
- EDAM\_PREFERENCE\_BUSINESS\_QUICKNOTE, 74
- EDAM\_PREFERENCE\_NAME\_LEN\_MAX, 75
- EDAM\_PREFERENCE\_NAME\_LEN\_MIN, 75
- EDAM\_PREFERENCE\_NAME\_REGEX, 75
- EDAM\_PREFERENCE\_ONLY\_ONE\_VALUE\_LEN\_MAX, 75
- EDAM\_PREFERENCE\_ONLY\_ONE\_VALUE\_REGEX, 75
- EDAM\_PREFERENCE\_SHORTCUTS\_MAX\_VALUES, 76



- EDAM\_PREFERENCE\_SHORTCUTS, 75
- EDAM\_PREFERENCE\_VALUE\_LEN\_MAX, 76
- EDAM\_PREFERENCE\_VALUE\_LEN\_MIN, 76
- EDAM\_PREFERENCE\_VALUE\_REGEX, 76
- EDAM\_PROMOTION\_ID\_LEN\_MAX, 76
- EDAM\_PROMOTION\_ID\_REGEX, 76
- EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MAX, 76
- EDAM\_PUBLISHING\_DESCRIPTION\_LEN\_MIN, 76
- EDAM\_PUBLISHING\_DESCRIPTION\_REGEX, 77
- EDAM\_PUBLISHING\_URI\_LEN\_MAX, 77
- EDAM\_PUBLISHING\_URI\_LEN\_MIN, 77
- EDAM\_PUBLISHING\_URI\_PROHIBITED, 77
- EDAM\_PUBLISHING\_URI\_REGEX, 77
- EDAM\_RELATED\_MAX\_EXPERTS, 77
- EDAM\_RELATED\_MAX\_NOTEBOOKS, 77
- EDAM\_RELATED\_MAX\_NOTES, 78
- EDAM\_RELATED\_MAX\_RELATED\_CONTENT, 78
- EDAM\_RELATED\_MAX\_TAGS, 78
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MAX, 78
- EDAM\_RELATED\_PLAINTEXT\_LEN\_MIN, 78
- EDAM\_RESOURCE\_SIZE\_MAX\_FREE, 78
- EDAM\_RESOURCE\_SIZE\_MAX\_PREMIUM, 78
- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MAX, 78
- EDAM\_SAVED\_SEARCH\_NAME\_LEN\_MIN, 79
- EDAM\_SAVED\_SEARCH\_NAME\_REGEX, 79
- EDAM\_SEARCH\_QUERY\_LEN\_MAX, 79
- EDAM\_SEARCH\_QUERY\_LEN\_MIN, 79
- EDAM\_SEARCH\_QUERY\_REGEX, 79
- EDAM\_SEARCH\_SUGGESTIONS\_MAX, 79
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MAX, 79
- EDAM\_SEARCH\_SUGGESTIONS\_PREFIX\_LEN\_MIN, 80
- EDAM\_SNIPPETS\_NOTES\_MAX, 80
- EDAM\_SOURCE\_APPLICATION\_ANDROID\_SHARE\_EXTENSION, 80
- EDAM\_SOURCE\_APPLICATION\_EN\_SCANS\_NAP, 80
- EDAM\_SOURCE\_APPLICATION\_EWC, 80
- EDAM\_SOURCE\_APPLICATION\_IOS\_SHARE\_EXTENSION, 80
- EDAM\_SOURCE\_APPLICATION\_MOLESKINE, 80
- EDAM\_SOURCE\_APPLICATION\_POSTIT, 80
- EDAM\_SOURCE\_APPLICATION\_WEB\_CLIPPER, 81
- EDAM\_SOURCE\_OUTLOOK\_CLIPPER, 81
- EDAM\_TAG\_NAME\_LEN\_MAX, 81
- EDAM\_TAG\_NAME\_LEN\_MIN, 81
- EDAM\_TAG\_NAME\_REGEX, 81
- EDAM\_TIMEZONE\_LEN\_MAX, 81
- EDAM\_TIMEZONE\_LEN\_MIN, 81
- EDAM\_TIMEZONE\_REGEX, 82
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX\_PREMIUM, 82
- EDAM\_USER\_LINKED\_NOTEBOOK\_MAX, 82
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_FREE, 82
- EDAM\_USER\_MAIL\_LIMIT\_DAILY\_PREMIUM, 82
- EDAM\_USER\_NAME\_LEN\_MAX, 82
- EDAM\_USER\_NAME\_LEN\_MIN, 83
- EDAM\_USER\_NAME\_REGEX, 83
- EDAM\_USER\_NOTEBOOKS\_MAX, 83
- EDAM\_USER\_NOTES\_MAX, 83
- EDAM\_USER\_PASSWORD\_LEN\_MAX, 83
- EDAM\_USER\_PASSWORD\_LEN\_MIN, 83
- EDAM\_USER\_PASSWORD\_REGEX, 83
- EDAM\_USER\_PROFILE\_PHOTO\_MAX\_BYTES, 83
- EDAM\_USER\_RECENT\_MAILED\_ADDRESSES\_MAX, 84
- EDAM\_USER\_SAVED\_SEARCHES\_MAX, 84
- EDAM\_USER\_TAGS\_MAX, 84
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_FIRST\_MONTH, 84
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_NEXT\_MONTH, 84
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS\_PER\_USER, 84
- EDAM\_USER\_UPLOAD\_LIMIT\_BUSINESS, 84
- EDAM\_USER\_UPLOAD\_LIMIT\_FREE, 85
- EDAM\_USER\_UPLOAD\_LIMIT\_PLUS, 85
- EDAM\_USER\_UPLOAD\_LIMIT\_PREMIUM, 85
- EDAM\_USER\_UPLOAD\_SURVEY\_THRESHOLD, 85
- EDAM\_USER\_USERNAME\_LEN\_MAX, 85
- EDAM\_USER\_USERNAME\_LEN\_MIN, 85
- EDAM\_USER\_USERNAME\_REGEX, 85
- EDAM\_USER\_WORKSPACES\_MAX, 86
- EDAM\_VAT\_REGEX, 86
- EDAM\_VERSION\_MAJOR, 86
- EDAM\_VERSION\_MINOR, 86
- EDAM\_WORKSPACE\_DESCRIPTION\_LEN\_MAX, 86
- EDAM\_WORKSPACE\_NAME\_LEN\_MAX, 86
- EDAM\_WORKSPACE\_NAME\_LEN\_MIN, 86
- EDAM\_WORKSPACE\_NAME\_REGEX, 87
- EDAMErrorCode, 34
- EDAMInvalidContactReason, 36
- EntityType, 36
- Error, 37
- EverCloudExceptionData, 87
- EverCloudExceptionDataPtr, 30
- evernoteNetworkProxy, 43
- Guid, 30
- IDurableServicePtr, 30
- ILoggerPtr, 30
- INoteStorePtr, 30
- IRequestContextPtr, 31
- IRetryPolicyPtr, 31
- IUserStorePtr, 31

- IdentityID, 30
- Info, 37
- initializeQEverCloud, 44
- InvalidationSequenceNumber, 30
- libraryVersion, 44
- LogLevel, 37
- logger, 44
- MessageEventID, 31
- MessageThreadID, 31
- newDurableService, 44
- newNoteStore, 44
- newRequestContext, 44
- newRetryPolicy, 45
- newStdErrLogger, 45
- newUserStore, 45
- NoteSortOrder, 37
- nullLogger, 45
- nullRetryPolicy, 45
- operator<<, 45–53
- PremiumOrderStatus, 37
- PrivilegeLevel, 38
- qHash, 53–57
- QueryFormat, 38
- RecipientStatus, 39
- RelatedContentAccess, 39
- RelatedContentType, 39
- ReminderEmailConfig, 40
- resetEvernoteNetworkProxy, 57
- ServiceLevel, 40
- setEvernoteNetworkProxy, 57
- setLogger, 57
- setNonceGenerator, 57
- ShareRelationshipPrivilegeLevel, 42
- SharedNotePrivilegeLevel, 42
- SharedNotebookInstanceRestrictions, 41
- SharedNotebookPrivilegeLevel, 41
- SponsoredGroupRole, 43
- Timestamp, 31
- toRange, 58
- Trace, 37
- UserID, 31
- UserIdentityType, 43
- Warn, 37
- qevercloud::AccountLimits, 94
  - localData, 95
  - noteResourceCountMax, 95
  - noteSizeMax, 95
  - noteTagCountMax, 95
  - operator!=, 94
  - operator==, 95
  - print, 95
  - resourceSizeMax, 96
  - uploadLimit, 96
  - userLinkedNotebookMax, 96
  - userMailLimitDaily, 96
  - userNoteCountMax, 96
  - userNotebookCountMax, 96
  - userSavedSearchesMax, 96
  - userTagCountMax, 97
- qevercloud::Accounting, 89
  - availablePoints, 90
  - businessId, 91
  - businessName, 91
  - businessRole, 91
  - currency, 91
  - lastFailedCharge, 91
  - lastFailedChargeReason, 91
  - lastRequestedCharge, 91
  - lastSuccessfulCharge, 91
  - localData, 92
  - nextChargeDate, 92
  - nextPaymentDue, 92
  - operator!=, 90
  - operator==, 90
  - premiumCommerceService, 92
  - premiumLockUntil, 92
  - premiumOrderNumber, 92
  - premiumServiceSKU, 92
  - premiumServiceStart, 92
  - premiumServiceStatus, 93
  - premiumSubscriptionNumber, 93
  - print, 90
  - unitDiscount, 93
  - unitPrice, 93
  - updated, 93
  - uploadLimitEnd, 93
  - uploadLimitNextMonth, 93
- qevercloud::AsyncResult, 98
  - ~AsyncResult, 100
  - asls, 100
  - AsyncResult, 99, 100
  - DurableService, 101
  - finished, 100
  - ReadFunctionType, 99
  - waitForFinished, 101
- qevercloud::AuthenticationResult, 101
  - authenticationToken, 103
  - currentTime, 103
  - expiration, 103
  - localData, 103
  - noteStoreUrl, 103
  - operator!=, 102
  - operator==, 102
  - print, 102
  - publicUserInfo, 103
  - secondFactorDeliveryHint, 103
  - secondFactorRequired, 104
  - urls, 104
  - user, 104
  - webApiUrlPrefix, 104
- qevercloud::BootstrapInfo, 104
  - localData, 105
  - operator!=, 105
  - operator==, 105
  - print, 105
  - profiles, 106

- qevercloud::BootstrapProfile, 106
  - localData, 107
  - name, 107
  - operator!=, 106
  - operator==, 107
  - print, 107
  - settings, 107
- qevercloud::BootstrapSettings, 108
  - accountEmailDomain, 109
  - enableFacebookSharing, 109
  - enableGiftSubscriptions, 109
  - enableGoogle, 109
  - enableLinkedInSharing, 109
  - enablePublicNotebooks, 110
  - enableSharedNotebooks, 110
  - enableSingleNoteSharing, 110
  - enableSponsoredAccounts, 110
  - enableSupportTickets, 110
  - enableTwitterSharing, 110
  - localData, 110
  - marketingUrl, 110
  - operator!=, 108
  - operator==, 108
  - print, 109
  - serviceHost, 111
  - supportUrl, 111
- qevercloud::BusinessInvitation, 111
  - businessId, 112
  - created, 112
  - email, 112
  - fromWorkChat, 113
  - localData, 113
  - mostRecentReminder, 113
  - operator!=, 112
  - operator==, 112
  - print, 112
  - requesterId, 113
  - role, 113
  - status, 113
- qevercloud::BusinessNotebook, 114
  - localData, 115
  - notebookDescription, 115
  - operator!=, 114
  - operator==, 114
  - print, 114
  - privilege, 115
  - recommended, 115
- qevercloud::BusinessUserAttributes, 115
  - companyStartDate, 117
  - department, 117
  - linkedinProfileUrl, 117
  - localData, 117
  - location, 117
  - mobilePhone, 117
  - operator!=, 116
  - operator==, 116
  - print, 116
  - title, 117
  - workPhone, 117
- qevercloud::BusinessUserInfo, 118
  - businessId, 119
  - businessName, 119
  - email, 119
  - localData, 119
  - operator!=, 118
  - operator==, 118
  - print, 119
  - role, 119
  - updated, 120
- qevercloud::CanMoveToContainerRestrictions, 120
  - canMoveToContainer, 121
  - localData, 121
  - operator!=, 120
  - operator==, 121
  - print, 121
- qevercloud::Contact, 121
  - id, 123
  - localData, 123
  - messagingPermit, 123
  - messagingPermitExpires, 123
  - name, 123
  - operator!=, 122
  - operator==, 122
  - photoLastUpdated, 123
  - photoUrl, 123
  - print, 122
  - type, 124
- qevercloud::CreateOrUpdateNotebookSharesResult, 124
  - localData, 125
  - matchingShares, 125
  - operator!=, 125
  - operator==, 125
  - print, 125
  - QList, 126
  - updateSequenceNum, 125
- qevercloud::Data, 126
  - body, 127
  - bodyHash, 127
  - localData, 127
  - operator!=, 127
  - operator==, 127
  - print, 127
  - size, 128
- qevercloud::EDAMInvalidContactsException, 128
  - ~EDAMInvalidContactsException, 129
  - contacts, 130
  - EDAMInvalidContactsException, 129
  - exceptionData, 129
  - operator!=, 130
  - operator==, 130
  - parameter, 130
  - print, 130
  - QList, 131
  - reasons, 131
  - what, 130

- qevercloud::EDAMInvalidContactsExceptionData, 131
  - EDAMInvalidContactsExceptionData, 132
  - m\_contacts, 132
  - m\_parameter, 132
  - m\_reasons, 132
  - throwException, 132
- qevercloud::EDAMNotFoundException, 133
  - ~EDAMNotFoundException, 134
  - EDAMNotFoundException, 134
  - exceptionData, 134
  - identifier, 135
  - key, 135
  - operator!=, 134
  - operator==, 134
  - print, 135
  - what, 135
- qevercloud::EDAMNotFoundExceptionData, 135
  - EDAMNotFoundExceptionData, 136
  - m\_identifier, 136
  - m\_key, 137
  - throwException, 136
- qevercloud::EDAMSystemException, 137
  - ~EDAMSystemException, 138
  - EDAMSystemException, 138
  - errorCode, 139
  - exceptionData, 138
  - message, 140
  - operator!=, 139
  - operator==, 139
  - print, 139
  - rateLimitDuration, 140
  - what, 139
- qevercloud::EDAMSystemExceptionAuthExpired, 140
  - exceptionData, 141
- qevercloud::EDAMSystemExceptionAuthExpiredData, 141
  - EDAMSystemExceptionAuthExpiredData, 141
  - throwException, 142
- qevercloud::EDAMSystemExceptionData, 142
  - EDAMSystemExceptionData, 143
  - m\_errorCode, 143
  - m\_message, 143
  - m\_rateLimitDuration, 144
  - throwException, 143
- qevercloud::EDAMSystemExceptionRateLimitReached, 144
  - exceptionData, 144
- qevercloud::EDAMSystemExceptionRateLimitReachedData, 145
  - EDAMSystemExceptionRateLimitReachedData, 145
  - throwException, 145
- qevercloud::EDAMUserException, 146
  - ~EDAMUserException, 147
  - EDAMUserException, 147
  - errorCode, 148
  - exceptionData, 147
  - operator!=, 148
  - operator==, 148
  - parameter, 148
  - print, 148
  - what, 148
- qevercloud::EDAMUserExceptionData, 149
  - EDAMUserExceptionData, 149
  - m\_errorCode, 150
  - m\_parameter, 150
  - throwException, 150
- qevercloud::EventLoopFinisher, 150
  - ~EventLoopFinisher, 151
  - EventLoopFinisher, 151
  - stopEventLoop, 151
- qevercloud::EverCloudException, 151
  - ~EverCloudException, 152
  - EverCloudException, 152
  - exceptionData, 153
  - m\_error, 153
  - what, 153
- qevercloud::EverCloudExceptionData, 153
  - errorMessage, 155
  - EverCloudExceptionData, 155
  - throwException, 155
- qevercloud::EverCloudLocalData, 156
  - ~EverCloudLocalData, 157
  - Dict, 157
  - dict, 158, 159
  - dirty, 158
  - EverCloudLocalData, 157
  - favorited, 158
  - id, 158
  - local, 159
  - operator!=, 157
  - operator==, 157
  - print, 158
- qevercloud::EvernoteException, 159
  - EvernoteException, 160
  - exceptionData, 160
- qevercloud::EvernoteExceptionData, 161
  - EvernoteExceptionData, 161
  - throwException, 161
- qevercloud::EvernoteOAuthDialog, 162
  - ~EvernoteOAuthDialog, 164
  - EvernoteOAuthDialog, 163
  - exec, 164
  - isSucceeded, 164
  - OAuthResult, 163
  - oauthError, 164
  - oauthResult, 164
  - open, 165
  - setWebViewSizeHint, 165
- qevercloud::EvernoteOAuthWebView, 165
  - authenticate, 166
  - authenticationFailed, 167
  - authenticationFinished, 167
  - authenticationSucceeded, 167
  - EvernoteOAuthWebView, 166
  - isSucceeded, 167

- oauthError, 167
- oauthResult, 167
- setSizeHint, 168
- sizeHint, 168
- qevercloud::EvernoteOAuthWebView::OAuthResult, 399
  - authenticationToken, 400
  - cookies, 400
  - expires, 400
  - noteStoreUrl, 400
  - print, 399
  - shardId, 400
  - userId, 400
  - webApiUrlPrefix, 401
- qevercloud::IDurableService, 171
  - AsyncServiceCall, 171
  - executeAsyncRequest, 172
  - executeSyncRequest, 172
  - SyncResult, 171
  - SyncServiceCall, 171
- qevercloud::IDurableService::AsyncRequest, 97
  - AsyncRequest, 97
  - m\_call, 97
  - m\_description, 98
  - m\_name, 98
- qevercloud::IDurableService::SyncRequest, 468
  - m\_call, 468
  - m\_description, 468
  - m\_name, 468
  - SyncRequest, 468
- qevercloud::ILogger, 172
  - level, 172
  - log, 172
  - setLevel, 173
  - shouldLog, 173
- qevercloud::INoteStore, 177
  - authenticateToSharedNote, 181
  - authenticateToSharedNoteAsync, 182
  - authenticateToSharedNotebook, 182
  - authenticateToSharedNotebookAsync, 183
  - copyNote, 183
  - copyNoteAsync, 184
  - createLinkedNotebook, 184
  - createLinkedNotebookAsync, 185
  - createNote, 185
  - createNoteAsync, 186
  - createNotebook, 187
  - createNotebookAsync, 188
  - createOrUpdateNotebookShares, 188
  - createOrUpdateNotebookSharesAsync, 189
  - createSearch, 189
  - createSearchAsync, 190
  - createTag, 190
  - createTagAsync, 191
  - deleteNote, 191
  - deleteNoteAsync, 192
  - emailNote, 192
  - emailNoteAsync, 193
  - expungeLinkedNotebook, 193
  - expungeLinkedNotebookAsync, 193
  - expungeNote, 194
  - expungeNoteAsync, 194
  - expungeNotebook, 194
  - expungeNotebookAsync, 195
  - expungeSearch, 195
  - expungeSearchAsync, 196
  - expungeTag, 196
  - expungeTagAsync, 197
  - findNoteCounts, 197
  - findNoteCountsAsync, 198
  - findNoteOffset, 198
  - findNoteOffsetAsync, 199
  - findNotesMetadata, 199
  - findNotesMetadataAsync, 200
  - findRelated, 200
  - findRelatedAsync, 201
  - getDefaultNotebook, 201
  - getDefaultNotebookAsync, 202
  - getFilteredSyncChunk, 202
  - getFilteredSyncChunkAsync, 202
  - getLinkedNotebookSyncChunk, 203
  - getLinkedNotebookSyncChunkAsync, 204
  - getLinkedNotebookSyncState, 204
  - getLinkedNotebookSyncStateAsync, 205
  - getNote, 205
  - getNoteApplicationData, 205
  - getNoteApplicationDataAsync, 205
  - getNoteApplicationDataEntry, 206
  - getNoteApplicationDataEntryAsync, 206
  - getNoteAsync, 206
  - getNoteContent, 208
  - getNoteContentAsync, 209
  - getNoteSearchText, 209
  - getNoteSearchTextAsync, 210
  - getNoteTagNames, 210
  - getNoteTagNamesAsync, 210
  - getNoteVersion, 211
  - getNoteVersionAsync, 211
  - getNoteWithResultSpec, 212
  - getNoteWithResultSpecAsync, 212
  - getNotebook, 206
  - getNotebookAsync, 208
  - getNotebookShares, 208
  - getNotebookSharesAsync, 208
  - getPublicNotebook, 212
  - getPublicNotebookAsync, 213
  - getResource, 213
  - getResourceAlternateData, 214
  - getResourceAlternateDataAsync, 214
  - getResourceApplicationData, 215
  - getResourceApplicationDataAsync, 215
  - getResourceApplicationDataEntry, 215
  - getResourceApplicationDataEntryAsync, 215
  - getResourceAsync, 216
  - getResourceAttributes, 216
  - getResourceAttributesAsync, 216

- getResourceByHash, 216
- getResourceByHashAsync, 217
- getResourceData, 217
- getResourceDataAsync, 218
- getResourceRecognition, 218
- getResourceRecognitionAsync, 219
- getResourceSearchText, 219
- getResourceSearchTextAsync, 220
- getSearch, 220
- getSearchAsync, 220
- getSharedNotebookByAuth, 220
- getSharedNotebookByAuthAsync, 221
- getSyncState, 221
- getSyncStateAsync, 221
- getTag, 221
- getTagAsync, 222
- INoteStore, 181
- listAccessibleBusinessNotebooks, 222
- listAccessibleBusinessNotebooksAsync, 222
- listLinkedNotebooks, 223
- listLinkedNotebooksAsync, 223
- listNoteVersions, 223
- listNoteVersionsAsync, 224
- listNotebooks, 223
- listNotebooksAsync, 223
- listSearches, 224
- listSearchesAsync, 224
- listSharedNotebooks, 224
- listSharedNotebooksAsync, 224
- listTags, 225
- listTagsAsync, 225
- listTagsByNotebook, 225
- listTagsByNotebookAsync, 225
- manageNotebookShares, 225
- manageNotebookSharesAsync, 226
- noteStoreUrl, 226
- setNoteApplicationDataEntry, 226
- setNoteApplicationDataEntryAsync, 226
- setNoteStoreUrl, 228
- setNotebookRecipientSettings, 227
- setNotebookRecipientSettingsAsync, 228
- setResourceApplicationDataEntry, 228
- setResourceApplicationDataEntryAsync, 228
- shareNote, 228
- shareNoteAsync, 229
- shareNotebook, 229
- shareNotebookAsync, 231
- stopSharingNote, 231
- stopSharingNoteAsync, 232
- unsetNoteApplicationDataEntry, 232
- unsetNoteApplicationDataEntryAsync, 232
- unsetResourceApplicationDataEntry, 232
- unsetResourceApplicationDataEntryAsync, 232
- untagAll, 233
- untagAllAsync, 233
- updateLinkedNotebook, 233
- updateLinkedNotebookAsync, 234
- updateNote, 234
- updateNoteAsync, 235
- updateNoteIfUsnMatches, 237
- updateNoteIfUsnMatchesAsync, 237
- updateNotebook, 235
- updateNotebookAsync, 236
- updateResource, 237
- updateResourceAsync, 238
- updateSearch, 238
- updateSearchAsync, 239
- updateSharedNotebook, 239
- updateSharedNotebookAsync, 239
- updateTag, 239
- updateTagAsync, 241
- qevercloud::IRequestContext, 243
  - ~IRequestContext, 244
  - authenticationToken, 244
  - clone, 244
  - cookies, 244
  - increaseRequestTimeoutExponentially, 244
  - maxRequestRetryCount, 244
  - maxRequestTimeout, 245
  - operator<=, 245
  - requestId, 245
  - requestTimeout, 245
- qevercloud::IRetryPolicy, 245
  - shouldRetry, 246
- qevercloud::IUserStore, 249
  - authenticateLongSession, 250
  - authenticateLongSessionAsync, 252
  - authenticateToBusiness, 252
  - authenticateToBusinessAsync, 253
  - checkVersion, 253
  - checkVersionAsync, 254
  - completeTwoFactorAuthentication, 254
  - completeTwoFactorAuthenticationAsync, 255
  - getAccountLimits, 255
  - getAccountLimitsAsync, 256
  - getBootstrapInfo, 256
  - getBootstrapInfoAsync, 256
  - getPublicUserInfo, 256
  - getPublicUserInfoAsync, 257
  - getUser, 257
  - getUserAsync, 257
  - getUserUrls, 257
  - getUserUrlsAsync, 257
  - IUserStore, 250
  - inviteToBusiness, 257
  - inviteToBusinessAsync, 258
  - listBusinessInvitations, 258
  - listBusinessInvitationsAsync, 259
  - listBusinessUsers, 259
  - listBusinessUsersAsync, 260
  - removeFromBusiness, 260
  - removeFromBusinessAsync, 260
  - revokeLongSession, 260
  - revokeLongSessionAsync, 261
  - setUserStoreUrl, 261
  - updateBusinessUserIdentifier, 261



- updateBusinessUserIdentifierAsync, 262
- userStoreUrl, 262
- qevercloud::Identity, 168
  - blocked, 169
  - contact, 169
  - deactivated, 169
  - eventId, 170
  - id, 170
  - localData, 170
  - operator!=, 169
  - operator==, 169
  - print, 169
  - sameBusiness, 170
  - userConnected, 170
  - userId, 170
- qevercloud::InkNoteImageDownloader, 173
  - ~InkNoteImageDownloader, 174
  - download, 175
  - InkNoteImageDownloader, 174
  - setAuthenticationToken, 175
  - setHeight, 176
  - setHost, 176
  - setShardId, 176
  - setWidth, 176
- qevercloud::InvitationShareRelationship, 241
  - displayName, 242
  - localData, 243
  - operator!=, 242
  - operator==, 242
  - print, 242
  - privilege, 243
  - recipientUserIdentity, 243
  - sharerUserId, 243
- qevercloud::LazyMap, 263
  - FullMap, 264
  - fullMap, 264, 265
  - keysOnly, 264
  - localData, 265
  - operator!=, 264
  - operator==, 264
  - print, 264
  - QSet, 265
- qevercloud::LinkedNotebook, 265
  - businessId, 266
  - guid, 267
  - localData, 267
  - noteStoreUrl, 267
  - operator!=, 266
  - operator==, 266
  - print, 266
  - shardId, 267
  - shareName, 267
  - sharedNotebookGlobalId, 267
  - stack, 268
  - updateSequenceNum, 268
  - uri, 268
  - username, 268
  - webApiUrlPrefix, 268
- qevercloud::ManageNoteSharesError, 275
  - identityID, 276
  - localData, 276
  - notFoundException, 276
  - operator!=, 276
  - operator==, 276
  - print, 276
  - userException, 276
  - userId, 277
- qevercloud::ManageNoteSharesParameters, 277
  - invitationsToUnshare, 278
  - invitationsToUpdate, 278
  - localData, 279
  - membershipsToUnshare, 279
  - membershipsToUpdate, 279
  - noteGuid, 279
  - operator!=, 278
  - operator==, 278
  - print, 278
  - QList, 279
- qevercloud::ManageNoteSharesResult, 280
  - errors, 281
  - localData, 281
  - operator!=, 280
  - operator==, 280
  - print, 281
  - QList, 281
- qevercloud::ManageNotebookSharesError, 269
  - localData, 270
  - notFoundException, 270
  - operator!=, 269
  - operator==, 269
  - print, 270
  - userException, 270
  - userIdentity, 270
- qevercloud::ManageNotebookSharesParameters, 271
  - invitationsToCreateOrUpdate, 272
  - inviteMessage, 272
  - localData, 272
  - membershipsToUpdate, 272
  - notebookGuid, 272
  - operator!=, 271
  - operator==, 271
  - print, 272
  - QList, 273
  - unshares, 273
- qevercloud::ManageNotebookSharesResult, 273
  - errors, 274
  - localData, 274
  - operator!=, 274
  - operator==, 274
  - print, 274
  - QList, 275
- qevercloud::MemberShareRelationship, 282
  - bestPrivilege, 283
  - displayName, 283
  - individualPrivilege, 283
  - localData, 283

- operator!=, 282
- operator==, 282
- print, 282
- recipientUserId, 283
- restrictions, 283
- sharerUserId, 284
- qevercloud::NetworkException, 284
  - ~NetworkException, 285
  - exceptionData, 285
  - m\_type, 286
  - NetworkException, 285
  - operator!=, 285
  - operator==, 286
  - type, 286
  - what, 286
- qevercloud::NetworkExceptionData, 286
  - m\_type, 287
  - NetworkExceptionData, 287
  - throwException, 287
- qevercloud::Note, 288
  - active, 289
  - attributes, 289
  - content, 289
  - contentHash, 290
  - contentLength, 290
  - created, 290
  - deleted, 290
  - guid, 290
  - limits, 290
  - localData, 291
  - notebookGuid, 291
  - operator!=, 289
  - operator==, 289
  - print, 289
  - QList, 292
  - resources, 291
  - restrictions, 291
  - sharedNotes, 291
  - tagGuids, 291
  - tagNames, 292
  - title, 292
  - updateSequenceNum, 292
  - updated, 292
- qevercloud::NoteAttributes, 293
  - altitude, 294
  - applicationData, 295
  - author, 295
  - Classifications, 294
  - classifications, 295, 299
  - conflictSourceNoteGuid, 295
  - contentClass, 295
  - creatorId, 296
  - lastEditedBy, 296
  - lastEditorId, 296
  - latitude, 296
  - localData, 296
  - longitude, 297
  - noteTitleQuality, 297
  - operator!=, 294
  - operator==, 294
  - placeName, 297
  - print, 294
  - reminderDoneTime, 297
  - reminderOrder, 297
  - reminderTime, 298
  - shareDate, 298
  - sharedWithBusiness, 298
  - source, 298
  - sourceApplication, 298
  - sourceURL, 299
  - subjectDate, 299
- qevercloud::NoteCollectionCounts, 317
  - localData, 319
  - notebookCounts, 319
  - operator!=, 318
  - operator==, 318
  - print, 318
  - TagCounts, 318
  - tagCounts, 319
  - trashCount, 319
- qevercloud::NoteEmailParameters, 320
  - ccAddresses, 321
  - guid, 321
  - localData, 321
  - message, 321
  - note, 321
  - operator!=, 320
  - operator==, 320
  - print, 321
  - subject, 322
  - toAddresses, 322
- qevercloud::NoteFilter, 322
  - ascending, 324
  - context, 324
  - emphasized, 324
  - inactive, 324
  - includeAllReadableNotebooks, 324
  - includeAllReadableWorkspaces, 324
  - localData, 325
  - notebookGuid, 325
  - operator!=, 323
  - operator==, 323
  - order, 325
  - print, 323
  - QList, 326
  - rawWords, 325
  - searchContextBytes, 325
  - tagGuids, 325
  - timeZone, 325
  - words, 326
- qevercloud::NoteInvitationShareRelationship, 326
  - displayName, 327
  - localData, 327
  - operator!=, 327
  - operator==, 327
  - print, 327



- privilege, 327
- recipientIdentityId, 328
- sharerUserId, 328
- qevercloud::NoteLimits, 328
  - localData, 329
  - noteResourceCountMax, 329
  - noteSizeMax, 330
  - operator!=, 329
  - operator==, 329
  - print, 329
  - resourceSizeMax, 330
  - uploadLimit, 330
  - uploaded, 330
- qevercloud::NoteList, 330
  - debugInfo, 331
  - localData, 332
  - notes, 332
  - operator!=, 331
  - operator==, 331
  - print, 331
  - searchContextBytes, 332
  - searchedWords, 332
  - startIndex, 332
  - stoppedWords, 332
  - totalNotes, 332
  - updateCount, 333
- qevercloud::NoteMemberShareRelationship, 333
  - displayName, 334
  - localData, 334
  - operator!=, 334
  - operator==, 334
  - print, 334
  - privilege, 334
  - recipientUserId, 334
  - restrictions, 335
  - sharerUserId, 335
- qevercloud::NoteMetadata, 335
  - attributes, 336
  - contentLength, 336
  - created, 337
  - deleted, 337
  - guid, 337
  - largestResourceMime, 337
  - largestResourceSize, 337
  - localData, 337
  - notebookGuid, 337
  - operator!=, 336
  - operator==, 336
  - print, 336
  - QList, 338
  - tagGuids, 338
  - title, 338
  - updateSequenceNum, 338
  - updated, 338
- qevercloud::NoteRestrictions, 338
  - localData, 340
  - noEmail, 340
  - noShare, 340
  - noSharePublicly, 340
  - noUpdateContent, 341
  - noUpdateTitle, 341
  - operator!=, 339
  - operator==, 340
  - print, 340
- qevercloud::NoteResultSpec, 341
  - includeAccountLimits, 342
  - includeContent, 342
  - includeNoteAppDataValues, 343
  - includeResourceAppDataValues, 343
  - includeResourcesAlternateData, 343
  - includeResourcesData, 343
  - includeResourcesRecognition, 343
  - includeSharedNotes, 343
  - localData, 343
  - operator!=, 342
  - operator==, 342
  - print, 342
- qevercloud::NoteShareRelationshipRestrictions, 344
  - localData, 345
  - noSetFullAccess, 345
  - noSetModifyNote, 345
  - noSetReadNote, 345
  - operator!=, 344
  - operator==, 344
  - print, 345
- qevercloud::NoteShareRelationships, 346
  - invitationRestrictions, 347
  - invitations, 347
  - localData, 347
  - memberships, 347
  - operator!=, 346
  - operator==, 346
  - print, 347
  - QList, 348
- qevercloud::NoteStoreServer, 354
  - authenticateToSharedNoteRequest, 360
  - authenticateToSharedNoteRequestReady, 360
  - authenticateToSharedNotebookRequest, 359
  - authenticateToSharedNotebookRequestReady, 360
  - copyNoteRequest, 360
  - copyNoteRequestReady, 360
  - createLinkedNotebookRequest, 360
  - createLinkedNotebookRequestReady, 361
  - createNoteRequest, 361
  - createNoteRequestReady, 361
  - createNotebookRequest, 361
  - createNotebookRequestReady, 361
  - createOrUpdateNotebookSharesRequest, 361
  - createOrUpdateNotebookSharesRequestReady, 362
  - createSearchRequest, 362
  - createSearchRequestReady, 362
  - createTagRequest, 362
  - createTagRequestReady, 362
  - deleteNoteRequest, 362

- deleteNoteRequestReady, 363
- emailNoteRequest, 363
- emailNoteRequestReady, 363
- expungeLinkedNotebookRequest, 363
- expungeLinkedNotebookRequestReady, 363
- expungeNoteRequest, 364
- expungeNoteRequestReady, 364
- expungeNotebookRequest, 363
- expungeNotebookRequestReady, 364
- expungeSearchRequest, 364
- expungeSearchRequestReady, 364
- expungeTagRequest, 364
- expungeTagRequestReady, 365
- findNoteCountsRequest, 365
- findNoteCountsRequestReady, 365
- findNoteOffsetRequest, 365
- findNoteOffsetRequestReady, 365
- findNotesMetadataRequest, 365
- findNotesMetadataRequestReady, 366
- findRelatedRequest, 366
- findRelatedRequestReady, 366
- getDefaultNotebookRequest, 366
- getDefaultNotebookRequestReady, 366
- getFilteredSyncChunkRequest, 366
- getFilteredSyncChunkRequestReady, 367
- getLinkedNotebookSyncChunkRequest, 367
- getLinkedNotebookSyncChunkRequestReady, 367
- getLinkedNotebookSyncStateRequest, 367
- getLinkedNotebookSyncStateRequestReady, 367
- getNoteApplicationDataEntryRequest, 367
- getNoteApplicationDataEntryRequestReady, 368
- getNoteApplicationDataRequest, 368
- getNoteApplicationDataRequestReady, 368
- getNoteContentRequest, 369
- getNoteContentRequestReady, 369
- getNoteRequest, 369
- getNoteRequestReady, 369
- getNoteSearchTextRequest, 369
- getNoteSearchTextRequestReady, 370
- getNoteTagNamesRequest, 370
- getNoteTagNamesRequestReady, 370
- getNoteVersionRequest, 370
- getNoteVersionRequestReady, 370
- getNoteWithResultSpecRequest, 370
- getNoteWithResultSpecRequestReady, 371
- getNotebookRequest, 368
- getNotebookRequestReady, 368
- getNotebookSharesRequest, 368
- getNotebookSharesRequestReady, 369
- getPublicNotebookRequest, 371
- getPublicNotebookRequestReady, 371
- getResourceAlternateDataRequest, 371
- getResourceAlternateDataRequestReady, 371
- getResourceApplicationDataEntryRequest, 371
- getResourceApplicationDataEntryRequestReady, 372
- getResourceApplicationDataRequest, 372
- getResourceApplicationDataRequestReady, 372
- getResourceAttributesRequest, 372
- getResourceAttributesRequestReady, 372
- getResourceByHashRequest, 372
- getResourceByHashRequestReady, 373
- getResourceDataRequest, 373
- getResourceDataRequestReady, 373
- getResourceRecognitionRequest, 373
- getResourceRecognitionRequestReady, 373
- getResourceRequest, 373
- getResourceRequestReady, 374
- getResourceSearchTextRequest, 374
- getResourceSearchTextRequestReady, 374
- getSearchRequest, 374
- getSearchRequestReady, 374
- getSharedNotebookByAuthRequest, 374
- getSharedNotebookByAuthRequestReady, 375
- getSyncStateRequest, 375
- getSyncStateRequestReady, 375
- getTagRequest, 375
- getTagRequestReady, 375
- listAccessibleBusinessNotebooksRequest, 375
- listAccessibleBusinessNotebooksRequestReady, 375
- listLinkedNotebooksRequest, 376
- listLinkedNotebooksRequestReady, 376
- listNoteVersionsRequest, 376
- listNoteVersionsRequestReady, 376
- listNotebooksRequest, 376
- listNotebooksRequestReady, 376
- listSearchesRequest, 376
- listSearchesRequestReady, 377
- listSharedNotebooksRequest, 377
- listSharedNotebooksRequestReady, 377
- listTagsByNotebookRequest, 377
- listTagsByNotebookRequestReady, 377
- listTagsRequest, 377
- listTagsRequestReady, 377
- manageNotebookSharesRequest, 378
- manageNotebookSharesRequestReady, 378
- NoteStoreServer, 359
- onAuthenticateToSharedNoteRequestReady, 378
- onAuthenticateToSharedNotebookRequestReady, 378
- onCopyNoteRequestReady, 378
- onCreateLinkedNotebookRequestReady, 378
- onCreateNoteRequestReady, 379
- onCreateNotebookRequestReady, 379
- onCreateOrUpdateNotebookSharesRequestReady, 379
- onCreateSearchRequestReady, 379
- onCreateTagRequestReady, 379
- onDeleteNoteRequestReady, 379
- onEmailNoteRequestReady, 380
- onExpungeLinkedNotebookRequestReady, 380
- onExpungeNoteRequestReady, 380
- onExpungeNotebookRequestReady, 380
- onExpungeSearchRequestReady, 380
- onExpungeTagRequestReady, 380

- onFindNoteCountsRequestReady, 381
- onFindNoteOffsetRequestReady, 381
- onFindNotesMetadataRequestReady, 381
- onFindRelatedRequestReady, 381
- onGetDefaultNotebookRequestReady, 381
- onGetFilteredSyncChunkRequestReady, 381
- onGetLinkedNotebookSyncChunkRequestReady, 382
- onGetLinkedNotebookSyncStateRequestReady, 382
- onGetNoteApplicationDataEntryRequestReady, 382
- onGetNoteApplicationDataRequestReady, 382
- onGetNoteContentRequestReady, 383
- onGetNoteRequestReady, 383
- onGetNoteSearchTextRequestReady, 383
- onGetNoteTagNameRequestReady, 383
- onGetNoteVersionRequestReady, 383
- onGetNoteWithResultSpecRequestReady, 383
- onGetNotebookRequestReady, 382
- onGetNotebookSharesRequestReady, 382
- onGetPublicNotebookRequestReady, 384
- onGetResourceAlternateDataRequestReady, 384
- onGetResourceApplicationDataEntryRequestReady, 384
- onGetResourceApplicationDataRequestReady, 384
- onGetResourceAttributesRequestReady, 384
- onGetResourceByHashRequestReady, 384
- onGetResourceDataRequestReady, 385
- onGetResourceRecognitionRequestReady, 385
- onGetResourceRequestReady, 385
- onGetResourceSearchTextRequestReady, 385
- onGetSearchRequestReady, 385
- onGetSharedNotebookByAuthRequestReady, 385
- onGetSyncStateRequestReady, 386
- onGetTagRequestReady, 386
- onListAccessibleBusinessNotebooksRequestReady, 386
- onListLinkedNotebooksRequestReady, 386
- onListNoteVersionsRequestReady, 386
- onListNotebooksRequestReady, 386
- onListSearchesRequestReady, 387
- onListSharedNotebooksRequestReady, 387
- onListTagsByNotebookRequestReady, 387
- onListTagsRequestReady, 387
- onManageNotebookSharesRequestReady, 387
- onRequest, 387
- onSetNoteApplicationDataEntryRequestReady, 388
- onSetNotebookRecipientSettingsRequestReady, 388
- onSetResourceApplicationDataEntryRequestReady, 388
- onShareNoteRequestReady, 388
- onShareNotebookRequestReady, 388
- onStopSharingNoteRequestReady, 388
- onUnsetNoteApplicationDataEntryRequestReady, 389
- onUnsetResourceApplicationDataEntryRequestReady, 389
- onUntagAllRequestReady, 389
- onUpdateLinkedNotebookRequestReady, 389
- onUpdateNoteIfUsnMatchesRequestReady, 389
- onUpdateNoteRequestReady, 390
- onUpdateNotebookRequestReady, 389
- onUpdateResourceRequestReady, 390
- onUpdateSearchRequestReady, 390
- onUpdateSharedNotebookRequestReady, 390
- onUpdateTagRequestReady, 390
- setNoteApplicationDataEntryRequest, 390
- setNoteApplicationDataEntryRequestReady, 391
- setNotebookRecipientSettingsRequest, 391
- setNotebookRecipientSettingsRequestReady, 391
- setResourceApplicationDataEntryRequest, 391
- setResourceApplicationDataEntryRequestReady, 391
- shareNoteRequest, 392
- shareNoteRequestReady, 392
- shareNotebookRequest, 391
- shareNotebookRequestReady, 392
- stopSharingNoteRequest, 392
- stopSharingNoteRequestReady, 392
- unsetNoteApplicationDataEntryRequest, 392
- unsetNoteApplicationDataEntryRequestReady, 393
- unsetResourceApplicationDataEntryRequest, 393
- unsetResourceApplicationDataEntryRequestReady, 393
- untagAllRequest, 393
- untagAllRequestReady, 393
- updateLinkedNotebookRequest, 393
- updateLinkedNotebookRequestReady, 394
- updateNoteIfUsnMatchesRequest, 394
- updateNoteIfUsnMatchesRequestReady, 394
- updateNoteRequest, 394
- updateNoteRequestReady, 395
- updateNotebookRequest, 394
- updateNotebookRequestReady, 394
- updateResourceRequest, 395
- updateResourceRequestReady, 395
- updateSearchRequest, 395
- updateSearchRequestReady, 395
- updateSharedNotebookRequest, 395
- updateSharedNotebookRequestReady, 396
- updateTagRequest, 396
- updateTagRequestReady, 396
- qevercloud::NoteVersionId, 396
- lastEditorId, 397
- localData, 398
- operator!=, 397
- operator==, 397
- print, 397
- saved, 398
- title, 398

- updateSequenceNum, 398
- updated, 398
- qevercloud::Notebook, 299
  - businessNotebook, 301
  - contact, 301
  - defaultNotebook, 301
  - guid, 301
  - localData, 301
  - name, 302
  - operator!=, 300
  - operator==, 300
  - print, 300
  - published, 302
  - publishing, 302
  - QList, 304
  - recipientSettings, 302
  - restrictions, 302
  - serviceCreated, 302
  - serviceUpdated, 303
  - sharedNotebookIds, 303
  - sharedNotebooks, 303
  - stack, 303
  - updateSequenceNum, 303
- qevercloud::NotebookDescriptor, 304
  - contactName, 305
  - guid, 305
  - hasSharedNotebook, 305
  - joinedUserCount, 305
  - localData, 306
  - notebookDisplayName, 306
  - operator!=, 304
  - operator==, 305
  - print, 305
- qevercloud::NotebookRecipientSettings, 306
  - inMyList, 307
  - localData, 307
  - operator!=, 307
  - operator==, 307
  - print, 307
  - recipientStatus, 308
  - reminderNotifyEmail, 308
  - reminderNotifyInApp, 308
  - stack, 308
- qevercloud::NotebookRestrictions, 308
  - canMoveToContainerRestrictions, 310
  - expungeWhichSharedNotebookRestrictions, 310
  - localData, 310
  - noCanMoveNote, 311
  - noChangeContact, 311
  - noCreateNotes, 311
  - noCreateSharedNotebooks, 311
  - noCreateTags, 311
  - noEmailNotes, 311
  - noExpungeNotebook, 311
  - noExpungeNotes, 312
  - noExpungeTags, 312
  - noPublishToBusinessLibrary, 312
  - noPublishToPublic, 312
  - noReadNotes, 312
  - noRenameNotebook, 312
  - noSendMessageToRecipients, 312
  - noSetDefaultNotebook, 313
  - noSetInMyList, 313
  - noSetNotebookStack, 313
  - noSetParentTag, 313
  - noSetRecipientSettingsStack, 313
  - noSetReminderNotifyEmail, 313
  - noSetReminderNotifyInApp, 313
  - noShareNotes, 314
  - noShareNotesWithBusiness, 314
  - noUpdateNotebook, 314
  - noUpdateNotes, 314
  - noUpdateTags, 314
  - operator!=, 310
  - operator==, 310
  - print, 310
  - updateWhichSharedNotebookRestrictions, 314
- qevercloud::NotebookShareTemplate, 315
  - localData, 316
  - notebookGuid, 316
  - operator!=, 315
  - operator==, 315
  - print, 316
  - privilege, 316
  - QList, 317
  - recipientContacts, 316
  - recipientThreadId, 316
- qevercloud::NotesMetadataList, 348
  - debugInfo, 349
  - localData, 349
  - notes, 349
  - operator!=, 348
  - operator==, 349
  - print, 349
  - searchContextBytes, 349
  - searchedWords, 350
  - startIndex, 350
  - stoppedWords, 350
  - totalNotes, 350
  - updateCount, 350
- qevercloud::NotesMetadataResultSpec, 351
  - includeAttributes, 352
  - includeContentLength, 352
  - includeCreated, 352
  - includeDeleted, 352
  - includeLargestResourceMime, 353
  - includeLargestResourceSize, 353
  - includeNotebookGuid, 353
  - includeTagGuids, 353
  - includeTitle, 353
  - includeUpdateSequenceNum, 353
  - includeUpdated, 353
  - localData, 353
  - operator!=, 351
  - operator==, 352
  - print, 352

- qevercloud::Optional
  - clear, [403](#)
  - init, [404](#)
  - isEqual, [404](#)
  - isSet, [404](#)
  - operator const T &, [405](#)
  - operator T &, [405](#)
  - operator!=, [405](#)
  - operator->, [405](#), [406](#)
  - operator=, [406](#), [407](#)
  - operator==, [407](#)
  - Optional, [402](#), [403](#), [409](#)
  - ref, [407](#), [408](#)
  - swap, [409](#)
  - value, [408](#)
- qevercloud::Optional< T >, [401](#)
- qevercloud::Printable, [409](#)
  - ~Printable, [410](#)
  - operator<<, [411](#), [412](#)
  - print, [411](#)
  - Printable, [410](#)
  - toString, [411](#)
- qevercloud::PublicUserInfo, [412](#)
  - localData, [413](#)
  - noteStoreUrl, [413](#)
  - operator!=, [413](#)
  - operator==, [413](#)
  - print, [413](#)
  - serviceLevel, [413](#)
  - userId, [414](#)
  - username, [414](#)
  - webApiUrlPrefix, [414](#)
- qevercloud::Publishing, [414](#)
  - ascending, [415](#)
  - localData, [415](#)
  - operator!=, [415](#)
  - operator==, [415](#)
  - order, [416](#)
  - print, [415](#)
  - publicDescription, [416](#)
  - uri, [416](#)
- qevercloud::QAssociativeContainerConstReference↔Wrapper
  - begin, [417](#)
  - end, [417](#)
  - QAssociativeContainerConstReferenceWrapper, [417](#)
- qevercloud::QAssociativeContainerConstReference↔Wrapper< Container >, [416](#)
- qevercloud::QAssociativeContainerConstReference↔Wrapper< Container >::iterator, [247](#)
- qevercloud::QAssociativeContainerConstReference↔Wrapper::iterator
  - iterator, [248](#)
  - m\_iterator, [248](#)
  - operator!=, [248](#)
  - operator\*, [248](#)
  - operator++, [248](#)
- qevercloud::QAssociativeContainerReferenceWrapper
  - begin, [418](#)
  - end, [418](#)
  - QAssociativeContainerReferenceWrapper, [418](#)
- qevercloud::QAssociativeContainerReferenceWrapper< Container >, [417](#)
- qevercloud::QAssociativeContainerReferenceWrapper< Container >::iterator, [246](#)
- qevercloud::QAssociativeContainerReferenceWrapper↔::iterator
  - iterator, [246](#)
  - m\_iterator, [247](#)
  - operator!=, [246](#)
  - operator\*, [247](#)
  - operator++, [247](#)
- qevercloud::RelatedContent, [418](#)
  - accessType, [420](#)
  - authors, [420](#)
  - clipUrl, [420](#)
  - contact, [420](#)
  - contentId, [420](#)
  - contentType, [420](#)
  - date, [420](#)
  - localData, [421](#)
  - operator!=, [419](#)
  - operator==, [419](#)
  - print, [419](#)
  - QList, [422](#)
  - sourceFaviconUrl, [421](#)
  - sourceId, [421](#)
  - sourceName, [421](#)
  - sourceUrl, [421](#)
  - teaser, [421](#)
  - thumbnails, [421](#)
  - title, [421](#)
  - url, [422](#)
  - visibleUrl, [422](#)
- qevercloud::RelatedContentImage, [422](#)
  - fileSize, [423](#)
  - height, [424](#)
  - localData, [424](#)
  - operator!=, [423](#)
  - operator==, [423](#)
  - pixelRatio, [424](#)
  - print, [423](#)
  - url, [424](#)
  - width, [424](#)
- qevercloud::RelatedQuery, [424](#)
  - cacheKey, [426](#)
  - context, [426](#)
  - filter, [426](#)
  - localData, [426](#)
  - noteGuid, [426](#)
  - operator!=, [425](#)
  - operator==, [425](#)
  - plainText, [426](#)
  - print, [425](#)
  - referenceUri, [426](#)

- qevercloud::RelatedResult, 427
  - cacheExpires, 428
  - cacheKey, 428
  - containingNotebooks, 429
  - debugInfo, 429
  - experts, 429
  - localData, 429
  - notebooks, 430
  - notes, 430
  - operator!=, 428
  - operator==, 428
  - print, 428
  - QList, 430
  - relatedContent, 430
  - tags, 430
- qevercloud::RelatedResultSpec, 430
  - includeContainingNotebooks, 432
  - includeDebugInfo, 432
  - localData, 432
  - maxExperts, 432
  - maxNotebooks, 432
  - maxNotes, 432
  - maxRelatedContent, 433
  - maxTags, 433
  - operator!=, 431
  - operator==, 431
  - print, 431
  - QSet, 433
  - relatedContentTypes, 433
  - writableNotebooksOnly, 433
- qevercloud::Resource, 434
  - active, 435
  - alternateData, 435
  - attributes, 435
  - data, 435
  - duration, 435
  - guid, 436
  - height, 436
  - localData, 436
  - mime, 436
  - noteGuid, 436
  - operator!=, 434
  - operator==, 434
  - print, 435
  - recognition, 436
  - updateSequenceNum, 436
  - width, 437
- qevercloud::ResourceAttributes, 437
  - altitude, 438
  - applicationData, 438
  - attachment, 439
  - cameraMake, 439
  - cameraModel, 439
  - clientWillIndex, 439
  - fileName, 439
  - latitude, 439
  - localData, 440
  - longitude, 440
  - operator!=, 438
  - operator==, 438
  - print, 438
  - recoType, 440
  - sourceURL, 440
  - timestamp, 440
- qevercloud::SavedSearch, 440
  - format, 442
  - guid, 442
  - localData, 442
  - name, 442
  - operator!=, 441
  - operator==, 441
  - print, 441
  - query, 442
  - scope, 442
  - updateSequenceNum, 442
- qevercloud::SavedSearchScope, 443
  - includeAccount, 444
  - includeBusinessLinkedNotebooks, 444
  - includePersonalLinkedNotebooks, 444
  - localData, 444
  - operator!=, 443
  - operator==, 443
  - print, 444
- qevercloud::ShareRelationshipRestrictions, 455
  - localData, 456
  - noSetFullAccess, 456
  - noSetModify, 456
  - noSetReadOnly, 457
  - noSetReadPlusActivity, 457
  - operator!=, 456
  - operator==, 456
  - print, 456
- qevercloud::ShareRelationships, 457
  - invitationRestrictions, 458
  - invitations, 458
  - localData, 459
  - memberships, 459
  - operator!=, 458
  - operator==, 458
  - print, 458
  - QList, 459
- qevercloud::SharedNote, 445
  - localData, 446
  - operator!=, 445
  - operator==, 445
  - print, 446
  - privilege, 446
  - recipientIdentity, 446
  - serviceAssigned, 446
  - serviceCreated, 446
  - serviceUpdated, 447
  - sharerUserID, 447
- qevercloud::SharedNoteTemplate, 453
  - localData, 454
  - noteGuid, 454
  - operator!=, 453



- operator==, 454
- print, 454
- privilege, 454
- QList, 455
- recipientContacts, 454
- recipientThreadId, 455
- qevercloud::SharedNotebook, 447
  - email, 448
  - globalId, 448
  - id, 448
  - localData, 449
  - notebookGuid, 449
  - notebookModifiable, 449
  - operator!=, 448
  - operator==, 448
  - print, 448
  - privilege, 449
  - recipientIdentityId, 449
  - recipientSettings, 449
  - recipientUserId, 449
  - recipientUsername, 450
  - serviceAssigned, 450
  - serviceCreated, 450
  - serviceUpdated, 450
  - sharerUserId, 450
  - userId, 450
  - username, 451
- qevercloud::SharedNotebookRecipientSettings, 451
  - localData, 452
  - operator!=, 452
  - operator==, 452
  - print, 452
  - reminderNotifyEmail, 452
  - reminderNotifyInApp, 452
- qevercloud::SyncChunk, 459
  - chunkHighUSN, 461
  - currentTime, 461
  - expungedLinkedNotebooks, 461
  - expungedNotebooks, 461
  - expungedNotes, 461
  - expungedSearches, 461
  - expungedTags, 462
  - linkedNotebooks, 462
  - localData, 462
  - notebooks, 462
  - notes, 462
  - operator!=, 460
  - operator==, 460
  - print, 460
  - QList, 463
  - resources, 462
  - searches, 462
  - tags, 463
  - updateCount, 463
- qevercloud::SyncChunkFilter, 463
  - includeExpunged, 465
  - includeLinkedNotebooks, 465
  - includeNoteApplicationDataFullMap, 465
  - includeNoteAttributes, 465
  - includeNoteResourceApplicationDataFullMap, 465
  - includeNoteResources, 466
  - includeNotebooks, 465
  - includeNotes, 466
  - includeResourceApplicationDataFullMap, 466
  - includeResources, 466
  - includeSearches, 466
  - includeSharedNotes, 466
  - includeTags, 466
  - localData, 467
  - notebookGuids, 467
  - omitSharedNotebooks, 467
  - operator!=, 464
  - operator==, 464
  - print, 464
  - QSet, 467
  - requireNoteContentClass, 467
- qevercloud::SyncState, 469
  - currentTime, 470
  - fullSyncBefore, 470
  - localData, 470
  - operator!=, 469
  - operator==, 469
  - print, 470
  - updateCount, 470
  - uploaded, 470
  - userLastUpdated, 471
  - userMaxMessageEventId, 471
- qevercloud::Tag, 471
  - guid, 472
  - localData, 473
  - name, 473
  - operator!=, 472
  - operator==, 472
  - parentGuid, 473
  - print, 472
  - updateSequenceNum, 473
- qevercloud::ThriftException, 473
  - ~ThriftException, 475
  - exceptionData, 475
  - m\_type, 476
  - operator!=, 475
  - operator<<, 476
  - operator==, 476
  - ThriftException, 475
  - Type, 474
  - type, 476
  - what, 476
- qevercloud::ThriftExceptionData, 477
  - m\_type, 478
  - ThriftExceptionData, 477
  - throwException, 477
- qevercloud::Thumbnail, 478
  - ~Thumbnail, 480
  - createPostRequest, 480
  - download, 481
  - downloadAsync, 481

- ImageType, 479
- operator<<, 483
- setAuthenticationToken, 481
- setHost, 482
- setImageType, 482
- setShardId, 482
- setSize, 482
- Thumbnail, 479
- qevercloud::UpdateNotelfUsnMatchesResult, 483
  - localData, 484
  - note, 484
  - operator!=, 484
  - operator==, 484
  - print, 484
  - updated, 484
- qevercloud::User, 485
  - accountLimits, 486
  - accounting, 486
  - active, 486
  - attributes, 486
  - businessUserInfo, 487
  - created, 487
  - deleted, 487
  - email, 487
  - id, 487
  - localData, 487
  - name, 487
  - operator!=, 486
  - operator==, 486
  - photoLastUpdated, 488
  - photoUrl, 488
  - print, 486
  - privilege, 488
  - serviceLevel, 488
  - shardId, 488
  - timezone, 488
  - updated, 489
  - username, 489
- qevercloud::UserAttributes, 489
  - businessAddress, 491
  - clipFullPage, 491
  - comments, 491
  - dailyEmailLimit, 491
  - dateAgreedToTermsOfService, 492
  - defaultLatitude, 492
  - defaultLocationName, 492
  - defaultLongitude, 492
  - educationalDiscount, 492
  - emailAddressLastConfirmed, 492
  - emailOptOutDate, 492
  - groupName, 493
  - hideSponsorBilling, 493
  - incomingEmailAddress, 493
  - localData, 493
  - maxReferrals, 493
  - operator!=, 490
  - operator==, 491
  - optOutMachineLearning, 493
  - partnerEmailOptInDate, 493
  - passwordUpdated, 494
  - preactivation, 494
  - preferredCountry, 494
  - preferredLanguage, 494
  - print, 491
  - recentMailedAddresses, 494
  - recognitionLanguage, 494
  - referrerCode, 495
  - referralCount, 495
  - referralProof, 495
  - reminderEmailConfig, 495
  - salesforcePushEnabled, 495
  - sentEmailCount, 495
  - sentEmailDate, 495
  - shouldLogClientEvent, 496
  - twitterId, 496
  - twitterUserName, 496
  - useEmailAutoFiling, 496
  - viewedPromotions, 496
- qevercloud::UserIdentity, 496
  - localData, 498
  - longIdentifier, 498
  - operator!=, 497
  - operator==, 497
  - print, 497
  - stringIdentifier, 498
  - type, 498
- qevercloud::UserProfile, 498
  - attributes, 500
  - email, 500
  - id, 500
  - joined, 500
  - localData, 500
  - name, 500
  - operator!=, 499
  - operator==, 499
  - photoLastUpdated, 500
  - photoUrl, 500
  - print, 499
  - role, 501
  - status, 501
  - username, 501
- qevercloud::UserStoreServer, 501
  - authenticateLongSessionRequest, 503
  - authenticateLongSessionRequestReady, 503
  - authenticateToBusinessRequest, 504
  - authenticateToBusinessRequestReady, 504
  - checkVersionRequest, 504
  - checkVersionRequestReady, 504
  - completeTwoFactorAuthenticationRequest, 504
  - completeTwoFactorAuthenticationRequestReady, 504
  - getAccountLimitsRequest, 505
  - getAccountLimitsRequestReady, 505
  - getBootstrapInfoRequest, 505
  - getBootstrapInfoRequestReady, 505
  - getPublicUserInfoRequest, 505



- getPublicUserInfoRequestReady, 505
- getUserRequest, 506
- getUserRequestReady, 506
- getUserUrlsRequest, 506
- getUserUrlsRequestReady, 506
- inviteToBusinessRequest, 506
- inviteToBusinessRequestReady, 506
- listBusinessInvitationsRequest, 506
- listBusinessInvitationsRequestReady, 507
- listBusinessUsersRequest, 507
- listBusinessUsersRequestReady, 507
- onAuthenticateLongSessionRequestReady, 507
- onAuthenticateToBusinessRequestReady, 507
- onCheckVersionRequestReady, 507
- onCompleteTwoFactorAuthenticationRequestReady, 508
- onGetAccountLimitsRequestReady, 508
- onGetBootstrapInfoRequestReady, 508
- onGetPublicUserInfoRequestReady, 508
- onGetUserRequestReady, 508
- onGetUserUrlsRequestReady, 508
- onInviteToBusinessRequestReady, 509
- onListBusinessInvitationsRequestReady, 509
- onListBusinessUsersRequestReady, 509
- onRemoveFromBusinessRequestReady, 509
- onRequest, 509
- onRevokeLongSessionRequestReady, 509
- onUpdateBusinessUserIdentifierRequestReady, 510
- removeFromBusinessRequest, 510
- removeFromBusinessRequestReady, 510
- revokeLongSessionRequest, 510
- revokeLongSessionRequestReady, 510
- updateBusinessUserIdentifierRequest, 510
- updateBusinessUserIdentifierRequestReady, 510
- UserStoreServer, 503
- qevercloud::UserUrls, 511
  - localData, 512
  - messageStoreUrl, 512
  - noteStoreUrl, 512
  - operator!=, 511
  - operator==, 511
  - print, 512
  - userStoreUrl, 512
  - userWebSocketUrl, 512
  - utilityUrl, 513
  - webApiUrlPrefix, 513
- query
  - qevercloud::SavedSearch, 442
- QueryFormat
  - qevercloud, 38
- README.md, 532
- rateLimitDuration
  - qevercloud::EDAMSystemException, 140
- rawWords
  - qevercloud::NoteFilter, 325
- ReadFunctionType
  - qevercloud::AsyncResult, 99
- reasons
  - qevercloud::EDAMInvalidContactsException, 131
- recentMailedAddresses
  - qevercloud::UserAttributes, 494
- recipientContacts
  - qevercloud::NotebookShareTemplate, 316
  - qevercloud::SharedNoteTemplate, 454
- recipientIdentity
  - qevercloud::SharedNote, 446
- recipientIdentityId
  - qevercloud::NoteInvitationShareRelationship, 328
  - qevercloud::SharedNotebook, 449
- recipientSettings
  - qevercloud::Notebook, 302
  - qevercloud::SharedNotebook, 449
- RecipientStatus
  - qevercloud, 39
- recipientStatus
  - qevercloud::NotebookRecipientSettings, 308
- recipientThreadId
  - qevercloud::NotebookShareTemplate, 316
  - qevercloud::SharedNoteTemplate, 455
- recipientUserId
  - qevercloud::MemberShareRelationship, 283
  - qevercloud::NoteMemberShareRelationship, 334
  - qevercloud::SharedNotebook, 449
- recipientUserIdentity
  - qevercloud::InvitationShareRelationship, 243
- recipientUsername
  - qevercloud::SharedNotebook, 450
- recoType
  - qevercloud::ResourceAttributes, 440
- recognition
  - qevercloud::Resource, 436
- recognitionLanguage
  - qevercloud::UserAttributes, 494
- recommended
  - qevercloud::BusinessNotebook, 115
- ref
  - qevercloud::Optional, 407, 408
- referenceUri
  - qevercloud::RelatedQuery, 426
- referrerCode
  - qevercloud::UserAttributes, 495
- referralCount
  - qevercloud::UserAttributes, 495
- referralProof
  - qevercloud::UserAttributes, 495
- relatedContent
  - qevercloud::RelatedResult, 430
- RelatedContentAccess
  - qevercloud, 39
- RelatedContentType
  - qevercloud, 39
- relatedContentTypes
  - qevercloud::RelatedResultSpec, 433
- reminderDoneTime
  - qevercloud::NoteAttributes, 297

- ReminderEmailConfig
  - qevercloud, 40
- reminderEmailConfig
  - qevercloud::UserAttributes, 495
- reminderNotifyEmail
  - qevercloud::NotebookRecipientSettings, 308
  - qevercloud::SharedNotebookRecipientSettings, 452
- reminderNotifyInApp
  - qevercloud::NotebookRecipientSettings, 308
  - qevercloud::SharedNotebookRecipientSettings, 452
- reminderOrder
  - qevercloud::NoteAttributes, 297
- reminderTime
  - qevercloud::NoteAttributes, 298
- removeFromBusiness
  - qevercloud::IUserStore, 260
- removeFromBusinessAsync
  - qevercloud::IUserStore, 260
- removeFromBusinessRequest
  - qevercloud::UserStoreServer, 510
- removeFromBusinessRequestReady
  - qevercloud::UserStoreServer, 510
- RequestContext.h, 532
- requestId
  - qevercloud::IRequestContext, 245
- requestTimeout
  - qevercloud::IRequestContext, 245
- requesterId
  - qevercloud::BusinessInvitation, 113
- requireNoteContentClass
  - qevercloud::SyncChunkFilter, 467
- resetEvernoteNetworkProxy
  - qevercloud, 57
- resourceSizeMax
  - qevercloud::AccountLimits, 96
  - qevercloud::NoteLimits, 330
- resources
  - qevercloud::Note, 291
  - qevercloud::SyncChunk, 462
- restrictions
  - qevercloud::MemberShareRelationship, 283
  - qevercloud::Note, 291
  - qevercloud::NoteMemberShareRelationship, 335
  - qevercloud::Notebook, 302
- revokeLongSession
  - qevercloud::IUserStore, 260
- revokeLongSessionAsync
  - qevercloud::IUserStore, 261
- revokeLongSessionRequest
  - qevercloud::UserStoreServer, 510
- revokeLongSessionRequestReady
  - qevercloud::UserStoreServer, 510
- role
  - qevercloud::BusinessInvitation, 113
  - qevercloud::BusinessUserInfo, 119
  - qevercloud::UserProfile, 501
- salesforcePushEnabled
  - qevercloud::UserAttributes, 495
- sameBusiness
  - qevercloud::Identity, 170
- saved
  - qevercloud::NoteVersionId, 398
- scope
  - qevercloud::SavedSearch, 442
- searchContextBytes
  - qevercloud::NoteFilter, 325
  - qevercloud::NoteList, 332
  - qevercloud::NotesMetadataList, 349
- searchedWords
  - qevercloud::NoteList, 332
  - qevercloud::NotesMetadataList, 350
- searches
  - qevercloud::SyncChunk, 462
- secondFactorDeliveryHint
  - qevercloud::AuthenticationResult, 103
- secondFactorRequired
  - qevercloud::AuthenticationResult, 104
- sentEmailCount
  - qevercloud::UserAttributes, 495
- sentEmailDate
  - qevercloud::UserAttributes, 495
- Servers.h, 533
- serviceAssigned
  - qevercloud::SharedNote, 446
  - qevercloud::SharedNotebook, 450
- serviceCreated
  - qevercloud::Notebook, 302
  - qevercloud::SharedNote, 446
  - qevercloud::SharedNotebook, 450
- serviceHost
  - qevercloud::BootstrapSettings, 111
- ServiceLevel
  - qevercloud, 40
- serviceLevel
  - qevercloud::PublicUserInfo, 413
  - qevercloud::User, 488
- serviceUpdated
  - qevercloud::Notebook, 303
  - qevercloud::SharedNote, 447
  - qevercloud::SharedNotebook, 450
- Services.h, 533
- setAuthenticationToken
  - qevercloud::InkNoteImageDownloader, 175
  - qevercloud::Thumbnail, 481
- setEvernoteNetworkProxy
  - qevercloud, 57
- setHeight
  - qevercloud::InkNoteImageDownloader, 176
- setHost
  - qevercloud::InkNoteImageDownloader, 176
  - qevercloud::Thumbnail, 482
- setImageType
  - qevercloud::Thumbnail, 482
- setLevel

- qevercloud::ILogger, 173
- setLogger
  - qevercloud, 57
- setNonceGenerator
  - qevercloud, 57
- setNoteApplicationDataEntry
  - qevercloud::INoteStore, 226
- setNoteApplicationDataEntryAsync
  - qevercloud::INoteStore, 226
- setNoteApplicationDataEntryRequest
  - qevercloud::NoteStoreServer, 390
- setNoteApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, 391
- setNoteStoreUrl
  - qevercloud::INoteStore, 228
- setNotebookRecipientSettings
  - qevercloud::INoteStore, 227
- setNotebookRecipientSettingsAsync
  - qevercloud::INoteStore, 228
- setNotebookRecipientSettingsRequest
  - qevercloud::NoteStoreServer, 391
- setNotebookRecipientSettingsRequestReady
  - qevercloud::NoteStoreServer, 391
- setResourceApplicationDataEntry
  - qevercloud::INoteStore, 228
- setResourceApplicationDataEntryAsync
  - qevercloud::INoteStore, 228
- setResourceApplicationDataEntryRequest
  - qevercloud::NoteStoreServer, 391
- setResourceApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, 391
- setShardId
  - qevercloud::InkNoteImageDownloader, 176
  - qevercloud::Thumbnail, 482
- setSize
  - qevercloud::Thumbnail, 482
- setSizeHint
  - qevercloud::EvernoteOAuthWebView, 168
- setUserStoreUrl
  - qevercloud::IUserStore, 261
- setWebViewSizeHint
  - qevercloud::EvernoteOAuthDialog, 165
- setWidth
  - qevercloud::InkNoteImageDownloader, 176
- settings
  - qevercloud::BootstrapProfile, 107
- shardId
  - qevercloud::EvernoteOAuthWebView::OAuthResult, 400
  - qevercloud::LinkedNotebook, 267
  - qevercloud::User, 488
- shareDate
  - qevercloud::NoteAttributes, 298
- shareName
  - qevercloud::LinkedNotebook, 267
- shareNote
  - qevercloud::INoteStore, 228
- shareNoteAsync
  - qevercloud::INoteStore, 229
- shareNoteRequest
  - qevercloud::NoteStoreServer, 392
- shareNoteRequestReady
  - qevercloud::NoteStoreServer, 392
- shareNotebook
  - qevercloud::INoteStore, 229
- shareNotebookAsync
  - qevercloud::INoteStore, 231
- shareNotebookRequest
  - qevercloud::NoteStoreServer, 391
- shareNotebookRequestReady
  - qevercloud::NoteStoreServer, 392
- ShareRelationshipPrivilegeLevel
  - qevercloud, 42
- SharedNotePrivilegeLevel
  - qevercloud, 42
- sharedNotebookGlobalId
  - qevercloud::LinkedNotebook, 267
- sharedNotebookIds
  - qevercloud::Notebook, 303
- SharedNotebookInstanceRestrictions
  - qevercloud, 41
- SharedNotebookPrivilegeLevel
  - qevercloud, 41
- sharedNotebooks
  - qevercloud::Notebook, 303
- sharedNotes
  - qevercloud::Note, 291
- sharedWithBusiness
  - qevercloud::NoteAttributes, 298
- sharerUserId
  - qevercloud::SharedNote, 447
- sharerUserId
  - qevercloud::InvitationShareRelationship, 243
  - qevercloud::MemberShareRelationship, 284
  - qevercloud::NoteInvitationShareRelationship, 328
  - qevercloud::NoteMemberShareRelationship, 335
  - qevercloud::SharedNotebook, 450
- shouldLog
  - qevercloud::ILogger, 173
- shouldLogClientEvent
  - qevercloud::UserAttributes, 496
- shouldRetry
  - qevercloud::IRetryPolicy, 246
- size
  - qevercloud::Data, 128
- sizeHint
  - qevercloud::EvernoteOAuthWebView, 168
- source
  - qevercloud::NoteAttributes, 298
- sourceApplication
  - qevercloud::NoteAttributes, 298
- sourceFaviconUrl
  - qevercloud::RelatedContent, 421
- sourceId
  - qevercloud::RelatedContent, 421
- sourceName

- qevercloud::RelatedContent, 421
- sourceURL
  - qevercloud::NoteAttributes, 299
  - qevercloud::ResourceAttributes, 440
- sourceUrl
  - qevercloud::RelatedContent, 421
- SponsoredGroupRole
  - qevercloud, 43
- stack
  - qevercloud::LinkedNotebook, 268
  - qevercloud::Notebook, 303
  - qevercloud::NotebookRecipientSettings, 308
- startIndex
  - qevercloud::NoteList, 332
  - qevercloud::NotesMetadataList, 350
- status
  - qevercloud::BusinessInvitation, 113
  - qevercloud::UserProfile, 501
- stopEventLoop
  - qevercloud::EventLoopFinisher, 151
- stopSharingNote
  - qevercloud::INoteStore, 231
- stopSharingNoteAsync
  - qevercloud::INoteStore, 232
- stopSharingNoteRequest
  - qevercloud::NoteStoreServer, 392
- stopSharingNoteRequestReady
  - qevercloud::NoteStoreServer, 392
- stoppedWords
  - qevercloud::NoteList, 332
  - qevercloud::NotesMetadataList, 350
- stringIdentifier
  - qevercloud::UserIdentity, 498
- subject
  - qevercloud::NoteEmailParameters, 322
- subjectDate
  - qevercloud::NoteAttributes, 299
- supportUrl
  - qevercloud::BootstrapSettings, 111
- swap
  - qevercloud::Optional, 409
- SyncRequest
  - qevercloud::IDurableService::SyncRequest, 468
- SyncResult
  - qevercloud::IDurableService, 171
- SyncServiceCall
  - qevercloud::IDurableService, 171
- TagCounts
  - qevercloud::NoteCollectionCounts, 318
- tagCounts
  - qevercloud::NoteCollectionCounts, 319
- tagGuids
  - qevercloud::Note, 291
  - qevercloud::NoteFilter, 325
  - qevercloud::NoteMetadata, 338
- tagNames
  - qevercloud::Note, 292
- tags
  - qevercloud::RelatedResult, 430
  - qevercloud::SyncChunk, 463
- teaser
  - qevercloud::RelatedContent, 421
- ThriftException
  - qevercloud::ThriftException, 475
- ThriftExceptionData
  - qevercloud::ThriftExceptionData, 477
- throwException
  - qevercloud::EDAMInvalidContactsExceptionData, 132
  - qevercloud::EDAMNotFoundExceptionData, 136
  - qevercloud::EDAMSystemExceptionAuthExpiredData, 142
  - qevercloud::EDAMSystemExceptionData, 143
  - qevercloud::EDAMSystemExceptionRateLimitReachedData, 145
  - qevercloud::EDAMUserExceptionData, 150
  - qevercloud::EverCloudExceptionData, 155
  - qevercloud::EvernoteExceptionData, 161
  - qevercloud::NetworkExceptionData, 287
  - qevercloud::ThriftExceptionData, 477
- Thumbnail
  - qevercloud::Thumbnail, 479
- Thumbnail.h, 534
- thumbnails
  - qevercloud::RelatedContent, 421
- timeZone
  - qevercloud::NoteFilter, 325
- Timestamp
  - qevercloud, 31
- timestamp
  - qevercloud::ResourceAttributes, 440
- timezone
  - qevercloud::User, 488
- title
  - qevercloud::BusinessUserAttributes, 117
  - qevercloud::Note, 292
  - qevercloud::NoteMetadata, 338
  - qevercloud::NoteVersionId, 398
  - qevercloud::RelatedContent, 421
- toAddresses
  - qevercloud::NoteEmailParameters, 322
- toRange
  - qevercloud, 58
- toString
  - qevercloud::Printable, 411
- totalNotes
  - qevercloud::NoteList, 332
  - qevercloud::NotesMetadataList, 350
- Trace
  - qevercloud, 37
- trashCount
  - qevercloud::NoteCollectionCounts, 319
- twitterId
  - qevercloud::UserAttributes, 496
- twitterUserName
  - qevercloud::UserAttributes, 496

- Type
  - qevercloud::ThriftException, [474](#)
- type
  - qevercloud::Contact, [124](#)
  - qevercloud::NetworkException, [286](#)
  - qevercloud::ThriftException, [476](#)
  - qevercloud::UserIdentity, [498](#)
- Types.h, [535](#)
- unitDiscount
  - qevercloud::Accounting, [93](#)
- unitPrice
  - qevercloud::Accounting, [93](#)
- unsetNoteApplicationDataEntry
  - qevercloud::INoteStore, [232](#)
- unsetNoteApplicationDataEntryAsync
  - qevercloud::INoteStore, [232](#)
- unsetNoteApplicationDataEntryRequest
  - qevercloud::NoteStoreServer, [392](#)
- unsetNoteApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, [393](#)
- unsetResourceApplicationDataEntry
  - qevercloud::INoteStore, [232](#)
- unsetResourceApplicationDataEntryAsync
  - qevercloud::INoteStore, [232](#)
- unsetResourceApplicationDataEntryRequest
  - qevercloud::NoteStoreServer, [393](#)
- unsetResourceApplicationDataEntryRequestReady
  - qevercloud::NoteStoreServer, [393](#)
- unshares
  - qevercloud::ManageNotebookSharesParameters, [273](#)
- untagAll
  - qevercloud::INoteStore, [233](#)
- untagAllAsync
  - qevercloud::INoteStore, [233](#)
- untagAllRequest
  - qevercloud::NoteStoreServer, [393](#)
- untagAllRequestReady
  - qevercloud::NoteStoreServer, [393](#)
- updateBusinessUserIdentifier
  - qevercloud::IUserStore, [261](#)
- updateBusinessUserIdentifierAsync
  - qevercloud::IUserStore, [262](#)
- updateBusinessUserIdentifierRequest
  - qevercloud::UserStoreServer, [510](#)
- updateBusinessUserIdentifierRequestReady
  - qevercloud::UserStoreServer, [510](#)
- updateCount
  - qevercloud::NoteList, [333](#)
  - qevercloud::NotesMetadataList, [350](#)
  - qevercloud::SyncChunk, [463](#)
  - qevercloud::SyncState, [470](#)
- updateLinkedNotebook
  - qevercloud::INoteStore, [233](#)
- updateLinkedNotebookAsync
  - qevercloud::INoteStore, [234](#)
- updateLinkedNotebookRequest
  - qevercloud::NoteStoreServer, [393](#)
- updateLinkedNotebookRequestReady
  - qevercloud::NoteStoreServer, [394](#)
- updateNote
  - qevercloud::INoteStore, [234](#)
- updateNoteAsync
  - qevercloud::INoteStore, [235](#)
- updateNoteIfUsnMatches
  - qevercloud::INoteStore, [237](#)
- updateNoteIfUsnMatchesAsync
  - qevercloud::INoteStore, [237](#)
- updateNoteIfUsnMatchesRequest
  - qevercloud::NoteStoreServer, [394](#)
- updateNoteIfUsnMatchesRequestReady
  - qevercloud::NoteStoreServer, [394](#)
- updateNoteRequest
  - qevercloud::NoteStoreServer, [394](#)
- updateNoteRequestReady
  - qevercloud::NoteStoreServer, [395](#)
- updateNotebook
  - qevercloud::INoteStore, [235](#)
- updateNotebookAsync
  - qevercloud::INoteStore, [236](#)
- updateNotebookRequest
  - qevercloud::NoteStoreServer, [394](#)
- updateNotebookRequestReady
  - qevercloud::NoteStoreServer, [394](#)
- updateResource
  - qevercloud::INoteStore, [237](#)
- updateResourceAsync
  - qevercloud::INoteStore, [238](#)
- updateResourceRequest
  - qevercloud::NoteStoreServer, [395](#)
- updateResourceRequestReady
  - qevercloud::NoteStoreServer, [395](#)
- updateSearch
  - qevercloud::INoteStore, [238](#)
- updateSearchAsync
  - qevercloud::INoteStore, [239](#)
- updateSearchRequest
  - qevercloud::NoteStoreServer, [395](#)
- updateSearchRequestReady
  - qevercloud::NoteStoreServer, [395](#)
- updateSequenceNum
  - qevercloud::CreateOrUpdateNotebookShares↔  
Result, [125](#)
  - qevercloud::LinkedNotebook, [268](#)
  - qevercloud::Note, [292](#)
  - qevercloud::NoteMetadata, [338](#)
  - qevercloud::NoteVersionId, [398](#)
  - qevercloud::Notebook, [303](#)
  - qevercloud::Resource, [436](#)
  - qevercloud::SavedSearch, [442](#)
  - qevercloud::Tag, [473](#)
- updateSharedNotebook
  - qevercloud::INoteStore, [239](#)
- updateSharedNotebookAsync
  - qevercloud::INoteStore, [239](#)
- updateSharedNotebookRequest

- qevercloud::NoteStoreServer, 395
- updateSharedNotebookRequestReady
  - qevercloud::NoteStoreServer, 396
- updateTag
  - qevercloud::INoteStore, 239
- updateTagAsync
  - qevercloud::INoteStore, 241
- updateTagRequest
  - qevercloud::NoteStoreServer, 396
- updateTagRequestReady
  - qevercloud::NoteStoreServer, 396
- updateWhichSharedNotebookRestrictions
  - qevercloud::NotebookRestrictions, 314
- updated
  - qevercloud::Accounting, 93
  - qevercloud::BusinessUserInfo, 120
  - qevercloud::Note, 292
  - qevercloud::NoteMetadata, 338
  - qevercloud::NoteVersionId, 398
  - qevercloud::UpdateNoteIfUsnMatchesResult, 484
  - qevercloud::User, 489
- uploadLimit
  - qevercloud::AccountLimits, 96
  - qevercloud::NoteLimits, 330
- uploadLimitEnd
  - qevercloud::Accounting, 93
- uploadLimitNextMonth
  - qevercloud::Accounting, 93
- uploaded
  - qevercloud::NoteLimits, 330
  - qevercloud::SyncState, 470
- uri
  - qevercloud::LinkedNotebook, 268
  - qevercloud::Publishing, 416
- url
  - qevercloud::RelatedContent, 422
  - qevercloud::RelatedContentImage, 424
- urls
  - qevercloud::AuthenticationResult, 104
- useEmailAutoFiling
  - qevercloud::UserAttributes, 496
- user
  - qevercloud::AuthenticationResult, 104
- userConnected
  - qevercloud::Identity, 170
- userException
  - qevercloud::ManageNoteSharesError, 276
  - qevercloud::ManageNotebookSharesError, 270
- UserID
  - qevercloud, 31
- userID
  - qevercloud::ManageNoteSharesError, 277
- userId
  - qevercloud::EvernoteOAuthWebView::OAuth←  
Result, 400
  - qevercloud::Identity, 170
  - qevercloud::PublicUserInfo, 414
  - qevercloud::SharedNotebook, 450
- userIdentity
  - qevercloud::ManageNotebookSharesError, 270
- UserIdentityType
  - qevercloud, 43
- userLastUpdated
  - qevercloud::SyncState, 471
- userLinkedNotebookMax
  - qevercloud::AccountLimits, 96
- userMailLimitDaily
  - qevercloud::AccountLimits, 96
- userMaxMessageEventId
  - qevercloud::SyncState, 471
- userNoteCountMax
  - qevercloud::AccountLimits, 96
- userNotebookCountMax
  - qevercloud::AccountLimits, 96
- userSavedSearchesMax
  - qevercloud::AccountLimits, 96
- UserStoreServer
  - qevercloud::UserStoreServer, 503
- userStoreUrl
  - qevercloud::IUserStore, 262
  - qevercloud::UserUrls, 512
- userTagCountMax
  - qevercloud::AccountLimits, 97
- userWebSocketUrl
  - qevercloud::UserUrls, 512
- username
  - qevercloud::LinkedNotebook, 268
  - qevercloud::PublicUserInfo, 414
  - qevercloud::SharedNotebook, 451
  - qevercloud::User, 489
  - qevercloud::UserProfile, 501
- utilityUrl
  - qevercloud::UserUrls, 513
- value
  - qevercloud::Optional, 408
- viewedPromotions
  - qevercloud::UserAttributes, 496
- visibleUrl
  - qevercloud::RelatedContent, 422
- waitForFinished
  - qevercloud::AsyncResult, 101
- Warn
  - qevercloud, 37
- webApiUrlPrefix
  - qevercloud::AuthenticationResult, 104
  - qevercloud::EvernoteOAuthWebView::OAuth←  
Result, 401
  - qevercloud::LinkedNotebook, 268
  - qevercloud::PublicUserInfo, 414
  - qevercloud::UserUrls, 513
- what
  - qevercloud::EDAMInvalidContactsException, 130
  - qevercloud::EDAMNotFoundException, 135
  - qevercloud::EDAMSystemException, 139
  - qevercloud::EDAMUserException, 148



- qevercloud::EverCloudException, [153](#)
- qevercloud::NetworkException, [286](#)
- qevercloud::ThriftException, [476](#)
- width
  - qevercloud::RelatedContentImage, [424](#)
  - qevercloud::Resource, [437](#)
- words
  - qevercloud::NoteFilter, [326](#)
- workPhone
  - qevercloud::BusinessUserAttributes, [117](#)
- writableNotebooksOnly
  - qevercloud::RelatedResultSpec, [433](#)