

SDK

2.1.4

Generated by Doxygen 1.9.1

1 API for C++	1
2 Class diagrams	3
2.1 Overview	3
2.2 CloudBackend - core object	4
2.3 Object	5
2.4 Group	6
3 Sync vs. Async	7
3.1 Choosing a Synchronous or Asynchronous coding style	7
4 Sequence diagrams - async	11
4.1 Login asynchronous UML	11
4.2 Query asynchronous UML	13
5 Example code	15
5.1 Example code - sync - exception	15
5.1.1 login - upload - download - query	15
5.1.2 Complete program	15
5.2 Example code - sync - non-throw	15
5.2.1 login - upload - download - query	16
5.2.2 Complete program	16
5.3 Example code - async	16
5.3.1 login - upload - download	16
5.3.2 query - select	16
5.3.3 other - misc	16
5.3.4 Complete program	16
5.4 Example code - combined lists grouped by call	16
5.4.1 login	16
5.4.2 upload	17
5.4.3 getStreams	17
5.4.4 uploadStream	17
5.4.5 downloadStream	17
5.4.6 query - select	17
5.4.7 databases - tenant	17
5.4.8 Complete program	17
5.5 Full example - sync [exception]	18
5.6 Full example - sync [non-throwing]	20
5.7 Full example - async	22
6 Document list	25
7 Namespace Index	27
7.1 Namespace List	27

8 Hierarchical Index	29
8.1 Class Hierarchy	29
9 Class Index	33
9.1 Class List	33
10 Namespace Documentation	39
10.1 cbe Namespace Reference	39
10.1.1 Detailed Description	42
10.1.2 Typedef Documentation	42
10.1.2.1 AclGroupId	42
10.1.2.2 Date	42
10.1.2.3 ErrorCode	42
10.1.2.4 AclMap	42
10.1.2.5 KeyValues	43
10.1.2.6 MemberBanInfo	43
10.1.3 Enumeration Type Documentation	43
10.1.3.1 AccountStatus	43
10.1.3.2 AclScope	43
10.1.3.3 DefaultCtor	44
10.1.3.4 FilterOrder	44
10.1.3.5 ItemType	44
10.1.3.6 ObjectType	44
10.1.3.7 Permissions	45
10.1.3.8 PublishAccess	45
10.1.3.9 PublishVisibility	46
10.1.3.10 Visibility	46
10.2 cbe::delegate Namespace Reference	46
10.2.1 Detailed Description	48
10.2.2 Typedef Documentation	48
10.2.2.1 AclDelegatePtr	49
10.2.2.2 AddRoleMemberDelegatePtr	49
10.2.2.3 BanDelegatePtr	49
10.2.2.4 CloudBackendListenerDelegatePtr	49
10.2.2.5 CreateAccountDelegatePtr	49
10.2.2.6 CreateContainerDelegatePtr	49
10.2.2.7 CreateGroupDelegatePtr	49
10.2.2.8 CreateObjectDelegatePtr	49
10.2.2.9 CreateRoleDelegatePtr	50
10.2.2.10 DownloadBinaryDelegatePtr	50
10.2.2.11 DownloadDelegatePtr	50
10.2.2.12 GetStreamsDelegatePtr	50
10.2.2.13 GetSubscriptionsDelegatePtr	50

10.2.2.14 ICloudBackendListenerPtr	50
10.2.2.15 JoinDelegatePtr	50
10.2.2.16 KickDelegatePtr	50
10.2.2.17 LeaveDelegatePtr	51
10.2.2.18 ListGroupsDelegatePtr	51
10.2.2.19 ListMembersDelegatePtr	51
10.2.2.20 ListRolesDelegatePtr	51
10.2.2.21 ListSharesDelegatePtr	51
10.2.2.22 LoginDelegatePtr	51
10.2.2.23 ProgressEventFn	51
10.2.2.24 PublishDelegatePtr	51
10.2.2.25 QueryDelegatePtr	52
10.2.2.26 QueryJoinDelegatePtr	52
10.2.2.27 RemoveRoleDelegatePtr	52
10.2.2.28 RemoveRoleMemberDelegatePtr	52
10.2.2.29 SearchGroupsDelegatePtr	52
10.2.2.30 ShareDelegatePtr	52
10.2.2.31 SubscribeDelegatePtr	52
10.2.2.32 UnBanDelegatePtr	53
10.2.2.33 UnPublishDelegatePtr	53
10.2.2.34 UnShareDelegatePtr	53
10.2.2.35 UnSubscribeDelegatePtr	53
10.2.2.36 UpdateKeyValuesDelegatePtr	53
10.2.2.37 UploadDelegatePtr	53
10.3 cbe::delegate::container Namespace Reference	53
10.3.1 Detailed Description	54
10.3.2 Typedef Documentation	54
10.3.2.1 MoveDelegatePtr	54
10.3.2.2 RemoveDelegatePtr	54
10.3.2.3 RenameDelegatePtr	54
10.4 cbe::delegate::group Namespace Reference	54
10.4.1 Detailed Description	55
10.4.2 Typedef Documentation	55
10.4.2.1 JoinDelegatePtr	55
10.4.2.2 RemoveDelegatePtr	55
10.4.2.3 RenameDelegatePtr	55
10.5 cbe::delegate::object Namespace Reference	55
10.5.1 Detailed Description	55
10.5.2 Typedef Documentation	56
10.5.2.1 MoveDelegatePtr	56
10.5.2.2 RemoveDelegatePtr	56
10.5.2.3 RenameDelegatePtr	56

10.6 cbe::util Namespace Reference	56
10.6.1 Detailed Description	56
11 Class Documentation	57
11.1 cbe::Account Class Reference	57
11.1.1 Detailed Description	58
11.1.2 Constructor & Destructor Documentation	58
11.1.2.1 Account()	58
11.1.3 Member Function Documentation	58
11.1.3.1 username()	58
11.1.3.2 password()	58
11.1.3.3 source()	58
11.1.3.4 client()	58
11.1.3.5 firstName()	58
11.1.3.6 rootContainer()	58
11.1.3.7 tenantContainerId()	59
11.1.3.8 libContainerId()	59
11.1.3.9 rootDatabase()	59
11.1.3.10 databases()	59
11.1.3.11 operator bool()	59
11.2 cbe::delegate::AclDelegate Class Reference	59
11.2.1 Detailed Description	60
11.2.2 Member Function Documentation	60
11.2.2.1 onAclSuccess()	60
11.2.2.2 onAclError()	60
11.3 cbe::delegate::AddRoleMemberDelegate Class Reference	60
11.3.1 Detailed Description	61
11.3.2 Member Function Documentation	61
11.3.2.1 onAddRoleMemberSuccess()	61
11.3.2.2 onAddRoleMemberError()	61
11.4 cbe::util::Optional< T >::BadAccess Class Reference	61
11.4.1 Detailed Description	62
11.5 cbe::delegate::BanDelegate Class Reference	62
11.5.1 Detailed Description	62
11.5.2 Member Function Documentation	62
11.5.2.1 onBanSuccess()	63
11.5.2.2 onBanError()	63
11.6 cbe::delegate::BanSuccess Class Reference	63
11.6.1 Detailed Description	63
11.7 cbe::delegate::ChunkTransferred Class Reference	63
11.7.1 Detailed Description	64
11.8 cbe::CloudBackend Class Reference	64

11.8.1 Detailed Description	67
11.8.2 Member Typedef Documentation	67
11.8.2.1 LoginDelegatePtr	67
11.8.2.2 LoginException	67
11.8.2.3 LoginError	67
11.8.2.4 CreateAccountDelegatePtr	67
11.8.2.5 CreateAccountException	67
11.8.2.6 CreateAccountError	68
11.8.2.7 CloudBackendListenerDelegatePtr	68
11.8.2.8 QueryDelegatePtr	68
11.8.2.9 QueryJoinDelegatePtr	68
11.8.2.10 QueryException	68
11.8.2.11 QueryJoinError	68
11.8.2.12 QueryError	69
11.8.2.13 SearchException	69
11.8.2.14 SearchError	69
11.8.3 Constructor & Destructor Documentation	69
11.8.3.1 CloudBackend()	69
11.8.4 Member Function Documentation	69
11.8.4.1 login() [1/3]	69
11.8.4.2 login() [2/3]	71
11.8.4.3 login() [3/3]	71
11.8.4.4 createAccount() [1/3]	72
11.8.4.5 createAccount() [2/3]	73
11.8.4.6 createAccount() [3/3]	73
11.8.4.7 addListener()	73
11.8.4.8 removeListener()	74
11.8.4.9 query() [1/8]	74
11.8.4.10 query() [2/8]	75
11.8.4.11 query() [3/8]	75
11.8.4.12 query() [4/8]	76
11.8.4.13 query() [5/8]	76
11.8.4.14 query() [6/8]	77
11.8.4.15 query() [7/8]	77
11.8.4.16 query() [8/8]	78
11.8.4.17 queryWithPath() [1/2]	78
11.8.4.18 queryWithPath() [2/2]	78
11.8.4.19 search() [1/6]	78
11.8.4.20 search() [2/6]	79
11.8.4.21 search() [3/6]	79
11.8.4.22 search() [4/6]	80
11.8.4.23 search() [5/6]	80

11.8.4.24 search() [6/6]	80
11.8.4.25 castContainer()	81
11.8.4.26 castObject()	81
11.8.4.27 clearCache()	81
11.8.4.28 account()	81
11.8.4.29 version()	81
11.8.4.30 groupManager()	82
11.8.4.31 shareManager()	82
11.8.4.32 publishManager()	82
11.8.4.33 terminate()	82
11.8.4.34 subscribeManager()	82
11.8.4.35 operator bool()	82
11.9 cbe::delegate::CloudBackendListenerDelegate Class Reference	83
11.9.1 Member Function Documentation	83
11.9.1.1 onRemoteObjectAdded()	83
11.9.1.2 onRemoteObjectMoved()	84
11.9.1.3 onRemoteObjectRemoved()	84
11.9.1.4 onRemoteObjectRenamed()	84
11.9.1.5 onRemoteContainerAdded()	84
11.9.1.6 onRemoteContainerMoved()	85
11.9.1.7 onRemoteContainerRemoved()	85
11.9.1.8 onRemoteContainerRenamed()	85
11.10 cbe::Container Class Reference	85
11.10.1 Detailed Description	90
11.10.2 Member Typedef Documentation	90
11.10.2.1 CreateContainerDelegatePtr	90
11.10.2.2 CreateContainerException	91
11.10.2.3 CreateContainerError	91
11.10.2.4 MoveDelegatePtr	91
11.10.2.5 MoveException	91
11.10.2.6 MoveError	91
11.10.2.7 RenameDelegatePtr	91
11.10.2.8 RenameException	91
11.10.2.9 RenameError	91
11.10.2.10 RemoveDelegatePtr	91
11.10.2.11 RemoveException	91
11.10.2.12 RemoveError	91
11.10.2.13 CreateObjectDelegatePtr	92
11.10.2.14 CreateObjectException	92
11.10.2.15 CreateObjectError	92
11.10.2.16 UploadDelegatePtr	92
11.10.2.17 UploadException	92

11.10.2.18 UploadError	92
11.10.2.19 QueryDelegatePtr	93
11.10.2.20 QueryJoinDelegatePtr	93
11.10.2.21 QueryException	93
11.10.2.22 QueryJoinError	93
11.10.2.23 QueryError	93
11.10.2.24 queryWithPathException	93
11.10.2.25 queryWithPathError	93
11.10.2.26 SearchException	93
11.10.2.27 SearchError	93
11.10.2.28 SetAclDelegatePtr	94
11.10.2.29 SetAclException	94
11.10.2.30 SetAclError	94
11.10.2.31 GetAclDelegatePtr	94
11.10.2.32 GetAclException	94
11.10.2.33 GetAclError	94
11.10.2.34 ShareDelegatePtr	94
11.10.2.35 ShareException	94
11.10.2.36 ShareError	94
11.10.2.37 UnShareDelegatePtr	94
11.10.2.38 UnShareException	94
11.10.2.39 UnShareError	95
11.10.2.40 PublishDelegatePtr	95
11.10.2.41 PublishException	95
11.10.2.42 PublishError	95
11.10.2.43 UnPublishDelegatePtr	95
11.10.2.44 UnPublishException	95
11.10.2.45 UnPublishError	95
11.10.2.46 UnSubscribeDelegatePtr	95
11.10.2.47 UnSubscribeException	95
11.10.2.48 UnSubscribeError	95
11.10.3 Member Function Documentation	95
11.10.3.1 createContainer() [1/3]	96
11.10.3.2 createContainer() [2/3]	96
11.10.3.3 createContainer() [3/3]	96
11.10.3.4 move() [1/3]	97
11.10.3.5 move() [2/3]	97
11.10.3.6 move() [3/3]	97
11.10.3.7 rename() [1/3]	98
11.10.3.8 rename() [2/3]	98
11.10.3.9 rename() [3/3]	98
11.10.3.10 remove() [1/3]	99

11.10.3.11 remove() [2/3]	99
11.10.3.12 remove() [3/3]	99
11.10.3.13 createObject() [1/4]	99
11.10.3.14 createObject() [2/4]	100
11.10.3.15 createObject() [3/4]	100
11.10.3.16 createObject() [4/4]	100
11.10.3.17 upload() [1/15]	101
11.10.3.18 upload() [2/15]	102
11.10.3.19 upload() [3/15]	102
11.10.3.20 upload() [4/15]	103
11.10.3.21 upload() [5/15]	104
11.10.3.22 upload() [6/15]	104
11.10.3.23 upload() [7/15]	104
11.10.3.24 upload() [8/15]	104
11.10.3.25 upload() [9/15]	105
11.10.3.26 upload() [10/15]	105
11.10.3.27 upload() [11/15]	106
11.10.3.28 upload() [12/15]	106
11.10.3.29 upload() [13/15]	106
11.10.3.30 upload() [14/15]	107
11.10.3.31 upload() [15/15]	107
11.10.3.32 query() [1/8]	107
11.10.3.33 query() [2/8]	107
11.10.3.34 query() [3/8]	108
11.10.3.35 query() [4/8]	108
11.10.3.36 query() [5/8]	108
11.10.3.37 query() [6/8]	108
11.10.3.38 query() [7/8]	109
11.10.3.39 query() [8/8]	109
11.10.3.40 queryWithPath() [1/3]	109
11.10.3.41 queryWithPath() [2/3]	109
11.10.3.42 queryWithPath() [3/3]	110
11.10.3.43 search() [1/6]	110
11.10.3.44 search() [2/6]	111
11.10.3.45 search() [3/6]	111
11.10.3.46 search() [4/6]	111
11.10.3.47 search() [5/6]	112
11.10.3.48 search() [6/6]	112
11.10.3.49 setAcl() [1/3]	112
11.10.3.50 setAcl() [2/3]	113
11.10.3.51 setAcl() [3/3]	113
11.10.3.52 getAcl() [1/3]	113

11.10.3.53 getAcl() [2/3]	114
11.10.3.54 getAcl() [3/3]	114
11.10.3.55 share() [1/3]	114
11.10.3.56 share() [2/3]	115
11.10.3.57 share() [3/3]	115
11.10.3.58 unShare() [1/3]	116
11.10.3.59 unShare() [2/3]	116
11.10.3.60 unShare() [3/3]	116
11.10.3.61 publish() [1/3]	117
11.10.3.62 publish() [2/3]	117
11.10.3.63 publish() [3/3]	117
11.10.3.64 unPublish() [1/3]	118
11.10.3.65 unPublish() [2/3]	118
11.10.3.66 unPublish() [3/3]	118
11.10.3.67 unsubscribe() [1/3]	119
11.10.3.68 unsubscribe() [2/3]	119
11.10.3.69 unsubscribe() [3/3]	119
11.11 cbe::util::Context Struct Reference	120
11.12 cbe::delegate::CreateAccountDelegate Class Reference	120
11.12.1 Detailed Description	121
11.12.2 Member Function Documentation	121
11.12.2.1 onCreateAccountSuccess()	121
11.12.2.2 onCreateAccountError()	121
11.13 cbe::delegate::CreateContainerDelegate Class Reference	121
11.13.1 Detailed Description	121
11.13.2 Member Function Documentation	122
11.13.2.1 onCreateContainerSuccess()	122
11.13.2.2 onCreateContainerError()	122
11.14 cbe::delegate::CreateGroupDelegate Class Reference	122
11.14.1 Detailed Description	122
11.14.2 Member Function Documentation	122
11.14.2.1 onCreateGroupSuccess()	123
11.14.2.2 onCreateGroupError()	123
11.15 cbe::delegate::CreateObjectDelegate Class Reference	123
11.15.1 Detailed Description	123
11.15.2 Member Function Documentation	123
11.15.2.1 onCreateObjectSuccess()	123
11.15.2.2 onCreateObjectError()	124
11.16 cbe::delegate::CreateRoleDelegate Class Reference	124
11.16.1 Detailed Description	124
11.16.2 Member Function Documentation	124
11.16.2.1 onCreateRoleSuccess()	124

11.16.2.2 onCreateRoleError()	125
11.17 cbe::Database Class Reference	125
11.17.1 Detailed Description	125
11.17.2 Member Function Documentation	125
11.17.2.1 name()	126
11.17.2.2 databaseld()	126
11.17.2.3 containerId()	126
11.17.2.4 ownerId()	126
11.17.2.5 created()	126
11.17.2.6 readLocked()	126
11.17.2.7 writeLocked()	127
11.17.2.8 rootContainer()	127
11.18 cbe::delegate::DownloadBinaryDelegate Class Reference	127
11.18.1 Detailed Description	127
11.18.2 Member Function Documentation	127
11.18.2.1 onDownloadBinarySuccess()	128
11.18.2.2 onDownloadBinaryError()	128
11.18.2.3 onChunkReceived()	128
11.19 cbe::delegate::DownloadBinarySuccess Class Reference	128
11.19.1 Detailed Description	129
11.19.2 Member Function Documentation	129
11.19.2.1 operator bool()	129
11.20 cbe::delegate::DownloadDelegate Class Reference	129
11.20.1 Detailed Description	130
11.20.2 Member Function Documentation	130
11.20.2.1 onDownloadSuccess()	130
11.20.2.2 onDownloadError()	130
11.20.2.3 onChunkReceived()	130
11.21 cbe::delegate::DownloadSuccess Class Reference	131
11.21.1 Detailed Description	131
11.21.2 Member Function Documentation	131
11.21.2.1 operator bool()	132
11.22 cbe::delegate::Error Class Reference	132
11.22.1 Detailed Description	132
11.22.2 Member Function Documentation	132
11.22.2.1 operator bool()	132
11.22.3 Friends And Related Function Documentation	133
11.22.3.1 operator<<	133
11.22.4 Member Data Documentation	133
11.22.4.1 errorCode	133
11.22.4.2 reason	133
11.22.4.3 message	133

11.23 cbe::delegate::ItemError::Error Struct Reference	134
11.24 cbe::delegate::JoinError::Error Class Reference	134
11.25 cbe::delegate::LoadError::Error Class Reference	135
11.25.1 Member Data Documentation	135
11.25.1.1 errorCode	135
11.26 cbe::delegate::AclDelegate::ErrorInfo Struct Reference	135
11.26.1 Detailed Description	136
11.27 cbe::delegate::AddRoleMemberDelegate::ErrorInfo Struct Reference	137
11.27.1 Detailed Description	137
11.28 cbe::delegate::BanDelegate::ErrorInfo Struct Reference	138
11.28.1 Detailed Description	138
11.29 cbe::delegate::container::MoveDelegate::ErrorInfo Struct Reference	139
11.29.1 Detailed Description	139
11.30 cbe::delegate::container::RemoveDelegate::ErrorInfo Struct Reference	140
11.30.1 Detailed Description	140
11.31 cbe::delegate::container::RenameDelegate::ErrorInfo Struct Reference	141
11.31.1 Detailed Description	141
11.32 cbe::delegate::CreateAccountDelegate::ErrorInfo Struct Reference	142
11.32.1 Detailed Description	142
11.33 cbe::delegate::CreateContainerDelegate::ErrorInfo Struct Reference	143
11.33.1 Detailed Description	143
11.34 cbe::delegate::CreateGroupDelegate::ErrorInfo Struct Reference	144
11.34.1 Detailed Description	144
11.35 cbe::delegate::CreateObjectDelegate::ErrorInfo Struct Reference	145
11.35.1 Detailed Description	145
11.36 cbe::delegate::CreateRoleDelegate::ErrorInfo Struct Reference	146
11.36.1 Detailed Description	146
11.37 cbe::delegate::DownloadBinaryDelegate::ErrorInfo Struct Reference	147
11.37.1 Detailed Description	147
11.38 cbe::delegate::DownloadDelegate::ErrorInfo Struct Reference	148
11.38.1 Detailed Description	148
11.39 cbe::delegate::GetStreamsDelegate::ErrorInfo Struct Reference	149
11.39.1 Detailed Description	149
11.40 cbe::delegate::GetSubscriptionsDelegate::ErrorInfo Struct Reference	150
11.40.1 Detailed Description	150
11.41 cbe::delegate::group::JoinDelegate::ErrorInfo Struct Reference	151
11.41.1 Detailed Description	151
11.42 cbe::delegate::group::RemoveDelegate::ErrorInfo Struct Reference	152
11.42.1 Detailed Description	152
11.43 cbe::delegate::group::RenameDelegate::ErrorInfo Struct Reference	153
11.43.1 Detailed Description	153
11.44 cbe::delegate::KickDelegate::ErrorInfo Struct Reference	154

11.44.1 Detailed Description	154
11.45 cbe::delegate::LeaveDelegate::ErrorInfo Struct Reference	155
11.45.1 Detailed Description	155
11.46 cbe::delegate::ListGroupDelegate::ErrorInfo Struct Reference	156
11.46.1 Detailed Description	156
11.47 cbe::delegate::ListMembersDelegate::ErrorInfo Struct Reference	157
11.47.1 Detailed Description	157
11.48 cbe::delegate::ListRolesDelegate::ErrorInfo Struct Reference	158
11.48.1 Detailed Description	158
11.49 cbe::delegate::ListSharesDelegate::ErrorInfo Struct Reference	159
11.49.1 Detailed Description	159
11.50 cbe::delegate::LoginDelegate::ErrorInfo Struct Reference	160
11.50.1 Detailed Description	160
11.51 cbe::delegate::object::MoveDelegate::ErrorInfo Struct Reference	161
11.51.1 Detailed Description	161
11.52 cbe::delegate::object::RemoveDelegate::ErrorInfo Struct Reference	162
11.52.1 Detailed Description	162
11.53 cbe::delegate::object::RenameDelegate::ErrorInfo Struct Reference	163
11.53.1 Detailed Description	163
11.54 cbe::delegate::PublishDelegate::ErrorInfo Struct Reference	164
11.54.1 Detailed Description	164
11.55 cbe::delegate::QueryDelegate::ErrorInfo Struct Reference	165
11.55.1 Detailed Description	165
11.56 cbe::delegate::RemoveRoleDelegate::ErrorInfo Struct Reference	166
11.56.1 Detailed Description	166
11.57 cbe::delegate::RemoveRoleMemberDelegate::ErrorInfo Struct Reference	167
11.57.1 Detailed Description	167
11.58 cbe::delegate::SearchGroupsDelegate::ErrorInfo Struct Reference	168
11.58.1 Detailed Description	168
11.59 cbe::delegate::ShareDelegate::ErrorInfo Struct Reference	169
11.59.1 Detailed Description	169
11.60 cbe::delegate::SubscribeDelegate::ErrorInfo Struct Reference	170
11.60.1 Detailed Description	170
11.61 cbe::delegate::UnBanDelegate::ErrorInfo Struct Reference	171
11.61.1 Detailed Description	171
11.62 cbe::delegate::UnPublishDelegate::ErrorInfo Struct Reference	172
11.62.1 Detailed Description	172
11.63 cbe::delegate::UnShareDelegate::ErrorInfo Struct Reference	173
11.63.1 Detailed Description	173
11.64 cbe::delegate::UnSubscribeDelegate::ErrorInfo Struct Reference	174
11.64.1 Detailed Description	174
11.65 cbe::delegate::UpdateKeyValuesDelegate::ErrorInfo Struct Reference	175

11.65.1 Detailed Description	175
11.66 cbe::delegate::UploadDelegate::ErrorInfo Struct Reference	176
11.66.1 Detailed Description	176
11.67 cbe::util::ErrorInfo Struct Reference	177
11.68 cbe::util::ErrorInfoImpl< ErrorT > Struct Template Reference	178
11.69 cbe::delegate::AclDelegate::Exception Struct Reference	179
11.69.1 Detailed Description	180
11.70 cbe::delegate::AddRoleMemberDelegate::Exception Struct Reference	180
11.70.1 Detailed Description	182
11.71 cbe::delegate::BanDelegate::Exception Struct Reference	182
11.71.1 Detailed Description	183
11.72 cbe::delegate::container::MoveDelegate::Exception Struct Reference	183
11.72.1 Detailed Description	185
11.73 cbe::delegate::container::RemoveDelegate::Exception Struct Reference	185
11.73.1 Detailed Description	186
11.74 cbe::delegate::container::RenameDelegate::Exception Struct Reference	186
11.74.1 Detailed Description	188
11.75 cbe::delegate::CreateAccountDelegate::Exception Struct Reference	188
11.75.1 Detailed Description	189
11.76 cbe::delegate::CreateContainerDelegate::Exception Struct Reference	189
11.76.1 Detailed Description	191
11.77 cbe::delegate::CreateGroupDelegate::Exception Struct Reference	191
11.77.1 Detailed Description	192
11.78 cbe::delegate::CreateObjectDelegate::Exception Struct Reference	192
11.78.1 Detailed Description	194
11.79 cbe::delegate::CreateRoleDelegate::Exception Struct Reference	194
11.79.1 Detailed Description	195
11.80 cbe::delegate::DownloadBinaryDelegate::Exception Struct Reference	195
11.80.1 Detailed Description	197
11.81 cbe::delegate::DownloadDelegate::Exception Struct Reference	197
11.81.1 Detailed Description	198
11.82 cbe::delegate::GetStreamsDelegate::Exception Struct Reference	198
11.82.1 Detailed Description	200
11.83 cbe::delegate::GetSubscriptionsDelegate::Exception Struct Reference	200
11.83.1 Detailed Description	201
11.84 cbe::delegate::group::JoinDelegate::Exception Struct Reference	201
11.84.1 Detailed Description	203
11.85 cbe::delegate::group::RemoveDelegate::Exception Struct Reference	203
11.85.1 Detailed Description	204
11.86 cbe::delegate::group::RenameDelegate::Exception Struct Reference	204
11.86.1 Detailed Description	206
11.87 cbe::delegate::JoinDelegate::Exception Struct Reference	206

11.87.1 Detailed Description	207
11.88 cbe::delegate::KickDelegate::Exception Struct Reference	207
11.88.1 Detailed Description	209
11.89 cbe::delegate::LeaveDelegate::Exception Struct Reference	209
11.89.1 Detailed Description	210
11.90 cbe::delegate::ListGroupDelegate::Exception Struct Reference	210
11.90.1 Detailed Description	212
11.91 cbe::delegate::ListMembersDelegate::Exception Struct Reference	212
11.91.1 Detailed Description	213
11.92 cbe::delegate::ListRolesDelegate::Exception Struct Reference	213
11.92.1 Detailed Description	215
11.93 cbe::delegate::ListSharesDelegate::Exception Struct Reference	215
11.93.1 Detailed Description	216
11.94 cbe::delegate::LoginDelegate::Exception Struct Reference	216
11.94.1 Detailed Description	218
11.95 cbe::delegate::object::MoveDelegate::Exception Struct Reference	218
11.95.1 Detailed Description	219
11.96 cbe::delegate::object::RemoveDelegate::Exception Struct Reference	219
11.96.1 Detailed Description	221
11.97 cbe::delegate::object::RenameDelegate::Exception Struct Reference	221
11.97.1 Detailed Description	222
11.98 cbe::delegate::PublishDelegate::Exception Struct Reference	222
11.98.1 Detailed Description	224
11.99 cbe::delegate::QueryDelegate::Exception Struct Reference	224
11.99.1 Detailed Description	225
11.100 cbe::delegate::QueryJoinDelegate::Exception Struct Reference	225
11.100.1 Detailed Description	227
11.101 cbe::delegate::RemoveRoleDelegate::Exception Struct Reference	227
11.101.1 Detailed Description	228
11.102 cbe::delegate::RemoveRoleMemberDelegate::Exception Struct Reference	228
11.102.1 Detailed Description	230
11.103 cbe::delegate::SearchGroupsDelegate::Exception Struct Reference	230
11.103.1 Detailed Description	231
11.104 cbe::delegate::ShareDelegate::Exception Struct Reference	231
11.104.1 Detailed Description	233
11.105 cbe::delegate::SubscribeDelegate::Exception Struct Reference	233
11.105.1 Detailed Description	234
11.106 cbe::delegate::UnBanDelegate::Exception Struct Reference	234
11.106.1 Detailed Description	236
11.107 cbe::delegate::UnPublishDelegate::Exception Struct Reference	236
11.107.1 Detailed Description	237
11.108 cbe::delegate::UnShareDelegate::Exception Struct Reference	237

11.108.1 Detailed Description	239
11.109 cbe::delegate::UnSubscribeDelegate::Exception Struct Reference	239
11.109.1 Detailed Description	240
11.110 cbe::delegate::UpdateKeyValuesDelegate::Exception Struct Reference	240
11.110.1 Detailed Description	242
11.111 cbe::delegate::UploadDelegate::Exception Struct Reference	242
11.111.1 Detailed Description	243
11.112 cbe::QueryChain::Exception Struct Reference	244
11.113 cbe::util::Exception Struct Reference	244
11.113.1 Detailed Description	246
11.113.2 Member Function Documentation	246
11.113.2.1 derivedCbeException()	246
11.114 cbe::util::ExceptionImpl< ErrorInfoT > Struct Template Reference	246
11.114.1 Detailed Description	248
11.115 cbe::Filter Class Reference	248
11.115.1 Detailed Description	249
11.115.2 Member Function Documentation	249
11.115.2.1 getDataType()	249
11.115.2.2 getQuery()	249
11.115.2.3 getOffset()	250
11.115.2.4 getItemOrder()	250
11.115.2.5 setDataType()	250
11.115.2.6 setQuery()	250
11.115.2.7 setAscending()	250
11.115.2.8 setDeleted()	251
11.115.2.9 setContainerFirst()	251
11.115.2.10 setOffset()	251
11.115.2.11 setCount()	251
11.115.2.12 setItemOrder()	252
11.115.2.13 setByPassCache()	252
11.116 cbe::delegate::GetStreamsDelegate Class Reference	252
11.116.1 Detailed Description	253
11.116.2 Member Function Documentation	253
11.116.2.1 onGetStreamsSuccess()	253
11.116.2.2 onGetStreamsError()	253
11.117 cbe::delegate::GetSubscriptionsDelegate Class Reference	253
11.117.1 Detailed Description	253
11.117.2 Member Function Documentation	254
11.117.2.1 onGetSubscriptionsSuccess()	254
11.117.2.2 onGetSubscriptionsError()	254
11.118 cbe::Group Class Reference	254
11.118.1 Detailed Description	257

11.118.2 Member Typedef Documentation	257
11.118.2.1 CreateGroupDelegatePtr	257
11.118.2.2 CreateGroupException	257
11.118.2.3 CreateGroupError	257
11.118.2.4 JoinDelegatePtr	257
11.118.2.5 JoinException	258
11.118.2.6 JoinError	258
11.118.2.7 LeaveDelegatePtr	258
11.118.2.8 LeaveException	258
11.118.2.9 LeaveError	258
11.118.2.10 RemoveDelegatePtr	258
11.118.2.11 RemoveException	258
11.118.2.12 RemoveError	258
11.118.2.13 RenameDelegatePtr	258
11.118.2.14 RenameException	258
11.118.2.15 RenameError	258
11.118.2.16 ListMembersDelegatePtr	259
11.118.2.17 ListMembersException	259
11.118.2.18 ListMembersError	259
11.118.2.19 ListBannedMembersDelegatePtr	259
11.118.2.20 ListBannedMembersException	259
11.118.2.21 ListBannedMembersError	259
11.118.2.22 ListRolesDelegatePtr	259
11.118.2.23 ListRolesException	259
11.118.2.24 ListRolesError	259
11.118.2.25 CreateRoleDelegatePtr	259
11.118.2.26 CreateRoleException	259
11.118.2.27 CreateRoleError	260
11.118.2.28 RemoveRoleDelegatePtr	260
11.118.2.29 RemoveRoleException	260
11.118.2.30 RemoveRoleError	260
11.118.3 Constructor & Destructor Documentation	260
11.118.3.1 Group()	260
11.118.4 Member Function Documentation	260
11.118.4.1 name()	260
11.118.4.2 id()	260
11.118.4.3 parentId()	260
11.118.4.4 groupContainer()	260
11.118.4.5 getVisibility()	261
11.118.4.6 joined()	261
11.118.4.7 requests()	261
11.118.4.8 createGroup() [1/3]	261

11.118.4.9 createGroup() [2/3]	261
11.118.4.10 createGroup() [3/3]	262
11.118.4.11 join() [1/3]	262
11.118.4.12 join() [2/3]	263
11.118.4.13 join() [3/3]	263
11.118.4.14 leave() [1/3]	263
11.118.4.15 leave() [2/3]	264
11.118.4.16 leave() [3/3]	264
11.118.4.17 remove() [1/3]	264
11.118.4.18 remove() [2/3]	265
11.118.4.19 remove() [3/3]	265
11.118.4.20 rename() [1/3]	265
11.118.4.21 rename() [2/3]	266
11.118.4.22 rename() [3/3]	266
11.118.4.23 listMembers() [1/3]	266
11.118.4.24 listMembers() [2/3]	267
11.118.4.25 listMembers() [3/3]	267
11.118.4.26 listBannedMembers() [1/3]	267
11.118.4.27 listBannedMembers() [2/3]	267
11.118.4.28 listBannedMembers() [3/3]	268
11.118.4.29 listRoles() [1/3]	268
11.118.4.30 listRoles() [2/3]	268
11.118.4.31 listRoles() [3/3]	269
11.118.4.32 createRole() [1/3]	269
11.118.4.33 createRole() [2/3]	269
11.118.4.34 createRole() [3/3]	270
11.118.4.35 removeRole() [1/3]	270
11.118.4.36 removeRole() [2/3]	270
11.118.4.37 removeRole() [3/3]	271
11.118.4.38 operator bool()	271
11.119 cbe::GroupFilter Class Reference	271
11.119.1 Detailed Description	272
11.119.2 Member Function Documentation	272
11.119.2.1 getQuery()	272
11.119.2.2 getFilter()	272
11.119.2.3 getAscending()	272
11.119.2.4 getDeleted()	272
11.119.2.5 getPublicFirst()	272
11.119.2.6 getOffset()	272
11.119.2.7 getCount()	272
11.119.2.8 getOrder()	273
11.119.2.9 getGroupOrder()	273

11.119.2.10 setQuery()	273
11.119.2.11 setFilter()	273
11.119.2.12 setAscending()	273
11.119.2.13 setDeleted()	273
11.119.2.14 setOffset()	273
11.119.2.15 setCount()	273
11.119.3 Friends And Related Function Documentation	273
11.119.3.1 operator<<	274
11.120 cbe::GroupManager Class Reference	274
11.120.1 Detailed Description	275
11.120.2 Member Typedef Documentation	275
11.120.2.1 ListGroupsDelegatePtr	275
11.120.2.2 ListGroupsException	275
11.120.2.3 ListGroupsError	275
11.120.2.4 SearchGroupsDelegatePtr	275
11.120.2.5 SearchGroupsException	275
11.120.2.6 SearchGroupsError	275
11.120.3 Constructor & Destructor Documentation	275
11.120.3.1 GroupManager()	275
11.120.4 Member Function Documentation	275
11.120.4.1 listGroups() [1/3]	276
11.120.4.2 listGroups() [2/3]	276
11.120.4.3 listGroups() [3/3]	276
11.120.4.4 searchGroups() [1/3]	276
11.120.4.5 searchGroups() [2/3]	277
11.120.4.6 searchGroups() [3/3]	277
11.120.4.7 getTenantId()	277
11.120.4.8 operator bool()	278
11.121 cbe::GroupQueryResult Class Reference	278
11.121.1 Detailed Description	278
11.121.2 Member Function Documentation	278
11.121.2.1 filter()	278
11.121.2.2 getGroupsSnapshot()	279
11.121.2.3 GroupsLoaded()	279
11.121.2.4 totalCount()	279
11.121.2.5 containsGroup()	279
11.122 cbe::delegate::ICloudBackendListener Class Reference	279
11.122.1 Member Function Documentation	280
11.122.1.1 onRemoteObjectAdded()	280
11.122.1.2 onRemoteObjectMoved()	280
11.122.1.3 onRemoteObjectRemoved()	280
11.122.1.4 onRemoteObjectRenamed()	281

11.122.1.5 onRemoteContainerAdded()	281
11.122.1.6 onRemoteContainerMoved()	281
11.122.1.7 onRemoteContainerRemoved()	281
11.122.1.8 onRemoteContainerRenamed()	281
11.123 cbe::Item Class Reference	282
11.123.1 Detailed Description	283
11.123.2 Member Function Documentation	283
11.123.2.1 getShareIds()	283
11.123.2.2 getShareFromUserId()	283
11.123.2.3 getUserFromShareId()	283
11.123.2.4 aclTag()	283
11.123.2.5 description()	283
11.123.2.6 id()	284
11.123.2.7 parentId()	284
11.123.2.8 oldParentId()	284
11.123.2.9 name()	284
11.123.2.10 path()	284
11.123.2.11 ownerId()	284
11.123.2.12 driveId()	284
11.123.2.13 username()	284
11.123.2.14 idLoaded()	284
11.123.2.15 dataLoaded()	284
11.123.2.16 created()	284
11.123.2.17 updated()	284
11.123.2.18 deleted()	285
11.123.2.19 type()	285
11.123.2.20 hasPublished()	285
11.123.2.21 getPublished()	285
11.123.2.22 hasSubscribe()	285
11.123.2.23 getSubscribe()	285
11.123.2.24 aclMap()	285
11.123.2.25 operator bool()	285
11.124 cbe::delegate::ItemError Class Reference	286
11.124.1 Detailed Description	286
11.124.2 Member Function Documentation	286
11.124.2.1 onItemError()	286
11.125 cbe::delegate::group::JoinDelegate Class Reference	286
11.125.1 Detailed Description	287
11.125.2 Member Function Documentation	287
11.125.2.1 onJoinSuccess()	287
11.125.2.2 onJoinError()	287
11.126 cbe::delegate::JoinDelegate Class Reference	287

11.126.1 Detailed Description	287
11.126.2 Member Typedef Documentation	288
11.126.2.1 ErrorInfo	288
11.126.3 Member Function Documentation	288
11.126.3.1 onJoinSuccess()	288
11.126.3.2 onJoinError()	288
11.127 cbe::delegate::JoinError Class Reference	288
11.127.1 Detailed Description	289
11.127.2 Member Function Documentation	289
11.127.2.1 onJoinError()	289
11.128 cbe::delegate::KickDelegate Class Reference	289
11.128.1 Detailed Description	289
11.128.2 Member Function Documentation	289
11.128.2.1 onKickSuccess()	290
11.128.2.2 onKickError()	290
11.129 cbe::delegate::KickSuccess Class Reference	290
11.129.1 Detailed Description	290
11.130 cbe::delegate::LeaveDelegate Class Reference	290
11.130.1 Detailed Description	291
11.130.2 Member Function Documentation	291
11.130.2.1 onLeaveSuccess()	291
11.130.2.2 onLeaveError()	291
11.131 cbe::delegate::LeaveSuccess Class Reference	291
11.131.1 Member Function Documentation	291
11.131.1.1 operator bool()	292
11.132 cbe::delegate::ListGroupDelegate Class Reference	292
11.132.1 Detailed Description	292
11.132.2 Member Function Documentation	292
11.132.2.1 onListGroupSuccess()	292
11.132.2.2 onListGroupError()	292
11.133 cbe::delegate::ListMembersDelegate Class Reference	293
11.133.1 Detailed Description	293
11.133.2 Member Function Documentation	293
11.133.2.1 onListMembersSuccess()	293
11.133.2.2 onListMembersError()	293
11.134 cbe::delegate::ListRolesDelegate Class Reference	294
11.134.1 Detailed Description	294
11.134.2 Member Function Documentation	294
11.134.2.1 onListRolesSuccess()	294
11.134.2.2 onListRolesError()	294
11.135 cbe::delegate::ListSharesDelegate Class Reference	295
11.135.1 Detailed Description	295

11.135.2 Member Function Documentation	295
11.135.2.1 onListSharesSuccess()	295
11.135.2.2 onListSharesError()	295
11.136 cbe::delegate::LoadError Class Reference	295
11.136.1 Detailed Description	296
11.136.2 Member Function Documentation	296
11.136.2.1 onLoadError()	296
11.137 cbe::delegate::LoginDelegate Class Reference	296
11.137.1 Detailed Description	297
11.137.2 Member Typedef Documentation	297
11.137.2.1 Success	297
11.137.3 Member Function Documentation	297
11.137.3.1 onLoginSuccess()	297
11.137.3.2 onLoginError()	297
11.138 cbe::MatchesID< IdT > Class Template Reference	297
11.138.1 Detailed Description	298
11.139 cbe::Member Class Reference	298
11.139.1 Detailed Description	299
11.139.2 Member Typedef Documentation	299
11.139.2.1 KickDelegatePtr	299
11.139.2.2 KickException	299
11.139.2.3 KickError	299
11.139.2.4 BanDelegatePtr	299
11.139.2.5 BanException	300
11.139.2.6 BanError	300
11.139.2.7 UnBanDelegatePtr	300
11.139.2.8 UnBanException	300
11.139.2.9 UnBanError	300
11.139.3 Constructor & Destructor Documentation	300
11.139.3.1 Member()	300
11.139.4 Member Function Documentation	300
11.139.4.1 kick() [1/3]	300
11.139.4.2 kick() [2/3]	301
11.139.4.3 kick() [3/3]	301
11.139.4.4 ban() [1/3]	301
11.139.4.5 ban() [2/3]	302
11.139.4.6 ban() [3/3]	302
11.139.4.7 unBan() [1/3]	302
11.139.4.8 unBan() [2/3]	302
11.139.4.9 unBan() [3/3]	303
11.139.4.10 memberId()	303
11.139.4.11 name()	303

11.139.4.12 visibility()	303
11.139.4.13 groupId()	303
11.139.4.14 operator bool()	304
11.140 cbe::delegate::container::MoveDelegate Class Reference	304
11.140.1 Detailed Description	304
11.140.2 Member Function Documentation	304
11.140.2.1 onMoveSuccess()	304
11.140.2.2 onMoveError()	305
11.141 cbe::delegate::object::MoveDelegate Class Reference	305
11.141.1 Detailed Description	305
11.141.2 Member Function Documentation	305
11.141.2.1 onMoveSuccess()	305
11.141.2.2 onMoveError()	305
11.142 cbe::Object Class Reference	306
11.142.1 Detailed Description	309
11.142.2 Member Typedef Documentation	309
11.142.2.1 MoveDelegatePtr	310
11.142.2.2 MoveException	310
11.142.2.3 MoveError	310
11.142.2.4 RenameDelegatePtr	310
11.142.2.5 RenameException	310
11.142.2.6 RenameError	310
11.142.2.7 RemoveDelegatePtr	310
11.142.2.8 RemoveException	310
11.142.2.9 RemoveError	310
11.142.2.10 DownloadDelegatePtr	310
11.142.2.11 DownloadException	311
11.142.2.12 DownloadError	311
11.142.2.13 DownloadBinaryDelegatePtr	311
11.142.2.14 DownloadBinaryException	311
11.142.2.15 DownloadBinaryError	311
11.142.2.16 UploadDelegatePtr	311
11.142.2.17 UploadException	311
11.142.2.18 UploadError	312
11.142.2.19 UpdateKeyValuesDelegatePtr	312
11.142.2.20 UpdateKeyValuesException	312
11.142.2.21 UpdateKeyValuesError	312
11.142.2.22 GetStreamsDelegatePtr	312
11.142.2.23 Streams	312
11.142.2.24 GetStreamsException	312
11.142.2.25 GetStreamsError	312
11.142.2.26 GetAclDelegatePtr	312

11.142.2.27 GetAclException	312
11.142.2.28 GetAclError	313
11.142.2.29 SetAclDelegatePtr	313
11.142.2.30 SetAclException	313
11.142.2.31 SetAclError	313
11.142.2.32 ShareDelegatePtr	313
11.142.2.33 ShareException	313
11.142.2.34 ShareError	313
11.142.2.35 UnShareDelegatePtr	313
11.142.2.36 UnShareException	313
11.142.2.37 UnShareError	313
11.142.2.38 PublishDelegatePtr	313
11.142.2.39 PublishException	314
11.142.2.40 PublishError	314
11.142.2.41 UnPublishDelegatePtr	314
11.142.2.42 UnPublishException	314
11.142.2.43 UnPublishError	314
11.142.2.44 UnSubscribeDelegatePtr	314
11.142.2.45 UnSubscribeException	314
11.142.2.46 UnSubscribeError	314
11.142.3 Member Function Documentation	314
11.142.3.1 move() [1/3]	314
11.142.3.2 move() [2/3]	315
11.142.3.3 move() [3/3]	315
11.142.3.4 rename() [1/3]	315
11.142.3.5 rename() [2/3]	316
11.142.3.6 rename() [3/3]	316
11.142.3.7 remove() [1/3]	316
11.142.3.8 remove() [2/3]	316
11.142.3.9 remove() [3/3]	317
11.142.3.10 download() [1/10]	317
11.142.3.11 download() [2/10]	317
11.142.3.12 download() [3/10]	318
11.142.3.13 download() [4/10]	318
11.142.3.14 download() [5/10]	319
11.142.3.15 download() [6/10]	319
11.142.3.16 download() [7/10]	319
11.142.3.17 download() [8/10]	320
11.142.3.18 download() [9/10]	320
11.142.3.19 download() [10/10]	320
11.142.3.20 downloadStream() [1/5]	320
11.142.3.21 downloadStream() [2/5]	321

11.142.3.22 downloadStream() [3/5]	322
11.142.3.23 downloadStream() [4/5]	322
11.142.3.24 downloadStream() [5/5]	323
11.142.3.25 uploadStream() [1/5]	323
11.142.3.26 uploadStream() [2/5]	324
11.142.3.27 uploadStream() [3/5]	325
11.142.3.28 uploadStream() [4/5]	326
11.142.3.29 uploadStream() [5/5]	326
11.142.3.30 updateKeyValues() [1/6]	327
11.142.3.31 updateKeyValues() [2/6]	327
11.142.3.32 updateKeyValues() [3/6]	327
11.142.3.33 updateKeyValues() [4/6]	328
11.142.3.34 updateKeyValues() [5/6]	328
11.142.3.35 updateKeyValues() [6/6]	328
11.142.3.36 getStreams() [1/3]	328
11.142.3.37 getStreams() [2/3]	329
11.142.3.38 getStreams() [3/3]	330
11.142.3.39 getMimeType()	330
11.142.3.40 getObjectType()	330
11.142.3.41 getAcl() [1/3]	331
11.142.3.42 getAcl() [2/3]	331
11.142.3.43 getAcl() [3/3]	331
11.142.3.44 setAcl() [1/3]	331
11.142.3.45 setAcl() [2/3]	332
11.142.3.46 setAcl() [3/3]	332
11.142.3.47 share() [1/3]	332
11.142.3.48 share() [2/3]	333
11.142.3.49 share() [3/3]	333
11.142.3.50 unShare() [1/3]	333
11.142.3.51 unShare() [2/3]	334
11.142.3.52 unShare() [3/3]	334
11.142.3.53 publish() [1/3]	334
11.142.3.54 publish() [2/3]	335
11.142.3.55 publish() [3/3]	335
11.142.3.56 unPublish() [1/3]	336
11.142.3.57 unPublish() [2/3]	336
11.142.3.58 unPublish() [3/3]	336
11.142.3.59 unsubscribe() [1/3]	337
11.142.3.60 unsubscribe() [2/3]	337
11.142.3.61 unsubscribe() [3/3]	337
11.143 cbe::util::Optional< T > Class Template Reference	338
11.143.1 Detailed Description	339

11.143.2 Member Function Documentation	339
11.143.2.1 operator bool()	339
11.144 cbe::Publish Class Reference	339
11.144.1 Detailed Description	340
11.144.2 Constructor & Destructor Documentation	340
11.144.2.1 Publish()	340
11.144.3 Member Function Documentation	340
11.144.3.1 setTitle()	340
11.144.3.2 setSecurity()	340
11.144.3.3 setPrivacy()	341
11.144.3.4 setPassword()	341
11.144.3.5 setDescription()	341
11.144.3.6 getTitle()	341
11.144.3.7 getSecurity()	341
11.144.3.8 getPrivacy()	341
11.144.3.9 getPassword()	341
11.144.3.10 getDescription()	342
11.144.3.11 getPublishId()	342
11.144.3.12 operator bool()	342
11.145 cbe::delegate::PublishDelegate Class Reference	342
11.145.1 Detailed Description	342
11.145.2 Member Function Documentation	343
11.145.2.1 onPublishSuccess()	343
11.145.2.2 onPublishError()	343
11.146 cbe::PublishManager Class Reference	343
11.146.1 Detailed Description	343
11.146.2 Constructor & Destructor Documentation	343
11.146.2.1 PublishManager()	343
11.146.3 Member Function Documentation	344
11.146.3.1 getPublished()	344
11.146.3.2 operator bool()	344
11.147 cbe::delegate::PublishSuccess Class Reference	344
11.147.1 Detailed Description	344
11.147.2 Member Function Documentation	344
11.147.2.1 operator bool()	345
11.148 cbe::QueryChain Class Reference	345
11.148.1 Detailed Description	346
11.148.2 Member Function Documentation	346
11.148.2.1 join() [1 / 4]	346
11.148.2.2 join() [2 / 4]	346
11.148.2.3 join() [3 / 4]	346
11.148.2.4 join() [4 / 4]	347

11.148.2.5 getResult()	347
11.149 cbe::QueryChainExt Class Reference	347
11.149.1 Detailed Description	348
11.149.2 Member Function Documentation	348
11.149.2.1 join() [1/8]	349
11.149.2.2 join() [2/8]	349
11.149.2.3 join() [3/8]	349
11.149.2.4 join() [4/8]	349
11.149.2.5 join() [5/8]	350
11.149.2.6 join() [6/8]	350
11.149.2.7 join() [7/8]	350
11.149.2.8 join() [8/8]	350
11.150 cbe::QueryChainSync Class Reference	351
11.150.1 Detailed Description	352
11.150.2 Member Typedef Documentation	352
11.150.2.1 JoinException	352
11.150.3 Member Function Documentation	352
11.150.3.1 join() [1/4]	352
11.150.3.2 join() [2/4]	352
11.150.3.3 join() [3/4]	353
11.150.3.4 join() [4/4]	353
11.150.3.5 getResult()	353
11.151 cbe::delegate::QueryDelegate Class Reference	354
11.151.1 Detailed Description	354
11.151.2 Member Function Documentation	355
11.151.2.1 onQuerySuccess()	355
11.151.2.2 onQueryError()	355
11.152 cbe::delegate::QueryError Class Reference	355
11.152.1 Detailed Description	356
11.153 cbe::delegate::QueryJoinDelegate Class Reference	356
11.153.1 Detailed Description	357
11.153.2 Member Typedef Documentation	357
11.153.2.1 ErrorInfo	357
11.153.3 Member Function Documentation	357
11.153.3.1 onQueryJoinSuccess()	357
11.153.3.2 onQueryJoinError()	357
11.154 cbe::delegate::QueryLoaded Struct Reference	358
11.154.1 Member Function Documentation	358
11.154.1.1 onQuerySuccess()	358
11.155 cbe::QueryResult Class Reference	358
11.155.1 Detailed Description	359
11.155.2 Constructor & Destructor Documentation	359

11.155.2.1 QueryResult()	359
11.155.3 Member Function Documentation	359
11.155.3.1 filter()	359
11.155.3.2 getItemsSnapshot()	360
11.155.3.3 itemsLoaded()	360
11.155.3.4 totalCount()	360
11.155.3.5 objectsLoaded()	360
11.155.3.6 containersLoaded()	360
11.155.3.7 containsItem()	360
11.155.3.8 operator bool()	360
11.156 cbe::delegate::container::RemoveDelegate Class Reference	361
11.156.1 Detailed Description	361
11.156.2 Member Function Documentation	361
11.156.2.1 onRemoveSuccess()	361
11.156.2.2 onRemoveError()	361
11.157 cbe::delegate::group::RemoveDelegate Class Reference	362
11.157.1 Detailed Description	362
11.157.2 Member Function Documentation	362
11.157.2.1 onRemoveSuccess()	362
11.157.2.2 onRemoveError()	362
11.158 cbe::delegate::object::RemoveDelegate Class Reference	362
11.158.1 Detailed Description	363
11.158.2 Member Function Documentation	363
11.158.2.1 onRemoveSuccess()	363
11.158.2.2 onRemoveError()	363
11.159 cbe::delegate::RemoveRoleDelegate Class Reference	363
11.159.1 Detailed Description	364
11.159.2 Member Function Documentation	364
11.159.2.1 onRemoveRoleSuccess()	364
11.159.2.2 onRemoveRoleError()	364
11.160 cbe::delegate::RemoveRoleMemberDelegate Class Reference	364
11.160.1 Detailed Description	365
11.160.2 Member Function Documentation	365
11.160.2.1 onRemoveRoleMemberSuccess()	365
11.160.2.2 onRemoveRoleMemberError()	365
11.161 cbe::delegate::container::RemoveSuccess Class Reference	365
11.161.1 Detailed Description	365
11.162 cbe::delegate::group::RemoveSuccess Class Reference	365
11.162.1 Detailed Description	366
11.163 cbe::delegate::object::RemoveSuccess Class Reference	366
11.163.1 Detailed Description	366
11.163.2 Member Function Documentation	366

11.163.2.1 operator bool()	366
11.164 cbe::delegate::container::RenameDelegate Class Reference	366
11.164.1 Detailed Description	367
11.164.2 Member Function Documentation	367
11.164.2.1 onRenameSuccess()	367
11.164.2.2 onRenameError()	367
11.165 cbe::delegate::group::RenameDelegate Class Reference	367
11.165.1 Detailed Description	368
11.165.2 Member Function Documentation	368
11.165.2.1 onRenameSuccess()	368
11.165.2.2 onRenameError()	368
11.166 cbe::delegate::object::RenameDelegate Class Reference	368
11.166.1 Detailed Description	369
11.166.2 Member Function Documentation	369
11.166.2.1 onRenameSuccess()	369
11.166.2.2 onRenameError()	369
11.167 cbe::delegate::group::RenameSuccess Class Reference	369
11.167.1 Detailed Description	370
11.167.2 Member Function Documentation	370
11.167.2.1 operator bool()	370
11.168 cbe::Request Struct Reference	370
11.168.1 Detailed Description	370
11.169 cbe::Role Class Reference	370
11.169.1 Detailed Description	372
11.169.2 Member Typedef Documentation	372
11.169.2.1 ListMembersDelegatePtr	372
11.169.2.2 ListMembersException	372
11.169.2.3 ListMembersError	372
11.169.2.4 AddRoleMemberDelegatePtr	372
11.169.2.5 AddRoleMemberException	372
11.169.2.6 AddRoleMemberError	372
11.169.2.7 RemoveRoleMemberDelegatePtr	372
11.169.2.8 RemoveRoleMemberException	372
11.169.2.9 RemoveRoleMemberError	373
11.169.3 Constructor & Destructor Documentation	373
11.169.3.1 Role()	373
11.169.4 Member Function Documentation	373
11.169.4.1 name()	373
11.169.4.2 id()	373
11.169.4.3 groupId()	373
11.169.4.4 listMembers() [1/3]	373
11.169.4.5 listMembers() [2/3]	373

11.169.4.6 listMembers() [3/3]	374
11.169.4.7 addRoleMember() [1/3]	374
11.169.4.8 addRoleMember() [2/3]	374
11.169.4.9 addRoleMember() [3/3]	375
11.169.4.10 removeRoleMember() [1/3]	375
11.169.4.11 removeRoleMember() [2/3]	375
11.169.4.12 removeRoleMember() [3/3]	376
11.169.4.13 operator bool()	376
11.170 cbe::delegate::SearchGroupsDelegate Class Reference	376
11.170.1 Detailed Description	377
11.170.2 Member Function Documentation	377
11.170.2.1 onSearchGroupsSuccess()	377
11.170.2.2 onSearchGroupsError()	377
11.171 cbe::ShareData Struct Reference	377
11.171.1 Detailed Description	377
11.172 cbe::delegate::ShareDelegate Class Reference	377
11.172.1 Detailed Description	378
11.172.2 Member Function Documentation	378
11.172.2.1 onShareSuccess()	378
11.172.2.2 onShareError()	378
11.173 cbe::ShareManager Class Reference	378
11.173.1 Detailed Description	379
11.173.2 Member Typedef Documentation	379
11.173.2.1 ListSharesDelegatePtr	379
11.173.2.2 ListSharesException	380
11.173.2.3 ListSharesError	380
11.173.2.4 ListMySharesException	380
11.173.2.5 ListMySharesError	380
11.173.3 Constructor & Destructor Documentation	380
11.173.3.1 ShareManager()	380
11.173.4 Member Function Documentation	380
11.173.4.1 listAvailableShares() [1/3]	380
11.173.4.2 listAvailableShares() [2/3]	380
11.173.4.3 listAvailableShares() [3/3]	381
11.173.4.4 listMyShares() [1/3]	381
11.173.4.5 listMyShares() [2/3]	381
11.173.4.6 listMyShares() [3/3]	382
11.173.4.7 operator bool()	382
11.174 cbe::delegate::ShareSuccess Class Reference	382
11.174.1 Detailed Description	383
11.174.2 Member Function Documentation	383
11.174.2.1 operator bool()	383

11.175 cbe::Stream Class Reference	383
11.175.1 Detailed Description	383
11.175.2 Member Data Documentation	383
11.175.2.1 streamId	383
11.175.2.2 length	383
11.176 cbe::Subscribe Class Reference	383
11.176.1 Detailed Description	384
11.176.2 Member Function Documentation	384
11.176.2.1 getDate()	384
11.176.2.2 getTitle()	384
11.176.2.3 getDescription()	384
11.176.2.4 getPassword()	384
11.176.2.5 getSecurity()	384
11.176.2.6 getPrivacy()	385
11.176.2.7 getSubscribeId()	385
11.176.2.8 getPublishId()	385
11.176.2.9 getOwner()	385
11.176.2.10 unsubscribe()	385
11.177 cbe::delegate::SubscribeDelegate Class Reference	385
11.177.1 Detailed Description	385
11.177.2 Member Function Documentation	385
11.177.2.1 onSubscribeSuccess()	386
11.177.2.2 onSubscribeError()	386
11.178 cbe::SubscribeManager Class Reference	386
11.178.1 Detailed Description	387
11.178.2 Member Typedef Documentation	387
11.178.2.1 GetSubscriptionsDelegatePtr	387
11.178.2.2 GetSubscriptionsException	387
11.178.2.3 GetSubscriptionsError	387
11.178.2.4 SubscribeDelegatePtr	387
11.178.2.5 SubscribeException	388
11.178.2.6 SubscribeError	388
11.178.3 Member Function Documentation	388
11.178.3.1 getSubscriptions() [1/3]	388
11.178.3.2 getSubscriptions() [2/3]	388
11.178.3.3 getSubscriptions() [3/3]	388
11.178.3.4 subscribe() [1/5]	389
11.178.3.5 subscribe() [2/5]	389
11.178.3.6 subscribe() [3/5]	390
11.178.3.7 subscribe() [4/5]	390
11.178.3.8 subscribe() [5/5]	391
11.179 cbe::delegate::TransferError Class Reference	391

11.179.1 Detailed Description	392
11.180 cbe::delegate::UnBanDelegate Class Reference	392
11.180.1 Detailed Description	392
11.180.2 Member Function Documentation	392
11.180.2.1 onUnBanSuccess()	392
11.180.2.2 onUnBanError()	393
11.181 cbe::delegate::UnBanSuccess Class Reference	393
11.181.1 Detailed Description	393
11.182 cbe::delegate::UnPublishDelegate Class Reference	393
11.182.1 Detailed Description	394
11.182.2 Member Function Documentation	394
11.182.2.1 onUnPublishSuccess()	394
11.182.2.2 onUnPublishError()	394
11.183 cbe::delegate::UnPublishSuccess Class Reference	394
11.183.1 Detailed Description	394
11.183.2 Member Function Documentation	395
11.183.2.1 operator bool()	395
11.184 cbe::delegate::UnShareDelegate Class Reference	395
11.184.1 Detailed Description	395
11.184.2 Member Function Documentation	395
11.184.2.1 onUnShareSuccess()	395
11.184.2.2 onUnShareError()	396
11.185 cbe::delegate::UnShareSuccess Class Reference	396
11.185.1 Detailed Description	396
11.185.2 Member Function Documentation	396
11.185.2.1 operator bool()	396
11.186 cbe::delegate::UnSubscribeDelegate Class Reference	396
11.186.1 Detailed Description	397
11.186.2 Member Function Documentation	397
11.186.2.1 onUnSubscribeSuccess()	397
11.186.2.2 onUnSubscribeError()	397
11.187 cbe::delegate::UnSubscribeSuccess Class Reference	397
11.187.1 Detailed Description	398
11.187.2 Member Function Documentation	398
11.187.2.1 operator bool()	398
11.188 cbe::delegate::UpdateKeyValuesDelegate Class Reference	398
11.188.1 Detailed Description	398
11.188.2 Member Function Documentation	398
11.188.2.1 onUpdateKeyValuesSuccess()	398
11.188.2.2 onUpdateKeyValuesError()	399
11.189 cbe::delegate::UploadDelegate Class Reference	399
11.189.1 Detailed Description	399

11.189.2 Member Function Documentation	399
11.189.2.1 onUploadSuccess()	399
11.189.2.2 onUploadError()	400
11.189.2.3 onChunkSent()	400
Index	401

Chapter 1

API for C++

Software Development Kit with the full Synchronous and Asynchronous API for the CloudBackend Singularity database service managing Data in Motion & Data at Rest.



For more information, see also the [CloudBackend web site](#)

Document	
Version	2.1.4
Extension	Sync & Async API
Date	2025-02-13

Copyright © CloudBackend AB

Chapter 2

Class diagrams

2.1 Overview

These are the main classes available when writing programs using the CloudBackend SDK.

The CloudBackend class is the core object and entry point into using the SDK.

The classes are represented as UML diagrams below.

2.2 CloudBackend - core object

Creating an instance of CloudBackend will be the first thing to do.

In order to initiate the core CloudBackend object, you need to call the `login()` method on the CloudBackend class. This will return the CloudBackend object instance that holds the currently logged in identity (user) and the databases available for that identity.

Once signed in the CloudBackend object will be able to access the identity's account, containers, objects and other functionality.

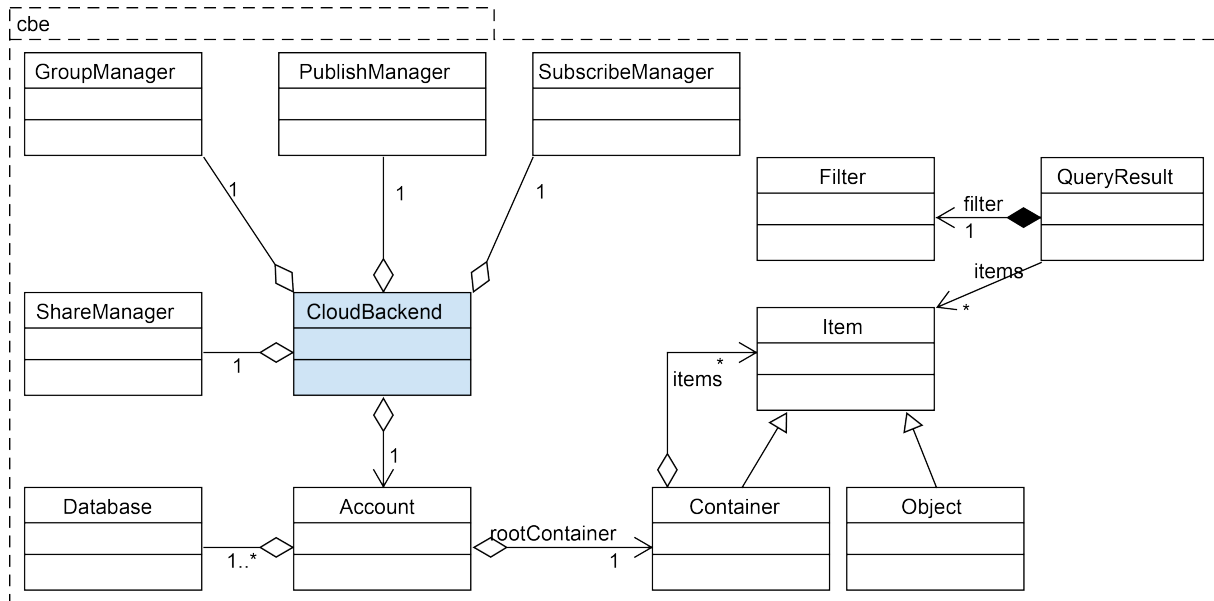


Figure 2.1 UML representation of the major classes in the SDK

2.3 Object

More details around the Object class.

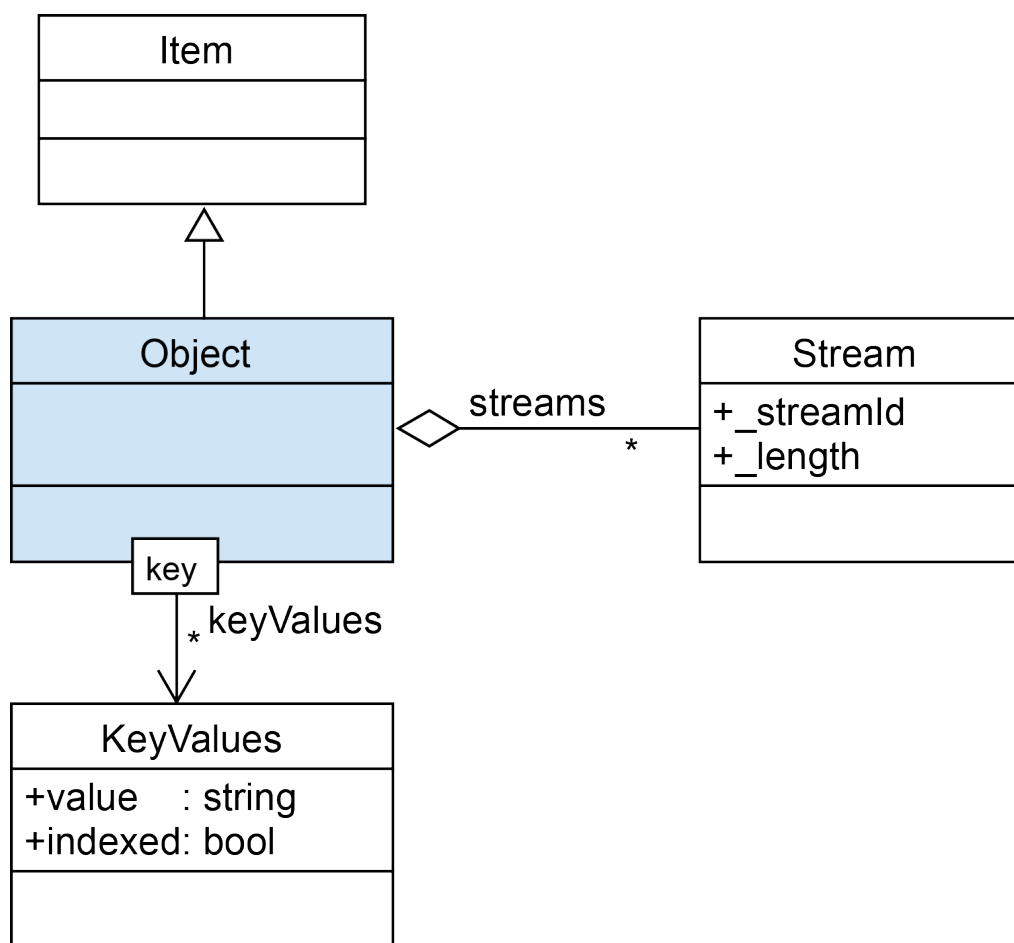


Figure 2.2 Object related classes

2.4 Group

More details around the Group class.

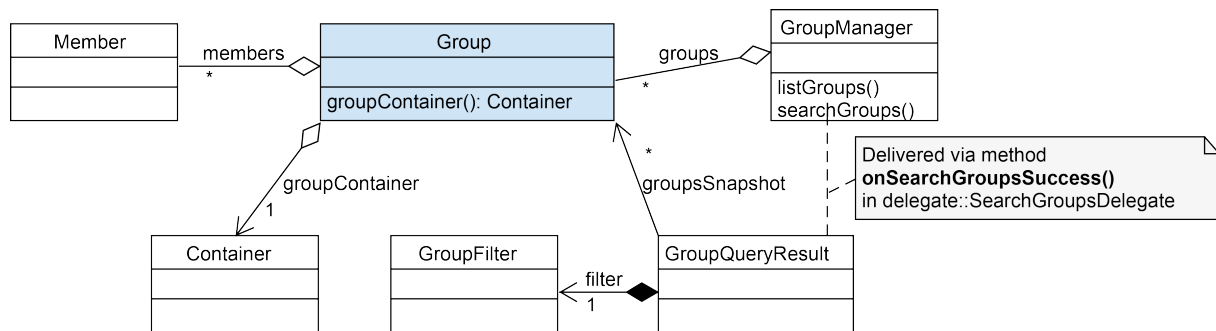


Figure 2.3 Group related classes

Chapter 3

Sync vs. Async

3.1 Choosing a Synchronous or Asynchronous coding style

Many calls to the SDK can return either a synchronous or an asynchronous response. This leads to a decision on how you wish to work based on priorities. All remote communication within the Singularity Database internally takes place asynchronously. Any updates are first made to the local cache of data, then asynchronously applied to the rest of the Singularity Database outside the SDK. When the remote request completes, a callback is made to the delegate you have implemented.

Synchronous

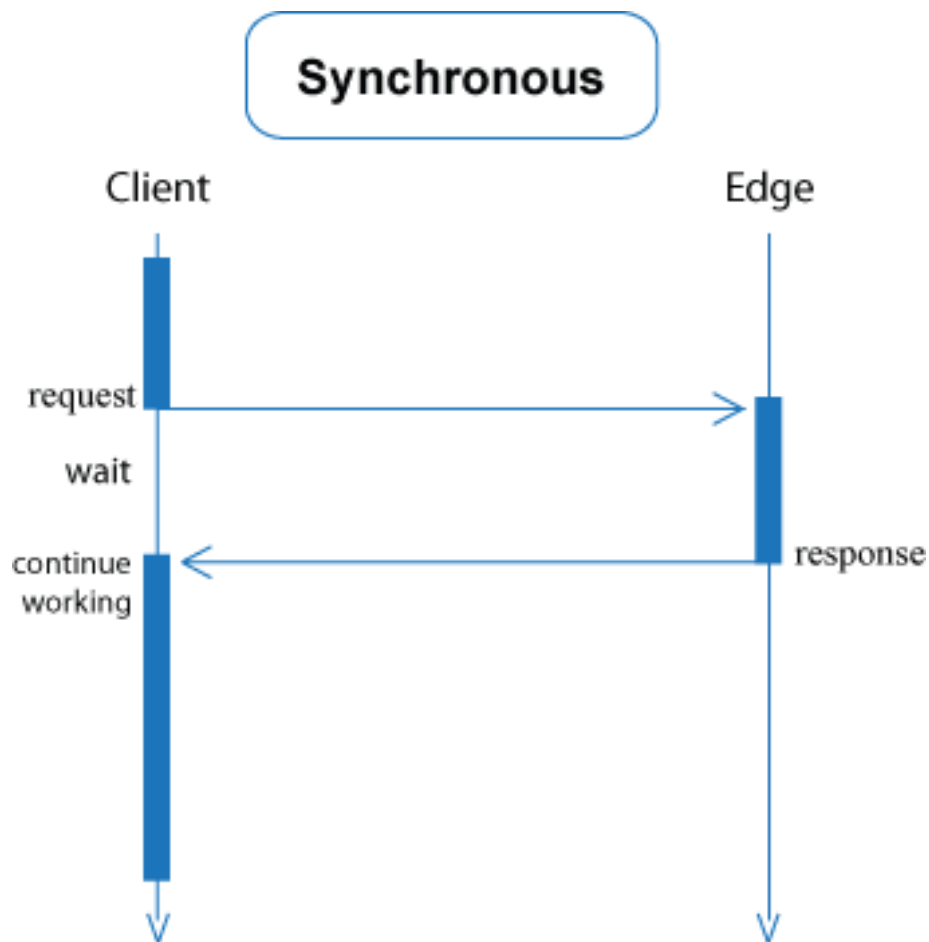


Figure 3.1 Synchronous program sequence

Working synchronously allows one to immediately act on data that has been updated. This, however, means that it is possible someone else is looking at the same data without the updates you just made (inconsistency). The Singularity Database is built on the concept of eventual consistency, meaning that all SDKs viewing or holding the same object will eventually be consistent. Additionally, if a request fails, requests dependent on the success of that request, that appear complete, will be rolled back.

Asynchronous

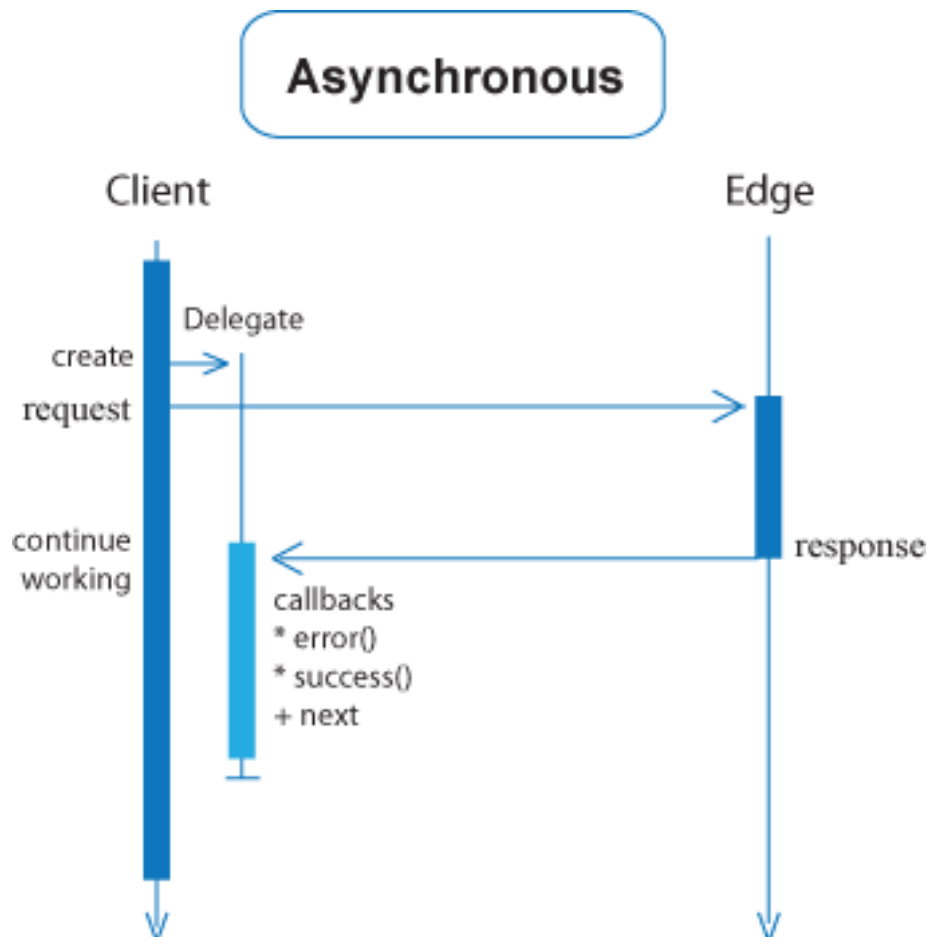


Figure 3.2 Asynchronous program sequence

Working asynchronously means that the main thread may continue to work while there is a separate parallel thread delegate created that will wait for the response, process the response, and optionally make any follow up processing or hand over. This means that the applications you write on top of the SDK will be event-driven instead of being triggered by a main loop. The end user experience will be a user interface that is responsive and does not freeze while waiting for a server response. Working this way guarantees everything presented to the end user has been propagated to the Singularity Database and will be what is retrieved if another client communicates with the database.

Pros and cons

With this SDK the developer is provided with sets of API with both Synchronous and Asynchronous variants of the calls that may communicate with the service in the cloud.

The Synchronous calls will be easier to use, but will freeze operations waiting for the cloud to respond. For a sequential program like a report or a sensor feeding data this is probably a good choice.

For a good interactive user experience in a user interface where multiple events happens in an unpredictable order, it is probably more desirable to write asynchronously on top of the SDK.

It is possible to mix Synchronous and Asynchronous calls in the same program.

Error handling

API calls have two possible outcomes: either success or error. The Synchronous API is provided in two variants that either uses exceptions or returns a result that can be of two types: either the error information or the expected result.

In the Asynchronous API the callback delegate needs to cater for both cases: success and error.

Chapter 4

Sequence diagrams - async

4.1 Login asynchronous UML

Sequence diagram for login using a delegate for asynchronous callbacks.

User can log in by calling login() submitting parameters with the credentials of:

- username
- password
- tenant

Create a Delegate that will get an input-callback of either:

- success
- error

Call the login() with credentials and the pointer to the delegate that is going to handle the callback.

Normally you need to wait for the completion and then, depending on if the result was success or error, take an appropriate action.

4.2 Query asynchronous UML

Sequence diagram for a query [optional filter] using a delegate for asynchronous callbacks.

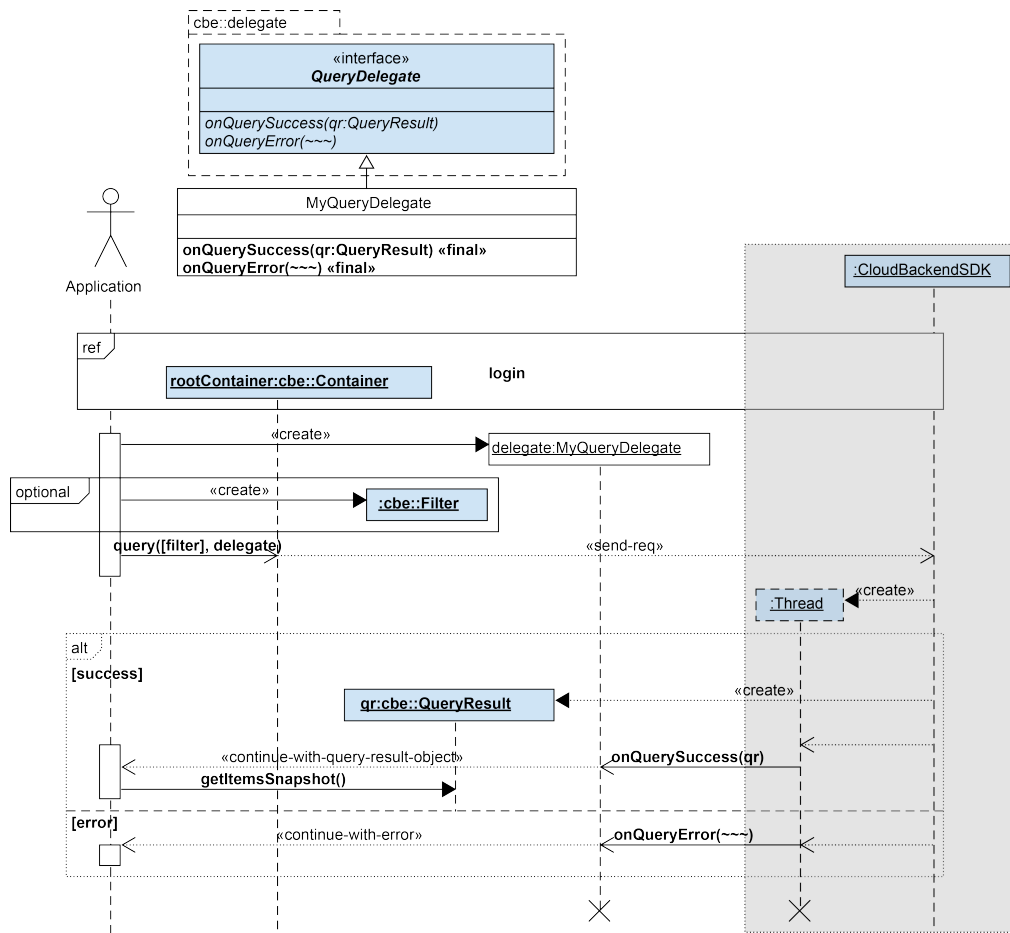


Figure 4.2 Query process sequence diagram

Chapter 5

Example code

For an overview about different coding styles, see the chapter [Sync vs. Async](#) and the [pros_cons](#)

Index of subpages — to sections that contains examples — for inspiration

[Example code - sync - exception](#)

[Example code - sync - non-throw](#)

[Example code - async](#)

[Example code - combined lists grouped by call](#)

A line in a code example with "~~~" indicates that additional code should be added there to handle next action.

Fully working example programs

[Full example - sync \[exception\]](#)

[Full example - sync \[non-throwing\]](#)

[Full example - async](#)

5.1 Example code - sync - exception

A line in a code example with "~~~" indicates that additional code should be added there to handle next action.

5.1.1 login - upload - download - query

1. [login - sync \[exception\] example](#)
2. [upload - sync \[exception\] example](#)
3. [getStreams - sync \[exception\] example](#)
4. [uploadStream - sync \[exception\] example](#)
5. [downloadStream - sync \[exception\] example](#)
6. [query - sync \[exception\] example](#)
7. [databases - tenant container example](#)

5.1.2 Complete program

- [all code - sync \[exception\] example](#)

5.2 Example code - sync - non-throw

A line in a code example with "~~~" indicates that additional code should be added there to handle next action.

5.2.1 login - upload - download - query

1. [login - sync \[non-throwing\] example](#)
2. [upload - sync \[non-throwing\] example](#)
3. [getStreams - sync \[non-throwing\] example](#)
4. [uploadStream - sync \[non-throwing\] example](#)
5. [downloadStream - sync \[non-throwing\] example](#)
6. [query - sync \[non-throwing\] example](#)
7. [databases - tenant container example](#)

5.2.2 Complete program

- [all code - sync \[non-throwing\] example](#)

5.3 Example code - async

Index of links — to sections that contains examples — for inspiration A line in a code example with "~~~" indicates that additional code should be added there to handle next action.

5.3.1 login - upload - download

1. [login - async example](#)
2. [upload - async example](#)
3. [getStreams - async example](#)
4. [uploadStream - async example](#)
5. [downloadStream - async example](#)

5.3.2 query - select

1. [query - async example](#)
2. [QueryDelegate implementation class - async](#)

5.3.3 other - misc

1. [databases - tenant container example](#)

5.3.4 Complete program

- [all code - async example](#)

5.4 Example code - combined lists grouped by call

Index of links — to sections that contains examples — for inspiration A line in a code example with "~~~" indicates that additional code should be added there to handle next action.

5.4.1 login

1. [login - sync \[exception\] example](#)
2. [login - sync \[non-throwing\] example](#)
3. [login - async example](#)

5.4.2 upload

1. [upload - sync \[exception\] example](#)
2. [upload - sync \[non-throwing\] example](#)
3. [upload - async example](#)

5.4.3 getStreams

1. [getStreams - sync \[exception\] example](#)
2. [getStreams - sync \[non-throwing\] example](#)
3. [getStreams - async example](#)

5.4.4 uploadStream

1. [uploadStream - sync \[exception\] example](#)
2. [uploadStream - sync \[non-throwing\] example](#)
3. [uploadStream - async example](#)

5.4.5 downloadStream

1. [downloadStream - sync \[exception\] example](#)
2. [downloadStream - sync \[non-throwing\] example](#)
3. [downloadStream - async example](#)

5.4.6 query - select

1. [query - sync \[exception\] example](#)
2. [query - sync \[non-throwing\] example](#)
3. [query - async example](#)
4. [QueryDelegate implementation class - async only](#)

5.4.7 databases - tenant

1. [databases - tenant container example](#)

5.4.8 Complete program

1. [all code - sync \[exception\] example](#)
2. [all code - sync \[non-throwing\] example](#)
3. [all code - async example](#)

5.5 Full example - sync [exception]

Example program - synchronous - exception

```
#include <algorithm>
#include <fstream>
#include <iostream>
#include "cbe/Account.h"
#include "cbe/CloudBackend.h"
#include "cbe/Container.h"
#include "cbe/Stream.h"
#include "cbe/Object.h"
#include "cbe/QueryChainSync.h"
#include "cbe/delegate/DownloadSuccess.h"
#include "cbe/util/Exception.h"
#include "cbe/util/ErrorInfo.h"
#include "../user_credentials.cpp"

int main(int argc, const char** argv) {
    static constexpr const char testContainer[] = "z-MediumExample";
    cbe::Container rootContainer{cbe::DefaultCtor{}};
    cbe::Container myContainer{cbe::DefaultCtor{}};
    cbe::Filter objectFilter;
    objectFilter.setDataType(cbe::ItemType::Object);
    // ----- login -----
    cbe::CloudBackend cloudBackend{ cbe::DefaultCtor{}};
    try {
        cloudBackend = cbe::CloudBackend::login(username, password, tenant, client);
    }
    // Catching specific LoginException
    catch (cbe::CloudBackend::LoginException& e) {
        std::cout << "Failed to login: error={" << e.what() << '}'
                  << std::endl;
        return 1;
    }
    catch (...) {
        std::cout << "Got other exception!"
                  << std::endl;
        return 9;
    }
    rootContainer = cloudBackend.account().rootContainer();
    std::cout << "login: " << cloudBackend.account().username() << std::endl;
    // ----- check for existing container -----
    cbe::QueryResult resultSet = cloudBackend.account().rootContainer().query();
    for (auto entry : resultSet.getItemsSnapshot()) {
        if (entry.type() == cbe::ItemType::Container &&
            entry.name() == testContainer) {
            std::cout << "Using existing container!" << std::endl;
            myContainer = cbe::CloudBackend::castContainer(entry);
        }
    }
    if (!myContainer) {
        std::cout << "Creating a new container!" << std::endl;
        myContainer = rootContainer.createContainer(testContainer);
    }
    // ----- upload_syncExcept -----
    constexpr const char* const uploadPath = "/tmp/";
    constexpr const char* const myObjectFileName = "Medium_B_a_message";
    const std::string qualFileName = std::string{uploadPath} +
                                     myObjectFileName;
    cbe::Object anObject{cbe::DefaultCtor{}};
    cbe::Object myObject{cbe::DefaultCtor{}};
    cbe::Object::Streams streams;
    std::cout << "Create local file " << qualFileName << std::endl;
    std::ofstream ofs{qualFileName};
    ofs << "Line a\n"
        << "Line b\n"
        << "Line c\n"
        << std::flush;
    ofs.close();
    try {
        {
            std::cout << "Uploading object from "
                      << qualFileName
                      << std::endl;
            myObject = myContainer.upload(qualFileName);
            streams = myObject.getStreams();
        }
        // Using generic Exception for convenience
        catch (const cbe::util::Exception& e) {
            {
                // typeAsString is used to print what type of exception was thrown
                std::cout << "Error!\n" << e.what()
                          << "\nType: " << e.typeAsString() << std::endl;
                cloudBackend.terminate();
                return 2;
            }
        }
    }
    // ----- uploadStream_syncExcept -----
```

```

constexpr const char* const myFile2Name = "Medium_B_an_attachment";
const std::string qualFile2Name = std::string(uploadPath) + myFile2Name;
std::ofstream ofs3{qualFile2Name};
ofs3 << "line 21\n"
      << "line 22\n"
      << "line 23\n"
      << "line 24\n"
      << std::flush;
ofs3.close();
std::cout << "Uploading additional stream from: "
          << qualFile2Name
          << std::endl;
const auto nextStreamId =
    std::max_element(std::begin(streams), std::end(streams),
        [](const cbe::Stream& stream1,
            const cbe::Stream& stream2) {
            return stream1.streamId < stream2.streamId;
        })->streamId + 1;
try
{
    myObject.uploadStream(qualFile2Name, nextStreamId);
    std::cout << "home://"
              << myContainer.name()
              << "/"
              << myObject.name()
              << " now has two streams."
              << std::endl;
    streams = myObject.getStreams();
}
catch (const cbe::util::Exception& e)
{
    std::cout << "Error!\n" << e.what()
              << "\nType: " << e.typeAsString() << std::endl;
    cloudBackend.terminate();
    return 3;
}
// ----- downloadStream_syncExcept -----
std::cout << "Downloading stream:" << std::endl;
constexpr const char* const downloadPath = "/tmp/Medium_B_download_stream_";
for (const auto& stream : streams) {
    const auto path = std::string(downloadPath) +
        std::to_string(stream.streamId) + "_of_";

    try
    {
        std::cout << stream.streamId
                  << " to: "
                  << path
                  << myObject.name()
                  << std::endl;
        auto result = myObject.downloadStream(path, stream);
    }
    catch (const cbe::util::Exception& e)
    {
        std::cout << "Error!\n" << e.what()
                  << "\nType: " << e.typeAsString() << std::endl;
        cloudBackend.terminate();
        return 4;
    }
}
// ----- cloudbackend_query_syncExcept -----
try
{
    std::cout << "Content of "
              << myContainer.name()
              << std::endl;
    auto queryResult = myContainer.query(objectFilter).getQueryResult();
    cbe::QueryResult::ItemsSnapshot itemsSnapshot =
        queryResult.getItemsSnapshot();
    std::cout << "----- table content -----" << std::endl;
    for (auto& item : itemsSnapshot) {
        anObject = cloudBackend.castObject(item);
        std::cout << " sum "
                  << anObject.length()
                  << " Bytes\t"
                  << anObject.name()
                  << std::endl;
    }
    std::cout << "-----" << std::endl;
}
catch (const cbe::util::Exception& e)
{
    std::cout << "Error!\n" << e.what()
              << "\nType: " << e.typeAsString() << std::endl;
    cloudBackend.terminate();
    return 5;
}
std::cout << "Example Medium done" << std::endl;

```

```

    cloudBackend.terminate();
    return 0;
} // int main()

```

5.6 Full example - sync [non-throwing]

Example program - synchronous - using no exception

```

#include <algorithm>
#include <fstream>
#include <iostream>
#include "cbe/Account.h"
#include "cbe/CloudBackend.h"
#include "cbe/Container.h"
#include "cbe/Stream.h"
#include "cbe/Object.h"
#include "cbe/QueryChainSync.h"
#include "cbe/delegate/DownloadSuccess.h"
#include "cbe/util/Exception.h"
#include "cbe/util/ErrorInfo.h"
#include "../user_credentials.cpp"
int main(int argc, const char** argv) {
    static constexpr const char testContainer[] = "z-MediumExample";
    cbe::Container rootContainer{cbe::DefaultCtor{}};
    cbe::Container myContainer{cbe::DefaultCtor{}};
    cbe::Filter objectFilter;
    objectFilter.setDataTypes(cbe::ItemType::Object);
    // ----- login -----
    cbe::CloudBackend cloudBackend{ cbe::DefaultCtor{} };
    cbe::CloudBackend::LoginError loginError{};
    cloudBackend = cbe::CloudBackend::login(username,
                                           password,
                                           tenant,
                                           client,
                                           loginError);

    if (loginError) {
        std::cout << "Failed to login: loginError={" << loginError << "}'
                  << std::endl;
        return 1;
    }
    rootContainer = cloudBackend.account().rootContainer();
    std::cout << "login: " << cloudBackend.account().username() << std::endl;
    // ----- check for existing container -----
    cbe::CloudBackend::QueryError queryError;
    cbe::QueryResult resultSet = rootContainer.query(queryError);
    if (queryError) {
        std::cout << "Error! " << queryError << std::endl;
        cloudBackend.terminate();
        return 2;
    }
    for (auto entry : resultSet.getItemsSnapshot()) {
        if (entry.type() == cbe::ItemType::Container &&
            entry.name() == testContainer) {
            std::cout << "Using existing container!" << std::endl;
            myContainer = cbe::CloudBackend::castContainer(entry);
        }
    }
    if (!myContainer) {
        std::cout << "Creating a new container!" << std::endl;
        cbe::Container::CreateContainerError createContainerError;
        myContainer =
            *rootContainer.createContainer(testContainer, createContainerError);
        if (createContainerError) {
            std::cout << "Error! " << createContainerError << std::endl;
            cloudBackend.terminate();
            return 3;
        }
    }
    // ----- upload_syncNonThrowing -----
    constexpr const char* const uploadPath = "/tmp/";
    constexpr const char* const myObjectFileName = "Medium_C_a_message";
    const std::string qualFileName = std::string(uploadPath) +
                                     myObjectFileName;

    cbe::Object anObject{cbe::DefaultCtor{}};
    cbe::Object myObject{cbe::DefaultCtor{}};
    cbe::Object::Streams streams;
    std::cout << "Create local file " << qualFileName << std::endl;
    std::ofstream ofs{qualFileName};
    ofs << "Line a\n"
        << "Line b\n"
        << "Line c\n"
        << std::flush;
    ofs.close();
    cbe::Container::UploadError uploadError;
    std::cout << "Uploading object from "

```

```

        « qualFileName
        « std::endl;
myObject = *myContainer.upload(qualFileName, uploadError);
if (uploadError) {
    std::cout « "Error! " « uploadError « std::endl;
    cloudBackend.terminate();
    return 4;
}
streams = myObject.getStreams();
// - - - - - uploadStream_syncNonThrowing - - - - -
constexpr const char* const myFile2Name = "Medium_C_an_attachment";
const std::string qualFile2Name = std::string{uploadPath} + myFile2Name;
std::ofstream ofs3{qualFile2Name};
ofs3 « "line 21\n"
     « "line 22\n"
     « "line 23\n"
     « "line 24\n"
     « std::flush;
ofs3.close();
std::cout « "Uploading additional stream from: "
          « qualFile2Name
          « std::endl;
const auto nextStreamId =
    std::max_element(std::begin(streams), std::end(streams),
        [](const cbe::Stream& stream1,
            const cbe::Stream& stream2) {
            return stream1.streamId < stream2.streamId;
        })->streamId + 1;
cbe::Object::UploadError uploadStreamError;
myObject.uploadStream(qualFile2Name, nextStreamId, uploadStreamError);
if (uploadStreamError) {
    std::cout « "Error! " « uploadStreamError « std::endl;
    cloudBackend.terminate();
    return 5;
}
std::cout « "home://"
          « myContainer.name()
          « "/"
          « myObject.name()
          « " now has two streams."
          « std::endl;
streams = myObject.getStreams();
// - - - - - downloadStream_syncNonThrowing - - - - -
constexpr const char* const downloadPath = "/tmp/Medium_C_download_stream_";
for (const auto& stream : streams) {
    const auto path = std::string{downloadPath} +
        std::to_string(stream.streamId) + "_of_";
    cbe::Object::DownloadError downloadStreamError;
    std::cout « stream.streamId
              « " to: "
              « path
              « myObject.name()
              « std::endl;
    auto result = *myObject.downloadStream(path, stream, downloadStreamError);
    if (downloadStreamError) {
        std::cout « "Error! " « downloadStreamError « std::endl;
        cloudBackend.terminate();
        return 6;
    }
}
// - - - - - cloudbackend_query_syncNonThrowing - - - - -
std::cout « "Content of "
          « myContainer.name()
          « std::endl;
auto queryResult = myContainer.query(objectFilter, queryError).getQueryResult();
if (queryError) {
    std::cout « "Error! " « queryError « std::endl;
    cloudBackend.terminate();
    return 7;
} else {
    cbe::QueryResult::ItemsSnapshot itemsSnapshot =
        queryResult.getItemsSnapshot();
    std::cout « "----- table content -----" « std::endl;
    for (auto& item : itemsSnapshot) {
        anObject = cloudBackend.castObject(item);
        std::cout « " sum "
                  « anObject.length()
                  « " Bytes\t"
                  « anObject.name()
                  « std::endl;
    }
    std::cout « "-----" « std::endl;
}
std::cout « "Example Medium done" « std::endl;
cloudBackend.terminate();
return 0;
} // int main()

```

5.7 Full example - async

Example program - asynchronous

```
#include <algorithm>
#include <fstream>
#include <iostream>
#include "cbe/Account.h"
#include "cbe/CloudBackend.h"
#include "cbe/Container.h"
#include "cbe/Stream.h"
#include "cbe/Object.h"
#include "cbe/QueryChain.h"
#include "cbe/util/ErrorInfo.h"
#include "MyDelegates.cpp"
#include "../user_credentials.cpp"

int main(int argc, const char** argv) {
    static constexpr const char testContainer[] = "z-MediumExample";
    cbe::Container rootContainer{cbe::DefaultCtor{}};
    cbe::Container myContainer{cbe::DefaultCtor{}};
    cbe::Filter objectFilter;
    objectFilter.setDataType(cbe::ItemType::Object);
    // ----- login_async -----
    std::shared_ptr<MyLogInDelegate> myLogInDelegate =
        std::make_shared<MyLogInDelegate>();
    cbe::CloudBackend cloudBackend = cbe::CloudBackend::login(
        username,
        password,
        tenant,
        client,
        myLogInDelegate);

    myLogInDelegate->waitForRsp();
    if (myLogInDelegate->error) {
        std::cout << "Failed to login: error={" << myLogInDelegate->errorInfo << '}'
            << std::endl;
        cloudBackend.terminate();
        return 1;
    }
    cloudBackend = myLogInDelegate->cloudBackend;
    myLogInDelegate.reset();
    rootContainer = cloudBackend.account().rootContainer();
    std::cout << "login: " << cloudBackend.account().username() << std::endl;
    // ----- check for existing container -----
    std::shared_ptr<MyQueryDelegate> myQueryDelegate =
        std::make_shared<MyQueryDelegate>();
    rootContainer.query(myQueryDelegate).getQueryResult();
    myQueryDelegate->waitForRsp();
    cbe::QueryResult resultSet = myQueryDelegate->queryResult;
    for (auto entry : resultSet.getItemsSnapshot()) {
        if (entry.type() == cbe::ItemType::Container &&
            entry.name() == testContainer) {
            std::cout << "Using existing container:" << std::endl;
            myContainer = cbe::CloudBackend::castContainer(entry);
        }
    }
    if (!myContainer) {
        std::cout << "Creating a new container:" << std::endl;
        std::shared_ptr<MyCreateContainerDelegate> myCreateContainerDelegate =
            std::make_shared<MyCreateContainerDelegate>();
        rootContainer.createContainer(testContainer, myCreateContainerDelegate);
        myCreateContainerDelegate->waitForRsp();
        myContainer = myCreateContainerDelegate->container;
    }
    std::cout << "home://"
        << myContainer.name()
        << "/"
        << std::endl;
    // ----- upload_async -----
    constexpr const char* const uploadPath = "/tmp/";
    constexpr const char* const myObjectFileName = "Medium_A_a_message";
    const std::string qualFileName = std::string(uploadPath) +
        myObjectFileName;

    cbe::Object anObject{cbe::DefaultCtor{}};
    cbe::Object myObject{cbe::DefaultCtor{}};
    cbe::Object::Streams streams;
    std::cout << "Create local file " << qualFileName << std::endl;
    std::ofstream ofs{qualFileName};
    ofs << "Line a\n"
        << "Line b\n"
        << "Line c\n"
        << std::flush;
    ofs.close();
    std::cout << "Uploading object from "
        << qualFileName
        << std::endl;
    std::shared_ptr<MyUploadDelegate> myUploadDelegate =
        std::make_shared<MyUploadDelegate>();
    myContainer.upload(qualFileName, myUploadDelegate);
}
```

```

myObject = myUploadDelegate->waitForRsp();
if (!myObject) {
    std::cout << "Error! " << myUploadDelegate->errorInfo << std::endl;
}
std::shared_ptr<MyGetStreamsDelegate> getStreamsDelegate =
    std::make_shared<MyGetStreamsDelegate>();
myObject.getStreams(getStreamsDelegate);
getStreamsDelegate->waitForRsp();
if (getStreamsDelegate->errorInfo) {
    std::cout << "Error! " << myLogInDelegate->errorInfo << std::endl;
}
streams = *getStreamsDelegate->streams;
// ----- uploadStream_async -----
constexpr const char* const myFile2Name = "Medium_A_an_attachment";
const std::string qualFile2Name = std::string{uploadPath} + myFile2Name;
std::ofstream ofs2{qualFile2Name};
ofs2 << "line 21\n"
    << "line 22\n"
    << "line 23\n"
    << "line 24\n"
    << std::flush;
ofs2.close();
std::cout << "Uploading additional stream from: "
    << qualFile2Name
    << std::endl;
const auto nextStreamId =
    std::max_element(std::begin(streams), std::end(streams),
        [](const cbe::Stream& stream1,
            const cbe::Stream& stream2) {
            return stream1.streamId < stream2.streamId;
        })->streamId + 1;
std::shared_ptr<MyUploadStreamDelegate> myUploadStreamDelegate =
    std::make_shared<MyUploadStreamDelegate>();
myObject.uploadStream(qualFile2Name, nextStreamId, myUploadStreamDelegate);
myUploadStreamDelegate->waitForRsp();
if (myUploadStreamDelegate->errorInfo) {
    std::cout << "Error! " << myLogInDelegate->errorInfo << std::endl;
}
std::cout << "home://"
    << myContainer.name()
    << "/"
    << myObject.name()
    << " now has two streams."
    << std::endl;
myObject.getStreams(getStreamsDelegate);
getStreamsDelegate->waitForRsp();
if (getStreamsDelegate->errorInfo) {
    std::cout << "Error! " << myLogInDelegate->errorInfo << std::endl;
}
streams = *getStreamsDelegate->streams;
// ----- downloadStream_async -----
std::cout << "Downloading stream:" << std::endl;
std::shared_ptr<MyDownloadDelegate> myDownloadDelegate =
    std::make_shared<MyDownloadDelegate>();
constexpr const char* const downloadPath = "/tmp/Medium_A_download_stream_";
for (const auto& stream : streams) {
    const auto path = std::string{downloadPath} +
        std::to_string(stream.streamId) + "_of_";
    std::cout << stream.streamId
        << " to: "
        << path
        << myObject.name()
        << std::endl;
    myObject.downloadStream(path, stream, myDownloadDelegate);
    myDownloadDelegate->waitForRsp();
    if (!myDownloadDelegate) {
        std::cout << "Error! " << myDownloadDelegate->errorInfo << std::endl;
    }
}
// ----- cloudbackend_query_async -----
std::cout << "Content of "
    << myContainer.name()
    << std::endl;
myContainer.query(objectFilter, myQueryDelegate);
myQueryDelegate->waitForRsp();
if (myQueryDelegate->errorInfo) {
    std::cout << "Error! " << myQueryDelegate->errorInfo << std::endl;
} else {
    cbe::QueryResult::ItemsSnapshot itemsSnapshot =
        myQueryDelegate->queryResult.getItemsSnapshot();
    std::cout << "----- table content -----" << std::endl;
    for (auto& item : itemsSnapshot) {
        anObject = cloudBackend.castObject(item);
        std::cout << " sum "
            << anObject.length()
            << " Bytes\t"
            << anObject.name()

```

```
        « std::endl;
    }
    std::cout « "-----" « std::endl;
}
std::cout « "Example Medium done" « std::endl;
cloudBackend.terminate();
} // int main()
```

Chapter 6

Document list

C++ latest version

- [HTML version](#) of the Synchronous and Asynchronous reference document
- [PDF version](#) of this reference document
- [HTML version](#) of the Asynchronous only reference document
- [PDF version](#) of the Asynchronous only reference document

Java latest version

- [Java and Android API](#) reference document

Swift latest version

- [iOS Swift API](#) reference document

Other

- [CloudBackend web site](#) document index

Chapter 7

Namespace Index

7.1 Namespace List

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

cbe	Root namespace for the CloudBackend SDK API	39
cbe::delegate	Root namespace for the delegate interfaces	46
cbe::delegate::container	Namespace for cbe::Container specific delegate interfaces	53
cbe::delegate::group	Namespace for cbe::Group specific delegate interfaces	54
cbe::delegate::object	Namespace for cbe::Object specific delegate interfaces	55
cbe::util	General support program utilities	56

Chapter 8

Hierarchical Index

8.1 Class Hierarchy

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

cbe::Account	57
cbe::delegate::AclDelegate	59
cbe::delegate::AddRoleMemberDelegate	60
cbe::delegate::BanDelegate	62
cbe::delegate::BanSuccess	63
cbe::delegate::ChunkTransferred	63
cbe::CloudBackend	64
cbe::util::Context	120
cbe::delegate::CreateAccountDelegate	120
cbe::delegate::CreateContainerDelegate	121
cbe::delegate::CreateGroupDelegate	122
cbe::delegate::CreateObjectDelegate	123
cbe::delegate::CreateRoleDelegate	124
cbe::Database	125
cbe::delegate::DownloadBinaryDelegate	127
cbe::delegate::DownloadBinarySuccess	128
cbe::delegate::DownloadDelegate	129
cbe::delegate::DownloadSuccess	131
cbe::delegate::Error	132
cbe::delegate::QueryError	355
cbe::delegate::TransferError	391
cbe::delegate::ItemError::Error	134
cbe::delegate::JoinError::Error	134
cbe::delegate::LoadError::Error	135
cbe::util::ErrorInfo	177
cbe::util::ErrorInfoImpl< Error >	178
cbe::delegate::AclDelegate::ErrorInfo	135
cbe::delegate::AddRoleMemberDelegate::ErrorInfo	137
cbe::delegate::BanDelegate::ErrorInfo	138
cbe::delegate::CreateAccountDelegate::ErrorInfo	142
cbe::delegate::CreateContainerDelegate::ErrorInfo	143
cbe::delegate::CreateGroupDelegate::ErrorInfo	144
cbe::delegate::CreateObjectDelegate::ErrorInfo	145
cbe::delegate::CreateRoleDelegate::ErrorInfo	146
cbe::delegate::DownloadBinaryDelegate::ErrorInfo	147
cbe::delegate::DownloadDelegate::ErrorInfo	148
cbe::delegate::GetStreamsDelegate::ErrorInfo	149
cbe::delegate::GetSubscriptionsDelegate::ErrorInfo	150
cbe::delegate::KickDelegate::ErrorInfo	154
cbe::delegate::LeaveDelegate::ErrorInfo	155

cbe::delegate::ListGroupDelegate::ErrorInfo	156
cbe::delegate::ListMembersDelegate::ErrorInfo	157
cbe::delegate::ListRolesDelegate::ErrorInfo	158
cbe::delegate::ListSharesDelegate::ErrorInfo	159
cbe::delegate::LoginDelegate::ErrorInfo	160
cbe::delegate::PublishDelegate::ErrorInfo	164
cbe::delegate::QueryDelegate::ErrorInfo	165
cbe::delegate::RemoveRoleDelegate::ErrorInfo	166
cbe::delegate::RemoveRoleMemberDelegate::ErrorInfo	167
cbe::delegate::SearchGroupsDelegate::ErrorInfo	168
cbe::delegate::ShareDelegate::ErrorInfo	169
cbe::delegate::SubscribeDelegate::ErrorInfo	170
cbe::delegate::UnBanDelegate::ErrorInfo	171
cbe::delegate::UnPublishDelegate::ErrorInfo	172
cbe::delegate::UnShareDelegate::ErrorInfo	173
cbe::delegate::UnSubscribeDelegate::ErrorInfo	174
cbe::delegate::UpdateKeyValuesDelegate::ErrorInfo	175
cbe::delegate::UploadDelegate::ErrorInfo	176
cbe::delegate::container::MoveDelegate::ErrorInfo	139
cbe::delegate::container::RemoveDelegate::ErrorInfo	140
cbe::delegate::container::RenameDelegate::ErrorInfo	141
cbe::delegate::group::JoinDelegate::ErrorInfo	151
cbe::delegate::group::RemoveDelegate::ErrorInfo	152
cbe::delegate::group::RenameDelegate::ErrorInfo	153
cbe::delegate::object::MoveDelegate::ErrorInfo	161
cbe::delegate::object::RemoveDelegate::ErrorInfo	162
cbe::delegate::object::RenameDelegate::ErrorInfo	163
cbe::util::ErrorInfoImpl< ErrorT >	178
cbe::Filter	248
cbe::delegate::GetStreamsDelegate	252
cbe::delegate::GetSubscriptionsDelegate	253
cbe::Group	254
cbe::GroupFilter	271
cbe::GroupManager	274
cbe::GroupQueryResult	278
cbe::delegate::ICloudBackendListener	279
cbe::delegate::CloudBackendListenerDelegate	83
cbe::Item	282
cbe::Container	85
cbe::Object	306
cbe::delegate::ItemError	286
cbe::delegate::group::JoinDelegate	286
cbe::delegate::JoinDelegate	287
cbe::delegate::JoinError	288
cbe::delegate::KickDelegate	289
cbe::delegate::KickSuccess	290
cbe::delegate::LeaveDelegate	290
cbe::delegate::LeaveSuccess	291
cbe::delegate::ListGroupDelegate	292
cbe::delegate::ListMembersDelegate	293
cbe::delegate::ListRolesDelegate	294
cbe::delegate::ListSharesDelegate	295
cbe::delegate::LoadError	295
cbe::delegate::LoginDelegate	296
cbe::MatchesID< IdT >	297
cbe::Member	298
cbe::delegate::container::MoveDelegate	304
cbe::delegate::object::MoveDelegate	305

cbe::util::Optional< T >	338
cbe::Publish	339
cbe::delegate::PublishDelegate	342
cbe::PublishManager	343
cbe::delegate::PublishSuccess	344
cbe::QueryChain	345
cbe::QueryChainExt	347
cbe::delegate::QueryDelegate	354
cbe::delegate::QueryJoinDelegate	356
cbe::delegate::QueryLoaded	358
cbe::QueryResult	358
cbe::QueryChainSync	351
cbe::delegate::container::RemoveDelegate	361
cbe::delegate::group::RemoveDelegate	362
cbe::delegate::object::RemoveDelegate	362
cbe::delegate::RemoveRoleDelegate	363
cbe::delegate::RemoveRoleMemberDelegate	364
cbe::delegate::container::RemoveSuccess	365
cbe::delegate::group::RemoveSuccess	365
cbe::delegate::object::RemoveSuccess	366
cbe::delegate::container::RenameDelegate	366
cbe::delegate::group::RenameDelegate	367
cbe::delegate::object::RenameDelegate	368
cbe::delegate::group::RenameSuccess	369
cbe::Request	370
cbe::Role	370
std::runtime_error	
cbe::util::Exception	244
cbe::util::ExceptionImpl< ErrorInfo >	246
cbe::delegate::AclDelegate::Exception	179
cbe::delegate::AddRoleMemberDelegate::Exception	180
cbe::delegate::BanDelegate::Exception	182
cbe::delegate::CreateAccountDelegate::Exception	188
cbe::delegate::CreateContainerDelegate::Exception	189
cbe::delegate::CreateGroupDelegate::Exception	191
cbe::delegate::CreateObjectDelegate::Exception	192
cbe::delegate::CreateRoleDelegate::Exception	194
cbe::delegate::DownloadBinaryDelegate::Exception	195
cbe::delegate::DownloadDelegate::Exception	197
cbe::delegate::GetStreamsDelegate::Exception	198
cbe::delegate::GetSubscriptionsDelegate::Exception	200
cbe::delegate::JoinDelegate::Exception	206
cbe::delegate::KickDelegate::Exception	207
cbe::delegate::LeaveDelegate::Exception	209
cbe::delegate::ListGroupDelegate::Exception	210
cbe::delegate::ListMembersDelegate::Exception	212
cbe::delegate::ListRolesDelegate::Exception	213
cbe::delegate::ListSharesDelegate::Exception	215
cbe::delegate::LoginDelegate::Exception	216
cbe::delegate::PublishDelegate::Exception	222
cbe::delegate::QueryDelegate::Exception	224
cbe::delegate::QueryJoinDelegate::Exception	225
cbe::delegate::RemoveRoleDelegate::Exception	227
cbe::delegate::RemoveRoleMemberDelegate::Exception	228
cbe::delegate::SearchGroupsDelegate::Exception	230
cbe::delegate::ShareDelegate::Exception	231
cbe::delegate::SubscribeDelegate::Exception	233
cbe::delegate::UnBanDelegate::Exception	234

cbe::delegate::UnPublishDelegate::Exception	236
cbe::delegate::UnShareDelegate::Exception	237
cbe::delegate::UnSubscribeDelegate::Exception	239
cbe::delegate::UpdateKeyValuesDelegate::Exception	240
cbe::delegate::UploadDelegate::Exception	242
cbe::delegate::container::MoveDelegate::Exception	183
cbe::delegate::container::RemoveDelegate::Exception	185
cbe::delegate::container::RenameDelegate::Exception	186
cbe::delegate::group::JoinDelegate::Exception	201
cbe::delegate::group::RemoveDelegate::Exception	203
cbe::delegate::group::RenameDelegate::Exception	204
cbe::delegate::object::MoveDelegate::Exception	218
cbe::delegate::object::RemoveDelegate::Exception	219
cbe::delegate::object::RenameDelegate::Exception	221
cbe::QueryChain::Exception	244
cbe::util::ExceptionImpl< ErrorInfoT >	246
cbe::util::Optional< T >::BadAccess	61
cbe::delegate::SearchGroupsDelegate	376
cbe::ShareData	377
cbe::delegate::ShareDelegate	377
cbe::ShareManager	378
cbe::delegate::ShareSuccess	382
cbe::Stream	383
cbe::Subscribe	383
cbe::delegate::SubscribeDelegate	385
cbe::SubscribeManager	386
cbe::delegate::UnBanDelegate	392
cbe::delegate::UnBanSuccess	393
cbe::delegate::UnPublishDelegate	393
cbe::delegate::UnPublishSuccess	394
cbe::delegate::UnShareDelegate	395
cbe::delegate::UnShareSuccess	396
cbe::delegate::UnSubscribeDelegate	396
cbe::delegate::UnSubscribeSuccess	397
cbe::delegate::UpdateKeyValuesDelegate	398
cbe::delegate::UploadDelegate	399

Chapter 9

Class Index

9.1 Class List

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

cbe::Account	
Login account information	57
cbe::delegate::AclDelegate	59
cbe::delegate::AddRoleMemberDelegate	60
cbe::util::Optional< T >::BadAccess	
Thrown by value() method when accessing an object that does not contain a value	61
cbe::delegate::BanDelegate	62
cbe::delegate::BanSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::BanDelegate::onBanSuccess	63
cbe::delegate::ChunkTransferred	
Bundles the progress event information in connection with download and upload	63
cbe::CloudBackend	
The session that holds the connection with the cloud	64
cbe::delegate::CloudBackendListenerDelegate	83
cbe::Container	
A collection of Item , can also represent a table or folder	85
cbe::util::Context	120
cbe::delegate::CreateAccountDelegate	120
cbe::delegate::CreateContainerDelegate	121
cbe::delegate::CreateGroupDelegate	122
cbe::delegate::CreateObjectDelegate	123
cbe::delegate::CreateRoleDelegate	124
cbe::Database	
A database	125
cbe::delegate::DownloadBinaryDelegate	127
cbe::delegate::DownloadBinarySuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::DownloadBinaryDelegate::onDownloadBinarySuccess	128
cbe::delegate::DownloadDelegate	129
cbe::delegate::DownloadSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::DownloadDelegate::onDownloadSuccess	131
cbe::delegate::Error	132
cbe::delegate::ItemError::Error	134
cbe::delegate::JoinError::Error	134
cbe::delegate::LoadError::Error	135
cbe::delegate::AclDelegate::ErrorInfo	135
cbe::delegate::AddRoleMemberDelegate::ErrorInfo	137
cbe::delegate::BanDelegate::ErrorInfo	138

cbe::delegate::container::MoveDelegate::ErrorInfo	139
cbe::delegate::container::RemoveDelegate::ErrorInfo	140
cbe::delegate::container::RenameDelegate::ErrorInfo	141
cbe::delegate::CreateAccountDelegate::ErrorInfo	142
cbe::delegate::CreateContainerDelegate::ErrorInfo	143
cbe::delegate::CreateGroupDelegate::ErrorInfo	144
cbe::delegate::CreateObjectDelegate::ErrorInfo	145
cbe::delegate::CreateRoleDelegate::ErrorInfo	146
cbe::delegate::DownloadBinaryDelegate::ErrorInfo	147
cbe::delegate::DownloadDelegate::ErrorInfo	148
cbe::delegate::GetStreamsDelegate::ErrorInfo	149
cbe::delegate::GetSubscriptionsDelegate::ErrorInfo	150
cbe::delegate::group::JoinDelegate::ErrorInfo	151
cbe::delegate::group::RemoveDelegate::ErrorInfo	152
cbe::delegate::group::RenameDelegate::ErrorInfo	153
cbe::delegate::KickDelegate::ErrorInfo	154
cbe::delegate::LeaveDelegate::ErrorInfo	155
cbe::delegate::ListGroupDelegate::ErrorInfo	156
cbe::delegate::ListMembersDelegate::ErrorInfo	157
cbe::delegate::ListRolesDelegate::ErrorInfo	158
cbe::delegate::ListSharesDelegate::ErrorInfo	159
cbe::delegate::LoginDelegate::ErrorInfo	160
cbe::delegate::object::MoveDelegate::ErrorInfo	161
cbe::delegate::object::RemoveDelegate::ErrorInfo	162
cbe::delegate::object::RenameDelegate::ErrorInfo	163
cbe::delegate::PublishDelegate::ErrorInfo	164
cbe::delegate::QueryDelegate::ErrorInfo	165
cbe::delegate::RemoveRoleDelegate::ErrorInfo	166
cbe::delegate::RemoveRoleMemberDelegate::ErrorInfo	167
cbe::delegate::SearchGroupsDelegate::ErrorInfo	168
cbe::delegate::ShareDelegate::ErrorInfo	169
cbe::delegate::SubscribeDelegate::ErrorInfo	170
cbe::delegate::UnBanDelegate::ErrorInfo	171
cbe::delegate::UnPublishDelegate::ErrorInfo	172
cbe::delegate::UnShareDelegate::ErrorInfo	173
cbe::delegate::UnSubscribeDelegate::ErrorInfo	174
cbe::delegate::UpdateKeyValuesDelegate::ErrorInfo	175
cbe::delegate::UploadDelegate::ErrorInfo	176
cbe::util::ErrorInfo	177
cbe::util::ErrorInfoImpl< ErrorT >	178
cbe::delegate::AclDelegate::Exception	
Exception thrown by cbe::Object::getAcl() if the request fails	179
cbe::delegate::AddRoleMemberDelegate::Exception	
Exception thrown by cbe::Group::AddMember() if the request fails	180
cbe::delegate::BanDelegate::Exception	
Exception thrown by cbe::Member::ban(std::string) if the request fails	182
cbe::delegate::container::MoveDelegate::Exception	
Exception thrown by cbe::Container::move() if the request fails	183
cbe::delegate::container::RemoveDelegate::Exception	
Exception thrown by cbe::Container::remove() if the request fails	185
cbe::delegate::container::RenameDelegate::Exception	
Exception thrown by cbe::Container::rename() if the request fails	186
cbe::delegate::CreateAccountDelegate::Exception	
Exception thrown by cbe::CloudBackend::createAccount() if the request fails	188
cbe::delegate::CreateContainerDelegate::Exception	
Exception thrown by cbe::Container::createContainer() if the request fails	189
cbe::delegate::CreateGroupDelegate::Exception	
Exception thrown by cbe::Group::createGroup() if the request fails	191

cbe::delegate::CreateObjectDelegate::Exception	
Exception thrown by cbe::Container::createObject() if the request fails	192
cbe::delegate::CreateRoleDelegate::Exception	
Exception thrown by cbe::Role::createRole() if the request fails	194
cbe::delegate::DownloadBinaryDelegate::Exception	
Exception thrown by cbe::Object::download() if the request fails	195
cbe::delegate::DownloadDelegate::Exception	
Exception thrown by:	197
cbe::delegate::GetStreamsDelegate::Exception	
Exception thrown by cbe::Object::getStreams() if the request fails	198
cbe::delegate::GetSubscriptionsDelegate::Exception	
Exception thrown by cbe::SubscribeManager::getSubscriptions() if the request fails	200
cbe::delegate::group::JoinDelegate::Exception	
Exception thrown by cbe::Group::join() if the request fails	201
cbe::delegate::group::RemoveDelegate::Exception	
Exception thrown by cbe::Group::remove() if the request fails	203
cbe::delegate::group::RenameDelegate::Exception	
Exception thrown by cbe::Group::rename() if the request fails	204
cbe::delegate::JoinDelegate::Exception	
Exception thrown by cbe::QueryChain::join() if the request fails	206
cbe::delegate::KickDelegate::Exception	
Exception thrown by cbe::Member::kick(std::string) if the request fails	207
cbe::delegate::LeaveDelegate::Exception	
Exception thrown by cbe::Group::leave() if the if the request fails	209
cbe::delegate::ListGroupDelegate::Exception	
Exception thrown by cbe::GroupManager::listGroups() if the request fails	210
cbe::delegate::ListMembersDelegate::Exception	
Exception thrown by cbe::Group::listMembers() if the request fails	212
cbe::delegate::ListRolesDelegate::Exception	
Exception thrown by cbe::Group::listRoles() if the request fails	213
cbe::delegate::ListSharesDelegate::Exception	
Exception thrown by cbe::ShareManager::listAvailableShares() or cbe::ShareManager::listMyShares() if the request fails	215
cbe::delegate::LoginDelegate::Exception	
Exception thrown by cbe::CloudBackend::login(const std::string&,const std::string&, const std::string&) if the request fails	216
cbe::delegate::object::MoveDelegate::Exception	
Exception thrown by cbe::Object::move() if the request fails	218
cbe::delegate::object::RemoveDelegate::Exception	
Exception thrown by cbe::Object::remove() if the request fails	219
cbe::delegate::object::RenameDelegate::Exception	
Exception thrown by cbe::Object::rename() if the request fails	221
cbe::delegate::PublishDelegate::Exception	
Exception thrown by:	222
cbe::delegate::QueryDelegate::Exception	
Exception thrown by	224
cbe::delegate::QueryJoinDelegate::Exception	
Exception thrown by	225
cbe::delegate::RemoveRoleDelegate::Exception	
Exception thrown by cbe::Role::removeRole() if the request fails	227
cbe::delegate::RemoveRoleMemberDelegate::Exception	
Exception thrown by cbe::Group::RemoveMember() if the request fails	228
cbe::delegate::SearchGroupsDelegate::Exception	
Exception thrown by cbe::GroupManager::searchGroups() if the request fails	230
cbe::delegate::ShareDelegate::Exception	
Exception thrown by	231
cbe::delegate::SubscribeDelegate::Exception	
Exception thrown by cbe::SubscribeManager::subscribe() and its overloads if the request fails	233

cbe::delegate::UnBanDelegate::Exception	
Exception thrown by cbe::Member::unBan() if the request fails	234
cbe::delegate::UnPublishDelegate::Exception	
Exception thrown by	236
cbe::delegate::UnShareDelegate::Exception	
Exception thrown by	237
cbe::delegate::UnSubscribeDelegate::Exception	
Exception thrown by	239
cbe::delegate::UpdateKeyValuesDelegate::Exception	
Exception thrown by cbe::Object::updateKeyValues() if the request fails	240
cbe::delegate::UploadDelegate::Exception	
Exception thrown by cbe::Container::upload(const std::string&,const std::string&) and its overloads if the request fails	242
cbe::QueryChain::Exception	244
cbe::util::Exception	244
cbe::util::ExceptionImpl< ErrorInfoT >	246
cbe::Filter	
Use to select Item that meets specific criterias when doing a query	248
cbe::delegate::GetStreamsDelegate	252
cbe::delegate::GetSubscriptionsDelegate	253
cbe::Group	
A group of members	254
cbe::GroupFilter	
To filter when searching a list of Group	271
cbe::GroupManager	
For managing the groups	274
cbe::GroupQueryResult	
Resultset of data retrieved in a search for Group	278
cbe::delegate::ICloudBackendListener	279
cbe::Item	
A set made up of Container and Object	282
cbe::delegate::ItemError	286
cbe::delegate::group::JoinDelegate	286
cbe::delegate::JoinDelegate	287
cbe::delegate::JoinError	288
cbe::delegate::KickDelegate	289
cbe::delegate::KickSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::KickDelegate::onKickSuccess	290
cbe::delegate::LeaveDelegate	290
cbe::delegate::LeaveSuccess	291
cbe::delegate::ListGroupDelegate	292
cbe::delegate::ListMembersDelegate	293
cbe::delegate::ListRolesDelegate	294
cbe::delegate::ListSharesDelegate	295
cbe::delegate::LoadError	295
cbe::delegate::LoginDelegate	296
cbe::MatchesID< IdT >	
Search lists with ID	297
cbe::Member	
A of member of a Group	298
cbe::delegate::container::MoveDelegate	304
cbe::delegate::object::MoveDelegate	305
cbe::Object	
Holder of a set of data, can represent a table row	306
cbe::util::Optional< T >	
Class template Optional manages an optional contained value - i.e., a value that is either present or absent	338

cbe::Publish	
Managing a published Item	339
cbe::delegate::PublishDelegate	342
cbe::PublishManager	
Managing the list of web shares	343
cbe::delegate::PublishSuccess	
Convenience type that bundles the parameter passed to method cbe::delegate::ShareDelegate::onShareSuccess	344
cbe::QueryChain	
To do a search for Object combining more than one Container table	345
cbe::QueryChainExt	
Extension of class QueryChain	347
cbe::QueryChainSync	
Synchronous version of querychain	351
cbe::delegate::QueryDelegate	354
cbe::delegate::QueryError	355
cbe::delegate::QueryJoinDelegate	356
cbe::delegate::QueryLoaded	358
cbe::QueryResult	
Resultset of data retrieved	358
cbe::delegate::container::RemoveDelegate	361
cbe::delegate::group::RemoveDelegate	362
cbe::delegate::object::RemoveDelegate	362
cbe::delegate::RemoveRoleDelegate	363
cbe::delegate::RemoveRoleMemberDelegate	364
cbe::delegate::container::RemoveSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::container::RemoveDelegate::onRemoveSuccess	365
cbe::delegate::group::RemoveSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::group::RemoveDelegate::onRemoveSuccess	365
cbe::delegate::object::RemoveSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::object::onRemoveSuccess	366
cbe::delegate::container::RenameDelegate	366
cbe::delegate::group::RenameDelegate	367
cbe::delegate::object::RenameDelegate	368
cbe::delegate::group::RenameSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::group::RenameDelegate::onRenameSuccess	369
cbe::Request	370
cbe::Role	
User role information	370
cbe::delegate::SearchGroupsDelegate	376
cbe::ShareData	377
cbe::delegate::ShareDelegate	377
cbe::ShareManager	
Managing Shares	378
cbe::delegate::ShareSuccess	
Convenience type that bundles the parameter passed to method cbe::delegate::ShareDelegate::onShareSuccess	382
cbe::Stream	
A data file attached to Object	383
cbe::Subscribe	
Managing a subscribed Item	383
cbe::delegate::SubscribeDelegate	385
cbe::SubscribeManager	
For managing subscriptions	386

cbe::delegate::TransferError	391
cbe::delegate::UnBanDelegate	392
cbe::delegate::UnBanSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::UnBanDelegate::onUnBanSuccess	
393	
cbe::delegate::UnPublishDelegate	393
cbe::delegate::UnPublishSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::UnPublishDelegate::onUnPublishError	
394	
cbe::delegate::UnShareDelegate	395
cbe::delegate::UnShareSuccess	
Convenience type that bundles the parameter passed to method cbe::delegate::ShareDelegate::onShareSuccess	
396	
cbe::delegate::UnSubscribeDelegate	396
cbe::delegate::UnSubscribeSuccess	
Convenience type that bundles all parameters passed to method cbe::delegate::UnSubscribeDelegate::onUnSubscribeS	
397	
cbe::delegate::UpdateKeyValuesDelegate	398
cbe::delegate::UploadDelegate	399

Chapter 10

Namespace Documentation

10.1 cbe Namespace Reference

Root namespace for the CloudBackend SDK API.

Namespaces

- [delegate](#)
Root namespace for the delegate interfaces.
- [util](#)
General support program utilities.

Classes

- class [Account](#)
Login account information.
- class [CloudBackend](#)
The session that holds the connection with the cloud.
- class [Container](#)
A collection of [Item](#), can also represent a table or folder.
- class [Database](#)
A database.
- class [Filter](#)
Use to select [Item](#) that meets specific criterias when doing a query.
- class [Group](#)
A group of members.
- class [GroupFilter](#)
To filter when searching a list of [Group](#).
- class [GroupManager](#)
For managing the groups.
- class [GroupQueryResult](#)
Resultset of data retrieved in a search for [Group](#).
- class [Item](#)
A set made up of [Container](#) and [Object](#).
- struct [Request](#)
- class [Member](#)
A of member of a [Group](#).
- class [Object](#)
Holder of a set of data, can represent a table row.
- class [Publish](#)

- Managing a published [Item](#).*

 - class [PublishManager](#)
- Managing the list of web shares.*

 - class [QueryChain](#)
- To do a search for [Object](#) combining more than one [Container](#) table.*

 - class [QueryChainExt](#)
- Extension of class [QueryChain](#).*

 - class [QueryChainSync](#)
- Synchronous version of querychain.*

 - class [QueryResult](#)
- resultset of data retrieved.*

 - class [Role](#)
- User role information.*

 - class [ShareManager](#)
- Managing Shares.*

 - class [Stream](#)
- A data file attached to [Object](#).*

 - class [Subscribe](#)
- Managing a subscribed [Item](#).*

 - class [SubscribeManager](#)
- For managing subscriptions.*

 - struct [ShareData](#)
- class [MatchesID](#)

Search lists with ID.

Typedefs

- using [Streams](#) = std::vector< [cbe::Stream](#) >

Collection of [Stream](#) objects.
- using [AclGroupId](#) = std::uint64_t

Identifies ACL-group.
- using [ContainerId](#) = std::uint64_t

Unique Id of a [cbe::Container](#).
- using [DatabaseId](#) = std::uint64_t

The id of a [Database](#).
- using [Date](#) = std::uint64_t

A time-stamp in the unix epoch format.
- using [GroupId](#) = std::uint64_t

Uniquely identifies the [Group](#).
- using [ItemId](#) = std::uint64_t

Id of a [cbe::Container](#) or [cbe::Object](#).
- using [MemberId](#) = std::uint64_t

Represents the [cbe::Group](#) membership id.
- using [ListenerHandle](#) = std::uint64_t

Unique Id for added Listener.
- using [ObjectId](#) = std::uint64_t

Unique Id of a [cbe::Object](#).
- using [PublishId](#) = std::uint64_t

Id of a subscribed [cbe::Container](#) or [cbe::Object](#).
- using [RoleId](#) = std::uint64_t

Uniquely identifies the [Role](#).

- using [ShareId](#) = std::uint64_t
Uniquely identifies a sharing of a [cbe::Container](#) or [cbe::Object](#).
- using [StreamId](#) = std::uint64_t
Uniquely identifies a [cbe::Stream](#).
- using [SubscribeId](#) = std::uint64_t
Id of a subscription of a [cbe::Container](#) or [cbe::Object](#).
- using [UserId](#) = std::uint64_t
Uniquely identifies the CBE user number.
- using [account_status_t](#) = std::uint32_t
- using [failed_status_t](#) = std::uint32_t
- using [http_t](#) = std::uint32_t
- using [publish_access_t](#) = std::uint32_t
- using [publish_visibility_t](#) = std::uint32_t
- using [sync_direction_t](#) = std::uint32_t
- using [sync_status_t](#) = std::uint32_t
- using [transfer_t](#) = std::uint32_t
- using [ErrorCode](#) = std::uint32_t
Mimics the general error code encoding in the www.
- using [application_t](#) = int
- using [item_t](#) = int
- using [object_t](#) = int
- using [permission_status_t](#) = int
- using [stream_t](#) = int
- using [visibility](#) = int
- using [AclMap](#) = std::map< [cbe::AclGroupId](#), std::pair< [cbe::Permissions](#), [AclScope](#) > >
*ACL map (**A**ccess **C**ontrol **L**ist) relating to users and groups.*
- using [DataBases](#) = std::map< std::string, [cbe::Database](#) >
[Databases](#) available for the [account](#).
- using [Items](#) = std::vector< [cbe::Item](#) >
Collection of [items](#).
- using [KeyValues](#) = std::map< std::string, std::pair< std::string, bool > >
Map with key/value pairs, a.k.a. metadata.
- using [MemberBanInfo](#) = std::map< std::pair< [cbe::MemberId](#), [cbe::MemberId](#) >, std::pair< [cbe::Date](#), std::string > >
Map of a pair of [members](#), associated with ban information.
- using [ShareIds](#) = std::map< [cbe::ShareId](#), std::vector< [cbe::ShareData](#) > >
Map of [cbe::ShareData](#) for a specific [cbe::ShareId](#).

Enumerations

- enum class [AccountStatus](#) : [cbe::account_status_t](#) { **NotLoggedIn** = 1 , **LoggedIn** = 2 , **Failed** = 3 }
- enum class [AclScope](#) : [uint64_t](#) { **User** = 1 , **Group** = 2 , **Role** = 3 }
- enum [DefaultCtor](#)
Default constructor marker.
- enum class [FilterOrder](#) : [uint32_t](#) {
 Title = 1 , **Relevance** = 2 , **Published** = 3 , **Updated** = 4 ,
 Length = 5 , **S1** = 6 , **S2** = 7 , **S3** = 8 ,
 S4 = 9 }
- enum class [ItemType](#) : [cbe::item_t](#) {
 Unapplicable = 1 , **Unknown** = 2 , **Object** = 4 , **Container** = 8 ,
 Tag = 16 , **Group** = 32 }
- enum class [ObjectType](#) : [object_t](#) { **Other** = 1 , **GroupInvites** = 2 , **ShareInvite** = 3 }

- enum class [Permissions](#) : permission_status_t {
Read = 1 , **Write** = 2 , **ReadWrite** = 3 , **Delete** = 4 ,
ReadDelete = 5 , **WriteDelete** = 6 , **ReadWriteDelete** = 7 , **ChangeACL** = 8 ,
ReadChangeACL = 9 , **WriteChangeACL** = 10 , **ReadWriteChangeACL** = 11 , **DeleteChangeACL** = 12 ,
ReadDeleteChangeACL = 13 , **WriteDeleteChangeACL** = 14 , **AllPermissions** = 15 , **NoPermissions** = 0
}

Represents the access permission that can be set for any [cbe::Object](#) or [cbe::Container](#).

- enum class [PublishAccess](#) : cbe::publish_access_t { **Read** = 1 , **Update** = 2 , **Create** = 3 }
- enum class [PublishVisibility](#) : cbe::publish_visibility_t { **Public** = 1 , **Friends** = 2 , **Private** = 3 , **Invited** = 4 }
- enum class [Visibility](#) : visibility { **Public** = 1 , **Private** = 2 }

Functions

- [cbe::ItemType](#) **operator|** ([cbe::ItemType](#) lh, [cbe::ItemType](#) rh)
- std::ostream & **operator<<** (std::ostream &os, [ItemType](#) itemType)

10.1.1 Detailed Description

Root namespace for the CloudBackend SDK API.
 Located in [Types.h](#) that should be included in programs.

10.1.2 Typedef Documentation

10.1.2.1 AclGroupId

using [cbe::AclGroupId](#) = typedef std::uint64_t

Identifies ACL-group.

Representation that can be a UserId or GroupId.

I.e., the union set:

$\text{AclGroupId} = \text{UserId} \cup \text{GroupId}$.

10.1.2.2 Date

using [cbe::Date](#) = typedef std::uint64_t

A time-stamp in the unix epoch format.

A.k.a. POSIX time or time-stamp.

The equivalent in Linux shell can be found using e.g.,

```
date +%s --utc --date='now'
```

```
date +%s --utc --date='today 17:30:00'
```

To convert a time-stamp to human readable format use e.g.,

```
date --utc --date=@1678902345
```

10.1.2.3 ErrorCode

using [cbe::ErrorCode](#) = typedef std::uint32_t

Mimics the general error code encoding in the www.

see [Wikipedia: List of HTTP status codes](#)

10.1.2.4 AclMap

using [cbe::AclMap](#) = typedef std::map<[cbe::AclGroupId](#), std::pair<[cbe::Permissions](#), [AclScope](#)> >

ACL map (**A**ccess **C**ontrol **L**ist) relating to users and groups.

Map of a specific user or group, in terms of [cbe::UserId](#) or [cbe::GroupId](#) and its [cbe::Permission](#) on a [Container](#) or [Object](#), to represent access permissions for specific users or groups.

10.1.2.5 KeyValues

```
using cbe::KeyValues = typedef std::map<std::string, std::pair<std::string, bool> >
```

Map with key/value pairs, a.k.a. metadata.

Can be applied on [cbe::Object](#) but not on [cbe::Container](#).

- **key** uniquely identifies the value; name must start with a letter or `_`
- **value** and **index flag**
 - a string value
 - a boolean flag indicating whether this key/value entry is index (`true`) or not indexed (`false`).

10.1.2.6 MemberBanInfo

```
using cbe::MemberBanInfo = typedef std::map<std::pair<cbe::MemberId, cbe::MemberId>, std::pair<cbe::Date, std::string> >
```

Map of a pair of [members](#), associated with ban information.

Structure:

- **Key** is formed by a `std::pair` of [members](#), where:
 - `first` represents the banned member.
 - `second` represents the banning member with administration privileges.
- **Value** is formed by another pair:
 - `date` of the ban
 - a free text reason message.

10.1.3 Enumeration Type Documentation

10.1.3.1 AccountStatus

```
enum cbe::AccountStatus : cbe::account_status_t [strong]
```

Callback for login returns one of these three in callback.

1. NotLoggedIn
2. LoggedIn
3. Failed

10.1.3.2 AclScope

```
enum cbe::AclScope : uint64_t [strong]
```

Enum that determines different ACL categories

10.1.3.3 DefaultCtor

enum [cbe::DefaultCtor](#)

Default constructor marker.

To default construct objects from most of the CloudBackend classes, this marker type is required.

Example use

```
~~~  
// Conceptually, a default construction of an instance of cbe::Container  
cbe::Container myContainer{ cbe::DefaultCtor{ } };  
// Ditto  
cbe::Object myObject{ cbe::DefaultCtor{ } };  
~~~
```

10.1.3.4 FilterOrder

enum [cbe::FilterOrder](#) : uint32_t [strong]

Set the filter order in which the search or query will be sorted after.

1. Title
2. Relevance
3. Published
4. Updated
5. Length
6. S1
7. S2
8. S3
9. S4

10.1.3.5 ItemType

enum [cbe::ItemType](#) : [cbe::item_t](#) [strong]

ItemType can be used to sort out cbe objects if the user would like to create a container to put all different kinds of cbe objects in. E.g.,

- [Object](#)
- [Container](#)
- [Group](#)

10.1.3.6 ObjectType

enum [cbe::ObjectType](#) : [object_t](#) [strong]

Type of invite.

10.1.3.7 Permissions

```
enum cbe::Permissions : permission_status_t [strong]
```

Represents the access permission that can be set for any [cbe::Object](#) or [cbe::Container](#).

Forms the the possible different combinations of:

- 1: Read
- 2: Write
- 4: Delete
- 8: ChangeACL

combinations give access for users to use different API calls.

0. NoPermissions

1. Read
2. Write
3. ReadWrite
4. Delete
5. ReadDelete
6. WriteDelete
7. ReadWriteDelete
8. ChangeACL
9. ReadChangeACL
10. WriteChangeACL
11. ReadWriteChangeACL
12. DeleteChangeACL
13. ReadDeleteChangeACL
14. WriteDeleteChangeACL
15. AllPermissions

E.g.: `cbe::Permissions::ReadDelete` gives the ability to call `move()` on a [Container](#) or an [Object](#).

10.1.3.8 PublishAccess

```
enum cbe::PublishAccess : cbe::publish_access_t [strong]
```

Access permission for publish

1. Read
2. Update
3. Create

10.1.3.9 PublishVisibility

```
enum cbe::PublishVisibility : cbe::publish_visibility_t [strong]
```

Visibility for publish

1. Public
2. Friends
3. Private
4. Invited

10.1.3.10 Visibility

```
enum cbe::Visibility : visibility [strong]
```

Visibility is used for both groups and members, in this version the member visibility will be Public for all members who join a group.

Members will in the future also have the option of visibility friends.

1. Public
2. Private

10.2 cbe::delegate Namespace Reference

Root namespace for the delegate interfaces.

Namespaces

- [container](#)
Namespace for [cbe::Container](#) specific delegate interfaces.
- [group](#)
Namespace for [cbe::Group](#) specific delegate interfaces.
- [object](#)
Namespace for [cbe::Object](#) specific delegate interfaces.

Classes

- class [AclDelegate](#)
- class [AddRoleMemberDelegate](#)
- class [BanDelegate](#)
- class [BanSuccess](#)
Convenience type that bundles all parameters passed to method [cbe::delegate::BanDelegate::onBanSuccess](#).
- class [ChunkTransferred](#)
Bundles the progress event information in connection with download and upload.
- class [CloudBackendListenerDelegate](#)
- class [CreateAccountDelegate](#)
- class [CreateContainerDelegate](#)
- class [CreateGroupDelegate](#)
- class [CreateObjectDelegate](#)
- class [CreateRoleDelegate](#)
- class [DownloadBinaryDelegate](#)
- class [DownloadBinarySuccess](#)
Convenience type that bundles all parameters passed to method [cbe::delegate::DownloadBinaryDelegate::onDownloadBinarySuccess](#).
- class [DownloadDelegate](#)

- class [DownloadSuccess](#)

Convenience type that bundles all parameters passed to method `cbe::delegate::DownloadDelegate::onDownloadSuccess`.

- class [Error](#)
- class [GetStreamsDelegate](#)
- class [GetSubscriptionsDelegate](#)
- class [ICloudBackendListener](#)
- class [LoadError](#)
- class [JoinError](#)
- class [ItemError](#)
- class [JoinDelegate](#)
- class [KickDelegate](#)
- class [KickSuccess](#)

Convenience type that bundles all parameters passed to method `cbe::delegate::KickDelegate::onKickSuccess`.

- class [LeaveDelegate](#)
- class [LeaveSuccess](#)
- class [ListGroupDelegate](#)
- class [ListMembersDelegate](#)
- class [ListRolesDelegate](#)
- class [ListSharesDelegate](#)
- class [LoginDelegate](#)
- class [PublishDelegate](#)
- class [PublishSuccess](#)

Convenience type that bundles the parameter passed to method `cbe::delegate::ShareDelegate::onShareSuccess`.

- class [QueryDelegate](#)
- class [QueryError](#)
- class [QueryJoinDelegate](#)
- struct [QueryLoaded](#)
- class [RemoveRoleDelegate](#)
- class [RemoveRoleMemberDelegate](#)
- class [SearchGroupsDelegate](#)
- class [ShareDelegate](#)
- class [ShareSuccess](#)

Convenience type that bundles the parameter passed to method `cbe::delegate::ShareDelegate::onShareSuccess`.

- class [SubscribeDelegate](#)
- class [TransferError](#)
- class [UnBanDelegate](#)
- class [UnBanSuccess](#)

Convenience type that bundles all parameters passed to method `cbe::delegate::UnBanDelegate::onUnBanSuccess`.

- class [UnPublishDelegate](#)
- class [UnPublishSuccess](#)

Convenience type that bundles all parameters passed to method `cbe::delegate::UnPublishDelegate::onUnPublishError`.

- class [UnShareDelegate](#)
- class [UnShareSuccess](#)

Convenience type that bundles the parameter passed to method `cbe::delegate::ShareDelegate::onShareSuccess`.

- class [UnSubscribeDelegate](#)
- class [UnSubscribeSuccess](#)

Convenience type that bundles all parameters passed to method `cbe::delegate::UnSubscribeDelegate::onUnSubscribeSuccess`.

- class [UpdateKeyValuesDelegate](#)
- class [UploadDelegate](#)

Typedefs

- using [AclDelegatePtr](#) = std::shared_ptr< [AclDelegate](#) >
 - using [AddRoleMemberDelegatePtr](#) = std::shared_ptr< [AddRoleMemberDelegate](#) >
 - using [BanDelegatePtr](#) = std::shared_ptr< [BanDelegate](#) >
 - using [CloudBackendListenerDelegatePtr](#) = std::shared_ptr< [CloudBackendListenerDelegate](#) >
 - using [CreateAccountDelegatePtr](#) = std::shared_ptr< [CreateAccountDelegate](#) >
 - using [CreateContainerDelegatePtr](#) = std::shared_ptr< [CreateContainerDelegate](#) >
 - using [CreateGroupDelegatePtr](#) = std::shared_ptr< [CreateGroupDelegate](#) >
 - using [CreateObjectDelegatePtr](#) = std::shared_ptr< [CreateObjectDelegate](#) >
 - using [CreateRoleDelegatePtr](#) = std::shared_ptr< [CreateRoleDelegate](#) >
 - using [DownloadBinaryDelegatePtr](#) = std::shared_ptr< [DownloadBinaryDelegate](#) >
 - using [DownloadDelegatePtr](#) = std::shared_ptr< [DownloadDelegate](#) >
 - using [GetStreamsDelegatePtr](#) = std::shared_ptr< [GetStreamsDelegate](#) >
 - using [GetSubscriptionsDelegatePtr](#) = std::shared_ptr< [GetSubscriptionsDelegate](#) >
 - using [ICloudBackendListenerPtr](#) = std::shared_ptr< [ICloudBackendListener](#) >
 - using [JoinDelegatePtr](#) = std::shared_ptr< [JoinDelegate](#) >
 - using [KickDelegatePtr](#) = std::shared_ptr< [KickDelegate](#) >
 - using [LeaveDelegatePtr](#) = std::shared_ptr< [LeaveDelegate](#) >
 - using [ListGroupDelegatePtr](#) = std::shared_ptr< [ListGroupDelegate](#) >
 - using [ListMembersDelegatePtr](#) = std::shared_ptr< [ListMembersDelegate](#) >
 - using [ListRolesDelegatePtr](#) = std::shared_ptr< [ListRolesDelegate](#) >
 - using [ListSharesDelegatePtr](#) = std::shared_ptr< [ListSharesDelegate](#) >
 - using [LoginDelegatePtr](#) = std::shared_ptr< [LoginDelegate](#) >
 - using [ProgressEventFn](#) = std::function< void(const [ChunkTransferred](#) &)>
- Callback interface function that the [CloudBackend](#) SDK uses to indicate the progress of an upload/download.*
- using [PublishDelegatePtr](#) = std::shared_ptr< [PublishDelegate](#) >
 - using [QueryDelegatePtr](#) = std::shared_ptr< [QueryDelegate](#) >
 - using **Error** = [delegate::Error](#)
 - using [QueryJoinDelegatePtr](#) = std::shared_ptr< [QueryJoinDelegate](#) >
 - using [RemoveRoleDelegatePtr](#) = std::shared_ptr< [RemoveRoleDelegate](#) >
 - using [RemoveRoleMemberDelegatePtr](#) = std::shared_ptr< [RemoveRoleMemberDelegate](#) >
 - using [SearchGroupsDelegatePtr](#) = std::shared_ptr< [SearchGroupsDelegate](#) >
 - using [ShareDelegatePtr](#) = std::shared_ptr< [ShareDelegate](#) >
 - using [SubscribeDelegatePtr](#) = std::shared_ptr< [SubscribeDelegate](#) >
 - using [UnBanDelegatePtr](#) = std::shared_ptr< [UnBanDelegate](#) >
 - using [UnPublishDelegatePtr](#) = std::shared_ptr< [UnPublishDelegate](#) >
 - using [UnShareDelegatePtr](#) = std::shared_ptr< [UnShareDelegate](#) >
 - using [UnSubscribeDelegatePtr](#) = std::shared_ptr< [UnSubscribeDelegate](#) >
 - using [UpdateKeyValuesDelegatePtr](#) = std::shared_ptr< [UpdateKeyValuesDelegate](#) >
 - using [UploadDelegatePtr](#) = std::shared_ptr< [UploadDelegate](#) >

Functions

- CBI::ItemDelegatePtr **createCbiQueryDelegate** ([QueryDelegatePtr](#), [cbe::util::Context](#) &&)
- CBI::ItemDelegatePtr **createCbiQueryDelegate** ([QueryJoinDelegatePtr](#), [cbe::util::Context](#) &&)
- CBI::ItemDelegatePtr **createCbiJoinDelegate** ([JoinDelegatePtr](#), [cbe::util::Context](#) &&)

10.2.1 Detailed Description

Root namespace for the delegate interfaces.

10.2.2 Typedef Documentation

10.2.2.1 AclDelegatePtr

using `cbe::delegate::AclDelegatePtr` = typedef `std::shared_ptr<AclDelegate>`

Pointer to `AclDelegate` that is passed into:

- `cbe::Container::getAcl()`
- `cbe::Container::setAcl()`
- `cbe::Object::getAcl()`
- `cbe::Object::setAcl()`

10.2.2.2 AddRoleMemberDelegatePtr

using `cbe::delegate::AddRoleMemberDelegatePtr` = typedef `std::shared_ptr<AddRoleMemberDelegate>`

Pointer to `AddRoleMemberDelegate` that is passed into:

- `cbe::Group::AddMember(AddRoleMemberDelegatePtr).`

10.2.2.3 BanDelegatePtr

using `cbe::delegate::BanDelegatePtr` = typedef `std::shared_ptr<BanDelegate>`

Pointer to `BanDelegate` that is passed into:

- `cbe::Member::ban(std::string,BanDelegatePtr)`

10.2.2.4 CloudBackendListenerDelegatePtr

using `cbe::delegate::CloudBackendListenerDelegatePtr` = typedef `std::shared_ptr<CloudBackendListenerDelegate>`

Pointer to `CloudBackendListenerDelegate` that is passed into:

`CloudBackend::addListener(CloudBackendListenerDelegatePtr).`

10.2.2.5 CreateAccountDelegatePtr

using `cbe::delegate::CreateAccountDelegatePtr` = typedef `std::shared_ptr<CreateAccountDelegate>`

Pointer to `CreateAccountDelegate` that is passed into:

`CloudBackend::createAccount(const std::string&,const std::string&,const std::string&,const std::string&,const std::string&,const std::string&,CreateAccountDelegatePtr).`

10.2.2.6 CreateContainerDelegatePtr

using `cbe::delegate::CreateContainerDelegatePtr` = typedef `std::shared_ptr<CreateContainerDelegate>`

Pointer to `CreateContainerDelegate` that is passed into:

- `cbe::Container::createContainer()`

10.2.2.7 CreateGroupDelegatePtr

using `cbe::delegate::CreateGroupDelegatePtr` = typedef `std::shared_ptr<CreateGroupDelegate>`

Pointer to `CreateGroupDelegate` that is passed into:

`Group::createGroup(std::string,std::string,CreateGroupDelegatePtr,cbe::Visibility).`

10.2.2.8 CreateObjectDelegatePtr

using `cbe::delegate::CreateObjectDelegatePtr` = typedef `std::shared_ptr<CreateObjectDelegate>`

Pointer to `CreateObjectDelegate` that is passed into:

`Container::createObject(std::string,CreateObjectDelegatePtr,cbe::KeyValues).`

10.2.2.9 CreateRoleDelegatePtr

using `cbe::delegate::CreateRoleDelegatePtr` = typedef `std::shared_ptr<CreateRoleDelegate>`

Pointer to `CreateRoleDelegate` that is passed into:

`Group::createRole(std::string, CreateRoleDelegatePtr)`.

10.2.2.10 DownloadBinaryDelegatePtr

using `cbe::delegate::DownloadBinaryDelegatePtr` = typedef `std::shared_ptr<DownloadBinaryDelegate>`

Pointer to `DownloadBinaryDelegate` that is passed into:

- `cbe::Object::download(DownloadBinaryDelegatePtr)`

10.2.2.11 DownloadDelegatePtr

using `cbe::delegate::DownloadDelegatePtr` = typedef `std::shared_ptr<DownloadDelegate>`

Pointer to `DownloadDelegate` that is passed into:

- `cbe::Object::download(const std::string&, DownloadDelegatePtr)`
- `cbe::Object::downloadStream(const std::string&, cbe::Stream, DownloadDelegatePtr)`

10.2.2.12 GetStreamsDelegatePtr

using `cbe::delegate::GetStreamsDelegatePtr` = typedef `std::shared_ptr<GetStreamsDelegate>`

Pointer to `GetStreamsDelegate` that is passed into:

`Object::getStreams(GetStreamsDelegatePtr)`.

10.2.2.13 GetSubscriptionsDelegatePtr

using `cbe::delegate::GetSubscriptionsDelegatePtr` = typedef `std::shared_ptr<GetSubscriptionsDelegate>`

Pointer to `GetSubscriptionsDelegate` that is passed into: `SubscribeManager::getSubscriptions(GetSubscriptionsDelegatePtr)`.

10.2.2.14 ICloudBackendListenerPtr

using `cbe::delegate::ICloudBackendListenerPtr` = typedef `std::shared_ptr<ICloudBackendListener>`

Pointer to `ICloudBackendListener`.

10.2.2.15 JoinDelegatePtr

typedef `std::shared_ptr<JoinDelegate>` `cbe::delegate::JoinDelegatePtr`

Pointer to `JoinDelegate` that is passed into `cbe::QueryChain::join()`.

Note

This is different from the group join.

10.2.2.16 KickDelegatePtr

using `cbe::delegate::KickDelegatePtr` = typedef `std::shared_ptr<KickDelegate>`

Pointer to `KickDelegate` that is passed into:

- `cbe::Member::kick(std::string, KickDelegatePtr)`

10.2.2.17 LeaveDelegatePtr

```
using cbe::delegate::LeaveDelegatePtr = typedef std::shared_ptr<LeaveDelegate>
```

Pointer to [LeaveDelegate](#) that is passed into:

[Group::leave\(LeaveDelegatePtr\)](#).

10.2.2.18 ListGroupsDelegatePtr

```
using cbe::delegate::ListGroupsDelegatePtr = typedef std::shared_ptr<ListGroupsDelegate>
```

Pointer to [ListGroupsDelegate](#) that is passed into:

[GroupManager::listGroups\(ListGroupsDelegatePtr\)](#).

10.2.2.19 ListMembersDelegatePtr

```
using cbe::delegate::ListMembersDelegatePtr = typedef std::shared_ptr<ListMembersDelegate>
```

Pointer to [ListMembersDelegate](#) that is passed into:

- [cbe::Group::listMembers\(ListMembersDelegatePtr\)](#).

10.2.2.20 ListRolesDelegatePtr

```
using cbe::delegate::ListRolesDelegatePtr = typedef std::shared_ptr<ListRolesDelegate>
```

Pointer to [ListRolesDelegate](#) that is passed into:

- [cbe::Group::ListRoles\(ListRolesDelegatePtr\)](#).

10.2.2.21 ListSharesDelegatePtr

```
using cbe::delegate::ListSharesDelegatePtr = typedef std::shared_ptr<ListSharesDelegate>
```

Pointer to [ListSharesDelegate](#) that is passed into:

- [ShareManager::listAvailableShares\(ListSharesDelegatePtr\)](#)
- [ShareManager::listMyShares\(ListSharesDelegatePtr\)](#)

10.2.2.22 LoginDelegatePtr

```
using cbe::delegate::LoginDelegatePtr = typedef std::shared_ptr<LoginDelegate>
```

Pointer to [LoginDelegate](#) that is passed into:

- [cbe::CloudBackend::login\(const std::string&,const std::string&,const std::string&,LoginDelegatePtr\)](#)

10.2.2.23 ProgressEventFn

```
using cbe::delegate::ProgressEventFn = typedef std::function<void(const ChunkTransferred&)>
```

Callback interface function that the [CloudBackend](#) SDK uses to indicate the progress of an upload/download.

10.2.2.24 PublishDelegatePtr

```
using cbe::delegate::PublishDelegatePtr = typedef std::shared_ptr<PublishDelegate>
```

Pointer to [PublishDelegate](#) that is passed into:

- [cbe::Container::publish\(\)](#)
- [cbe::Object::publish\(\)](#)

10.2.2.25 QueryDelegatePtr

```
typedef std::shared_ptr< QueryDelegate > cbe::delegate::QueryDelegatePtr
```

Pointer to [QueryDelegate](#) that is passed into:

- [CloudBackend::query\(ContainerId,delegate::QueryDelegatePtr\)](#) and overloads.
- [Container::query\(delegate::QueryDelegatePtr\)](#) and overloads.

10.2.2.26 QueryJoinDelegatePtr

```
typedef std::shared_ptr< QueryJoinDelegate > cbe::delegate::QueryJoinDelegatePtr
```

Pointer to [QueryJoinDelegate](#) that is passed into:

- [CloudBackend::query\(ContainerId,delegate::QueryJoinDelegatePtr\)](#) and overloads.
- [Container::query\(delegate::QueryJoinDelegatePtr\)](#) and overloads.

10.2.2.27 RemoveRoleDelegatePtr

```
using cbe::delegate::RemoveRoleDelegatePtr = typedef std::shared_ptr<RemoveRoleDelegate>
```

Pointer to [RemoveRoleDelegate](#) that is passed into:

[Group::removeRole\(std::string, RemoveRoleDelegatePtr\)](#).

10.2.2.28 RemoveRoleMemberDelegatePtr

```
using cbe::delegate::RemoveRoleMemberDelegatePtr = typedef std::shared_ptr<RemoveRoleMemberDelegate>
```

Pointer to [RemoveRoleMemberDelegate](#) that is passed into:

- [cbe::Group::RemoveMember\(RemoveRoleMemberDelegatePtr\)](#).

10.2.2.29 SearchGroupsDelegatePtr

```
using cbe::delegate::SearchGroupsDelegatePtr = typedef std::shared_ptr<SearchGroupsDelegate>
```

Pointer to [SearchGroupsDelegate](#) that is passed into:

- [cbe::GroupManager::searchGroups\(\)](#)

10.2.2.30 ShareDelegatePtr

```
using cbe::delegate::ShareDelegatePtr = typedef std::shared_ptr<ShareDelegate>
```

Pointer to [ShareDelegate](#) that is passed into:

- [cbe::Container::share\(cbe::UserId,std::string,ShareDelegatePtr\)](#)
- [cbe::Object::share\(cbe::UserId,std::string,ShareDelegatePtr\)](#)

10.2.2.31 SubscribeDelegatePtr

```
using cbe::delegate::SubscribeDelegatePtr = typedef std::shared_ptr<SubscribeDelegate>
```

Pointer to [SubscribeDelegate](#) that is passed into:

- [cbe::SubscribeManager::subscribe\(cbe::UserId,std::string,cbe::PublishId,std::string,std::string,std::string,SubscribeDelegatePtr\)](#)

10.2.2.32 UnBanDelegatePtr

using `cbe::delegate::UnBanDelegatePtr` = typedef `std::shared_ptr<UnBanDelegate>`

Pointer to `UnBanDelegate` that is passed into:

- `cbe::Member::unBan(UnBanDelegatePtr)`

10.2.2.33 UnPublishDelegatePtr

using `cbe::delegate::UnPublishDelegatePtr` = typedef `std::shared_ptr<UnPublishDelegate>`

Pointer to `UnPublishDelegate` that is passed into:

- `Container::unPublish(UnPublishDelegatePtr)`
- `Object::unPublish(UnPublishDelegatePtr)`

10.2.2.34 UnShareDelegatePtr

using `cbe::delegate::UnShareDelegatePtr` = typedef `std::shared_ptr<UnShareDelegate>`

Pointer to `UnShareDelegate` that is passed into:

- `cbe::Container::unShare()`
- `cbe::Object::unShare()`

10.2.2.35 UnSubscribeDelegatePtr

using `cbe::delegate::UnSubscribeDelegatePtr` = typedef `std::shared_ptr<UnSubscribeDelegate>`

Pointer to `UnSubscribeDelegate` that is passed into:

- `Container::unSubscribe(UnSubscribeDelegatePtr)`
- `Object::unSubscribe(UnSubscribeDelegatePtr)`

10.2.2.36 UpdateKeyValuesDelegatePtr

using `cbe::delegate::UpdateKeyValuesDelegatePtr` = typedef `std::shared_ptr<UpdateKeyValuesDelegate>`

Pointer to `UpdateKeyValuesDelegate` that is passed into:

- `cbe::Object::updateKeyValues()`

10.2.2.37 UploadDelegatePtr

using `cbe::delegate::UploadDelegatePtr` = typedef `std::shared_ptr<UploadDelegate>`

Pointer to `UploadDelegate` that is passed into:

- `cbe::Container::upload(const std::string&, UploadDelegatePtr)`
- `cbe::Container::upload(const std::string&, const std::string&, UploadDelegatePtr)`
- `cbe::Object::uploadStream(const std::string&, cbe::StreamId, UploadDelegatePtr)`

10.3 cbe::delegate::container Namespace Reference

Namespace for `cbe::Container` specific delegate interfaces.

Classes

- class [MoveDelegate](#)
- class [RemoveDelegate](#)
- class [RemoveSuccess](#)

Convenience type that bundles all parameters passed to method [cbe::delegate::container::RemoveDelegate::onRemoveSuccess](#).

- class [RenameDelegate](#)

Typedefs

- using [MoveDelegatePtr](#) = std::shared_ptr< [MoveDelegate](#) >
- using [RemoveDelegatePtr](#) = std::shared_ptr< [RemoveDelegate](#) >
- using [RenameDelegatePtr](#) = std::shared_ptr< [RenameDelegate](#) >

10.3.1 Detailed Description

Namespace for [cbe::Container](#) specific delegate interfaces.

10.3.2 Typedef Documentation

10.3.2.1 MoveDelegatePtr

using [cbe::delegate::container::MoveDelegatePtr](#) = typedef std::shared_ptr<[MoveDelegate](#)>
 Pointer to [MoveDelegate](#) that is passed into: [Container::move](#)(void move([cbe::ContainerId](#),[MoveDelegatePtr](#)).

10.3.2.2 RemoveDelegatePtr

using [cbe::delegate::container::RemoveDelegatePtr](#) = typedef std::shared_ptr<[RemoveDelegate](#)>
 Pointer to [RemoveDelegate](#) that is passed into:

- [cbe::Container::remove](#)([RemoveDelegatePtr](#))

10.3.2.3 RenameDelegatePtr

using [cbe::delegate::container::RenameDelegatePtr](#) = typedef std::shared_ptr<[RenameDelegate](#)>
 Pointer to [RenameDelegate](#) that is passed into: [Container::rename](#)(const std::string&,[RenameDelegatePtr](#)).

10.4 cbe::delegate::group Namespace Reference

Namespace for [cbe::Group](#) specific delegate interfaces.

Classes

- class [JoinDelegate](#)
- class [RemoveDelegate](#)
- class [RemoveSuccess](#)

Convenience type that bundles all parameters passed to method [cbe::delegate::group::RemoveDelegate::onRemoveSuccess](#).

- class [RenameDelegate](#)
- class [RenameSuccess](#)

Convenience type that bundles all parameters passed to method [cbe::delegate::group::RenameDelegate::onRenameSuccess](#).

Typedefs

- using [JoinDelegatePtr](#) = std::shared_ptr< [JoinDelegate](#) >
- using [RemoveDelegatePtr](#) = std::shared_ptr< [RemoveDelegate](#) >
- using [RenameDelegatePtr](#) = std::shared_ptr< [RenameDelegate](#) >

10.4.1 Detailed Description

Namespace for [cbe::Group](#) specific delegate interfaces.

10.4.2 Typedef Documentation

10.4.2.1 JoinDelegatePtr

using [cbe::delegate::group::JoinDelegatePtr](#) = typedef std::shared_ptr<[JoinDelegate](#)>
 Pointer to [JoinDelegate](#) that is passed into: [cbe::Group::join\(\)](#)

Note

This is different from the query chain join.

10.4.2.2 RemoveDelegatePtr

using [cbe::delegate::group::RemoveDelegatePtr](#) = typedef std::shared_ptr<[RemoveDelegate](#)>
 Pointer to [RemoveDelegate](#) that is passed into:

- [cbe::Group::remove\(RemoveDelegatePtr\)](#)

10.4.2.3 RenameDelegatePtr

using [cbe::delegate::group::RenameDelegatePtr](#) = typedef std::shared_ptr<[RenameDelegate](#)>
 Pointer to [RenameDelegate](#) that is passed into: [cbe::Group::rename\(\)](#)

10.5 cbe::delegate::object Namespace Reference

Namespace for [cbe::Object](#) specific delegate interfaces.

Classes

- class [MoveDelegate](#)
- class [RemoveDelegate](#)
- class [RemoveSuccess](#)
Convenience type that bundles all parameters passed to method [cbe::delegate::object::onRemoveSuccess](#).
- class [RenameDelegate](#)

Typedefs

- using [MoveDelegatePtr](#) = std::shared_ptr< [MoveDelegate](#) >
- using [RemoveDelegatePtr](#) = std::shared_ptr< [RemoveDelegate](#) >
- using [RenameDelegatePtr](#) = std::shared_ptr< [RenameDelegate](#) >

10.5.1 Detailed Description

Namespace for [cbe::Object](#) specific delegate interfaces.

10.5.2 Typedef Documentation

10.5.2.1 MoveDelegatePtr

using `cbe::delegate::object::MoveDelegatePtr` = typedef `std::shared_ptr<MoveDelegate>`
 Pointer to `MoveDelegate` that is passed into: `cbe::Object::move()`

10.5.2.2 RemoveDelegatePtr

using `cbe::delegate::object::RemoveDelegatePtr` = typedef `std::shared_ptr<RemoveDelegate>`
 Pointer to `RemoveDelegate` that is passed into:

- `cbe::Object::remove(RemoveDelegatePtr)`

10.5.2.3 RenameDelegatePtr

using `cbe::delegate::object::RenameDelegatePtr` = typedef `std::shared_ptr<RenameDelegate>`
 Pointer to `RenameDelegate` that is passed into: `cbe::Object::rename()`

10.6 cbe::util Namespace Reference

General support program utilities.

Classes

- struct `Context`
- struct `ErrorInfo`
- struct `ErrorInfoImpl`
- struct `Exception`
- struct `ExceptionImpl`
- class `Optional`

Class template `Optional` manages an optional contained value - i.e., a value that is either present or absent.

Functions

- template<typename T, typename... Ts>
`std::ostream & buildMsg` (`std::ostream &os`, `T &&arg`, `Ts &&... restArgs`)
- template<typename T, typename... Ts>
`std::enable_if<!std::is_base_of< std::ostream, typename std::remove_reference< T >::type >::value, std::string >::type` `buildMsg` (`T &&arg`, `Ts &&... restArgs`)
- `std::ostream & operator<<` (`std::ostream &os`, const `ErrorInfo &ei`)

10.6.1 Detailed Description

General support program utilities.

Chapter 11

Class Documentation

11.1 cbe::Account Class Reference

Login account information.

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

Public Member Functions

- [cbe::UserId](#) [userId](#) () const
Returns the account id of the user.
- [std::string](#) [username](#) () const
Returns the username of the account.
- [std::string](#) [password](#) () const
Returns the password of the account.
- [std::string](#) [source](#) () const
Returns the tenant name of the account.
- [std::string](#) [client](#) () const
Returns the client of the account.
- [std::string](#) [firstName](#) () const
Returns the given name of the user.
- [std::string](#) [lastName](#) () const
Returns the surname of the user.
- [cbe::Container](#) [rootContainer](#) () const
Returns the rootContainer for the account.
- [cbe::ContainerId](#) [tenantContainerId](#) () const
Returns the tenant ContainerId.
- [cbe::ContainerId](#) [libContainerId](#) () const
Returns the library ContainerId.
- [cbe::DatabaseId](#) [rootDatabase](#) () const
Returns the DatabaseId for home://.
- [cbe::DataBases](#) [databases](#) () const
Returns the Databases available.
- [Account](#) ([cbe::DefaultCtor](#))
Default constructor.
- [operator bool](#) () const
Checks if the current instance is real.

Friends

- class **CloudBackend**

11.1.1 Detailed Description

Login account information.

11.1.2 Constructor & Destructor Documentation

11.1.2.1 Account()

```
cbe::Account::Account (
    cbe::DefaultCtor )
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the `operator bool()` test

11.1.3 Member Function Documentation

11.1.3.1 username()

```
std::string cbe::Account::username ( ) const
```

Returns the username of the account.

Stored in lowercase.

11.1.3.2 password()

```
std::string cbe::Account::password ( ) const
```

Returns the password of the account.

What was used to login.

11.1.3.3 source()

```
std::string cbe::Account::source ( ) const
```

Returns the tenant name of the account.

Note

Stored in lowercase.

11.1.3.4 client()

```
std::string cbe::Account::client ( ) const
```

Returns the client of the account.

If it was set when the account was created.

11.1.3.5 firstName()

```
std::string cbe::Account::firstName ( ) const
```

Returns the given name of the user.

Stored in lowercase.

11.1.3.6 rootContainer()

```
cbe::Container cbe::Account::rootContainer ( ) const
```

Returns the rootContainer for the account.

The top container is also known as home://

11.1.3.7 tenantContainerId()

```
cbe::ContainerId cbe::Account::tenantContainerId ( ) const
```

Returns the tenant ContainerId.

Returns the ContainerId of the tenant database for the account. The top container is also known as tenant://

To get the [Container](#), see [Databases](#)

11.1.3.8 libContainerId()

```
cbe::ContainerId cbe::Account::libContainerId ( ) const
```

Returns the library ContainerId.

The ContainerId of the library database for the account, which is where application specific settings and application state can be permanently stored in order to be available next time the app starts.

11.1.3.9 rootDatabase()

```
cbe::DatabaseId cbe::Account::rootDatabase ( ) const
```

Returns the DatabaseId for home://.

Returns the databaseId for root a.k.a. home:// of the account.

11.1.3.10 databases()

```
cbe::DataBases cbe::Account::databases ( ) const
```

Returns the Databases available.

Returns a std::map of databases available for the account.

11.1.3.11 operator bool()

```
cbe::Account::operator bool ( ) const [explicit]
```

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [Account\(cbe::DefaultCtor\)](#)

Returns

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/Account.h

11.2 cbe::delegate::AclDelegate Class Reference

```
#include <AclDelegate.h>
```

Classes

- struct [ErrorInfo](#)
- struct [Exception](#)
exception thrown by [cbe::Object::getAcl\(\)](#) if the request fails.

Public Types

- using **Success** = [AclMap](#)
- using **Error** = [delegate::Error](#)

Public Member Functions

- virtual void [onAclSuccess](#) ([AclMap](#) &&aclMap)=0
- virtual void [onAclError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

11.2.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Container::getAcl\(\)](#)
- [cbe::Container::setAcl\(\)](#)
- [cbe::Object::getAcl\(\)](#)
- [cbe::Object::setAcl\(\)](#)

11.2.2 Member Function Documentation

11.2.2.1 onAclSuccess()

```
virtual void cbe::delegate::AclDelegate::onAclSuccess (
    AclMap && aclMap ) [pure virtual]
```

Called upon successful GetACL.

Parameters

<i>aclMap</i>	The permissions for requested cbe::Container or cbe::Object .
---------------	---

11.2.2.2 onAclError()

```
virtual void cbe::delegate::AclDelegate::onAclError (
    Error && error,
    cbe::util::Context && context ) [pure virtual]
```

Called if an error is encountered.

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

- `include/cbe/delegate/AclDelegate.h`

11.3 cbe::delegate::AddRoleMemberDelegate Class Reference

```
#include <AddRoleMemberDelegate.h>
```

Classes

- struct [ErrorInfo](#)
- struct [Exception](#)
exception thrown by cbe::Group::AddMember() if the request fails.

Public Types

- using **Success** = [MemberId](#)
- using **Error** = [delegate::Error](#)

Public Member Functions

- virtual void [onAddRoleMemberSuccess](#) ([MemberId](#) memberId)=0
- virtual void [onAddRoleMemberError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

11.3.1 Detailed Description

Delegate class for the asynchronous version of method:

- cbe::Group::AddMember

11.3.2 Member Function Documentation

11.3.2.1 onAddRoleMemberSuccess()

```
virtual void cbe::delegate::AddRoleMemberDelegate::onAddRoleMemberSuccess (
    MemberId memberId ) [pure virtual]
```

Called upon successful AddMember.

11.3.2.2 onAddRoleMemberError()

```
virtual void cbe::delegate::AddRoleMemberDelegate::onAddRoleMemberError (
    Error && error,
    cbe::util::Context && context ) [pure virtual]
```

Called if an error is encountered.

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

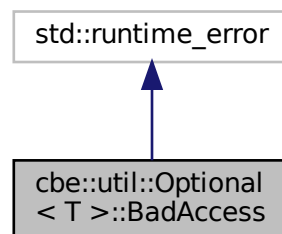
- include/cbe/delegate/AddRoleMemberDelegate.h

11.4 cbe::util::Optional< T >::BadAccess Class Reference

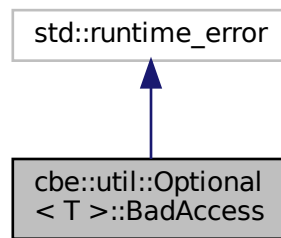
Thrown by [value\(\)](#) method when accessing an object that does not contain a value.

```
#include <Optional.h>
```

Inheritance diagram for cbe::util::Optional< T >::BadAccess:



Collaboration diagram for `cbe::util::Optional< T >::BadAccess`:



11.4.1 Detailed Description

```
template<typename T>
class cbe::util::Optional< T >::BadAccess
```

Thrown by [value\(\)](#) method when accessing an object that does not contain a value.
The documentation for this class was generated from the following file:

- `include/cbe/util/Optional.h`

11.5 cbe::delegate::BanDelegate Class Reference

```
#include <BanDelegate.h>
```

Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

exception thrown by `cbe::Member::ban(std::string)` if the request fails.

Public Types

- using **Success** = [BanSuccess](#)
- using **Error** = [delegate::Error](#)

Public Member Functions

- virtual void [onBanSuccess](#) (std::string &&memberName, [cbe::MemberId](#) memberId)=0
- virtual void [onBanError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

11.5.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Member::ban](#)

11.5.2 Member Function Documentation

11.5.2.1 onBanSuccess()

```
virtual void cbe::delegate::BanDelegate::onBanSuccess (
    std::string && memberName,
    cbe::MemberId memberId ) [pure virtual]
```

Called upon successful ban.

Parameters

<i>memberName</i>	Name of the member being banned.
<i>memberId</i>	Id of the member being banned.

11.5.2.2 onBanError()

```
virtual void cbe::delegate::BanDelegate::onBanError (
    Error && error,
    cbe::util::Context && context ) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/BanDelegate.h

11.6 cbe::delegate::BanSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::BanDelegate::onBanSuccess](#).

```
#include <BanDelegate.h>
```

Public Member Functions

- **BanSuccess** ([cbe::DefaultCtor](#))
- **BanSuccess** (std::string &&memberName, [cbe::MemberId](#) memberId)

Public Attributes

- std::string **memberName** {}
- [cbe::MemberId](#) **memberId** {}

11.6.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::BanDelegate::onBanSuccess](#).

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

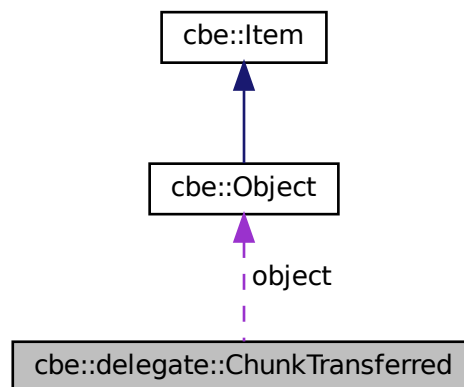
- include/cbe/delegate/BanDelegate.h

11.7 cbe::delegate::ChunkTransferred Class Reference

Bundles the progress event information in connection with download and upload.

```
#include <ChunkTransferred.h>
```

Collaboration diagram for `cbe::delegate::ChunkTransferred`:



Public Member Functions

- **ChunkTransferred** ([cbe::DefaultCtor](#))
- **ChunkTransferred** ([cbe::Object](#) &&object, std::uint64_t transferred, std::uint64_t total)

Public Attributes

- [cbe::Object](#) **object** {[cbe::DefaultCtor](#){}}
- std::uint64_t **transferred** {}
- std::uint64_t **total** {}

11.7.1 Detailed Description

Bundles the progress event information in connection with download and upload.
Convenience type that bundles all parameters passed to methods:

- [cbe::delegate::DownloadDelegate::onChunkReceived\(\)](#)
- [cbe::delegate::DownloadBinaryDelegate::onChunkReceived\(\)](#)
- [cbe::delegate::UploadDelegate::onChunkSent\(\)](#)
- [cbe::delegate::UploadBinaryDelegate::onChunkSent\(\)](#)

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

- `include/cbe/delegate/ChunkTransferred.h`

11.8 cbe::CloudBackend Class Reference

The session that holds the connection with the cloud.

```
#include <CloudBackend.h>
```

Public Types

- using `LoginDelegatePtr` = `delegate::LoginDelegatePtr`
- using `LoginException` = `delegate::LoginDelegate::Exception`
- using `LoginError` = `delegate::LoginDelegate::ErrorInfo`
- using `CreateAccountDelegatePtr` = `delegate::CreateAccountDelegatePtr`
- using `CreateAccountException` = `delegate::CreateAccountDelegate::Exception`
- using `CreateAccountError` = `delegate::CreateAccountDelegate::ErrorInfo`
- using `CloudBackendListenerDelegatePtr` = `std::shared_ptr< delegate::CloudBackendListenerDelegate >`
- using `ListenerHandle` = `std::uint64_t`
Handle identifying a registered Listener.
- using `QueryDelegatePtr` = `delegate::QueryDelegatePtr`
- using `QueryJoinDelegatePtr` = `delegate::QueryJoinDelegatePtr`
- using `QueryException` = `delegate::QueryDelegate::Exception`
- using `QueryJoinError` = `delegate::QueryJoinDelegate::ErrorInfo`
- using `QueryError` = `delegate::QueryDelegate::ErrorInfo`
- using `SearchException` = `delegate::QueryDelegate::Exception`
- using `SearchError` = `delegate::QueryDelegate::ErrorInfo`

Public Member Functions

- `ListenerHandle addListener (CloudBackendListenerDelegatePtr listener)`
Listen for changes to specific items.
- `void removeListener (ListenerHandle handle)`
Delete a specific listener.
- `cbe::QueryChain query (ContainerId containerId, QueryDelegatePtr queryDelegate)`
Select Item's of a container (table).
- `cbe::QueryChain query (ContainerId containerId, Filter filter, QueryDelegatePtr queryDelegate)`
Select Item's of a container (table) with a filter (where).
- `cbe::QueryChainExt query (ContainerId containerId, QueryJoinDelegatePtr queryJoinDelegate)`
Join multiple tables.
- `cbe::QueryChainExt query (ContainerId containerId, Filter filter, QueryJoinDelegatePtr queryJoinDelegate)`
Join multiple tables using filter (where).
- `cbe::QueryChainSync query (ContainerId containerId)`
Synchronous [exception].
- `cbe::QueryChainSync query (ContainerId containerId, Filter filter)`
Synchronous [exception] filtered.
- `cbe::QueryChainSync query (ContainerId containerId, QueryJoinError &error)`
Synchronous [non-throwing] query.
- `cbe::QueryChainSync query (ContainerId containerId, Filter filter, QueryJoinError &error)`
Synchronous [non-throwing] filtered query.
- `cbe::QueryChain queryWithPath (std::string relativePath, cbe::ContainerId queryRoot, QueryDelegatePtr delegate)`
Queries items of a container with a given path.
- `cbe::QueryChain queryWithPath (std::string relativePath, QueryDelegatePtr delegate)`
Queries items of a container with a given path relative the home root container.
- `cbe::QueryResult search (std::string tags, cbe::ContainerId containerId, QueryDelegatePtr delegate)`
Select Object's with specified key/values.
- `cbe::QueryResult search (std::string tags, cbe::ContainerId containerId)`
*Synchronous [exception] **Synchronous** version of `search(std::string, cbe::ContainerId, QueryDelegatePtr)` , and **throws an exception**, `SearchException`, in case of a failed call.
See `search(QueryDelegatePtr)`*

- [cbe::util::Optional](#)< [cbe::QueryResult](#) > [search](#) (std::string tags, [cbe::ContainerId](#) containerId, [SearchError](#) &error)
*Synchronous [non-throwing] search **Synchronous** version of [search\(tags, containerId, QueryDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*
See [search\(tags, containerId, QueryDelegatePtr\)](#)
- [cbe::QueryResult](#) [search](#) ([cbe::Filter](#) filter, [cbe::ContainerId](#) containerId, [QueryDelegatePtr](#) delegate)
Select [Object](#)'s with specified key/values using filter.
- [cbe::QueryResult](#) [search](#) ([cbe::Filter](#) filter, [cbe::ContainerId](#) containerId)
*Synchronous [exception] **Synchronous** version of [search\(cbe::Filter, cbe::ContainerId, QueryDelegatePtr\)](#) , and **throws an exception**, [SearchException](#), in case of a failed call.*
See [search\(QueryDelegatePtr\)](#)
- [cbe::util::Optional](#)< [cbe::QueryResult](#) > [search](#) ([cbe::Filter](#) filter, [cbe::ContainerId](#) containerId, [SearchError](#) &error)
*Synchronous [non-throwing] search **Synchronous** version of [search\(filter,containerId,QueryDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*
See [search\(filter,containerId, QueryDelegatePtr\)](#)
- bool [clearCache](#) ()
Clear the local cache.
- [cbe::Account](#) [account](#) ()
Returns an account object with information on the user.
- std::string [version](#) () const
Returns the version number of the SDK.
- [cbe::GroupManager](#) [groupManager](#) ()
Gets the group manager.
- [cbe::ShareManager](#) [shareManager](#) ()
Gets the share manager.
- [cbe::PublishManager](#) [publishManager](#) ()
Gets the publish manager.
- void [terminate](#) ()
Terminates the CloudBackend service.
- [cbe::SubscribeManager](#) [subscribeManager](#) ()
Gets the subscribe manager.
- [CloudBackend](#) ([cbe::DefaultCtor](#))
Default constructor.
- operator bool () const
Checks if the current instance is real.

Static Public Member Functions

- static [cbe::CloudBackend](#) [login](#) (const std::string &username, const std::string &password, const std::string &tenant, const std::string &client, [LoginDelegatePtr](#) delegate)
Logs in to the CloudBackend service.
- static [cbe::CloudBackend](#) [login](#) (const std::string &username, const std::string &password, const std::string &tenant, const std::string &client)
Synchronous [exception] login.
- static [cbe::CloudBackend](#) [login](#) (const std::string &username, const std::string &password, const std::string &tenant, const std::string &client, [LoginError](#) &error)
Synchronous [non-throwing] login.
- static [cbe::UserId](#) [createAccount](#) (const std::string &username, const std::string &password, const std::string &email, const std::string &firstName, const std::string &lastName, const std::string &tenant, const std::string &client, [CreateAccountDelegatePtr](#) delegate)
Creates a user account.

- static [cbe::UserId createAccount](#) (const std::string &username, const std::string &password, const std::string &email, const std::string &firstName, const std::string &lastName, const std::string &tenant, const std::string &client)
Synchronous [exception] Creates a user account.
- static [cbe::util::Optional< cbe::UserId > createAccount](#) (const std::string &username, const std::string &password, const std::string &email, const std::string &firstName, const std::string &lastName, const std::string &tenant, const std::string &client, [CreateAccountError](#) &error)
*Synchronous [no exception] **Synchronous** version of `createAccount(username,password,email,firstname,lastname,tenant,client,↔ CreateAccountDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*
See `createAccount(username,password,email,firstname,lastname,tenant,client,CreateAccountDelegatePtr)`
- static [cbe::Container castContainer](#) ([cbe::Item](#) item)
Casts an item to a container.
- static [cbe::Object castObject](#) ([cbe::Item](#) item)
Casts an item to an object.

11.8.1 Detailed Description

The session that holds the connection with the cloud.
 It ensures the connection is authenticated.

11.8.2 Member Typedef Documentation

11.8.2.1 LogInDelegatePtr

```
using cbe::CloudBackend::LogInDelegatePtr = delegate::LogInDelegatePtr
```

Pointer to [cbe::delegate::LogInDelegate\(\)](#) that is passed into asynchronous version of method `login()`

11.8.2.2 LogInException

```
using cbe::CloudBackend::LogInException = delegate::LogInDelegate::Exception
```

See also

[delegate::LogInDelegate::Exception](#)

11.8.2.3 LogInError

```
using cbe::CloudBackend::LogInError = delegate::LogInDelegate::ErrorInfo
```

Forms the type of the `error` return parameter for the synchronous version of method `login()`

See also

[delegate::LogInDelegate::ErrorInfo](#)

11.8.2.4 CreateAccountDelegatePtr

```
using cbe::CloudBackend::CreateAccountDelegatePtr = delegate::CreateAccountDelegatePtr
```

Pointer to [delegate::CreateAccountDelegate](#) that is passed into asynchronous version of method `createAccount()`

11.8.2.5 CreateAccountException

```
using cbe::CloudBackend::CreateAccountException = delegate::CreateAccountDelegate::Exception
```

See [delegate::CreateAccountDelegate::Exception](#)

11.8.2.6 CreateAccountError

using `cbe::CloudBackend::CreateAccountError = delegate::CreateAccountDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `createAccount()`

See [delegate::CreateAccountDelegate::ErrorInfo](#)

11.8.2.7 CloudBackendListenerDelegatePtr

using `cbe::CloudBackend::CloudBackendListenerDelegatePtr = std::shared_ptr<delegate::CloudBackendListenerDelegate>`

Pointer to [delegate::CloudBackendListenerDelegate](#) that is passed into asynchronous version of method `addListener(CloudBackendListenerDelegatePtr)`

11.8.2.8 QueryDelegatePtr

using `cbe::CloudBackend::QueryDelegatePtr = delegate::QueryDelegatePtr`

Pointer to [delegate::QueryDelegate](#) that is passed into asynchronous version of methods:

- [query\(ContainerId, QueryDelegatePtr\)](#),
- [query\(ContainerId, Filter, QueryDelegatePtr\)](#),
- [queryWithPath\(std::string, QueryDelegatePtr\)](#)
- [queryWithPath\(std::string, cbe::ContainerId, QueryDelegatePtr\)](#),
- [search\(std::string, cbe::ContainerId, QueryDelegatePtr\)](#),
- [search\(Filter, cbe::ContainerId, QueryDelegatePtr\)](#)

11.8.2.9 QueryJoinDelegatePtr

using `cbe::CloudBackend::QueryJoinDelegatePtr = delegate::QueryJoinDelegatePtr`

Pointer to [delegate::QueryJoinDelegate](#) that is passed into asynchronous version of methods

- [query\(ContainerId, QueryJoinDelegatePtr\)](#)
- [query\(ContainerId, Filter, QueryJoinDelegatePtr\)](#)

11.8.2.10 QueryException

using `cbe::CloudBackend::QueryException = delegate::QueryDelegate::Exception`

See also

[delegate::QueryDelegate::Exception](#)

11.8.2.11 QueryJoinError

using `cbe::CloudBackend::QueryJoinError = delegate::QueryJoinDelegate::ErrorInfo`

See also

[delegate::QueryJoinDelegate::ErrorInfo](#)

11.8.2.12 QueryError

using `cbe::CloudBackend::QueryError = delegate::QueryDelegate::ErrorInfo`

See also

[delegate::QueryDelegate::ErrorInfo](#)

11.8.2.13 SearchException

using `cbe::CloudBackend::SearchException = delegate::QueryDelegate::Exception`

Pointer to [cbe::delegate::QueryDelegate](#) that is passed into asynchronous version of method `search()` See [delegate::object::QueryDelegate::Exception](#)

11.8.2.14 SearchError

using `cbe::CloudBackend::SearchError = delegate::QueryDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `search()`

See [delegate::QueryDelegate::ErrorInfo](#)

11.8.3 Constructor & Destructor Documentation

11.8.3.1 CloudBackend()

```
cbe::CloudBackend::CloudBackend (
    cbe::DefaultCtor )
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the [operator bool\(\)](#) test

11.8.4 Member Function Documentation

11.8.4.1 logIn() [1/3]

```
static cbe::CloudBackend cbe::CloudBackend::logIn (
    const std::string & username,
    const std::string & password,
    const std::string & tenant,
    const std::string & client,
    LoginDelegatePtr delegate ) [static]
```

Logs in to the CloudBackend service.

When finished with the usage of the CloudBackend service, call method [CloudBackend::terminate\(\)](#) in all cases.

The authentication to the account is determined by `username`, `password` and `tenant`.

Asynchronous version of this service function.

Parameters

<i>username</i>	Name of the user to be signed in.
<i>password</i>	Password for the user to be signed in.
<i>tenant</i>	The identifier for the tenant, formerly known as <code>source</code> .
<i>client</i>	Identifier of what type of client the program is running on. This is used by the tenant for statistics and selective communication with the users. The tenant administrator defines what client names to use.

Parameters

<i>delegate</i>	Pointer to a <code>be::CloudBackend::LogInDelegate()</code> instance that is implemented by the user to receive the response of this login request. This is accomplished in terms of the <code>LogInDelegate</code> callback functions onLogInSuccess() and onLogInError() .
-----------------	---

Example

Async login

```

#include "cbe/CloudBackend.h"
~~~
#include <condition_variable>
#include <mutex>
~~~
int main() {
    // First, declare and implement a LogInDelegate class
    class MyLogInDelegate : public cbe::delegate::LogInDelegate {
        std::mutex          mutex{};
        std::condition_variable conditionVariable{};
        // Indicates operation completed - success or failure
        bool                called{};
    public:
        // Default construct this CloudBackend member. If the method
        // onLogInSuccess() will not not be called, this default constructed state
        // implies that this cloudBackend object is not valid due to a successful
        // log in
        cbe::CloudBackend cloudBackend{ cbe::DefaultCtor{} };
        // Default construct this cbe::delegate::Error member, If the method
        // onLogInError() will not not be called, this default constructed state
        // implies no error. Otherwise, thus error object will contain the error
        // information.
        cbe::delegate::Error error{};
        // Implement the cbe::delegate::LogInDelegate interface callbacks (private
        // cause they are only used by the SDK):
    private:
        // Callback called upon successful login.
        void onLogInSuccess(cbe::CloudBackend&& cloudBackend) final {
            {
                std::lock_guard<std::mutex> lock(mutex);
                // Put error member into the default constructed state to
                // indicate success (i.e., no error)
                error = cbe::delegate::Error{};
                // Change state of cloudBackend member to indicate success
                this->cloudBackend = std::move(cloudBackend);
                // Indicate operation completed, member cloudBackend non-default
                // constructed state indicates success
                called = true;
            }
            conditionVariable.notify_one();
        }
        // Callback called upon failed login.
        void onLogInError(Error&& error, cbe::util::Context&& context) final {
            {
                std::lock_guard<std::mutex> lock(mutex);
                // Put cloudBackend member into the default constructed state to
                // indicate no-success
                cloudBackend = cbe::CloudBackend{ cbe::DefaultCtor{} };
                // Change state of error member to indicate failure
                this->error = std::move(error);
                // Indicate operation completed, member member cloudBackend default
                // constructed state indicates failure
                called = true;
            }
            conditionVariable.notify_one();
        }
    };
    // -----
    public:
        void waitForRsp() {
            std::unique_lock<std::mutex> lock(mutex);
            conditionVariable.wait(lock, [this]{ return called; });
            // Reset called flag, so current delegate instance can be reused
            called = false;
        }
    }; // struct MyLogInDelegate
    ~~~
    ~~~
    // Second, initiate the login:
    static constexpr const char* const username = "~~~";
    static constexpr const char* const password = "~~~";
    static constexpr const char* const tenant   = "~~~";
    static constexpr const char* const client   = "linux_desktop";

```

```

    ~~~
    std::shared_ptr<MyLoginDelegate> myLoginDelegate =
        std::make_shared<MyLoginDelegate>();
    // Invoke the login (cast to return value to void to indicate is deliberately
    // discarded)
    cbe::CloudBackend cloudBackend = cbe::CloudBackend::login(username, password, tenant, client,
myLoginDelegate);
    // Await the login procedure to run to completion
    myLoginDelegate->waitForRsp();
    // Test if login failed
    if (myLoginDelegate->error) {
        std::cout << "Failed to login: error={" << myLoginDelegate->error << '}'
            << std::endl;
        // Bail out!
        ~~~
    }
    // Since we have a complete CloudBackend object now, copy it from the delegate
    cloudBackend = myLoginDelegate->cloudBackend;
    // Dispose the delegate object, not needed anymore
    myLoginDelegate.reset();
    // Do your stuff
    ~~~
    // End of program, tell cbe to terminate the session
    cloudBackend.terminate();
}

```

11.8.4.2 login() [2/3]

```

static cbe::CloudBackend cbe::CloudBackend::login (
    const std::string & username,
    const std::string & password,
    const std::string & tenant,
    const std::string & client ) [static]

```

Synchronous [exception] login.

Synchronous version of [login\(\)](#), and **throws an exception**, [LoginException](#), in case of a failed login.

Returns

A CloudBackend instance — if login was successful.

Exceptions

LoginException	
--------------------------------	--

Example

Synchronous [exception] login

```

cbe::CloudBackend cloudBackend{ cbe::DefaultCtor{} };
try {
    cloudBackend = cbe::CloudBackend::login(username, password, tenant, client);
}
catch (cbe::CloudBackend::LoginException& e) {
    std::cout << "Failed to login: error={" << e << '}'
        << std::endl;
    // Bail out!
    ~~~
}
catch (std::runtime_error& e) {
    std::cout << "Got runtime exception: {" << e << '}'
        << std::endl;
}
catch (...) {
    std::cout << "Got other exception!"
        << std::endl;
}

```

11.8.4.3 login() [3/3]

```

static cbe::CloudBackend cbe::CloudBackend::login (
    const std::string & username,

```

```

    const std::string & password,
    const std::string & tenant,
    const std::string & client,
    LoginError & error ) [static]

```

Synchronous [non-throwing] login.

Synchronous version of `login()`, and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information.

Parameters

<code>out</code>	<code>error</code>	Return parameter containing the error information in case of a failed login. The return value, of type <code>CloudBackend</code> , will indicate false on a failed login, and the <code>LoginError</code> object passed in will be populated with the error information.
------------------	--------------------	--

Returns

If empty, **false**, an error has occurred. I.e., the login has failed, and the error information has been passed out via the `error` out/return parameter.

Example

Synchronous [non-throwing] login

```

cbe::CloudBackend cloudBackend{ cbe::DefaultCtor{} };
cbe::CloudBackend::LoginError error{};
cloudBackend = cbe::CloudBackend::login(username, password, tenant, error, client);
if (!cloudBackend) {
    std::cout << "Failed to login: error={" << error << "}"
              << std::endl;
    // Bail out!
    ~~~
}

```

11.8.4.4 createAccount() [1/3]

```

static cbe::UserId cbe::CloudBackend::createAccount (
    const std::string & username,
    const std::string & password,
    const std::string & email,
    const std::string & firstName,
    const std::string & lastName,
    const std::string & tenant,
    const std::string & client,
    CreateAccountDelegatePtr delegate ) [static]

```

Creates a user account.

This method is only granted to the tenant administrator.

Asynchronous version of this service function.

Parameters

<code>username</code>	Username for the new account
<code>password</code>	Password for the new account
<code>email</code>	Email address to owner of the new account
<code>firstName</code>	First name of the owner of the new account
<code>lastName</code>	Last name of the owner of the new account
<code>tenant</code>	The identifier for the tenant, formerly known as <code>source</code> .
<code>client</code>	Describing platform running the software. Names defined by the tenant administrator.
<code>delegate</code>	Pointer to a <code>delegate::CreateAccountDelegate</code> instance that is implemented by the user.

11.8.4.5 createAccount() [2/3]

```
static cbe::UserId cbe::CloudBackend::createAccount (
    const std::string & username,
    const std::string & password,
    const std::string & email,
    const std::string & firstName,
    const std::string & lastName,
    const std::string & tenant,
    const std::string & client ) [static]
```

Synchronous [exception] Creates a user account.

Synchronous version of [createAccount\(\)](#), and **throws an exception**, [CreateAccountException](#), in case of a failed account creation.

See [createAccount\(\)](#)

Returns

A [CloudBackend](#) instance associated with the created account — if account creation was successful.

Exceptions

CreateAccountException	
--	--

11.8.4.6 createAccount() [3/3]

```
static cbe::util::Optional<cbe::UserId> cbe::CloudBackend::createAccount (
    const std::string & username,
    const std::string & password,
    const std::string & email,
    const std::string & firstName,
    const std::string & lastName,
    const std::string & tenant,
    const std::string & client,
    CreateAccountError & error ) [static]
```

Synchronous [no exception] **Synchronous** version of `createAccount(username,password,email,firstname,lastname,tenant,client,↵ CreateAccountDelegatePtr)`, and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `createAccount(username,password,email,firstname,lastname,tenant,client,CreateAccountDelegatePtr)`

Parameters

out	error	Return parameter containing the error information in case of a failed call. An empty return value will indicate failure, and the CreateAccountError object passed in will be populated with the error information.
-----	-------	---

Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

11.8.4.7 addListener()

```
ListenerHandle cbe::CloudBackend::addListener (
```

```
CloudBackendListenerDelegatePtr listener )
```

Listen for changes to specific items.

Adds a listener that will receive updates as changes occur on the account.

Note

`removeListener()` should always be called when no longer using the delegate.

Parameters

<i>listener</i>	Delegate is a shared pointer to the class <code>delegate::CloudBackendListenerDelegate</code> that the user has implemented
-----------------	---

Returns

Handle identifying the registration of the listener.

To be used when de-registering with method `removeListener(ListenerHandle)`.

11.8.4.8 removeListener()

```
void cbe::CloudBackend::removeListener (
    ListenerHandle handle )
```

Delete a specific listener.

Removes a listener that will receive updates as changes occur on the account

Parameters

<i>handle</i>	Handle previously retrieved from method <code>addListener(CloudBackendListenerDelegatePtr)</code> .
---------------	---

11.8.4.9 query() [1/8]

```
cbe::QueryChain cbe::CloudBackend::query (
    ContainerId containerId,
    QueryDelegatePtr queryDelegate )
```

Select Item's of a container (table).

Inquires for a set of `Items` from the provided Container, identified by `containerId`. Response is delivered asynchronously via the `delegate::QueryDelegate` callback interface passed in as argument via parameter `queryDelegate`.

This overload of `join()` accepting a `QueryDelegate`-pointer as parameter has two use cases:

1. To make a solely `query()` call **without** an additional chained `join()`-call.
2. To make a `query()` call **with** a desired chained `join()` call.
In this latter case, the overloaded `query()`-functions:

- `query(ContainerId,delegate::QueryJoinDelegatePtr)`
- `query(ContainerId,Filter,delegate::QueryJoinDelegatePtr)`

both accepting a `QueryJoinDelegate` as `delegate` argument, must be used.

Asynchronous version of this service function.

Parameters

<i>containerId</i>	Id of the container which contents will inquired.
--------------------	---

Parameters

<i>queryDelegate</i>	Pointer to a QueryDelegate instance, implemented by the user, that receives the response of this query request. I.e., either the QueryDelegate callback function onQuerySuccess() or onQueryError() will be called in the event of success, or failure respectively.
----------------------	---

Example

Async query

```

~~~
std::shared_ptr<MyQueryDelegate> queryDelegate =
    std::make_shared<MyQueryDelegate>();
const cbe::ContainerId myContainerId =
    cloudBackend.account().rootContainer().id();
cloudBackend.query(myContainerId, queryDelegate);
queryDelegate->waitForRsp();
if (queryDelegate->errorInfo) {
    std::cout << "Query failed:" << std::endl;
    std::cout << "ErrorInfo = " << queryDelegate->errorInfo << std::endl;
} else {
    cbe::QueryResult::ItemsSnapshot itemsSnapshot =
        queryDelegate->queryResult.getItemsSnapshot();
    for (auto& item : itemsSnapshot) {
        std::cout << item.name() << std::endl;
    }
}
~~~

```

To use the code above, we must first login and then declare a [QueryDelegate](#) class.

See also:

- [Async login](#)
Here, we retrieve the `cloudBackend` object.
- [Example QueryDelegatePtr of a QueryDelegate implementation class](#)
Here, class `MyQueryDelegate` is defined.

11.8.4.10 query() [2/8]

```

cbe::QueryChain cbe::CloudBackend::query (
    ContainerId containerId,
    Filter filter,
    QueryDelegatePtr queryDelegate )

```

Select Item's of a container (table) with a filter (where).

Same as [query\(ContainerId,QueryDelegatePtr\)](#), but with an additional parameter `filter`.

Parameters

<i>filter</i>	Filter specifying the constraints of the requested items.
---------------	---

11.8.4.11 query() [3/8]

```

cbe::QueryChainExt cbe::CloudBackend::query (
    ContainerId containerId,
    QueryJoinDelegatePtr queryJoinDelegate )

```

Join multiple tables.

Same as [query\(ContainerId,delegate::QueryDelegatePtr\)](#), but with a [QueryJoinDelegate](#) as delegate, `query←JoinDelegate`.

See also

[query\(ContainerId,delegate::QueryDelegatePtr\)](#)

Parameters

<i>queryJoinDelegate</i>	Pointer to a QueryJoinDelegate instance, implemented by the user, that receives the response of this query request. I.e., either its callback function onQueryJoinSuccess() or onQueryJoinError() will be called in the event of success or failure respectively.
--------------------------	--

11.8.4.12 query() [4/8]

```
cbe::QueryChainExt cbe::CloudBackend::query (
    ContainerId containerId,
    Filter filter,
    QueryJoinDelegatePtr queryJoinDelegate )
```

Join multiple tables using filter (where).

Same as [query\(ContainerId,delegate::QueryJoinDelegatePtr\)](#), but with an additional [Filter](#) parameter *filter*.

Parameters

<i>filter</i>	Filter specifying the constraints of the requested items.
---------------	---

11.8.4.13 query() [5/8]

```
cbe::QueryChainSync cbe::CloudBackend::query (
    ContainerId containerId )
```

Synchronous [exception].

Synchronous version of [query\(ContainerId,delegate::QueryDelegatePtr\)](#), and **throws an exception**, [QueryException](#), in case of a failed query.

Returns

Upon success, a [QueryResult](#) object - in fact an instance of [QueryChainSync](#) that is a subclass of [QueryResult](#).

Exceptions

QueryException	
--------------------------------	--

Example

Sync [exception] query

```
~~~~~
const cbe::ContainerId myContainerId = cloudBackend.account().rootContainer().id();
try
{
    auto queryResult = cloudBackend.query(myContainerId).getQueryResult();
    cbe::QueryResult::ItemsSnapshot itemsSnapshot = queryResult.getItemsSnapshot();
    for (auto& item : itemsSnapshot) {
        std::cout << item.name() << std::endl;
    }
}
catch (const cbe::CloudBackend::QueryException& e)
{
    std::cout << "Error!" << std::endl << e.what() << std::endl;
    ~~~~
}
```

~~~

To use the code above, we must first login.

See also:

- [Sync \[exception\] login](#)

Here, we retrieve the `cloudBackend` class object.

#### 11.8.4.14 query() [6/8]

```
cbe::QueryChainSync cbe::CloudBackend::query (
    ContainerId containerId,
    Filter filter )
```

Synchronous [exception] filtered.

Extended version of [query\(ContainerId\)](#) with an additional filter parameter `filter`.

##### Parameters

|               |                                                                           |
|---------------|---------------------------------------------------------------------------|
| <i>filter</i> | <a href="#">Filter</a> specifying the constraints of the requested items. |
|---------------|---------------------------------------------------------------------------|

#### 11.8.4.15 query() [7/8]

```
cbe::QueryChainSync cbe::CloudBackend::query (
    ContainerId containerId,
    QueryJoinError & error )
```

Synchronous [non-throwing] query.

In line with synchronous [query\(ContainerId\)](#), but **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information.

##### Parameters

|            |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed query.<br>The return value, of type <a href="#">QueryChainSync</a> , will indicate <b>false</b> on a failed query, and the passed in <a href="#">QueryJoinError</a> object will be populated with the error information.<br>Since there might be multiple chained calls after the return from this method, the error information will also contain the index of the failed chained <code>join()</code> call. |
|------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

##### Returns

An object of [QueryResult](#) - in fact an instance of [QueryChainSync](#) that is a subclass of [QueryResult](#).

If empty - i.e., **false** an error has occurred, i.e., a failed query, and the error information has been passed out via the `error` out/return parameter.

##### Example

##### Sync [non-throwing] query

```
~~~
const cbe::ContainerId myContainerId = cloudBackend.account().rootContainer().id();
cbe::CloudBackend::QueryError queryError;
auto queryResult = cloudBackend.query(myContainerId).getQueryResult();
if (queryError) {
 std::cout << "Error! " << queryError << std::endl;
    ~~~
} else {
    cbe::QueryResult::ItemsSnapshot itemsSnapshot = queryResult.getItemsSnapshot();
    for (auto& item : itemsSnapshot) {
        std::cout << item.name() << std::endl;
    }
}
~~~
```

**11.8.4.16 query()** [8/8]

```
cbe::QueryChainSync cbe::CloudBackend::query (
 ContainerId containerId,
 Filter filter,
 QueryJoinError & error)
```

Synchronous [non-throwing] filtered query.

Extended version of [query\(ContainerId,QueryJoinError&\)](#) with an additional filter parameter `filter`.

See also

[query\(ContainerId,QueryJoinError&\)](#) To use the code above, we must first login.

See also:

- [Sync \[non-throwing\] login](#)  
Here, we retrieve the `cloudBackend` class object.

**11.8.4.17 queryWithPath()** [1/2]

```
cbe::QueryChain cbe::CloudBackend::queryWithPath (
 std::string relativePath,
 cbe::ContainerId queryRoot,
 QueryDelegatePtr delegate)
```

Queries items of a container with a given path.

Note

`..` or `.` relative path options are not permitted.

Only top down search from start point, `queryRoot` id to downwards path in container tree.

Parameters

|                     |                                                                                                                                                                                                                  |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>relativePath</i> | The relative path from the <code>queryRoot</code> .<br>E.g.: <code>/Documents/Pictures</code><br>from a <code>queryRoot</code> that has the nested containers <code>Documents</code> and <code>Pictures</code> . |
| <i>queryRoot</i>    | The container id forming the root of this query.                                                                                                                                                                 |
| <i>delegate</i>     | Pointer to a <a href="#">delegate::QueryDelegate</a> instance that is implemented by the user.                                                                                                                   |

**11.8.4.18 queryWithPath()** [2/2]

```
cbe::QueryChain cbe::CloudBackend::queryWithPath (
 std::string relativePath,
 QueryDelegatePtr delegate)
```

Queries items of a container with a given path relative the home root container.

Same as [queryWithPath\(std::string, cbe::ContainerId, QueryDelegatePtr\)](#), but with the `queryRoot` parameter omitted. Instead, here the home root container is used as top container in the search tree.

**11.8.4.19 search()** [1/6]

```
cbe::QueryResult cbe::CloudBackend::search (
 std::string tags,
 cbe::ContainerId containerId,
 QueryDelegatePtr delegate)
```

Select [Object](#)'s with specified key/values.

Search the whole container for tags related to [Objects](#) in the container structure.

E.g., Key = Name, Value Contract/Object/Song => Name:Contract1 .

Search handles tags in combination / conjunction of keys and/or key values separated by |.

E.g., Name:\*|Country:Sweden|Country:Norway.

This would search for objects with key Name of any value and where key Country is either Sweden or Norway.

#### Parameters

|                    |                                                                                                                                                                                                                                  |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>tags</i>        | Is a string of key tags or key:value pairs that are separated by  .                                                                                                                                                              |
| <i>containerId</i> | Is the <a href="#">cbe::ContainerId</a> id of the rootContainer to start the search of <a href="#">objects</a> in.<br>E.g., if starting in the rootContainer, the whole account will be searched for matching tags, key/value's. |
| <i>delegate</i>    | Is the callback pointer to where the API returns from either cache or Server.                                                                                                                                                    |

#### 11.8.4.20 search() [2/6]

```
cbe::QueryResult cbe::CloudBackend::search (
 std::string tags,
 cbe::ContainerId containerId)
```

Synchronous [exception] **Synchronous** version of [search\(std::string, cbe::ContainerId, QueryDelegatePtr\)](#) , and **throws an exception**, [SearchException](#), in case of a failed call.

See [search\(QueryDelegatePtr\)](#)

#### Returns

Information about the search — if the call was successful.

See [cbe::cbe::QueryResult](#)

#### Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">SearchException</a> |  |
|---------------------------------|--|

#### 11.8.4.21 search() [3/6]

```
cbe::util::Optional<cbe::QueryResult> cbe::CloudBackend::search (
 std::string tags,
 cbe::ContainerId containerId,
 SearchError & error)
```

Synchronous [non-throwing] search **Synchronous** version of [search\(tags, containerId, QueryDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [search\(tags, containerId, QueryDelegatePtr\)](#)

#### Parameters

|            |              |                                                                                                                                                                                                                                |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SearchError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.8.4.22 search()** [4/6]

```
cbe::QueryResult cbe::CloudBackend::search (
 cbe::Filter filter,
 cbe::ContainerId containerId,
 QueryDelegatePtr delegate)
```

Select [Object](#)'s with specified key/values using filter.

Search the whole container with sub-containers related to Objects in the container hierarchy structure.

E.g., Key = Name, Value Contract/Object/Song => Name:Contract1 .

Search handles tags in combination / conjunction of keys and/or key values separated by |.

E.g., Name:\*|Country:Sweden|Country:Norway, this would search for objects with key Name of any value and where key Country is either Sweden or Norway.

See [Filter](#).

**Parameters**

|                    |                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>filter</i>      | is a <a href="#">cbe::Filter</a> on which you can set how you want data to be ordered when searching. Remember to set the queryString to be keys/tags or key:value pairs that are separated by  . |
| <i>containerId</i> | is the ContainerId of the <a href="#">Container</a> to start the search of Objects in. E.g., if starting in the rootContainer, the whole account will be searched for matching tags, key/value's. |
| <i>delegate</i>    | is the callback pointer to where the API returns from either cache or Server.                                                                                                                     |

**11.8.4.23 search()** [5/6]

```
cbe::QueryResult cbe::CloudBackend::search (
 cbe::Filter filter,
 cbe::ContainerId containerId)
```

Synchronous [exception] **Synchronous** version of [search\(cbe::Filter, cbe::ContainerId, QueryDelegatePtr\)](#) , and **throws an exception**, [SearchException](#), in case of a failed call.

See [search\(QueryDelegatePtr\)](#)

Pointer to [cbe::delegate::QueryDelegate](#) that is passed into asynchronous version of method [search\(\)](#) See [delegate::object::QueryDelegate::Exception](#)

**Returns**

Information about the search — if the call was successful.

See [cbe::cbe::QueryResult](#)

**Exceptions**

|                                 |  |
|---------------------------------|--|
| <a href="#">SearchException</a> |  |
|---------------------------------|--|

**11.8.4.24 search()** [6/6]

```
cbe::util::Optional<cbe::QueryResult> cbe::CloudBackend::search (
 cbe::Filter filter,
 cbe::ContainerId containerId,
 SearchError & error)
```

Synchronous [non-throwing] search **Synchronous** version of [search\(filter,containerId,QueryDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [search\(filter,containerId, QueryDelegatePtr\)](#)

Forms the type of the `error` return parameter for the synchronous version of method [search\(\)](#)

See [delegate::QueryDelegate::ErrorInfo](#)

#### Parameters

|     |       |                                                                                                                                                                                                                                |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SearchError</a> object passed in will be populated with the error information. |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.8.4.25 castContainer()

```
static cbe::Container cbe::CloudBackend::castContainer (
 cbe::Item item) [static]
```

Casts an item to a container.

If the provided [cbe::Item](#) "item" is not a [cbe::Container](#) "container", an empty container instance is returned

#### 11.8.4.26 castObject()

```
static cbe::Object cbe::CloudBackend::castObject (
 cbe::Item item) [static]
```

Casts an item to an object.

If the provided [cbe::Item](#) "item" is not a [cbe::Object](#) "object", an empty object instance is returned

#### 11.8.4.27 clearCache()

```
bool cbe::CloudBackend::clearCache ()
```

Clear the local cache.

Use if there is a local SDK memory issue.

#### Returns

true success

false failed

#### 11.8.4.28 account()

```
cbe::Account cbe::CloudBackend::account ()
```

Returns an account object with information on the user.

#### Returns

[cbe::Account](#)

#### 11.8.4.29 version()

```
std::string cbe::CloudBackend::version () const
```

Returns the version number of the SDK.

#### Returns

std::string

#### 11.8.4.30 groupManager()

`cbe::GroupManager cbe::CloudBackend::groupManager ( )`

Gets the group manager.

Returns

`cbe::GroupManager`

#### 11.8.4.31 shareManager()

`cbe::ShareManager cbe::CloudBackend::shareManager ( )`

Gets the share manager.

Returns

`cbe::ShareManager`

#### 11.8.4.32 publishManager()

`cbe::PublishManager cbe::CloudBackend::publishManager ( )`

Gets the publish manager.

Returns

`cbe::PublishManager`

#### 11.8.4.33 terminate()

`void cbe::CloudBackend::terminate ( )`

Terminates the CloudBackend service.

Shall also be called in connection with a failed log in.

#### 11.8.4.34 subscribeManager()

`cbe::SubscribeManager cbe::CloudBackend::subscribeManager ( )`

Gets the subscribe manager.

Returns

`cbe::SubscribeManager`

#### 11.8.4.35 operator bool()

`cbe::CloudBackend::operator bool ( ) const [explicit]`

Checks if the current instance is real.

An "unreal" instance implies typically a failed login event.

Relies on the Default constructor [CloudBackend\(cbe::DefaultCtor\)](#)

Returns

`true` : is real

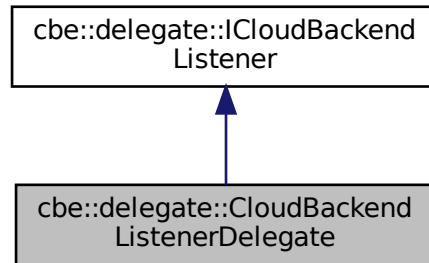
`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

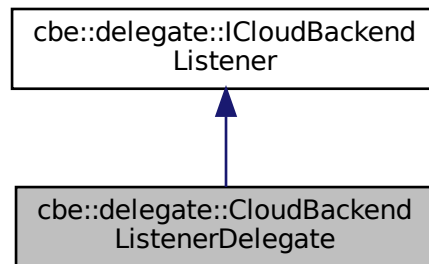
- include/cbe/CloudBackend.h

## 11.9 cbe::delegate::CloudBackendListenerDelegate Class Reference

Inheritance diagram for cbe::delegate::CloudBackendListenerDelegate:



Collaboration diagram for cbe::delegate::CloudBackendListenerDelegate:



### Public Member Functions

- virtual void [onRemoteObjectAdded](#) (cbe::Object &&object) override
- virtual void [onRemoteObjectMoved](#) (cbe::Object &&object) override
- virtual void [onRemoteObjectRemoved](#) (cbe::ItemId objectId, std::string name) override
- virtual void [onRemoteObjectRenamed](#) (cbe::Object &&object) override
- virtual void [onRemoteContainerAdded](#) (cbe::Container &&container) override
- virtual void [onRemoteContainerMoved](#) (cbe::Container &&container) override
- virtual void [onRemoteContainerRemoved](#) (cbe::ItemId containerId, std::string name) override
- virtual void [onRemoteContainerRenamed](#) (cbe::Container &&container) override

### 11.9.1 Member Function Documentation

#### 11.9.1.1 onRemoteObjectAdded()

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteObjectAdded (
 cbe::Object && object) [override], [virtual]
```

Called upon successful create of an [object](#).

#### Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>object</i> | Instance of object that is being created. |
|---------------|-------------------------------------------|

Implements [cbe::delegate::ICloudBackendListener](#).

#### 11.9.1.2 onRemoteObjectMoved()

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteObjectMoved (
 cbe::Object && object) [override], [virtual]
```

Called upon successful move of an [object](#).

#### Parameters

|               |                                         |
|---------------|-----------------------------------------|
| <i>object</i> | Instance of object that is being moved. |
|---------------|-----------------------------------------|

Implements [cbe::delegate::ICloudBackendListener](#).

#### 11.9.1.3 onRemoteObjectRemoved()

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteObjectRemoved (
 cbe::ItemId objectId,
 std::string name) [override], [virtual]
```

Called upon successful removal of an [object](#).

#### Parameters

|                       |                                               |
|-----------------------|-----------------------------------------------|
| <i>object↔<br/>Id</i> | Instance of the object that is being Removed. |
| <i>name</i>           | Name of the object that is being Removed.     |

Implements [cbe::delegate::ICloudBackendListener](#).

#### 11.9.1.4 onRemoteObjectRenamed()

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteObjectRenamed (
 cbe::Object && object) [override], [virtual]
```

Called upon successful renaming of an [object](#).

#### Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>object</i> | Instance of object that is being renamed. |
|---------------|-------------------------------------------|

Implements [cbe::delegate::ICloudBackendListener](#).

#### 11.9.1.5 onRemoteContainerAdded()

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteContainerAdded (
 cbe::Container && container) [override], [virtual]
```

Called upon successful Create.

## Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <i>container</i> | Instance of container that is being created. |
|------------------|----------------------------------------------|

Implements [cbe::delegate::ICloudBackendListener](#).

**11.9.1.6 onRemoteContainerMoved()**

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteContainerMoved (
 cbe::Container && container) [override], [virtual]
```

Called upon successful Move.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <i>container</i> | Instance of container that is being moved. |
|------------------|--------------------------------------------|

Implements [cbe::delegate::ICloudBackendListener](#).

**11.9.1.7 onRemoteContainerRemoved()**

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteContainerRemoved (
 cbe::ItemId containerId,
 std::string name) [override], [virtual]
```

Called upon successful Remove.

## Parameters

|                          |                                                  |
|--------------------------|--------------------------------------------------|
| <i>container↔<br/>Id</i> | Instance of the container that is being Removed. |
| <i>name</i>              | Name of the container that is being removed.     |

Implements [cbe::delegate::ICloudBackendListener](#).

**11.9.1.8 onRemoteContainerRenamed()**

```
virtual void cbe::delegate::CloudBackendListenerDelegate::onRemoteContainerRenamed (
 cbe::Container && container) [override], [virtual]
```

Called upon successful Rename.

## Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <i>container</i> | Instance of container that is being Renamed. |
|------------------|----------------------------------------------|

Implements [cbe::delegate::ICloudBackendListener](#).

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

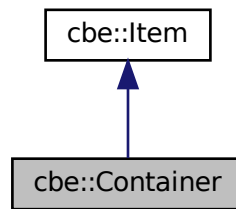
- include/cbe/delegate/CloudBackendListenerDelegate.h

**11.10 cbe::Container Class Reference**

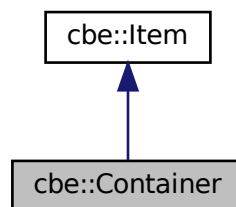
A collection of [Item](#), can also represent a table or folder.

```
#include <Container.h>
```

Inheritance diagram for cbe::Container:



Collaboration diagram for cbe::Container:



## Public Types

- using [CreateContainerDelegatePtr](#) = [delegate::CreateContainerDelegatePtr](#)
- using [CreateContainerException](#) = [delegate::CreateContainerDelegate::Exception](#)
- using [CreateContainerError](#) = [delegate::CreateContainerDelegate::ErrorInfo](#)
- using [MoveDelegatePtr](#) = [delegate::container::MoveDelegatePtr](#)
- using [MoveException](#) = [delegate::container::MoveDelegate::Exception](#)
- using [MoveError](#) = [delegate::container::MoveDelegate::ErrorInfo](#)
- using [RenameDelegatePtr](#) = [delegate::container::RenameDelegatePtr](#)
- using [RenameException](#) = [delegate::container::RenameDelegate::Exception](#)
- using [RenameError](#) = [delegate::container::RenameDelegate::ErrorInfo](#)
- using [RemoveDelegatePtr](#) = [delegate::container::RemoveDelegatePtr](#)
- using [RemoveException](#) = [delegate::container::RemoveDelegate::Exception](#)
- using [RemoveError](#) = [delegate::container::RemoveDelegate::ErrorInfo](#)
- using [CreateObjectDelegatePtr](#) = [delegate::CreateObjectDelegatePtr](#)
- using [CreateObjectException](#) = [delegate::CreateObjectDelegate::Exception](#)
- using [CreateObjectError](#) = [delegate::CreateObjectDelegate::ErrorInfo](#)
- using [UploadDelegatePtr](#) = [delegate::UploadDelegatePtr](#)
- using [UploadException](#) = [delegate::UploadDelegate::Exception](#)
- using [UploadError](#) = [delegate::UploadDelegate::ErrorInfo](#)
- using [QueryDelegatePtr](#) = [delegate::QueryDelegatePtr](#)
- using [QueryJoinDelegatePtr](#) = [std::shared\\_ptr< delegate::QueryJoinDelegate >](#)
- using [QueryException](#) = [delegate::QueryDelegate::Exception](#)

- using `QueryJoinError` = `delegate::QueryJoinDelegate::ErrorInfo`
- using `QueryError` = `delegate::QueryDelegate::ErrorInfo`
- using `queryWithPathException` = `delegate::QueryDelegate::Exception`
- using `queryWithPathError` = `delegate::QueryDelegate::ErrorInfo`
- using `SearchException` = `delegate::QueryDelegate::Exception`
- using `SearchError` = `delegate::QueryDelegate::ErrorInfo`
- using `SetAclDelegatePtr` = `delegate::AclDelegatePtr`
- using `SetAclException` = `delegate::AclDelegate::Exception`
- using `SetAclError` = `delegate::AclDelegate::ErrorInfo`
- using `GetAclDelegatePtr` = `delegate::AclDelegatePtr`
- using `GetAclException` = `delegate::AclDelegate::Exception`
- using `GetAclError` = `delegate::AclDelegate::ErrorInfo`
- using `ShareDelegatePtr` = `delegate::ShareDelegatePtr`
- using `ShareException` = `delegate::ShareDelegate::Exception`
- using `ShareError` = `delegate::ShareDelegate::ErrorInfo`
- using `UnShareDelegatePtr` = `std::shared_ptr< delegate::UnShareDelegate >`
- using `UnShareException` = `delegate::UnShareDelegate::Exception`
- using `UnShareError` = `delegate::UnShareDelegate::ErrorInfo`
- using `PublishDelegatePtr` = `delegate::PublishDelegatePtr`
- using `PublishException` = `delegate::PublishDelegate::Exception`
- using `PublishError` = `delegate::PublishDelegate::ErrorInfo`
- using `UnPublishDelegatePtr` = `delegate::UnPublishDelegatePtr`
- using `UnPublishException` = `delegate::UnPublishDelegate::Exception`
- using `UnPublishError` = `delegate::UnPublishDelegate::ErrorInfo`
- using `UnSubscribeDelegatePtr` = `delegate::UnSubscribeDelegatePtr`
- using `UnSubscribeException` = `delegate::UnSubscribeDelegate::Exception`
- using `UnSubscribeError` = `delegate::UnSubscribeDelegate::ErrorInfo`

## Public Member Functions

- `cbe::Container createContainer` (const std::string &name, `CreateContainerDelegatePtr` delegate)  
*Create a container.*
- `cbe::Container createContainer` (const std::string &name)  
*Synchronous [exception] createContainer.*
- `cbe::util::Optional< cbe::Container > createContainer` (const std::string &name, `CreateContainerError` &error)  
*Synchronous [non-throwing] createContainer.*
- void `move` (`cbe::ContainerId` dstId, `MoveDelegatePtr` delegate)  
*Move a container to a different parent.*
- `cbe::Container move` (`cbe::ContainerId` dstId)  
*Synchronous [exception] move.*
- `cbe::util::Optional< cbe::Container > move` (`cbe::ContainerId` dstId, `MoveError` &error)  
*Synchronous [non-throwing] move.*
- void `rename` (const std::string &name, `RenameDelegatePtr` delegate)  
*Change the name of the container.*
- `cbe::Container rename` (const std::string &name)  
*Synchronous [exception] rename.*
- `cbe::util::Optional< cbe::Container > rename` (const std::string &name, `RenameError` &error)  
*Synchronous [non-throwing] rename.*
- void `remove` (`RemoveDelegatePtr` delegate)  
*Delete the container.*
- `delegate::container::RemoveSuccess remove` ()  
*Synchronous [exception] remove/delete.*

- [cbe::util::Optional](#)< [delegate::container::RemoveSuccess](#) > [remove](#) ([RemoveError](#) &error)  
*Synchronous [non-throwing] remove/delete.*
- [cbe::Object](#) [createObject](#) (std::string [name](#), [cbe::KeyValues](#) keyValues, [CreateObjectDelegatePtr](#) delegate)  
*Create a new object.*
- [cbe::Object](#) [createObject](#) (std::string [name](#), [CreateObjectDelegatePtr](#) delegate)
- [cbe::Object](#) [createObject](#) (std::string [name](#), [cbe::KeyValues](#) keyValues)  
*Synchronous [exception] createObject.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [createObject](#) (std::string [name](#), [cbe::KeyValues](#) keyValues, [CreateObjectError](#) &error)  
*Synchronous [non-throwing] createObject.*
- [cbe::Object](#) [upload](#) (const std::string &filePath, [UploadDelegatePtr](#) delegate)  
*Upload object to container with file given by filePath.*
- [cbe::Object](#) [upload](#) (const std::string &name, const std::string &path, [UploadDelegatePtr](#) delegate)  
*Create an object in current container by uploading a file.*
- [cbe::Object](#) [upload](#) (const std::string &name, std::uint64\_t length, const char \*byteData, [UploadDelegatePtr](#) delegate)  
*Upload from local memory to an object.*
- [cbe::Object](#) [upload](#) (const std::string &filePath, [delegate::ProgressEventFn](#) &&progressEventFn)  
*Synchronous [exception] upload.*
- [cbe::Object](#) [upload](#) (const std::string &filePath)  
*Synchronous [exception] upload.*
- [cbe::Object](#) [upload](#) (const std::string &name, const std::string &path, [delegate::ProgressEventFn](#) &&progressEventFn)  
*Synchronous [exception] upload with progress.*
- [cbe::Object](#) [upload](#) (const std::string &name, const std::string &path)  
*Synchronous [exception] upload.*
- [cbe::Object](#) [upload](#) (const std::string &name, std::uint64\_t length, const char \*byteData, [delegate::ProgressEventFn](#) &&progressEventFn)  
*Synchronous [exception] upload from memory with progress.*
- [cbe::Object](#) [upload](#) (const std::string &name, std::uint64\_t length, const char \*byteData)  
*Synchronous [exception] upload from memory.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [upload](#) (const std::string &filePath, [delegate::ProgressEventFn](#) &&progressEventFn, [UploadError](#) &error)  
*Synchronous [non-throwing] upload with progress.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [upload](#) (const std::string &filePath, [UploadError](#) &error)  
*Synchronous [non-throwing] upload.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [upload](#) (const std::string &name, const std::string &path, [delegate::ProgressEventFn](#) &&progressEventFn, [UploadError](#) &error)  
*Synchronous [non-throwing] upload with progress.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [upload](#) (const std::string &name, const std::string &path, [UploadError](#) &error)  
*Synchronous [non-throwing] upload.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [upload](#) (const std::string &name, std::uint64\_t length, const char \*byteData, [delegate::ProgressEventFn](#) &&progressEventFn, [UploadError](#) &error)  
*Synchronous [non-throwing] upload from memory with progress.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [upload](#) (const std::string &name, std::uint64\_t length, const char \*byteData, [UploadError](#) &error)  
*Synchronous [non-throwing] upload from memory.*
- [cbe::QueryChain](#) [query](#) ([QueryDelegatePtr](#) queryDelegate)  
*Select list of objects.*
- [cbe::QueryChain](#) [query](#) ([Filter](#) filter, [QueryDelegatePtr](#) delegate)  
*Select list of objects using filter.*

- [cbe::QueryChainExt query](#) ([delegate::QueryJoinDelegatePtr](#) delegate)  
*Select list of objects, for join.*
- [cbe::QueryChainExt query](#) ([Filter](#) filter, [delegate::QueryJoinDelegatePtr](#) delegate)  
*Select list of objects using filter, for join.*
- [cbe::QueryChainSync query](#) ()  
*Synchronous [exception] Select list of objects.*
- [cbe::QueryChainSync query](#) ([Filter](#) filter)  
*Synchronous [exception] Select list of objects using filter.*
- [cbe::QueryChainSync query](#) ([QueryJoinError](#) &error)  
*Synchronous [non-throwing] Select list of objects, for join.*
- [cbe::QueryChainSync query](#) ([Filter](#) filter, [QueryJoinError](#) &error)  
*Synchronous [non-throwing] Select list of objects using filter, for join.*
- [cbe::QueryChain queryWithPath](#) (std::string relativePath, [QueryDelegatePtr](#) delegate)  
*Select list of objects in hierarchy.*
- [cbe::QueryResult queryWithPath](#) (std::string relativePath)  
*Synchronous [exception] queryWithPath.*
- [cbe::util::Optional< cbe::QueryResult > queryWithPath](#) (std::string relativePath, [queryWithPathError](#) &error)  
*Synchronous [non-throwing] queryWithPath.*
- [cbe::QueryResult search](#) (std::string tags, [QueryDelegatePtr](#) delegate)  
*Search by tags.*
- [cbe::QueryResult search](#) (std::string tags)  
*Synchronous [exception] search.*
- [cbe::util::Optional< cbe::QueryResult > search](#) (std::string tags, [SearchError](#) &error)  
*Synchronous [non-throwing] search.*
- [cbe::QueryResult search](#) ([cbe::Filter](#) filter, [QueryDelegatePtr](#) delegate)  
*Search using filter.*
- [cbe::QueryResult search](#) ([cbe::Filter](#))  
*Synchronous [exception] search using filter.*
- [cbe::util::Optional< cbe::QueryResult > search](#) ([cbe::Filter](#) filter, [SearchError](#) &error)  
*Synchronous [non-throwing] search.*
- void [setAcl](#) ([cbe::AclMap](#) aclMap, [SetAclDelegatePtr](#) delegate)  
*set ACL.*
- [cbe::AclMap setAcl](#) ([cbe::AclMap](#) aclMap)  
*Synchronous [exception] setAcl.*
- [cbe::util::Optional< cbe::AclMap > setAcl](#) ([cbe::AclMap](#) aclMap, [SetAclError](#) &error)  
*Synchronous [non-throwing] setAcl.*
- void [getAcl](#) ([GetAclDelegatePtr](#) delegate)  
*Retrieves its ACL map.*
- [cbe::AclMap getAcl](#) ()  
*Synchronous [exception] getAcl.*
- [cbe::util::Optional< cbe::AclMap > getAcl](#) ([GetAclError](#) &error)  
*Synchronous [non-throwing] getAcl.*
- void [share](#) ([cbe::UserId](#) toUserGroup, std::string description, [ShareDelegatePtr](#) delegate)  
*Make accessible by other user.*
- [delegate::ShareDelegate::Success share](#) ([cbe::UserId](#) toUserGroup, std::string description)  
*Synchronous [exception] share.*
- [cbe::util::Optional< delegate::ShareDelegate::Success > share](#) ([cbe::UserId](#) toUserGroup, std::string description, [ShareError](#) &error)  
*Synchronous [non-throwing] share.*
- void [unShare](#) ([cbe::ShareId](#) shareId, [UnShareDelegatePtr](#) delegate)  
*Revoke a previous share.*

- `delegate::UnShareDelegate::Success unShare (cbe::ShareId shareId)`  
*Synchronous [exception] unShare.*
- `cbe::util::Optional< delegate::UnShareDelegate::Success > unShare (cbe::ShareId shareId, UnShareError &error)`  
*Synchronous [non-throwing] unShare.*
- `void publish (cbe::PublishAccess security, cbe::PublishVisibility privacy, std::string description, std::string password, PublishDelegatePtr delegate)`  
*Publishes a container and its content to any user.*
- `delegate::PublishSuccess publish (cbe::PublishAccess security, cbe::PublishVisibility privacy, std::string description, std::string password)`  
*Synchronous [exception] publish.*
- `cbe::util::Optional< delegate::PublishSuccess > publish (cbe::PublishAccess security, cbe::PublishVisibility privacy, std::string description, std::string password, PublishError &error)`  
*Synchronous [non-throwing] publish.*
- `void unPublish (UnPublishDelegatePtr delegate)`  
*UnPublishes this container.*
- `delegate::UnPublishSuccess unPublish ()`  
*Synchronous [exception] unPublish.*
- `cbe::util::Optional< delegate::UnPublishSuccess > unPublish (UnPublishError &error)`  
*Synchronous [non-throwing] unPublish.*
- `void unSubscribe (UnSubscribeDelegatePtr delegate)`  
*UnSubscribes from this container.*
- `delegate::UnSubscribeSuccess unSubscribe ()`  
*Synchronous [exception] unSubscribe.*
- `cbe::util::Optional< delegate::UnSubscribeSuccess > unSubscribe (UnSubscribeError &error)`  
*Synchronous [non-throwing] unSubscribe.*
- **Container** (`cbe::DefaultCtor`)

## Friends

- class **Account**
- class **CloudBackend**
- class **Database**
- class **Filter**
- class **Group**
- class **QueryChain**

## Additional Inherited Members

### 11.10.1 Detailed Description

A collection of [Item](#), can also represent a table or folder.

### 11.10.2 Member Typedef Documentation

#### 11.10.2.1 CreateContainerDelegatePtr

using `cbe::Container::CreateContainerDelegatePtr = delegate::CreateContainerDelegatePtr`  
Pointer to `cbe::delegate::CreateContainerDelegate` that is passed into asynchronous version of method `create↔Container()`

### 11.10.2.2 CreateContainerException

using `cbe::Container::CreateContainerException` = `delegate::CreateContainerDelegate::Exception`  
See `delegate::container::CreateContainerDelegate::Exception`

### 11.10.2.3 CreateContainerError

using `cbe::Container::CreateContainerError` = `delegate::CreateContainerDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `createContainer()`  
See `delegate::CreateContainerDelegate::ErrorInfo`

### 11.10.2.4 MoveDelegatePtr

using `cbe::Container::MoveDelegatePtr` = `delegate::container::MoveDelegatePtr`  
Pointer to `cbe::delegate::MoveDelegate` that is passed into asynchronous version of method `move()`

### 11.10.2.5 MoveException

using `cbe::Container::MoveException` = `delegate::container::MoveDelegate::Exception`  
See `delegate::object::MoveDelegate::Exception`

### 11.10.2.6 MoveError

using `cbe::Container::MoveError` = `delegate::container::MoveDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `move()`  
See `delegate::container::MoveDelegate::ErrorInfo`

### 11.10.2.7 RenameDelegatePtr

using `cbe::Container::RenameDelegatePtr` = `delegate::container::RenameDelegatePtr`  
Pointer to `cbe::delegate::container::RenameDelegate` that is passed into asynchronous version of method `rename()`

### 11.10.2.8 RenameException

using `cbe::Container::RenameException` = `delegate::container::RenameDelegate::Exception`  
See `delegate::object::RenameDelegate::Exception`

### 11.10.2.9 RenameError

using `cbe::Container::RenameError` = `delegate::container::RenameDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `rename()`  
See `delegate::container::RenameDelegate::ErrorInfo`

### 11.10.2.10 RemoveDelegatePtr

using `cbe::Container::RemoveDelegatePtr` = `delegate::container::RemoveDelegatePtr`  
Pointer to `cbe::delegate::RemoveDelegate` that is passed into asynchronous version of method `remove()`

### 11.10.2.11 RemoveException

using `cbe::Container::RemoveException` = `delegate::container::RemoveDelegate::Exception`  
See `delegate::object::RemoveDelegate::Exception`

### 11.10.2.12 RemoveError

using `cbe::Container::RemoveError` = `delegate::container::RemoveDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `remove()`  
See `delegate::RemoveDelegate::ErrorInfo`

### 11.10.2.13 CreateObjectDelegatePtr

using `cbe::Container::CreateObjectDelegatePtr = delegate::CreateObjectDelegatePtr`

Pointer to `cbe::delegate::CreateObjectDelegate` that is passed into asynchronous version of method `createObject()`

### 11.10.2.14 CreateObjectException

using `cbe::Container::CreateObjectException = delegate::CreateObjectDelegate::Exception`

See `delegate::object::CreateObjectDelegate::Exception`

### 11.10.2.15 CreateObjectError

using `cbe::Container::CreateObjectError = delegate::CreateObjectDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `createObject()`

See `delegate::CreateObjectDelegate::ErrorInfo`

### 11.10.2.16 UploadDelegatePtr

using `cbe::Container::UploadDelegatePtr = delegate::UploadDelegatePtr`

Pointer to `cbe::delegate::UploadDelegate` that is passed into asynchronous version of methods:

- `upload(const std::string&, UploadDelegatePtr)`
- `upload(const std::string&, const std::string&, UploadDelegatePtr)`
- `upload(const std::string&, std::uint64_t, const char*, UploadDelegatePtr)`

### 11.10.2.17 UploadException

using `cbe::Container::UploadException = delegate::UploadDelegate::Exception`

The exception type thrown by the synchronous version of methods:

- `upload(const std::string&, delegate::ProgressEventFn&&)`
- `upload(const std::string&)`
- `upload(const std::string&, const std::string&, delegate::ProgressEventFn&&)`
- `upload(const std::string&, const std::string&)`
- `upload(const std::string&, std::uint64_t, const char*, delegate::ProgressEventFn&&)`
- `upload(const std::string&, std::uint64_t, const char*)`

See `delegate::UploadDelegate::Exception`

### 11.10.2.18 UploadError

using `cbe::Container::UploadError = delegate::UploadDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous versions of methods:

- `upload(const std::string&, delegate::ProgressEventFn&&, UploadError&)`
- `upload(const std::string&, UploadError&)`
- `upload(const std::string&, const std::string&, delegate::ProgressEventFn&&, UploadError&)`
- `upload(const std::string&, const std::string&, UploadError&)`
- `upload(const std::string&, std::uint64_t, const char*, delegate::ProgressEventFn&&, UploadError&)`
- `upload(const std::string&, std::uint64_t, const char*, UploadError&)`

See `delegate::UploadDelegate::ErrorInfo`

#### 11.10.2.19 QueryDelegatePtr

using `cbe::Container::QueryDelegatePtr` = `delegate::QueryDelegatePtr`

Pointer to `cbe::delegate::QueryDelegatePtr` that is passed into asynchronous version of method `query()`

#### 11.10.2.20 QueryJoinDelegatePtr

using `cbe::Container::QueryJoinDelegatePtr` = `std::shared_ptr<delegate::QueryJoinDelegate>`

Pointer to `cbe::delegate::QueryJoinDelegate` that is passed into asynchronous version of method `query()`

#### 11.10.2.21 QueryException

using `cbe::Container::QueryException` = `delegate::QueryDelegate::Exception`

See also

[delegate::QueryDelegate::Exception](#)

#### 11.10.2.22 QueryJoinError

using `cbe::Container::QueryJoinError` = `delegate::QueryJoinDelegate::ErrorInfo`

See also

[delegate::QueryJoinDelegate::ErrorInfo](#)

#### 11.10.2.23 QueryError

using `cbe::Container::QueryError` = `delegate::QueryDelegate::ErrorInfo`

See also

[delegate::QueryDelegate::ErrorInfo](#)

#### 11.10.2.24 queryWithPathException

using `cbe::Container::queryWithPathException` = `delegate::QueryDelegate::Exception`

See `delegate::object::queryWithPathDelegate::Exception`

#### 11.10.2.25 queryWithPathError

using `cbe::Container::queryWithPathError` = `delegate::QueryDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `queryWithPath()`

See `delegate::queryWithPathDelegate::ErrorInfo`

#### 11.10.2.26 SearchException

using `cbe::Container::SearchException` = `delegate::QueryDelegate::Exception`

See `delegate::container::QueryDelegate::Exception`

#### 11.10.2.27 SearchError

using `cbe::Container::SearchError` = `delegate::QueryDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `search()`

See [delegate::QueryDelegate::ErrorInfo](#)

#### 11.10.2.28 SetAclDelegatePtr

using `cbe::Container::SetAclDelegatePtr = delegate::AclDelegatePtr`

Pointer to `cbe::delegate::AclDelegate` that is passed into asynchronous version of method `setAcl()`

#### 11.10.2.29 SetAclException

using `cbe::Container::SetAclException = delegate::AclDelegate::Exception`

See `delegate::object::AclDelegate::Exception`

#### 11.10.2.30 SetAclError

using `cbe::Container::SetAclError = delegate::AclDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `setAcl()`

See `delegate::AclDelegate::ErrorInfo`

#### 11.10.2.31 GetAclDelegatePtr

using `cbe::Container::GetAclDelegatePtr = delegate::AclDelegatePtr`

Pointer to `cbe::delegate::AclDelegate` that is passed into asynchronous version of method `getAcl()`

#### 11.10.2.32 GetAclException

using `cbe::Container::GetAclException = delegate::AclDelegate::Exception`

See `delegate::container::AclDelegate::Exception`

#### 11.10.2.33 GetAclError

using `cbe::Container::GetAclError = delegate::AclDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `getAcl()`

See `delegate::AclDelegate::ErrorInfo`

#### 11.10.2.34 ShareDelegatePtr

using `cbe::Container::ShareDelegatePtr = delegate::ShareDelegatePtr`

Pointer to `cbe::delegate::ShareDelegate` that is passed into asynchronous version of method `share()`

#### 11.10.2.35 ShareException

using `cbe::Container::ShareException = delegate::ShareDelegate::Exception`

See `delegate::container::ShareDelegate::Exception`

#### 11.10.2.36 ShareError

using `cbe::Container::ShareError = delegate::ShareDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `share()`

See `delegate::ShareDelegate::ErrorInfo`

#### 11.10.2.37 UnShareDelegatePtr

using `cbe::Container::UnShareDelegatePtr = std::shared_ptr<delegate::UnShareDelegate>`

Pointer to `cbe::delegate::UnShareDelegate` that is passed into asynchronous version of method `unShare()`

#### 11.10.2.38 UnShareException

using `cbe::Container::UnShareException = delegate::UnShareDelegate::Exception`

See `delegate::object::UnShareDelegate::Exception`

### 11.10.2.39 UnShareError

```
using cbe::Container::UnShareError = delegate::UnShareDelegate::ErrorInfo
```

Forms the type of the `error` return parameter for the synchronous version of method `unShare()`

See [delegate::UnShareDelegate::ErrorInfo](#)

### 11.10.2.40 PublishDelegatePtr

```
using cbe::Container::PublishDelegatePtr = delegate::PublishDelegatePtr
```

Pointer to [cbe::delegate::PublishDelegate](#) that is passed into asynchronous version of method `publish()`

### 11.10.2.41 PublishException

```
using cbe::Container::PublishException = delegate::PublishDelegate::Exception
```

See `delegate::object::PublishDelegate::Exception`

### 11.10.2.42 PublishError

```
using cbe::Container::PublishError = delegate::PublishDelegate::ErrorInfo
```

Forms the type of the `error` return parameter for the synchronous version of method `publish()`

See [delegate::PublishDelegate::ErrorInfo](#)

### 11.10.2.43 UnPublishDelegatePtr

```
using cbe::Container::UnPublishDelegatePtr = delegate::UnPublishDelegatePtr
```

Pointer to [cbe::delegate::UnPublishDelegate](#) that is passed into asynchronous version of method `unPublish()`

### 11.10.2.44 UnPublishException

```
using cbe::Container::UnPublishException = delegate::UnPublishDelegate::Exception
```

Pointer to [cbe::delegate::UnPublishDelegate](#) that is passed into asynchronous version of method `unPublish()` See `delegate::object::UnPublishDelegate::Exception`

### 11.10.2.45 UnPublishError

```
using cbe::Container::UnPublishError = delegate::UnPublishDelegate::ErrorInfo
```

Forms the type of the `error` return parameter for the synchronous version of method `unPublish()`

See [delegate::UnPublishDelegate::ErrorInfo](#)

### 11.10.2.46 UnSubscribeDelegatePtr

```
using cbe::Container::UnSubscribeDelegatePtr = delegate::UnSubscribeDelegatePtr
```

Pointer to [cbe::delegate::UnSubscribeDelegate](#) that is passed into asynchronous version of method `unSubscribe()`

### 11.10.2.47 UnSubscribeException

```
using cbe::Container::UnSubscribeException = delegate::UnSubscribeDelegate::Exception
```

See `delegate::object::UnSubscribeDelegate::Exception`

### 11.10.2.48 UnSubscribeError

```
using cbe::Container::UnSubscribeError = delegate::UnSubscribeDelegate::ErrorInfo
```

Forms the type of the `error` return parameter for the synchronous version of method `unSubscribe()`

See [delegate::UnSubscribeDelegate::ErrorInfo](#)

## 11.10.3 Member Function Documentation

**11.10.3.1 createContainer()** [1/3]

```
cbe::Container cbe::Container::createContainer (
 const std::string & name,
 CreateContainerDelegatePtr delegate)
```

Create a container.

Creates a container inside this container to be used for adding objects.

**Parameters**

|                 |                                                                                                          |
|-----------------|----------------------------------------------------------------------------------------------------------|
| <i>name</i>     | Name of the container to be created.                                                                     |
| <i>delegate</i> | Pointer to a <a href="#">delegate::CreateContainerDelegate</a> instance that is implemented by the user. |

**11.10.3.2 createContainer()** [2/3]

```
cbe::Container cbe::Container::createContainer (
 const std::string & name)
```

Synchronous [exception] createContainer.

**Synchronous** version of [createContainer\(const std::string&, CreateContainerDelegatePtr\)](#), and **throws an exception**, [CreateContainerException](#), in case of a failed call.

See [createContainer\(CreateContainerDelegatePtr\)](#)

**Returns**

Information about the createContainer object — if the call was successful.

See [cbe::delegate::CreateContainerDelegate::Success](#)

**Exceptions**

|                                          |  |
|------------------------------------------|--|
| <a href="#">CreateContainerException</a> |  |
|------------------------------------------|--|

**11.10.3.3 createContainer()** [3/3]

```
cbe::util::Optional<cbe::Container> cbe::Container::createContainer (
 const std::string & name,
 CreateContainerError & error)
```

Synchronous [non-throwing] createContainer.

**Synchronous** version of [createContainer\(const std::string&, CreateContainerDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [createContainer\(const std::string& , CreateContainerDelegatePtr\)](#)

**Parameters**

|            |              |                                                                                                                                                                                                                                         |
|------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">CreateContainerError</a> object passed in will be populated with the error information. |
|------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.4 move()** [1/3]

```
void cbe::Container::move (
 cbe::ContainerId dstId,
 MoveDelegatePtr delegate)
```

Move a container to a different parent.

Used to move container with its content to user specified location e.g., other container or to root container.

**Parameters**

|                 |                                                                                                          |
|-----------------|----------------------------------------------------------------------------------------------------------|
| <i>dstId</i>    | id of the container to which it should be moved to.                                                      |
| <i>delegate</i> | Pointer to a <a href="#">delegate::container::MoveDelegate</a> instance that is implemented by the user. |

**11.10.3.5 move()** [2/3]

```
cbe::Container cbe::Container::move (
 cbe::ContainerId dstId)
```

Synchronous [exception] move.

**Synchronous** version of [move\(cbe::ContainerId, MoveDelegatePtr\)](#) , and **throws an exception**, [MoveException](#), in case of a failed call.

See [move\(MoveDelegatePtr\)](#)

**Returns**

Information about the moved object — if the call was successful.

See [cbe::delegate::container::MoveDelegate::Success](#)

**Exceptions**

|                               |  |
|-------------------------------|--|
| <a href="#">MoveException</a> |  |
|-------------------------------|--|

**11.10.3.6 move()** [3/3]

```
cbe::util::Optional<cbe::Container> cbe::Container::move (
 cbe::ContainerId dstId,
 MoveError & error)
```

Synchronous [non-throwing] move.

**Synchronous** version of [move\(dstId, MoveDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [move\(dstId, MoveDelegatePtr\)](#)

**Parameters**

|            |              |                                                                                                                                                                                                                              |
|------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">MoveError</a> object passed in will be populated with the error information. |
|------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.7 rename()** [1/3]

```
void cbe::Container::rename (
 const std::string & name,
 RenameDelegatePtr delegate)
```

Change the name of the container.

Rename the container.

**Parameters**

|                 |                                                                                                            |
|-----------------|------------------------------------------------------------------------------------------------------------|
| <i>name</i>     | New name of the container.                                                                                 |
| <i>delegate</i> | Pointer to a <a href="#">delegate::container::RenameDelegate</a> instance that is implemented by the user. |

**11.10.3.8 rename()** [2/3]

```
cbe::Container cbe::Container::rename (
 const std::string & name)
```

Synchronous [exception] rename.

**Synchronous** version of [rename\(const std::string&, RenameDelegatePtr\)](#) , and **throws an exception**, [RenameException](#), in case of a failed call.

See [rename\(RenameDelegatePtr\)](#)

**Returns**

Information about the renamed object — if the call was successful.

See [cbe::delegate::container::RenameDelegate::Success](#)

**Exceptions**

|                                 |  |
|---------------------------------|--|
| <a href="#">RenameException</a> |  |
|---------------------------------|--|

**11.10.3.9 rename()** [3/3]

```
cbe::util::Optional<cbe::Container> cbe::Container::rename (
 const std::string & name,
 RenameError & error)
```

Synchronous [non-throwing] rename.

**Synchronous** version of [rename\(name, RenameDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [rename\(name, RenameDelegatePtr\)](#)

**Parameters**

|            |              |                                                                                                                                                                                                                                |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">RenameError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.10 remove()** [1/3]

```
void cbe::Container::remove (
 RemoveDelegatePtr delegate)
```

Delete the container.

Removes/deletes the container and all its content.

**Parameters**

|                 |                                                                                                            |
|-----------------|------------------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::container::RemoveDelegate</a> instance that is implemented by the user. |
|-----------------|------------------------------------------------------------------------------------------------------------|

**11.10.3.11 remove()** [2/3]

```
delegate::container::RemoveSuccess cbe::Container::remove ()
```

Synchronous [exception] remove/delete.

**Synchronous** version of [remove\(RemoveDelegatePtr\)](#) , and **throws an exception**, [RemoveException](#), in case of a failed call.

See [remove\(RemoveDelegatePtr\)](#)

**Returns**

Information about the removed object — if the call was successful.

See [cbe::delegate::container::RemoveSuccess](#)

**Exceptions**

|                                 |  |
|---------------------------------|--|
| <a href="#">RemoveException</a> |  |
|---------------------------------|--|

**11.10.3.12 remove()** [3/3]

```
cbe::util::Optional<delegate::container::RemoveSuccess> cbe::Container::remove (
 RemoveError & error)
```

Synchronous [non-throwing] remove/delete.

**Synchronous** version of [remove\(RemoveDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [remove\(RemoveDelegatePtr\)](#)

**Parameters**

|            |              |                                                                                                                                                                                                                                |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">RemoveError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.13 createObject()** [1/4]

```
cbe::Object cbe::Container::createObject (
 std::string name,
```

```
cbe::KeyValues keyValues,
CreateObjectDelegatePtr delegate)
```

Create a new object.

Creates an object with indexed tags or indexed tags + non indexed tags a.k.a. metadata, key/value pairs.

#### Parameters

|                  |                                                                                                       |
|------------------|-------------------------------------------------------------------------------------------------------|
| <i>name</i>      | Name of the object.                                                                                   |
| <i>keyValues</i> | Optional map of key/value pairs (metadata).                                                           |
| <i>delegate</i>  | Pointer to a <a href="#">delegate::CreateObjectDelegate</a> instance that is implemented by the user. |

#### Note

If an old object with the same name already exists that will be removed.

The object name may not contain characters < & : /

Any key name must start with a letter or \_

The following key names are reserved and should not be used: category, content, id, link and date

Key names are case sensitive, hence variations with uppercase are permitted.

#### 11.10.3.14 createObject() [2/4]

```
cbe::Object cbe::Container::createObject (
 std::string name,
 CreateObjectDelegatePtr delegate)
```

Same as [createObject\(std::string, cbe::KeyValues, CreateObjectDelegatePtr\)](#), but without the `keyValues` parameter.

#### 11.10.3.15 createObject() [3/4]

```
cbe::Object cbe::Container::createObject (
 std::string name,
 cbe::KeyValues keyValues)
```

Synchronous [exception] createObject.

**Synchronous** version of [createObject\(std::string, cbe::KeyValues, CreateObjectDelegatePtr\)](#) , and **throws an exception**, [CreateObjectException](#), in case of a failed call.

See [createObject\(CreateObjectDelegatePtr\)](#)

#### Returns

Information about the created object — if the call was successful.

See `cbe::delegate::CreateContainerDelegate::Success`

#### Exceptions

|                                       |
|---------------------------------------|
| <a href="#">CreateObjectException</a> |
|---------------------------------------|

#### 11.10.3.16 createObject() [4/4]

```
cbe::util::Optional<cbe::Object> cbe::Container::createObject (
 std::string name,
 cbe::KeyValues keyValues,
 CreateObjectError & error)
```

Synchronous [non-throwing] createObject.

**Synchronous** version of [createObject\(name, keyValues, CreateObjectDelegatePtr\)](#) , and **throws no exception** on

error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `createObject(name, keyValues, CreateObjectDelegatePtr)`

#### Parameters

|                  |                    |                                                                                                                                                                                                                                      |
|------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">CreateObjectError</a> object passed in will be populated with the error information. |
|------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

### 11.10.3.17 upload() [1/15]

```
cbe::Object cbe::Container::upload (
 const std::string & filePath,
 UploadDelegatePtr delegate)
```

Upload object to container with file given by `filePath`.

See [upload\(const std::string&,const std::string& path,UploadDelegatePtr\)](#)

#### Parameters

|                       |                                                                                       |
|-----------------------|---------------------------------------------------------------------------------------|
| <code>filePath</code> | Fully qualified file name. I.e., the path, relative or absolute, including file name. |
|-----------------------|---------------------------------------------------------------------------------------|

#### Example

##### Async creation of a [cbe::Object](#) by using `Container::upload()`

```
#include "cbe/Object.h"
#include "cbe/Container.h"
#include "cbe/delegate/UploadDelegate.h"
#include <condition_variable>
#include <fstream> // std::ofstream
#include <mutex>
~~~
struct MyUploadDelegate : cbe::delegate::UploadDelegate {
    std::mutex          mutex{};
    std::condition_variable conditionVariable{};
    // Indicates operation completed - success or failure
    bool                called{};
    // Default construct the result object. If the method
    // onUploadSuccess() has not been called, this default constructed state
    // implies that the object is not valid
    cbe::Object object{ cbe::DefaultCtor{} };
    void onUploadSuccess(cbe::Object&& object) final {
        {
            std::lock_guard<std::mutex> lock{mutex};
            // Change state of object member to indicate success
            this->object = std::move(object);
            // Indicate operation completed, member object indicates success
            called = true;
        }
        conditionVariable.notify_one();
    }
    void onUploadError(cbe::delegate::TransferError&& transferError,
                      cbe::util::Context&& context) final {
        {
            std::lock_guard<std::mutex> lock{mutex};
            // Put object into the default constructed state to indicate no-success
            object = cbe::Object{ cbe::DefaultCtor{} };
            // Indicate operation completed, member object indicates failure
            called = true;
        }
        conditionVariable.notify_one();
    }
}
cbe::Object waitForRsp() {
    std::unique_lock<std::mutex> lock{mutex};
```

```

        conditionVariable.wait(lock, [this]{ return called; });
        // Reset called flag, so current delegate instance can be reused
        called = false;
        // If object is still in its default constructed state, this implies a
        // failed cbe::object::upload() operation
        return object;
    }
}; // struct MyUploadDelegate
~~~
constexpr const char* const uploadPath = "/tmp/upload/";
constexpr const char* const myObjectFileName = "myObject";
const std::string
 qualFileName =
 std::string{uploadPath} + myObjectFileName;
std::ofstream ofs{qualFileName};
ofs << "Line 11\n" << "Line 12\n" << "Line 13\n" << std::flush;
ofs.close();
// Access container previously created with cbe::Container::createContainer()
cbe::Container myContainer = ~~~;
std::shared_ptr<MyUploadDelegate> myUploadDelegate =
 std::make_shared<MyUploadDelegate>();
// Create an object named after file name in variable qualFileName
myContainer.upload(qualFileName, myUploadDelegate);
// Continuously, use the Object instance passed into the delegate.
cbe::Object object = myUploadDelegate->waitForRsp();
// Check if upload was successful
if (!object) {
 // Not, bail out
    ~~~
}
// }
~~~

```

Continues at: [Async retrieving the Stream attached to a cbe::Object with the Object::getStreams\(\)](#)"

### 11.10.3.18 upload() [2/15]

```

cbe::Object cbe::Container::upload (
 const std::string & name,
 const std::string & path,
 UploadDelegatePtr delegate)

```

Create an object in current container by uploading a file.

**Object** is named by name, residing at path.

The **object** being created is instantly returned with a temporary id. When the response is retrieved from the server, via callback method [onUploadSuccess\(\)](#) the object will be updated with its final unique id.

#### Parameters

|                 |                                                                                                                        |
|-----------------|------------------------------------------------------------------------------------------------------------------------|
| <i>name</i>     | Name of local file name. The object, that is created, will be given the same name.                                     |
| <i>path</i>     | Path to local folder where the file is located. The can be relative or absolute and <b>must end with a slash (/)</b> . |
| <i>delegate</i> | Pointer to a <a href="#">delegate::UploadDelegate</a> instance that is implemented by the user.                        |

#### Returns

The created **Object**, first with its temporary id, and after successful with is final id.

### 11.10.3.19 upload() [3/15]

```

cbe::Object cbe::Container::upload (
 const std::string & name,
 std::uint64_t length,
 const char * byteData,
 UploadDelegatePtr delegate)

```

Upload from local memory to an object.

## Parameters

|                 |                                                                                                 |
|-----------------|-------------------------------------------------------------------------------------------------|
| <i>name</i>     | name that the uploaded object will get                                                          |
| <i>length</i>   | size of file in Bytes                                                                           |
| <i>byteData</i> | (char pointer to an array containing the data).                                                 |
| <i>delegate</i> | Pointer to a <a href="#">delegate::UploadDelegate</a> instance that is implemented by the user. |

## 11.10.3.20 upload() [4/15]

```
cbe::Object cbe::Container::upload (
 const std::string & filePath,
 delegate::ProgressEventFn && progressEventFn)
```

Synchronous [exception] upload.

**Synchronous** version of [upload\(const std::string&, UploadDelegatePtr\)](#) , and **throws an exception**, [UploadException](#), in case of a failed call.

See [upload\(const std::string&, UploadDelegatePtr\)](#)

## Parameters

|                        |                                                                                                                                                                                                                                                                                                          |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>progressEventFn</i> | Callback function that is called for each memory chunk uploaded.<br>The callback function will be executed the calling thread's context.<br>Also see <a href="#">cbe::delegate::ProgressEventFn</a> .<br>For the other parameters, see <a href="#">upload(const std::string&amp;, UploadDelegatePtr)</a> |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

The uploaded and created object — if the call was successful.

## Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">UploadException</a> |  |
|---------------------------------|--|

## Example

**Sync** [exception] creation of a [cbe::Object](#) by using **Container::upload()**

```
#include "cbe/Object.h"
#include "cbe/Container.h"
#include <condition_variable>
#include <fstream> // std::ofstream
#include <mutex>
~~~
constexpr const char* const uploadPath = "/tmp/upload/";
constexpr const char* const myObjectFileName = "myObject";
const std::string      qualFileName =
    std::string(uploadPath) + myObjectFileName;
std::ofstream ofs{qualFileName};
ofs << "Line 11\n" << "Line 12\n" << "Line 13\n" << std::flush;
ofs.close();
// Access container previously created with cbe::Container::createContainer()
cbe::Container myContainer = ~~~;
// Create an object named after file name in variable qualFileName
try
{
    myContainer.upload(qualFileName);
}
catch (const cbe::Container::UploadException& e)
{
    std::cout << "Error!" << std::endl << e.what() << std::endl;
    ~~~
}
~~~
```

Continues at: [Sync exception example of retrieving the Stream attached to a cbe::Object with the Object::getStreams\(\)](#)

**11.10.3.21 upload()** [5/15]

```
cbe::Object cbe::Container::upload (
    const std::string & filePath )
```

Synchronous [exception] upload.

Same as [upload\(const std::string&,delegate::ProgressEventFn&&\)](#), but without the parameter, `progressEventFn`.

**11.10.3.22 upload()** [6/15]

```
cbe::Object cbe::Container::upload (
    const std::string & name,
    const std::string & path,
    delegate::ProgressEventFn && progressEventFn )
```

Synchronous [exception] upload with progress.

**Synchronous** version of [upload\(const std::string&,const std::string&,UploadDelegatePtr\)](#) , and **throws an exception**, [UploadException](#), in case of a failed call.

See [upload\(const std::string&,const std::string&,UploadDelegatePtr\)](#)

**Parameters**

|                        |                                                                                                                                                                                            |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>progressEventFn</i> | See <a href="#">upload(const std::string&amp;,delegate::ProgressEventFn&amp;&amp;)</a> .<br>For the other parameters, see <a href="#">upload(const std::string&amp;,UploadDelegatePtr)</a> |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

The uploaded and created object — if the call was successful.

**Exceptions**

|                                 |  |
|---------------------------------|--|
| <a href="#">UploadException</a> |  |
|---------------------------------|--|

**11.10.3.23 upload()** [7/15]

```
cbe::Object cbe::Container::upload (
    const std::string & name,
    const std::string & path )
```

Synchronous [exception] upload.

Same as [upload\(const std::string&, const std::string&, delegate::ProgressEventFn&&, UploadError&\)](#) , but without the parameter, `progressEventFn`.

**11.10.3.24 upload()** [8/15]

```
cbe::Object cbe::Container::upload (
    const std::string & name,
    std::uint64_t length,
    const char * byteData,
    delegate::ProgressEventFn && progressEventFn )
```

Synchronous [exception] upload from memory with progress.

**Synchronous** version of [upload\(const std::string&,std::uint64\\_t,const char\\*,UploadDelegatePtr\)](#) , and **throws an exception**, [UploadException](#), in case of a failed call.

See [upload\(const std::string&,std::uint64\\_t,const char\\*,UploadDelegatePtr\)](#)

## Parameters

|                        |                                                                                                                                                                                            |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>progressEventFn</i> | See <a href="#">upload(const std::string&amp;,delegate::ProgressEventFn&amp;&amp;)</a> .<br>For the other parameters, see <a href="#">upload(const std::string&amp;,UploadDelegatePtr)</a> |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

The uploaded and created object — if the call was successful.

## Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">UploadException</a> |  |
|---------------------------------|--|

## 11.10.3.25 upload() [9/15]

```
cbe::Object cbe::Container::upload (
    const std::string & name,
    std::uint64_t length,
    const char * byteData )
```

Synchronous [exception] upload from memory.

Same as [upload\(const std::string&,std::uint64\\_t,const char\\*,delegate::ProgressEventFn&&\)](#) , but without the parameter, *progressEventFn*.

## 11.10.3.26 upload() [10/15]

```
cbe::util::Optional<cbe::Object> cbe::Container::upload (
    const std::string & filePath,
    delegate::ProgressEventFn && progressEventFn,
    UploadError & error )
```

Synchronous [non-throwing] upload with progress.

Similar to synchronous method [upload\(const std::string&,delegate::ProgressEventFn&&\)](#) , but **throws no exception** on error, instead the out/return parameter *error* is used to provide the error information in connection with a failed call.

See [upload\(const std::string&,delegate::ProgressEventFn&&\)](#)

## Parameters

|     |              |                                                                                                                                                                                                                                                                                                                                                    |
|-----|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UploadError</a> object passed in will be populated with the error information.<br>For the other parameters, see <a href="#">upload(const std::string&amp;,delegate::ProgressEventFn&amp;&amp;)</a> |
|-----|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the *error* out/return parameter.

## Example

**Sync** [non-throwing] creation of a [cbe::Object](#) by using **Container::upload()**

```
#include "cbe/Object.h"
#include "cbe/Container.h"
#include <condition_variable>
#include <fstream>           // std::ofstream
#include <mutex>
~~~
constexpr const char* const uploadPath = "/tmp/upload/";
```

```
constexpr const char* const myObjectFileName = "myObject";
const std::string qualFileName =
 std::string(uploadPath) + myObjectFileName;
std::ofstream ofs(qualFileName);
ofs << "Line 11\n"
 << "Line 12\n"
 << "Line 13\n"
 << std::flush;
ofs.close();
// Access container previously created with cbe::Container::createContainer()
cbe::Container myContainer = ~~~;
cbe::Object myObject{cbe::DefaultCtor{}};
cbe::Container::UploadError uploadError;
// Create an object named after file name in variable qualFileName
myObject = *myContainer.upload(qualFileName, uploadError);
if (uploadError) {
 std::cout << "Error! " << uploadError << std::endl;
    ~~~
}
~~~
```

Continues at: [Sync \[non-throwing\] retrieving the Stream attached to a cbe::Object with the Object::getStreams\(\)](#)"

### 11.10.3.27 upload() [11/15]

```
cbe::util::Optional<cbe::Object> cbe::Container::upload (
 const std::string & filePath,
 UploadError & error)
```

Synchronous [non-throwing] upload.

Same as [upload\(const std::string&, delegate::ProgressEventFn&&, UploadError&\)](#) , but without the parameter, `progressEventFn`.

### 11.10.3.28 upload() [12/15]

```
cbe::util::Optional<cbe::Object> cbe::Container::upload (
 const std::string & name,
 const std::string & path,
 delegate::ProgressEventFn && progressEventFn,
 UploadError & error)
```

Synchronous [non-throwing] upload with progress.

Similar to synchronous method [upload\(const std::string&, const std::string&, delegate::ProgressEventFn&&\)](#) , but **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [upload\(const std::string&, const std::string&, delegate::ProgressEventFn&&\)](#)

#### Parameters

|     |       |                                                                                                                                                                                                                                                          |
|-----|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | See <a href="#">upload(const std::string&amp;, delegate::ProgressEventFn&amp;&amp;, UploadError&amp;)</a> .<br>For the other parameters, see <a href="#">upload(const std::string&amp;, const std::string&amp;, delegate::ProgressEventFn&amp;&amp;)</a> |
|-----|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

See [upload\(const std::string&, delegate::ProgressEventFn&&, UploadError&\)](#)

### 11.10.3.29 upload() [13/15]

```
cbe::util::Optional<cbe::Object> cbe::Container::upload (
 const std::string & name,
 const std::string & path,
 UploadError & error)
```

Synchronous [non-throwing] upload.

Same as `upload(const std::string&, const std::string&, delegate::ProgressEventFn&&, UploadError&)` , but without the parameter, `progressEventFn`.

### 11.10.3.30 `upload()` [14/15]

```
cbe::util::Optional<cbe::Object> cbe::Container::upload (
 const std::string & name,
 std::uint64_t length,
 const char * byteData,
 delegate::ProgressEventFn && progressEventFn,
 UploadError & error)
```

Synchronous [non-throwing] upload from memory with progress.

Similar to synchronous method `upload(const std::string&, std::uint64_t, const char*, delegate::ProgressEventFn&&)` , but **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `upload(const std::string&, std::uint64_t, const char*, delegate::ProgressEventFn&&)`

#### Parameters

|     |              |                                                                                                                          |
|-----|--------------|--------------------------------------------------------------------------------------------------------------------------|
| out | <i>error</i> | See <code>upload(const std::string&amp;, <a href="#">delegate::ProgressEventFn&amp;&amp;</a>, UploadError&amp;)</code> . |
|-----|--------------|--------------------------------------------------------------------------------------------------------------------------|

#### Returns

See `upload(const std::string&, delegate::ProgressEventFn&&, UploadError&)`

### 11.10.3.31 `upload()` [15/15]

```
cbe::util::Optional<cbe::Object> cbe::Container::upload (
 const std::string & name,
 std::uint64_t length,
 const char * byteData,
 UploadError & error)
```

Synchronous [non-throwing] upload from memory.

Same as `upload(const std::string&, std::uint64_t, const char*, delegate::ProgressEventFn&&, UploadError&)` , but without the parameter, `progressEventFn`.

### 11.10.3.32 `query()` [1/8]

```
cbe::QueryChain cbe::Container::query (
 QueryDelegatePtr queryDelegate)
```

Select list of objects.

In line with function `CloudBackend::query(ContainerId, QueryDelegatePtr)`, but with its `ContainerId` parameter excluded.

#### See also

[CloudBackend::query\(ContainerId, QueryDelegatePtr\)](#)

### 11.10.3.33 `query()` [2/8]

```
cbe::QueryChain cbe::Container::query (
 Filter filter,
 QueryDelegatePtr delegate)
```

Select list of objects using filter.

In line with function `CloudBackend::query(ContainerId, Filter, QueryDelegatePtr)`, but with its `ContainerId` parameter excluded.

See also

[CloudBackend::query\(ContainerId, Filter, QueryDelegatePtr\)](#)

#### 11.10.3.34 query() [3/8]

```
cbe::QueryChainExt cbe::Container::query (
 delegate::QueryJoinDelegatePtr delegate)
```

Select list of objects, for join.

In line with function [CloudBackend::query\(ContainerId, delegate::QueryJoinDelegatePtr\)](#), but with its ContainerId parameter excluded.

See also

[CloudBackend::query\(ContainerId, delegate::QueryJoinDelegatePtr\)](#)

#### 11.10.3.35 query() [4/8]

```
cbe::QueryChainExt cbe::Container::query (
 Filter filter,
 delegate::QueryJoinDelegatePtr delegate)
```

Select list of objects using filter, for join.

In line with function [CloudBackend::query\(ContainerId, Filter, delegate::QueryJoinDelegatePtr\)](#), but with its ContainerId parameter excluded.

See also

[CloudBackend::query\(ContainerId, Filter, delegate::QueryJoinDelegatePtr\)](#)

#### 11.10.3.36 query() [5/8]

```
cbe::QueryChainSync cbe::Container::query ()
```

Synchronous [exception] Select list of objects.

In line with function [CloudBackend::query\(ContainerId\)](#), but with its ContainerId parameter excluded.

See also

[CloudBackend::query\(ContainerId\)](#)

Exceptions

|                                |  |
|--------------------------------|--|
| <a href="#">QueryException</a> |  |
|--------------------------------|--|

#### 11.10.3.37 query() [6/8]

```
cbe::QueryChainSync cbe::Container::query (
 Filter filter)
```

Synchronous [exception] Select list of objects using filter.

In line with function [CloudBackend::query\(ContainerId, Filter\)](#), but with its ContainerId parameter excluded.

See also

[CloudBackend::query\(ContainerId, Filter\)](#)

## Exceptions

|                                |  |
|--------------------------------|--|
| <a href="#">QueryException</a> |  |
|--------------------------------|--|

**11.10.3.38 query()** [7/8]

```
cbe::QueryChainSync cbe::Container::query (
 QueryJoinError & error)
```

Synchronous [non-throwing] Select list of objects, for join.

In line with function [CloudBackend::query\(ContainerId, QueryJoinError&\)](#), but with its `ContainerId` parameter excluded.

See also

[CloudBackend::query\(ContainerId, QueryJoinError&\)](#)

**11.10.3.39 query()** [8/8]

```
cbe::QueryChainSync cbe::Container::query (
 Filter filter,
 QueryJoinError & error)
```

Synchronous [non-throwing] Select list of objects using filter, for join.

In line with function [CloudBackend::query\(ContainerId, Filter, QueryJoinError&\)](#), but with its `ContainerId` parameter excluded.

See also

[CloudBackend::query\(ContainerId, Filter, QueryJoinError&\)](#)

**11.10.3.40 queryWithPath()** [1/3]

```
cbe::QueryChain cbe::Container::queryWithPath (
 std::string relativePath,
 QueryDelegatePtr delegate)
```

Select list of objects in hierarchy.

Pointer to `cbe::delegate::queryWithPathDelegate` that is passed into asynchronous version of method `queryWithPath()`

Queries the container with a given relative path, returns container with objects.

E.g. `/Documents/Pictures` will return objects and subContainers for Pictures.

## Note

`..` or `.` path options are not available, top down Paths in the container tree are available.

## Parameters

|                     |                                                                                                |
|---------------------|------------------------------------------------------------------------------------------------|
| <i>relativePath</i> | container path, e.g. <code>/Documents/Pictures</code>                                          |
| <i>delegate</i>     | Pointer to a <a href="#">delegate::QueryDelegate</a> instance that is implemented by the user. |

**11.10.3.41 queryWithPath()** [2/3]

```
cbe::QueryResult cbe::Container::queryWithPath (
```

```
std::string relativePath)
```

Synchronous [exception] `queryWithPath`.

**Synchronous** version of `queryWithPath(relativePath, queryWithPathDelegatePtr)` , and **throws an exception**, [queryWithPathException](#), in case of a failed call.

See `queryWithPath(queryWithPathDelegatePtr)`

#### Returns

Information about the query — if the call was successful.

See `cbe::delegate::queryWithPathSuccess`

#### Exceptions

|                                        |  |
|----------------------------------------|--|
| <a href="#">queryWithPathException</a> |  |
|----------------------------------------|--|

### 11.10.3.42 queryWithPath() [3/3]

```
cbe::util::Optional<cbe::QueryResult> cbe::Container::queryWithPath (
 std::string relativePath,
 queryWithPathError & error)
```

Synchronous [non-throwing] `queryWithPath`.

**Synchronous** version of `queryWithPath(relativePath, queryWithPathDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `queryWithPath(relativePath, queryWithPathDelegatePtr)`

#### Parameters

|     |       |                                                                                                                                                                                                                                       |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">queryWithPathError</a> object passed in will be populated with the error information. |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

### 11.10.3.43 search() [1/6]

```
cbe::QueryResult cbe::Container::search (
 std::string tags,
 QueryDelegatePtr delegate)
```

Search by tags.

Pointer to [cbe::delegate::QueryDelegate](#) that is passed into asynchronous version of method [search\(\)](#)

Search the whole container for tags related to Objects in the container structure.

E.g. Key = Name, Value Contract/Object/Song => Name:Contract1.

Search handles tags in combination of conjunctions of keys and/or key values separated by |.

E.g. Name:\*|Country:Sweden|Country:Norway, this would search for objects with key Name of any value and where key Country is either Sweden or Norway.

#### Parameters

|          |                                                                                                |
|----------|------------------------------------------------------------------------------------------------|
| tags     | is a string of key tags or key:value pairs that are separated by  .                            |
| delegate | Pointer to a <a href="#">delegate::QueryDelegate</a> instance that is implemented by the user. |

**11.10.3.44 search()** [2/6]

```
cbe::QueryResult cbe::Container::search (
 std::string tags)
```

Synchronous [exception] search.

**Synchronous** version of [search\(std::string, QueryDelegatePtr\)](#) , and **throws an exception**, [SearchException](#), in case of a failed call.

See [search\(QueryDelegatePtr\)](#)

**Returns**

Information about the search — if the call was successful.

See [cbe::cbe::QueryResult](#)

**Exceptions**

|                                 |
|---------------------------------|
| <a href="#">SearchException</a> |
|---------------------------------|

**11.10.3.45 search()** [3/6]

```
cbe::util::Optional<cbe::QueryResult> cbe::Container::search (
 std::string tags,
 SearchError & error)
```

Synchronous [non-throwing] search.

**Synchronous** version of [search\(tags, QueryDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [search\(tags, QueryDelegatePtr\)](#)

**Parameters**

|     |       |                                                                                                                                                                                                                                |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SearchError</a> object passed in will be populated with the error information. |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.46 search()** [4/6]

```
cbe::QueryResult cbe::Container::search (
 cbe::Filter filter,
 QueryDelegatePtr delegate)
```

Search using filter.

Search the whole container with sub-containers related to Objects in the container hierarchy structure.

E.g. Key = Name, Value Contract/Object/Song => Name:Contract1.

Search handles tags in combination / conjunction of keys and/or key values separated by |.

E.g. Name:\*|Country:Sweden|Country:Norway, this would search for objects with key Name of any value and where key Country is either Sweden or Norway.

## Parameters

|                 |                                                                                                                                                                                                   |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>filter</i>   | is a <a href="#">cbe::Filter</a> on which you can set how you want data to be ordered when searching. Remember to set the queryString to be keys/tags or key:value pairs that are separated by  . |
| <i>delegate</i> | Pointer to a <a href="#">delegate::QueryDelegate</a> instance that is implemented by the user.                                                                                                    |

**11.10.3.47 search()** [5/6]

```
cbe::QueryResult cbe::Container::search (
 cbe::Filter)
```

Synchronous [exception] search using filter.

Pointer to cbe::delegate::SearchDelegate that is passed into asynchronous version of method search()

**Synchronous** version of search(cbe::Filter, SearchDelegatePtr) , and **throws an exception**, [SearchException](#), in case of a failed call.

See search(cbe::Filter, SearchDelegatePtr)

## Returns

Information about the searched object — if the call was successful.

See cbe::cbe::QueryResult

## Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">SearchException</a> |  |
|---------------------------------|--|

**11.10.3.48 search()** [6/6]

```
cbe::util::Optional<cbe::QueryResult> cbe::Container::search (
 cbe::Filter filter,
 SearchError & error)
```

Synchronous [non-throwing] search.

**Synchronous** version of search(filter, SearchDelegatePtr) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See search(filter, SearchDelegatePtr)

## Parameters

|            |              |                                                                                                                                                                                                                                |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SearchError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.49 setAcl()** [1/3]

```
void cbe::Container::setAcl (
 cbe::AclMap aclMap,
 SetAclDelegatePtr delegate)
```

set ACL.

Set the Access Control List `ACL` for the container. For containers `set` does set the whole container tree, so all its sub items as well. Remember this is `set` and not `update` so every time you set all ids that should be there should be added.

#### Parameters

|                       |                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------|
| <code>aclMap</code>   | The desired <a href="#">permissions</a> for current container.                               |
| <code>delegate</code> | Pointer to a <a href="#">delegate::AcIDelegate</a> instance that is implemented by the user. |

#### 11.10.3.50 `setAcl()` [2/3]

```
cbe::AcIMap cbe::Container::setAcl (
 cbe::AcIMap aclMap)
```

Synchronous [exception] `setAcl`.

**Synchronous** version of `setAcl(cbe::AcIMap, SetAclDelegatePtr)` , and **throws an exception**, [SetAclException](#), in case of a failed call.

See `setAcl(SetAclDelegatePtr)`

#### Returns

Information about the shared object — if the call was successful.

See `cbe::delegate::AcIDelegate::Success`

#### Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">SetAclException</a> |  |
|---------------------------------|--|

#### 11.10.3.51 `setAcl()` [3/3]

```
cbe::util::Optional<cbe::AcIMap> cbe::Container::setAcl (
 cbe::AcIMap aclMap,
 SetAclError & error)
```

Synchronous [non-throwing] `setAcl`.

**Synchronous** version of `setAcl(aclMap, SetAclDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `setAcl(aclMap, SetAclDelegatePtr)`

#### Parameters

|                  |                    |                                                                                                                                                                                                                                |
|------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SetAclError</a> object passed in will be populated with the error information. |
|------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.10.3.52 `getAcl()` [1/3]

```
void cbe::Container::getAcl (
 GetAclDelegatePtr delegate)
```

Retrieves its ACL map.  
get the Access Control List ACL of the [Container](#).

#### Parameters

|                 |                                                                                              |
|-----------------|----------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::AcIDelegate</a> instance that is implemented by the user. |
|-----------------|----------------------------------------------------------------------------------------------|

#### 11.10.3.53 getAcl() [2/3]

```
cbe::AcIMap cbe::Container::getAcl ()
```

Synchronous [exception] getAcl.

**Synchronous** version of [getAcl\(GetAcIDelegatePtr\)](#) , and **throws an exception**, [GetAcIException](#), in case of a failed call.

See [getAcl\(GetAcIDelegatePtr\)](#)

#### Returns

Information about the shared object — if the call was successful.  
See `cbe::delegate::AcIDelegate::Success`

#### Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">GetAcIException</a> |  |
|---------------------------------|--|

#### 11.10.3.54 getAcl() [3/3]

```
cbe::util::Optional<cbe::AcIMap> cbe::Container::getAcl (
 GetAcIError & error)
```

Synchronous [non-throwing] getAcl.

**Synchronous** version of [getAcl\(GetAcIDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [getAcl\(GetAcIDelegatePtr\)](#)

#### Parameters

|            |              |                                                                                                                                                                                                                                |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">GetAcIError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.10.3.55 share() [1/3]

```
void cbe::Container::share (
 cbe::UserId toUserGroup,
 std::string description,
 ShareDelegatePtr delegate)
```

Make accessible by other user.

Shares a container and its content to a user. This provides the user the ability to access what has been shared to them via the `listAvailableShares` command. To allow users to view and change shared information see [ACL](#) .

**Note**

At present Sharing the container gives the user read permissions for the container and all its sub-items, this might change in the future.

**Parameters**

|                    |                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------|
| <i>toUserGroup</i> | takes a user id or group id to share to.                                                       |
| <i>description</i> | names the specific share between you and the user/group.                                       |
| <i>delegate</i>    | Pointer to a <a href="#">delegate::ShareDelegate</a> instance that is implemented by the user. |

**11.10.3.56 share() [2/3]**

```
delegate::ShareDelegate::Success cbe::Container::share (
 cbe::UserId toUserGroup,
 std::string description)
```

Synchronous [exception] share.

**Synchronous** version of [share\(cbe::UserId, std::string, ShareDelegatePtr\)](#) , and **throws an exception**, [ShareException](#), in case of a failed call.

See [share\(ShareDelegatePtr\)](#)

**Returns**

Information about the shared container — if the call was successful.

See [cbe::delegate::ShareDelegate::Success](#)

**Exceptions**

|                                |  |
|--------------------------------|--|
| <a href="#">ShareException</a> |  |
|--------------------------------|--|

**11.10.3.57 share() [3/3]**

```
cbe::util::Optional<delegate::ShareDelegate::Success> cbe::Container::share (
 cbe::UserId toUserGroup,
 std::string description,
 ShareError & error)
```

Synchronous [non-throwing] share.

**Synchronous** version of [share\(toUserGroup, description, ShareDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [share\(toUserGroup, description, ShareDelegatePtr\)](#)

**Parameters**

|            |              |                                                                                                                                                                                                                               |
|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ShareError</a> object passed in will be populated with the error information. |
|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.58 unShare()** [1/3]

```
void cbe::Container::unShare (
 cbe::ShareId shareId,
 UnShareDelegatePtr delegate)
```

Revoke a previous share.

unShare the container to a specific shareId created when sharing. Each share is unique between user/group and the one sharing. This is represented with a unique share id.

**Parameters**

|                 |                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------|
| <i>shareId</i>  | is as mentioned the unique id for a share between the owner and other user/group.                |
| <i>delegate</i> | Pointer to a <a href="#">delegate::UnShareDelegate</a> instance that is implemented by the user. |

**11.10.3.59 unShare()** [2/3]

```
delegate::UnShareDelegate::Success cbe::Container::unShare (
 cbe::ShareId shareId)
```

Synchronous [exception] unShare.

**Synchronous** version of unShare(UnShareDelegatePtr) , and **throws an exception**, [UnShareException](#), in case of a failed call.

See unShare(UnShareDelegatePtr)

**Returns**

Information about the unshared object — if the call was successful.

See [cbe::delegate::UnShareSuccess](#)

**Exceptions**

|                                  |  |
|----------------------------------|--|
| <a href="#">UnShareException</a> |  |
|----------------------------------|--|

**11.10.3.60 unShare()** [3/3]

```
cbe::util::Optional<delegate::UnShareDelegate::Success> cbe::Container::unShare (
 cbe::ShareId shareId,
 UnShareError & error)
```

Synchronous [non-throwing] unShare.

**Synchronous** version of unShare(UnShareDelegatePtr) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See unShare(UnShareDelegatePtr)

**Parameters**

|            |              |                                                                                                                                                                                                                                 |
|------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UnShareError</a> object passed in will be populated with the error information. |
|------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.61 publish()** [1/3]

```
void cbe::Container::publish (
 cbe::PublishAccess security,
 cbe::PublishVisibility privacy,
 std::string description,
 std::string password,
 PublishDelegatePtr delegate)
```

Publishes a container and its content to any user.

**Asynchronous** version of this service function.

Can be revoked with [unPublish\(\)](#)

**Parameters**

|                    |                                                                                                  |
|--------------------|--------------------------------------------------------------------------------------------------|
| <i>security</i>    | A <a href="#">cbe::PublishAccess</a> enum                                                        |
| <i>privacy</i>     | A <a href="#">cbe::PublishVisibility</a> enum                                                    |
| <i>description</i> | Free text                                                                                        |
| <i>password</i>    | Password                                                                                         |
| <i>delegate</i>    | Pointer to a <a href="#">delegate::PublishDelegate</a> instance that is implemented by the user. |

**11.10.3.62 publish()** [2/3]

```
delegate::PublishSuccess cbe::Container::publish (
 cbe::PublishAccess security,
 cbe::PublishVisibility privacy,
 std::string description,
 std::string password)
```

Synchronous [exception] publish.

**Synchronous** version of [publish\(cbe::PublishAccess, cbe::PublishVisibility, std::string, std::string, PublishDelegatePtr\)](#) , and **throws an exception**, [PublishException](#), in case of a failed call.

See [publish\(PublishDelegatePtr\)](#)

**Returns**

Information about the published object — if the call was successful.

See [cbe::delegate::PublishSuccess](#)

**Exceptions**

|                                  |  |
|----------------------------------|--|
| <a href="#">PublishException</a> |  |
|----------------------------------|--|

**11.10.3.63 publish()** [3/3]

```
cbe::util::Optional<delegate::PublishSuccess> cbe::Container::publish (
 cbe::PublishAccess security,
 cbe::PublishVisibility privacy,
 std::string description,
 std::string password,
 PublishError & error)
```

Synchronous [non-throwing] publish.

**Synchronous** version of [publish\(security, privacy, description, password, PublishDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [publish\(security, privacy, description, password, PublishDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                 |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">PublishError</a> object passed in will be populated with the error information. |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.64 unPublish()** [1/3]

```
void cbe::Container::unPublish (
 UnPublishDelegatePtr delegate)
```

UnPublishes this container.

**Asynchronous** version of this service function.

Revokes previous [publish\(\)](#).

## Parameters

|                 |                                                                                  |
|-----------------|----------------------------------------------------------------------------------|
| <i>delegate</i> | Gets notified when the container has been unPublished (or if there was an error) |
|-----------------|----------------------------------------------------------------------------------|

**11.10.3.65 unPublish()** [2/3]

```
delegate::UnPublishSuccess cbe::Container::unPublish ()
```

Synchronous [exception] unPublish.

**Synchronous** version of [unPublish\(UnPublishDelegatePtr\)](#) , and **throws an exception**, [UnPublishException](#), in case of a failed call.

See [unPublish\(UnPublishDelegatePtr\)](#)

## Returns

Information about the unPublished object — if the call was successful.

See [cbe::delegate::UnPublishSuccess](#)

## Exceptions

|                                    |  |
|------------------------------------|--|
| <a href="#">UnPublishException</a> |  |
|------------------------------------|--|

**11.10.3.66 unPublish()** [3/3]

```
cbe::util::Optional<delegate::UnPublishSuccess> cbe::Container::unPublish (
 UnPublishError & error)
```

Synchronous [non-throwing] unPublish.

**Synchronous** version of [unPublish\(UnPublishDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [unPublish\(UnPublishDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                   |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UnPublishError</a> object passed in will be populated with the error information. |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.10.3.67 unSubscribe()** [1/3]

```
void cbe::Container::unSubscribe (
 UnSubscribeDelegatePtr delegate)
```

UnSubscribes from this container.

**Asynchronous** version of this service function.

Revokes the subscription previously established with [cbe::SubscribeManager::subscribe\(\)](#)

## Parameters

|          |                                                                                   |
|----------|-----------------------------------------------------------------------------------|
| delegate | Gets notified when the container has been unSubscribed (or if there was an error) |
|----------|-----------------------------------------------------------------------------------|

**11.10.3.68 unSubscribe()** [2/3]

```
delegate::UnSubscribeSuccess cbe::Container::unSubscribe ()
```

Synchronous [exception] unSubscribe.

**Synchronous** version of [unSubscribe\(UnSubscribeDelegatePtr\)](#), and **throws an exception**, [UnSubscribeException](#), in case of a failed call.

See [unSubscribe\(UnSubscribeDelegatePtr\)](#)

## Returns

Information about the unSubscribe object — if the call was successful.

See [cbe::delegate::UnSubscribeSuccess](#)

## Exceptions

|                                      |
|--------------------------------------|
| <a href="#">UnSubscribeException</a> |
|--------------------------------------|

**11.10.3.69 unSubscribe()** [3/3]

```
cbe::util::Optional<delegate::UnSubscribeSuccess> cbe::Container::unSubscribe (
 UnSubscribeError & error)
```

Synchronous [non-throwing] unSubscribe.

**Synchronous** version of [unSubscribe\(UnSubscribeDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [unSubscribe\(UnSubscribeDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                     |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UnSubscribeError</a> object passed in will be populated with the error information. |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

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

- include/cbe/Container.h

## 11.11 cbe::util::Context Struct Reference

### Public Types

- using **ReportFn** = std::function< void(std::ostream &)>

### Public Member Functions

- **Context** (ReportFn &&reportFn, const char fnName[])
- std::string **report** () const

### Public Attributes

- std::string **fnName** {}

### Friends

- std::ostream & **operator**<< (std::ostream &os, const [Context](#) &context)

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

- include/cbe/util/Context.h

## 11.12 cbe::delegate::CreateAccountDelegate Class Reference

```
#include <CreateAccountDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::CloudBackend::createAccount\(\)](#) if the request fails.*

### Public Types

- using **Success** = [UserId](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onCreateAccountSuccess](#) ([UserId](#) &&userId)=0
- virtual void [onCreateAccountError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.12.1 Detailed Description

Delegate class for the asynchronous version of method:

- CloudBackend::CreateAccount

### 11.12.2 Member Function Documentation

#### 11.12.2.1 onCreateAccountSuccess()

```
virtual void cbe::delegate::CreateAccountDelegate::onCreateAccountSuccess (
 UserId && userId) [pure virtual]
```

Called upon successful account creation.

#### 11.12.2.2 onCreateAccountError()

```
virtual void cbe::delegate::CreateAccountDelegate::onCreateAccountError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called upon failed log in.

#### Parameters

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| <i>error</i>   | <a href="#">Error</a> information passed from CloudBackend SDK.                 |
| <i>context</i> | Additional context information about the original service call that has failed. |

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

- include/cbe/delegate/CreateAccountDelegate.h

## 11.13 cbe::delegate::CreateContainerDelegate Class Reference

```
#include <CreateContainerDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::Container::createContainer\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::Container](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onCreateContainerSuccess](#) ([cbe::Container](#) &&container)=0
- virtual void [onCreateContainerError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.13.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Container::createContainer\(\)](#)

## 11.13.2 Member Function Documentation

### 11.13.2.1 onCreateContainerSuccess()

```
virtual void cbe::delegate::CreateContainerDelegate::onCreateContainerSuccess (
 cbe::Container && container) [pure virtual]
```

Called upon successful CreateContainer.

#### Parameters

|                  |                                                              |
|------------------|--------------------------------------------------------------|
| <i>container</i> | Instance of <a href="#">Container</a> that is being created. |
|------------------|--------------------------------------------------------------|

### 11.13.2.2 onCreateContainerError()

```
virtual void cbe::delegate::CreateContainerDelegate::onCreateContainerError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/CreateContainerDelegate.h

## 11.14 cbe::delegate::CreateGroupDelegate Class Reference

```
#include <CreateGroupDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Group::createGroup\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::Group](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onCreateGroupSuccess](#) ([cbe::Group](#) &&group)=0
- virtual void [onCreateGroupError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.14.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Group::createGroup](#)

### 11.14.2 Member Function Documentation

### 11.14.2.1 onCreateGroupSuccess()

```
virtual void cbe::delegate::CreateGroupDelegate::onCreateGroupSuccess (
 cbe::Group && group) [pure virtual]
```

Called upon successful create group.

#### Parameters

|              |                                            |
|--------------|--------------------------------------------|
| <i>group</i> | Instance of a <a href="#">cbe::Group</a> . |
|--------------|--------------------------------------------|

### 11.14.2.2 onCreateGroupError()

```
virtual void cbe::delegate::CreateGroupDelegate::onCreateGroupError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/CreateGroupDelegate.h

## 11.15 cbe::delegate::CreateObjectDelegate Class Reference

```
#include <CreateObjectDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Container::createObject\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::Object](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onCreateObjectSuccess](#) ([cbe::Object](#) &&object)=0
- virtual void [onCreateObjectError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.15.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Container::createObject](#)

### 11.15.2 Member Function Documentation

#### 11.15.2.1 onCreateObjectSuccess()

```
virtual void cbe::delegate::CreateObjectDelegate::onCreateObjectSuccess (
 cbe::Object && object) [pure virtual]
```

Called upon successful CreateObject.

## Parameters

|               |                                                           |
|---------------|-----------------------------------------------------------|
| <i>object</i> | Instance of <a href="#">Object</a> that is being created. |
|---------------|-----------------------------------------------------------|

**11.15.2.2 onCreateObjectError()**

```
virtual void cbe::delegate::CreateObjectDelegate::onCreateObjectError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/CreateObjectDelegate.h

**11.16 cbe::delegate::CreateRoleDelegate Class Reference**

```
#include <CreateRoleDelegate.h>
```

**Classes**

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by cbe::Role::createRole() if the request fails.*

**Public Types**

- using **Success** = [cbe::Role](#)
- using **Error** = [delegate::Error](#)

**Public Member Functions**

- virtual void [onCreateRoleSuccess](#) ([cbe::Role](#) &&role)=0
- virtual void [onCreateRoleError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

**11.16.1 Detailed Description**

Delegate class for the asynchronous version of method:

- [cbe::Role::createRole](#)

**11.16.2 Member Function Documentation****11.16.2.1 onCreateRoleSuccess()**

```
virtual void cbe::delegate::CreateRoleDelegate::onCreateRoleSuccess (
 cbe::Role && role) [pure virtual]
```

Called upon successful create [Role](#).

## Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <a href="#">Role</a> | the role id of the created role. |
|----------------------|----------------------------------|

### 11.16.2.2 onCreateRoleError()

```
virtual void cbe::delegate::CreateRoleDelegate::onCreateRoleError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/CreateRoleDelegate.h

## 11.17 cbe::Database Class Reference

A database.

```
#include <Database.h>
```

### Public Member Functions

- std::string **name** () const  
*The database name.*
- cbe::DatabaseId **databaseId** () const  
*The databaseId number.*
- cbe::ContainerId **containerId** () const  
*The containerId number.*
- cbe::UserId **ownerId** () const  
*The userId number of the owner.*
- cbe::Date **created** () const  
*Timestamp of when it was created.*
- bool **readLocked** () const  
*Checks if it has a read lock.*
- bool **writeLocked** () const  
*Checks if it has a write lock.*
- cbe::Container **rootContainer** () const  
*The top container of the database.*
- **Database** (cbe::DefaultCtor)
- **operator bool** () const

### 11.17.1 Detailed Description

A database.

#### Example

Get the tenant **Container** object

```
cbe::DataBases myDatabases = cloudBackend.account().databases();
cbe::Container tenantContainer{ cbe::DefaultCtor{} };
for (auto myEntry : myDatabases) {
 if (myEntry.first == "tenant") {
 tenantContainer = myEntry.second.rootContainer();
 }
}
return tenantContainer;
```

### 11.17.2 Member Function Documentation

### 11.17.2.1 name()

```
std::string cbe::Database::name () const
```

The database name.

Returns

std::string

### 11.17.2.2 databaseId()

```
cbe::DatabaseId cbe::Database::databaseId () const
```

The databaseId number.

Returns

cbe::DatabaseId

### 11.17.2.3 containerId()

```
cbe::ContainerId cbe::Database::containerId () const
```

The containerId number.

Refers to the top rootContainer of the database.

Returns

cbe::ContainerId

### 11.17.2.4 ownerId()

```
cbe::UserId cbe::Database::ownerId () const
```

The userId number of the owner.

Returns

cbe::UserId

### 11.17.2.5 created()

```
cbe::Date cbe::Database::created () const
```

Timestamp of when it was created.

Given in unix epoch number format.

Returns

cbe::Date

### 11.17.2.6 readLocked()

```
bool cbe::Database::readLocked () const
```

Checks if it has a read lock.

Returns

true

false

### 11.17.2.7 writeLocked()

```
bool cbe::Database::writeLocked () const
```

Checks if it has a write lock.

Returns

true  
false

### 11.17.2.8 rootContainer()

```
cbe::Container cbe::Database::rootContainer () const
```

The top container of the database.

The whole container object.

Returns

[cbe::Container](#)

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

- include/cbe/Database.h

## 11.18 cbe::delegate::DownloadBinaryDelegate Class Reference

```
#include <DownloadBinaryDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Object::download\(\)](#) if the request fails.*

### Public Types

- using **Success** = [DownloadBinarySuccess](#)
- using **Error** = [TransferError](#)

### Public Member Functions

- virtual void [onDownloadBinarySuccess](#) (cbe::Object &&object, std::unique\_ptr< char[]> data)=0
- virtual void [onDownloadBinaryError](#) (cbe::delegate::TransferError &&transferError, cbe::util::Context &&context)=0
- virtual void [onChunkReceived](#) (cbe::Object &&object, std::uint64\_t received, std::uint64\_t total)

### 11.18.1 Detailed Description

Delegate class for the asynchronous version of method:

- cbe::Object::download(DownloadBinaryDelegatePtr)

### 11.18.2 Member Function Documentation

### 11.18.2.1 onDownloadBinarySuccess()

```
virtual void cbe::delegate::DownloadBinaryDelegate::onDownloadBinarySuccess (
 cbe::Object && object,
 std::unique_ptr< char[]> data) [pure virtual]
```

Called upon successful Download.

#### Parameters

|               |                                                                                                                                                                                                                                                                        |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>object</i> | Instance of <a href="#">cbe::Object</a> that is being downloaded.                                                                                                                                                                                                      |
| <i>data</i>   | The downloaded data of the object.<br>The provided memory is an array of bytes ( <code>char</code> ) dynamically allocated by the SDK, and managed by the <code>std::unique_ptr</code> . So, no further precautions needed concerning the the disposal of this memory. |

### 11.18.2.2 onDownloadBinaryError()

```
virtual void cbe::delegate::DownloadBinaryDelegate::onDownloadBinaryError (
 cbe::delegate::TransferError && transferError,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

### 11.18.2.3 onChunkReceived()

```
virtual void cbe::delegate::DownloadBinaryDelegate::onChunkReceived (
 cbe::Object && object,
 std::uint64_t received,
 std::uint64_t total) [virtual]
```

Called when a chunk of data has been received.

#### Parameters

|                 |                                                          |
|-----------------|----------------------------------------------------------|
| <i>object</i>   | <a href="#">Object</a> associated with current download. |
| <i>received</i> | Size, in number of bytes, of the received chunk.         |
| <i>total</i>    | Total number of bytes received so far.                   |

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

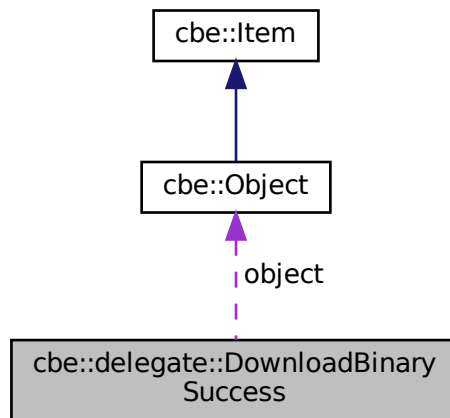
- `include/cbe/delegate/DownloadBinaryDelegate.h`

## 11.19 cbe::delegate::DownloadBinarySuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::DownloadBinaryDelegate::onDownloadBinarySuccess](#)

```
#include <DownloadBinarySuccess.h>
```

Collaboration diagram for cbe::delegate::DownloadBinarySuccess:



## Public Member Functions

- **DownloadBinarySuccess** ([cbe::DefaultCtor](#))
- **DownloadBinarySuccess** ([cbe::Object](#) &&object, std::unique\_ptr< char[]> data)
- **operator bool** () const  
*Checks if current instance is valid.*

## Public Attributes

- [cbe::Object](#) **object** {[cbe::DefaultCtor](#){}}
- std::unique\_ptr< char[]> **data** {}

### 11.19.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::DownloadBinaryDelegate::onDownloadBinarySuccess](#)

### 11.19.2 Member Function Documentation

#### 11.19.2.1 operator bool()

```
cbe::delegate::DownloadBinarySuccess::operator bool () const [explicit]
```

Checks if current instance is valid.

#### Returns

`true`: is valid

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

- include/cbe/delegate/DownloadBinarySuccess.h

## 11.20 cbe::delegate::DownloadDelegate Class Reference

```
#include <DownloadDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by:*

## Public Types

- using **Success** = [DownloadSuccess](#)
- using **Error** = [cbe::delegate::TransferError](#)

## Public Member Functions

- virtual void [onDownloadSuccess](#) ([cbe::Object](#) &&object, std::string path)=0
- virtual void [onDownloadError](#) ([cbe::delegate::TransferError](#) &&transferError, [cbe::util::Context](#) &&context)=0
- virtual void [onChunkReceived](#) ([cbe::Object](#) &&object, std::uint64\_t received, std::uint64\_t total)

### 11.20.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Object::download\(const std::string&, DownloadDelegatePtr\)](#)
- [cbe::Object::downloadStream\(const std::string&,cbe::Stream,DownloadDelegatePtr\)](#)

### 11.20.2 Member Function Documentation

#### 11.20.2.1 onDownloadSuccess()

```
virtual void cbe::delegate::DownloadDelegate::onDownloadSuccess (
 cbe::Object && object,
 std::string path) [pure virtual]
```

Called upon successful Download.

##### Parameters

|               |                                              |
|---------------|----------------------------------------------|
| <i>object</i> | Instance of object that is being Downloaded. |
| <i>path</i>   | Path of object to be downloaded.             |

#### 11.20.2.2 onDownloadError()

```
virtual void cbe::delegate::DownloadDelegate::onDownloadError (
 cbe::delegate::TransferError && transferError,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

#### 11.20.2.3 onChunkReceived()

```
virtual void cbe::delegate::DownloadDelegate::onChunkReceived (
 cbe::Object && object,
 std::uint64_t received,
 std::uint64_t total) [virtual]
```

Called when a chunk of data has been received.

## Parameters

|                 |                                                          |
|-----------------|----------------------------------------------------------|
| <i>object</i>   | <a href="#">Object</a> associated with current download. |
| <i>received</i> | Size, in number of bytes, of the received chunk.         |
| <i>total</i>    | Total number of bytes received so far.                   |

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

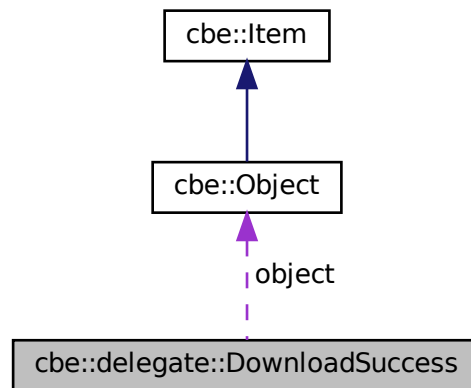
- include/cbe/delegate/DownloadDelegate.h

## 11.21 cbe::delegate::DownloadSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::DownloadDelegate::onDownloadSuccess](#).

```
#include <DownloadSuccess.h>
```

Collaboration diagram for cbe::delegate::DownloadSuccess:



### Public Member Functions

- **DownloadSuccess** ([cbe::DefaultCtor](#))
- **DownloadSuccess** ([cbe::Object](#) &&object, std::string path)
- **operator bool** () const  
*Checks if current instance is valid.*

### Public Attributes

- [cbe::Object](#) **object** {[cbe::DefaultCtor](#){}}
- std::string **path** {}

#### 11.21.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::DownloadDelegate::onDownloadSuccess](#).

#### 11.21.2 Member Function Documentation

### 11.21.2.1 operator bool()

`cbe::delegate::DownloadSuccess::operator bool ( ) const [explicit]`  
 Checks if current instance is valid.

#### Returns

`true`: is valid

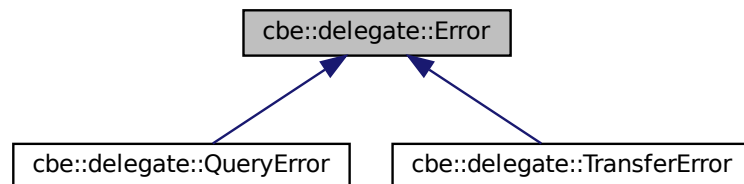
The documentation for this class was generated from the following file:

- `include/cbe/delegate/DownloadSuccess.h`

## 11.22 cbe::delegate::Error Class Reference

`#include <Error.h>`

Inheritance diagram for `cbe::delegate::Error`:



### Public Member Functions

- `operator bool ( ) const`

### Public Attributes

- `ErrorCode errorCode {}`
- `std::string reason {}`
- `std::string message {}`

### Friends

- `std::ostream & operator<< (std::ostream &os, const Error &error)`

### 11.22.1 Detailed Description

Entity class containing the error information

### 11.22.2 Member Function Documentation

#### 11.22.2.1 operator bool()

`cbe::delegate::Error::operator bool ( ) const [explicit]`  
 Checks whether current object holds error information.

## Returns

`true` implies that current object contains an error, while `false` indicates no error.

## 11.22.3 Friends And Related Function Documentation

### 11.22.3.1 operator<<

```
std::ostream& operator<< (
 std::ostream & os,
 const Error & error) [friend]
```

Output stream operator for CloudBackend::LoginDelegate::Error.

## 11.22.4 Member Data Documentation

### 11.22.4.1 errorCode

```
ErrorCode cbe::delegate::Error::errorCode {}
```

Mimics the general error code encoding in the www.

See also

[Wikipedia: List of HTTP status codes](#)

### 11.22.4.2 reason

```
std::string cbe::delegate::Error::reason {}
```

Human readable description of the reason for the failure.

### 11.22.4.3 message

```
std::string cbe::delegate::Error::message {}
```

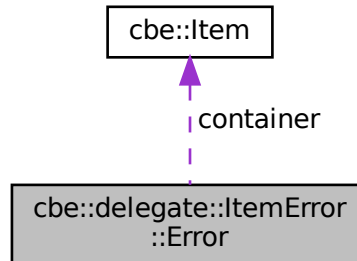
Human readable additional information about the error.

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

- include/cbe/delegate/Error.h

## 11.23 cbe::delegate::ItemError::Error Struct Reference

Collaboration diagram for cbe::delegate::ItemError::Error:



### Public Member Functions

- **Error** ([Item](#) container, [ItemType](#) type, [ErrorCode](#) errorCode, std::string &&reason, std::string &&message)

### Public Attributes

- [Item](#) **container** {[cbe::DefaultCtor](#){}}
- [ItemType](#) **type** {}
- [ErrorCode](#) **errorCode** {}
- std::string **reason** {}
- std::string **message** {}

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

- include/cbe/delegate/ItemDelegateErrors.h

## 11.24 cbe::delegate::JoinError::Error Class Reference

### Public Member Functions

- **Error** ([ErrorCode](#) errorCode, std::string &&reason, std::string &&message)

### Public Attributes

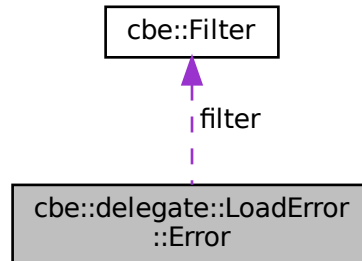
- [ErrorCode](#) **errorCode** {}
- std::string **reason** {}
- std::string **message** {}

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

- include/cbe/delegate/ItemDelegateErrors.h

## 11.25 cbe::delegate::LoadError::Error Class Reference

Collaboration diagram for cbe::delegate::LoadError::Error:



### Public Member Functions

- **Error** ([Filter](#) &&filter, [ErrorCode](#) errorCode, std::string &&reason, std::string &&message)

### Public Attributes

- [Filter](#) filter {}
- [ErrorCode](#) errorCode {}
- std::string reason {}
- std::string message {}

### 11.25.1 Member Data Documentation

#### 11.25.1.1 errorCode

```
ErrorCode cbe::delegate::LoadError::Error::errorCode {}
```

Mimics the general error code encoding in the www.

See also

[Wikipedia: List of HTTP status codes](#)

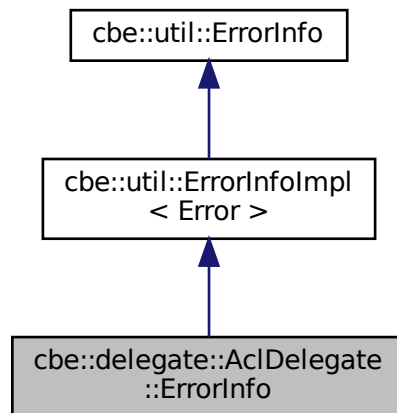
The documentation for this class was generated from the following file:

- include/cbe/delegate/ItemDelegateErrors.h

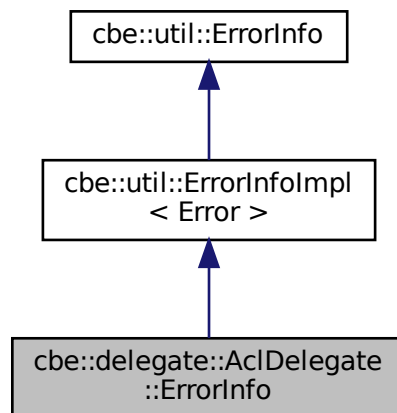
## 11.26 cbe::delegate::AclDelegate::ErrorInfo Struct Reference

```
#include <AclDelegate.h>
```

Inheritance diagram for cbe::delegate::AcIDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::AcIDelegate::ErrorInfo:



## Additional Inherited Members

### 11.26.1 Detailed Description

Contains all information about a failed GetACL.

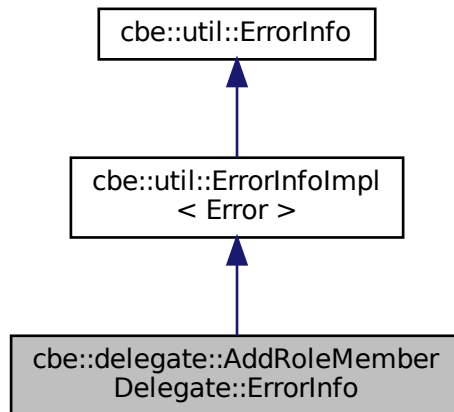
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/AcIDelegate.h`

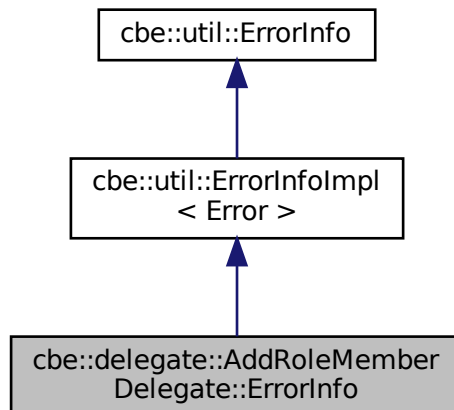
## 11.27 cbe::delegate::AddRoleMemberDelegate::ErrorInfo Struct Reference

```
#include <AddRoleMemberDelegate.h>
```

Inheritance diagram for cbe::delegate::AddRoleMemberDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::AddRoleMemberDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.27.1 Detailed Description

Contains all information about a failed AddMember.

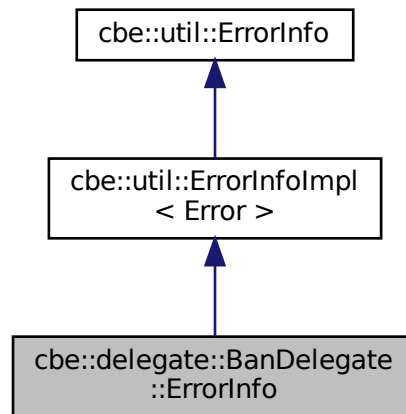
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/AddRoleMemberDelegate.h`

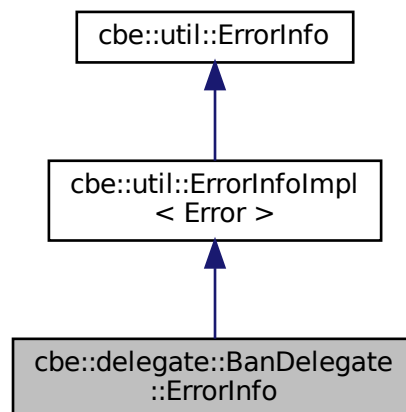
## 11.28 cbe::delegate::BanDelegate::ErrorInfo Struct Reference

```
#include <BanDelegate.h>
```

Inheritance diagram for cbe::delegate::BanDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::BanDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.28.1 Detailed Description

Contains all information about a failed Ban.

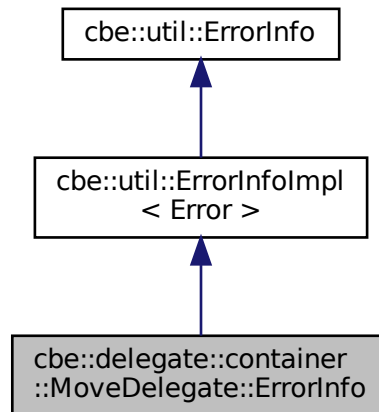
The documentation for this struct was generated from the following file:

- include/cbe/delegate/BanDelegate.h

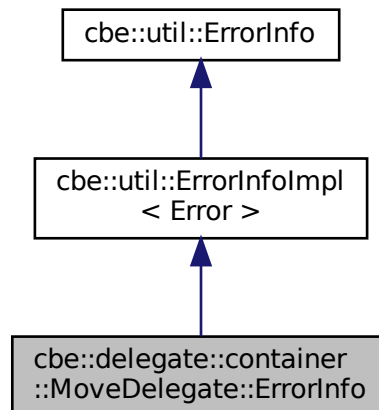
## 11.29 cbe::delegate::container::MoveDelegate::ErrorInfo Struct Reference

```
#include <MoveDelegate.h>
```

Inheritance diagram for cbe::delegate::container::MoveDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::container::MoveDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.29.1 Detailed Description

Contains all information about a failed Move.

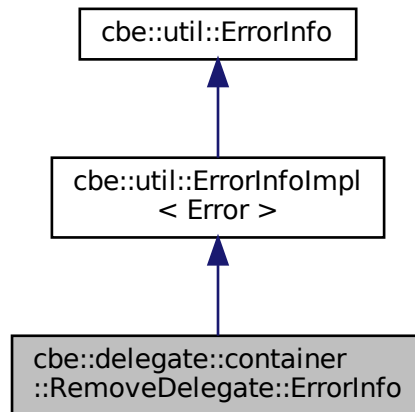
The documentation for this struct was generated from the following file:

- include/cbe/delegate/container/MoveDelegate.h

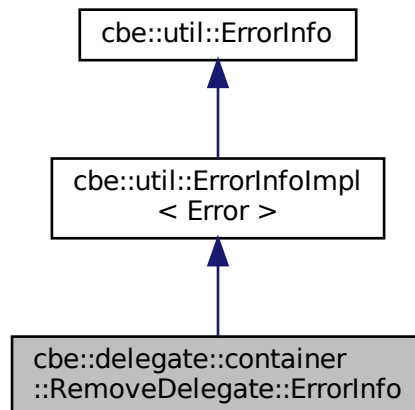
## 11.30 cbe::delegate::container::RemoveDelegate::ErrorInfo Struct Reference

```
#include <RemoveDelegate.h>
```

Inheritance diagram for cbe::delegate::container::RemoveDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::container::RemoveDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.30.1 Detailed Description

Contains all information about a failed Remove.

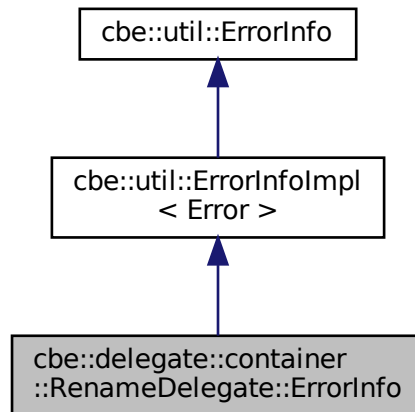
The documentation for this struct was generated from the following file:

- include/cbe/delegate/container/RemoveDelegate.h

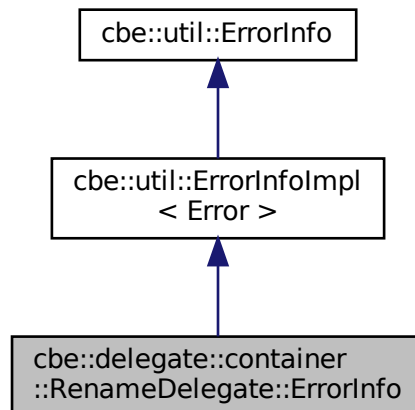
## 11.31 cbe::delegate::container::RenameDelegate::ErrorInfo Struct Reference

```
#include <RenameDelegate.h>
```

Inheritance diagram for cbe::delegate::container::RenameDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::container::RenameDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.31.1 Detailed Description

Contains all information about a failed Rename.

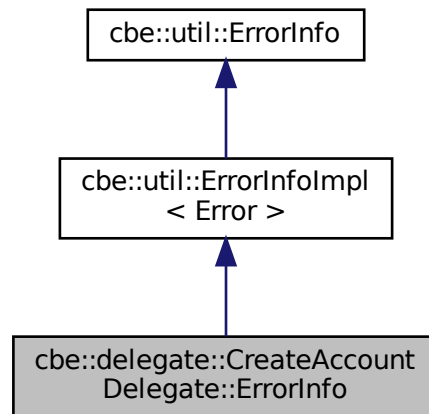
The documentation for this struct was generated from the following file:

- include/cbe/delegate/container/RenameDelegate.h

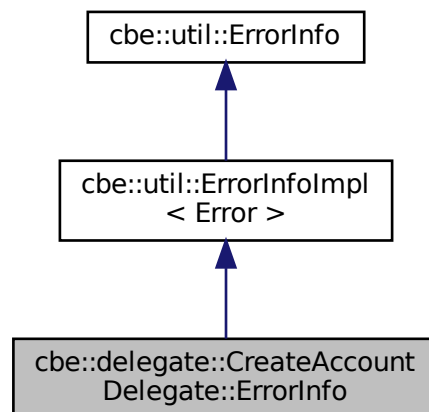
## 11.32 cbe::delegate::CreateAccountDelegate::ErrorInfo Struct Reference

```
#include <CreateAccountDelegate.h>
```

Inheritance diagram for cbe::delegate::CreateAccountDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::CreateAccountDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.32.1 Detailed Description

Contains all information about a failed CreateAccount.

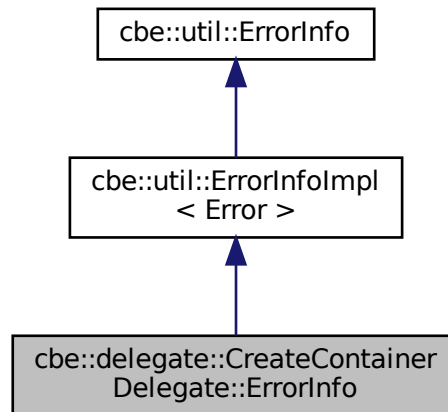
The documentation for this struct was generated from the following file:

- include/cbe/delegate/CreateAccountDelegate.h

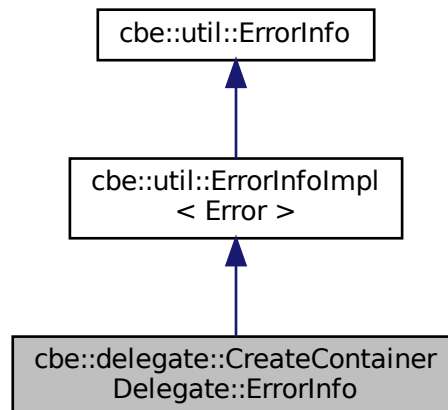
## 11.33 cbe::delegate::CreateContainerDelegate::ErrorInfo Struct Reference

```
#include <CreateContainerDelegate.h>
```

Inheritance diagram for cbe::delegate::CreateContainerDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::CreateContainerDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.33.1 Detailed Description

Contains all information about a failed CreateContainer.

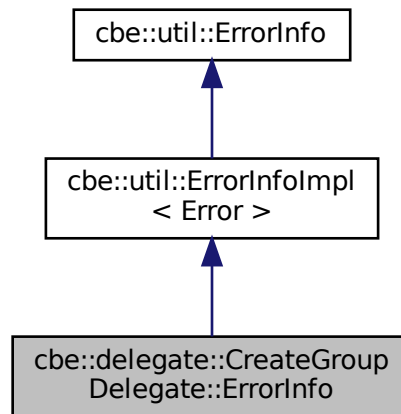
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/CreateContainerDelegate.h`

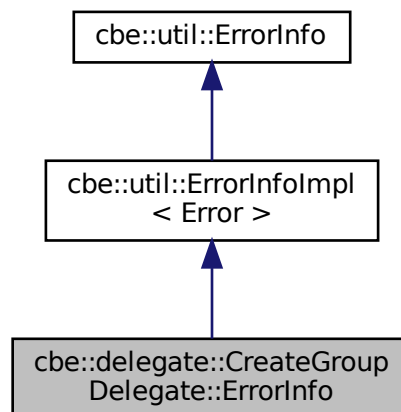
## 11.34 cbe::delegate::CreateGroupDelegate::ErrorInfo Struct Reference

```
#include <CreateGroupDelegate.h>
```

Inheritance diagram for cbe::delegate::CreateGroupDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::CreateGroupDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.34.1 Detailed Description

Contains all information about a failed CreateGroup.

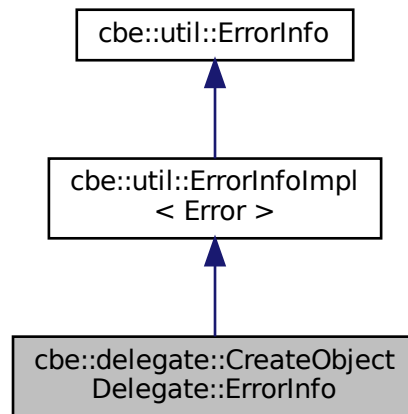
The documentation for this struct was generated from the following file:

- include/cbe/delegate/CreateGroupDelegate.h

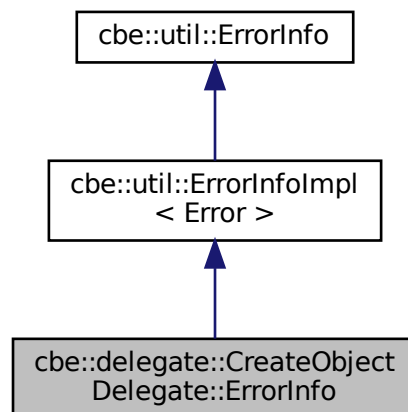
## 11.35 cbe::delegate::CreateObjectDelegate::ErrorInfo Struct Reference

```
#include <CreateObjectDelegate.h>
```

Inheritance diagram for cbe::delegate::CreateObjectDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::CreateObjectDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.35.1 Detailed Description

Contains all information about a failed CreateObject.

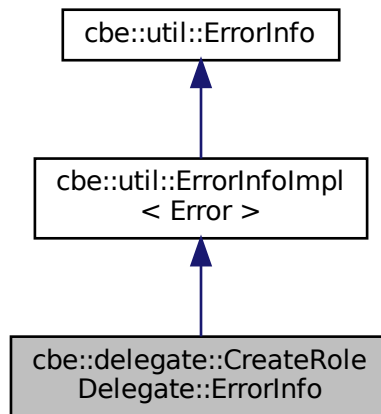
The documentation for this struct was generated from the following file:

- include/cbe/delegate/CreateObjectDelegate.h

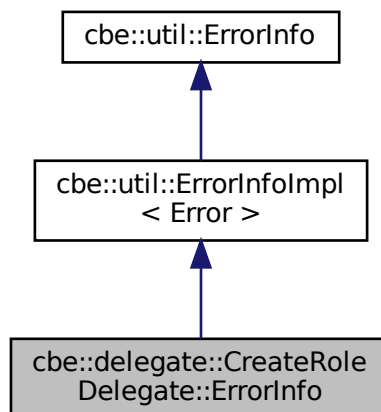
## 11.36 cbe::delegate::CreateRoleDelegate::ErrorInfo Struct Reference

```
#include <CreateRoleDelegate.h>
```

Inheritance diagram for cbe::delegate::CreateRoleDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::CreateRoleDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.36.1 Detailed Description

Contains all information about a failed `CreateRole`.

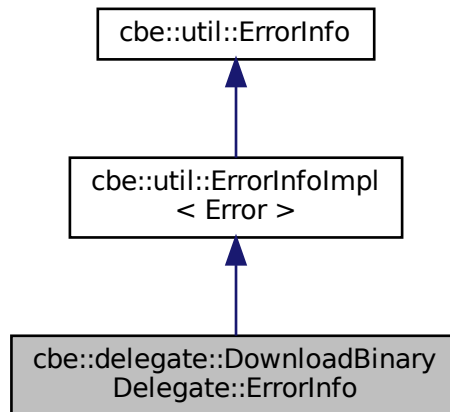
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/CreateRoleDelegate.h`

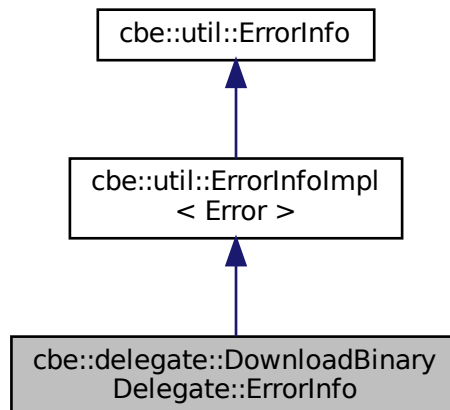
## 11.37 cbe::delegate::DownloadBinaryDelegate::ErrorInfo Struct Reference

```
#include <DownloadBinaryDelegate.h>
```

Inheritance diagram for cbe::delegate::DownloadBinaryDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::DownloadBinaryDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.37.1 Detailed Description

Contains all information about a failed DownloadBinary.

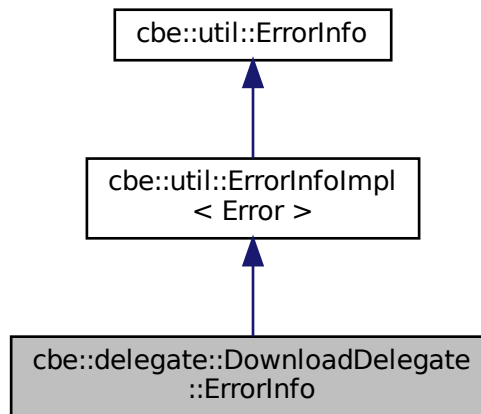
The documentation for this struct was generated from the following file:

- include/cbe/delegate/DownloadBinaryDelegate.h

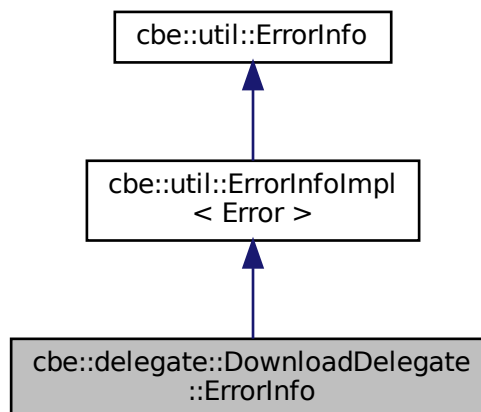
## 11.38 cbe::delegate::DownloadDelegate::ErrorInfo Struct Reference

```
#include <DownloadDelegate.h>
```

Inheritance diagram for cbe::delegate::DownloadDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::DownloadDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.38.1 Detailed Description

Contains all information about a failed download.

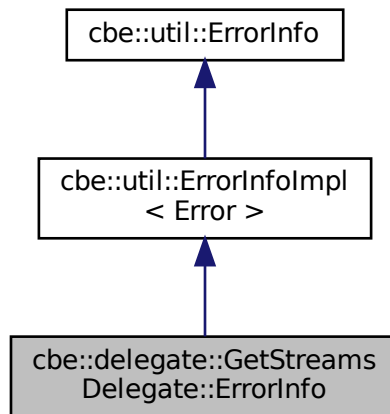
The documentation for this struct was generated from the following file:

- include/cbe/delegate/DownloadDelegate.h

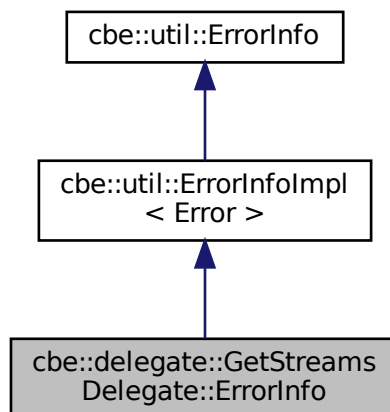
## 11.39 cbe::delegate::GetStreamsDelegate::ErrorInfo Struct Reference

```
#include <GetStreamsDelegate.h>
```

Inheritance diagram for cbe::delegate::GetStreamsDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::GetStreamsDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.39.1 Detailed Description

Contains all information about a failed GetStreams.

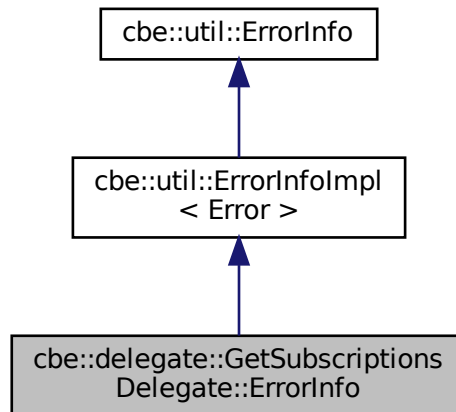
The documentation for this struct was generated from the following file:

- include/cbe/delegate/GetStreamsDelegate.h

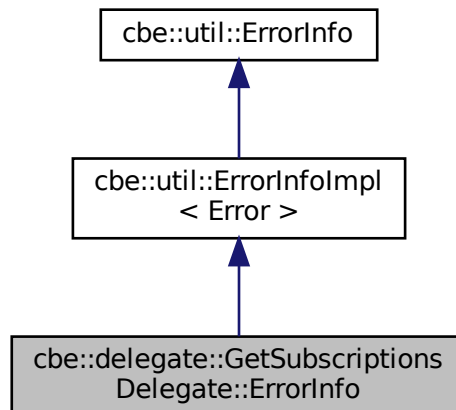
## 11.40 cbe::delegate::GetSubscriptionsDelegate::ErrorInfo Struct Reference

```
#include <GetSubscriptionsDelegate.h>
```

Inheritance diagram for cbe::delegate::GetSubscriptionsDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::GetSubscriptionsDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.40.1 Detailed Description

Contains all information about a failed `GetSubscriptions`.

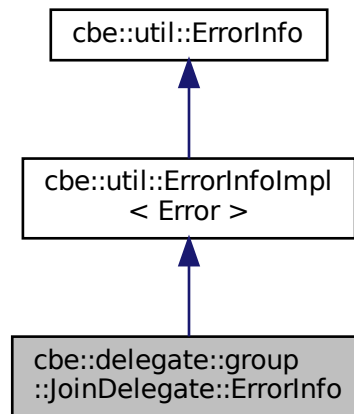
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/GetSubscriptionsDelegate.h`

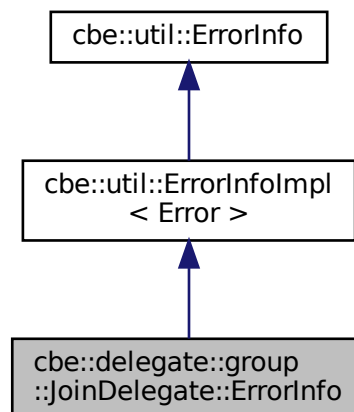
## 11.41 cbe::delegate::group::JoinDelegate::ErrorInfo Struct Reference

```
#include <JoinDelegate.h>
```

Inheritance diagram for cbe::delegate::group::JoinDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::group::JoinDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.41.1 Detailed Description

Contains all information about a failed Join.

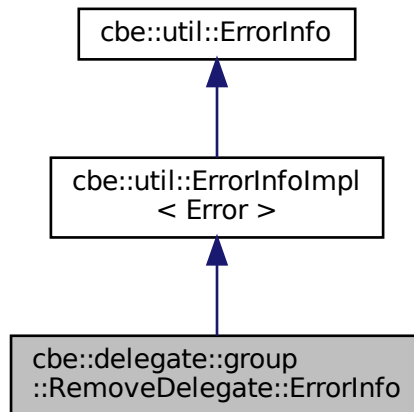
The documentation for this struct was generated from the following file:

- include/cbe/delegate/group/JoinDelegate.h

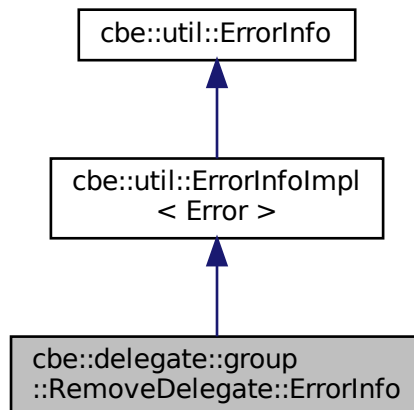
## 11.42 cbe::delegate::group::RemoveDelegate::ErrorInfo Struct Reference

```
#include <RemoveDelegate.h>
```

Inheritance diagram for cbe::delegate::group::RemoveDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::group::RemoveDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.42.1 Detailed Description

Contains all information about a failed Remove.

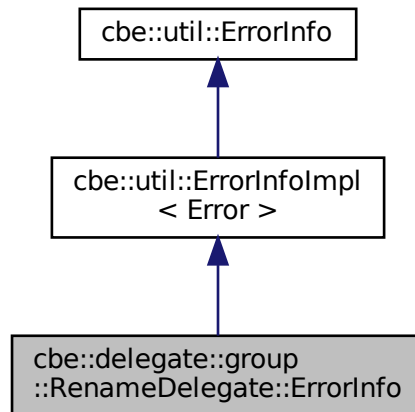
The documentation for this struct was generated from the following file:

- include/cbe/delegate/group/RemoveDelegate.h

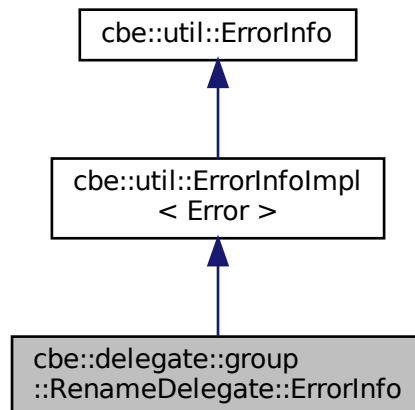
## 11.43 cbe::delegate::group::RenameDelegate::ErrorInfo Struct Reference

```
#include <RenameDelegate.h>
```

Inheritance diagram for cbe::delegate::group::RenameDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::group::RenameDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.43.1 Detailed Description

Contains all information about a failed Rename.

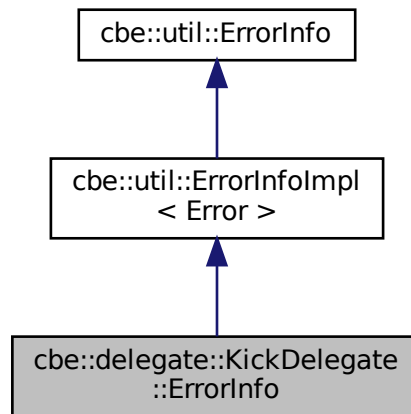
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/group/RenameDelegate.h`

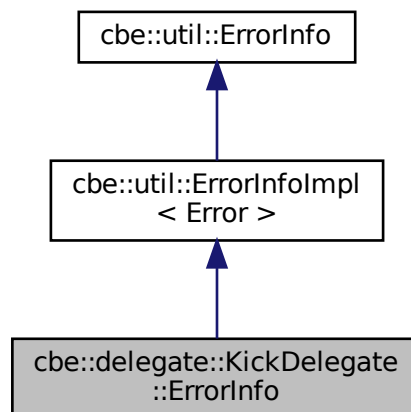
## 11.44 cbe::delegate::KickDelegate::ErrorInfo Struct Reference

```
#include <KickDelegate.h>
```

Inheritance diagram for cbe::delegate::KickDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::KickDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.44.1 Detailed Description

Contains all information about a failed Kick.

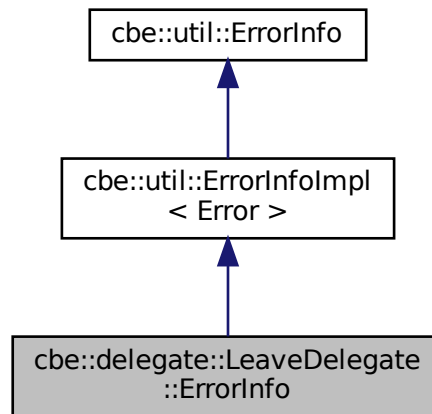
The documentation for this struct was generated from the following file:

- include/cbe/delegate/KickDelegate.h

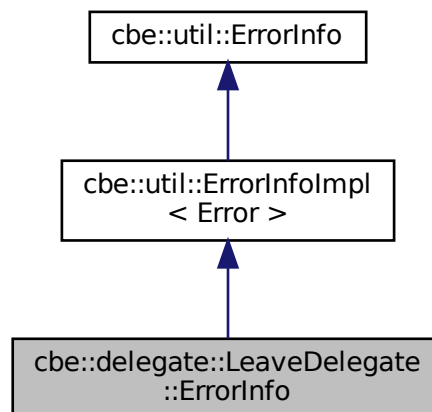
## 11.45 cbe::delegate::LeaveDelegate::ErrorInfo Struct Reference

```
#include <LeaveDelegate.h>
```

Inheritance diagram for cbe::delegate::LeaveDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::LeaveDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.45.1 Detailed Description

Contains all information about a failed Leave.

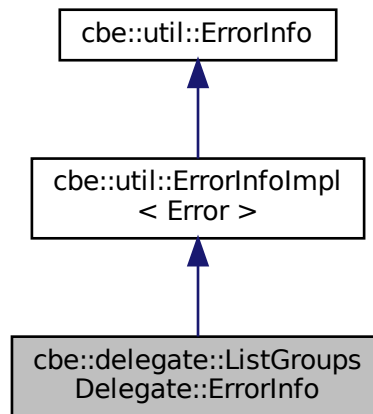
The documentation for this struct was generated from the following file:

- include/cbe/delegate/LeaveDelegate.h

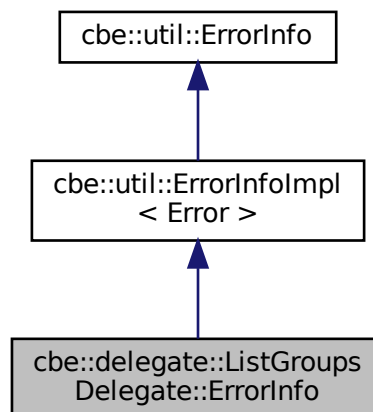
## 11.46 cbe::delegate::ListGroupsDelegate::ErrorInfo Struct Reference

```
#include <ListGroupsDelegate.h>
```

Inheritance diagram for cbe::delegate::ListGroupsDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::ListGroupsDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.46.1 Detailed Description

Contains all information about a failed ListGroup.

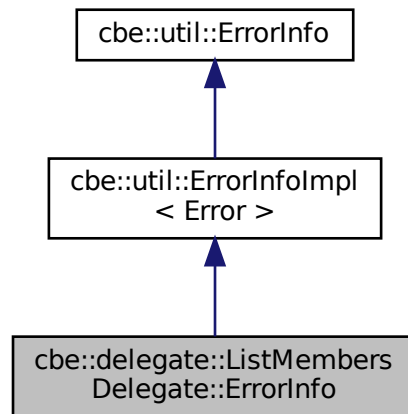
The documentation for this struct was generated from the following file:

- include/cbe/delegate/ListGroupsDelegate.h

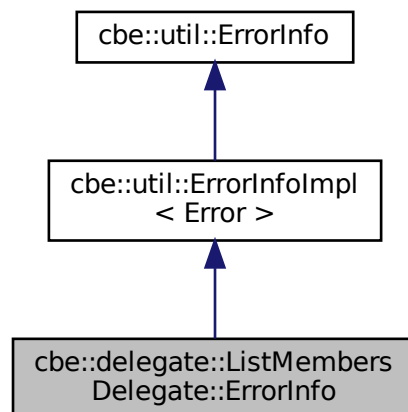
## 11.47 cbe::delegate::ListMembersDelegate::ErrorInfo Struct Reference

```
#include <ListMembersDelegate.h>
```

Inheritance diagram for cbe::delegate::ListMembersDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::ListMembersDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.47.1 Detailed Description

Contains all information about a failed ListMembers.

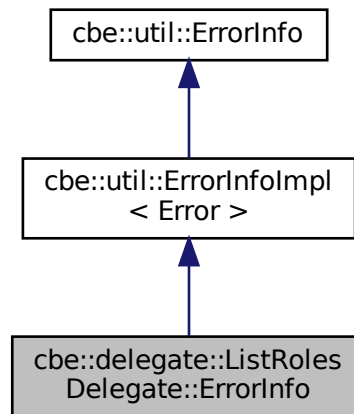
The documentation for this struct was generated from the following file:

- include/cbe/delegate/ListMembersDelegate.h

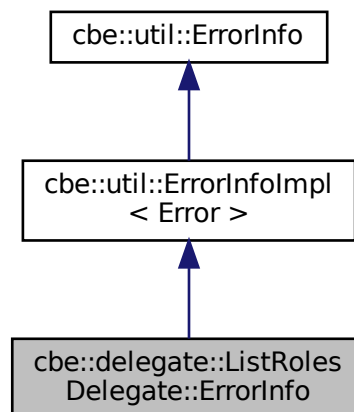
## 11.48 cbe::delegate::ListRolesDelegate::ErrorInfo Struct Reference

```
#include <ListRolesDelegate.h>
```

Inheritance diagram for cbe::delegate::ListRolesDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::ListRolesDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.48.1 Detailed Description

Contains all information about a failed ListRoles.

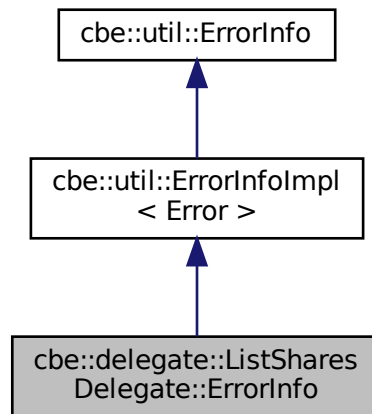
The documentation for this struct was generated from the following file:

- include/cbe/delegate/ListRolesDelegate.h

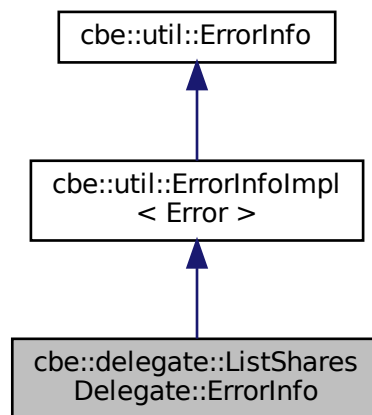
## 11.49 cbe::delegate::ListSharesDelegate::ErrorInfo Struct Reference

```
#include <ListSharesDelegate.h>
```

Inheritance diagram for cbe::delegate::ListSharesDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::ListSharesDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.49.1 Detailed Description

Contains all information about a failed Share.

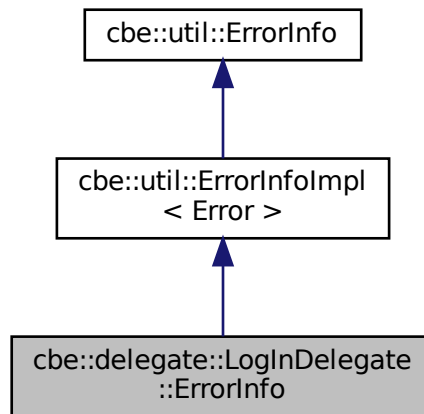
The documentation for this struct was generated from the following file:

- include/cbe/delegate/ListSharesDelegate.h

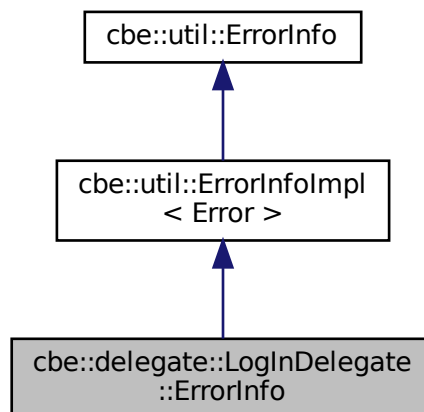
## 11.50 cbe::delegate::LogInDelegate::ErrorInfo Struct Reference

```
#include <LogInDelegate.h>
```

Inheritance diagram for cbe::delegate::LogInDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::LogInDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.50.1 Detailed Description

Contains all information about a failed login.

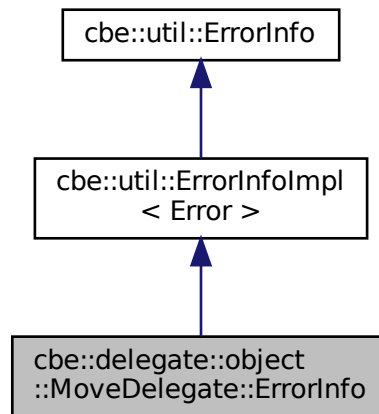
The documentation for this struct was generated from the following file:

- include/cbe/delegate/LogInDelegate.h

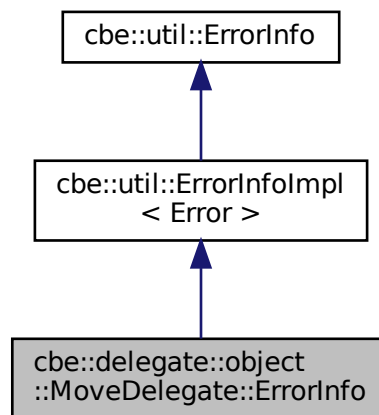
## 11.51 cbe::delegate::object::MoveDelegate::ErrorInfo Struct Reference

```
#include <MoveDelegate.h>
```

Inheritance diagram for cbe::delegate::object::MoveDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::object::MoveDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.51.1 Detailed Description

Contains all information about a failed Move.

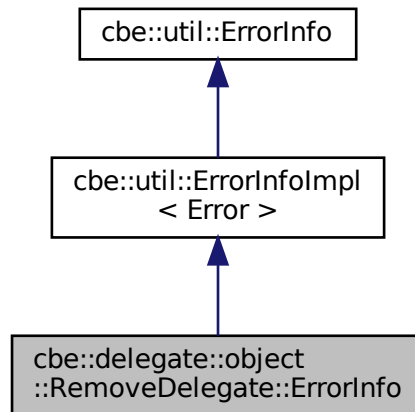
The documentation for this struct was generated from the following file:

- include/cbe/delegate/object/MoveDelegate.h

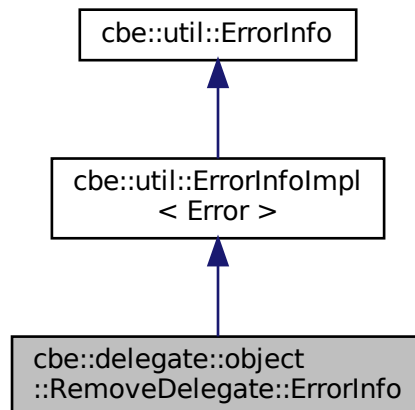
## 11.52 cbe::delegate::object::RemoveDelegate::ErrorInfo Struct Reference

```
#include <RemoveDelegate.h>
```

Inheritance diagram for cbe::delegate::object::RemoveDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::object::RemoveDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.52.1 Detailed Description

Contains all information about a failed Remove.

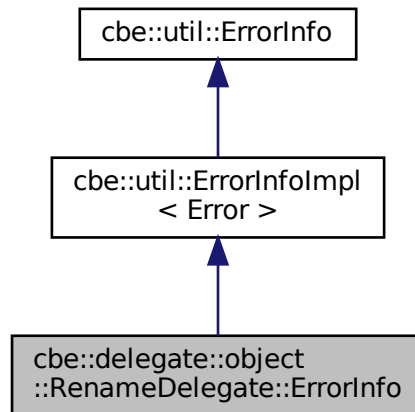
The documentation for this struct was generated from the following file:

- include/cbe/delegate/object/RemoveDelegate.h

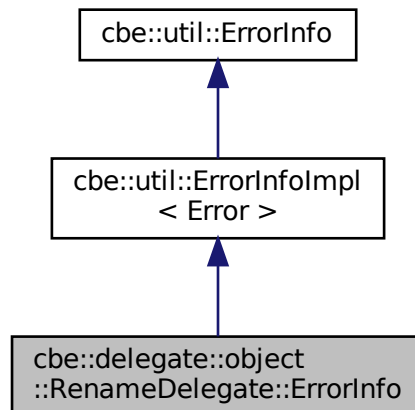
## 11.53 cbe::delegate::object::RenameDelegate::ErrorInfo Struct Reference

```
#include <RenameDelegate.h>
```

Inheritance diagram for cbe::delegate::object::RenameDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::object::RenameDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.53.1 Detailed Description

Contains all information about a failed Rename.

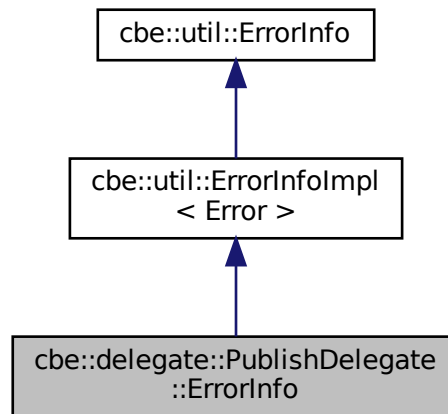
The documentation for this struct was generated from the following file:

- include/cbe/delegate/object/RenameDelegate.h

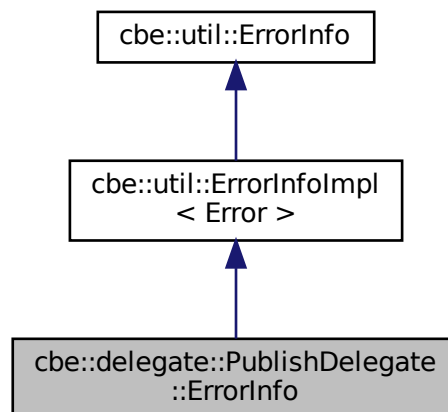
## 11.54 cbe::delegate::PublishDelegate::ErrorInfo Struct Reference

```
#include <PublishDelegate.h>
```

Inheritance diagram for cbe::delegate::PublishDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::PublishDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.54.1 Detailed Description

Contains all information about a failed [Publish](#).

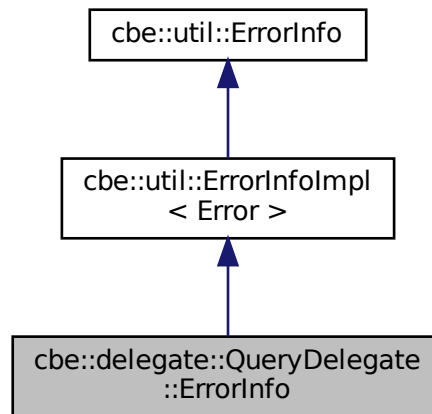
The documentation for this struct was generated from the following file:

- include/cbe/delegate/PublishDelegate.h

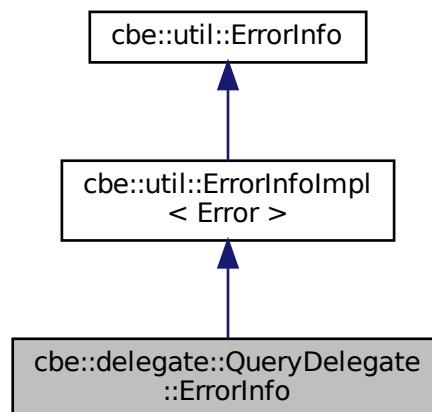
## 11.55 cbe::delegate::QueryDelegate::ErrorInfo Struct Reference

```
#include <QueryDelegate.h>
```

Inheritance diagram for cbe::delegate::QueryDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::QueryDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.55.1 Detailed Description

Contains all information about a failed query.

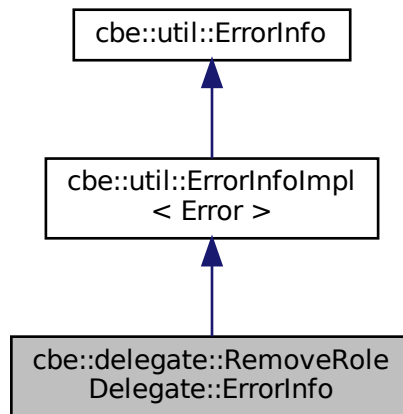
The documentation for this struct was generated from the following file:

- include/cbe/delegate/QueryDelegate.h

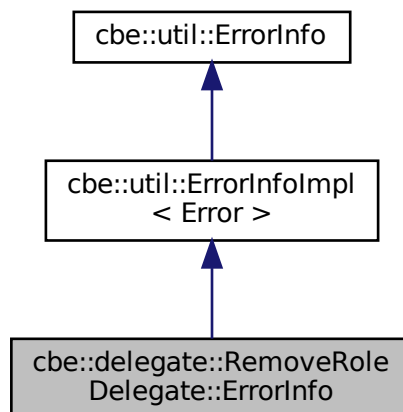
## 11.56 cbe::delegate::RemoveRoleDelegate::ErrorInfo Struct Reference

```
#include <RemoveRoleDelegate.h>
```

Inheritance diagram for cbe::delegate::RemoveRoleDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::RemoveRoleDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.56.1 Detailed Description

Contains all information about a failed `RemoveRole`.

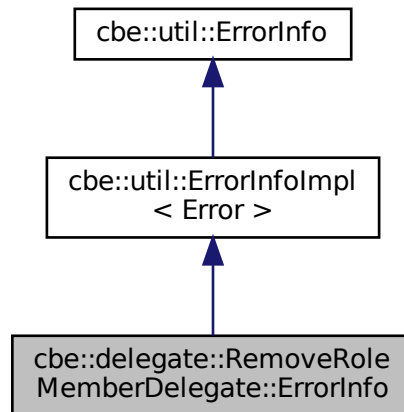
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/RemoveRoleDelegate.h`

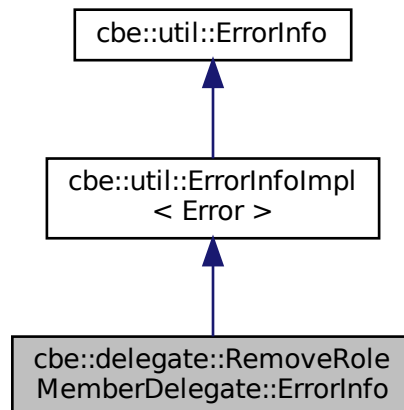
## 11.57 cbe::delegate::RemoveRoleMemberDelegate::ErrorInfo Struct Reference

```
#include <RemoveRoleMemberDelegate.h>
```

Inheritance diagram for cbe::delegate::RemoveRoleMemberDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::RemoveRoleMemberDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.57.1 Detailed Description

Contains all information about a failed RemoveMember.

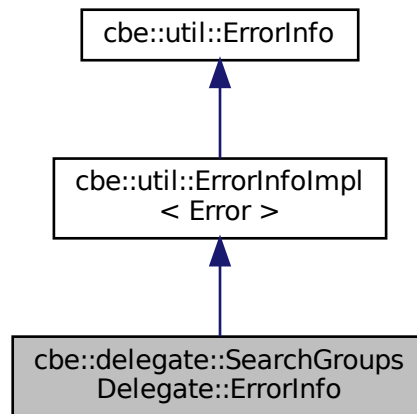
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/RemoveRoleMemberDelegate.h`

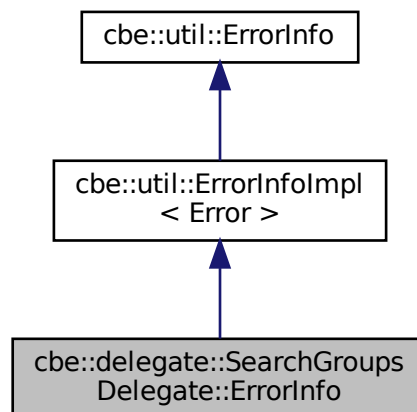
## 11.58 cbe::delegate::SearchGroupsDelegate::ErrorInfo Struct Reference

```
#include <SearchGroupsDelegate.h>
```

Inheritance diagram for cbe::delegate::SearchGroupsDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::SearchGroupsDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.58.1 Detailed Description

Contains all information about a failed SearchGroups.

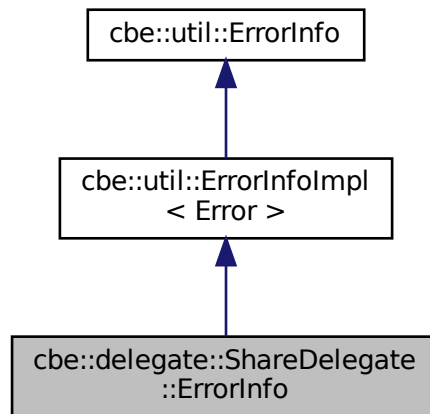
The documentation for this struct was generated from the following file:

- include/cbe/delegate/SearchGroupsDelegate.h

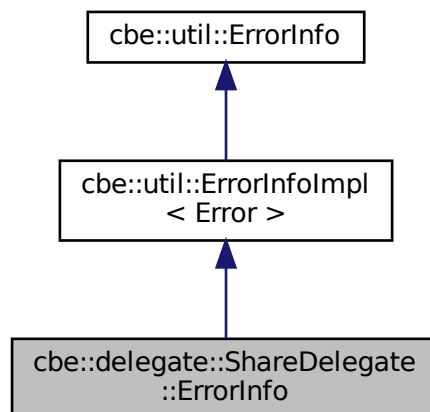
## 11.59 cbe::delegate::ShareDelegate::ErrorInfo Struct Reference

```
#include <ShareDelegate.h>
```

Inheritance diagram for cbe::delegate::ShareDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::ShareDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.59.1 Detailed Description

Contains all information about a failed Share.

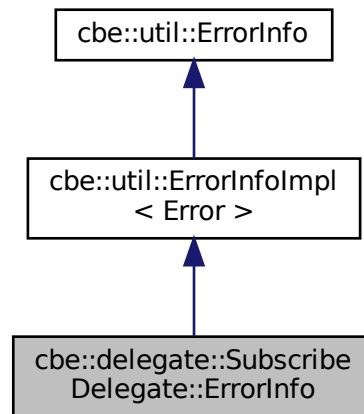
The documentation for this struct was generated from the following file:

- include/cbe/delegate/ShareDelegate.h

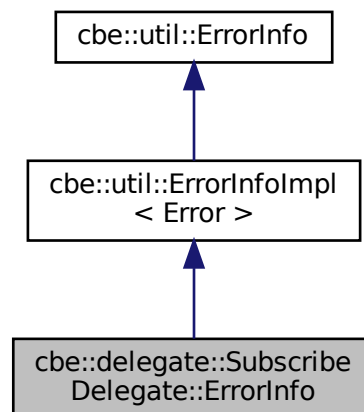
## 11.60 cbe::delegate::SubscribeDelegate::ErrorInfo Struct Reference

```
#include <SubscribeDelegate.h>
```

Inheritance diagram for cbe::delegate::SubscribeDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::SubscribeDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.60.1 Detailed Description

Contains all information about a failed subscribe.

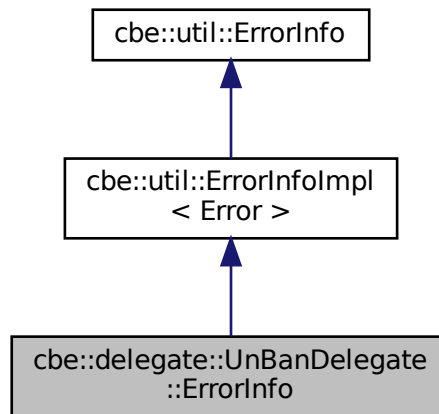
The documentation for this struct was generated from the following file:

- include/cbe/delegate/SubscribeDelegate.h

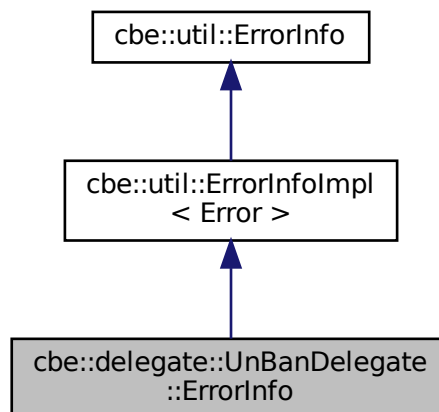
## 11.61 cbe::delegate::UnBanDelegate::ErrorInfo Struct Reference

```
#include <UnBanDelegate.h>
```

Inheritance diagram for cbe::delegate::UnBanDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::UnBanDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.61.1 Detailed Description

Contains all information about a failed UnBan.

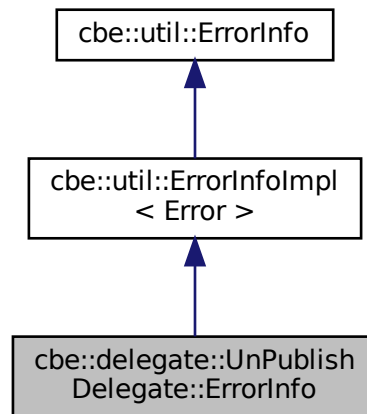
The documentation for this struct was generated from the following file:

- include/cbe/delegate/UnBanDelegate.h

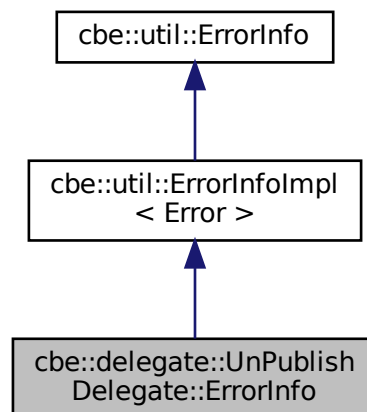
## 11.62 cbe::delegate::UnPublishDelegate::ErrorInfo Struct Reference

```
#include <UnPublishDelegate.h>
```

Inheritance diagram for cbe::delegate::UnPublishDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::UnPublishDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.62.1 Detailed Description

Contains all information about a failed UnPublish.

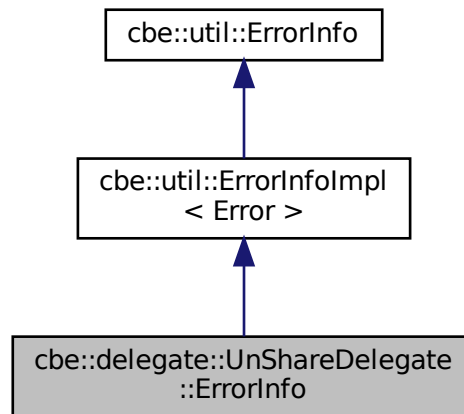
The documentation for this struct was generated from the following file:

- include/cbe/delegate/UnPublishDelegate.h

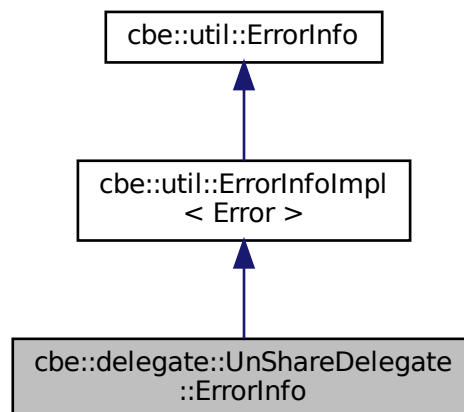
## 11.63 cbe::delegate::UnShareDelegate::ErrorInfo Struct Reference

```
#include <UnShareDelegate.h>
```

Inheritance diagram for cbe::delegate::UnShareDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::UnShareDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.63.1 Detailed Description

Contains all information about a failed UnShare.

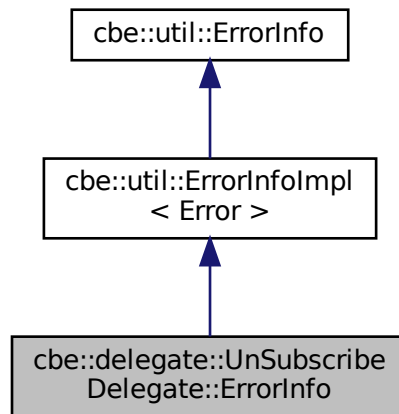
The documentation for this struct was generated from the following file:

- include/cbe/delegate/UnShareDelegate.h

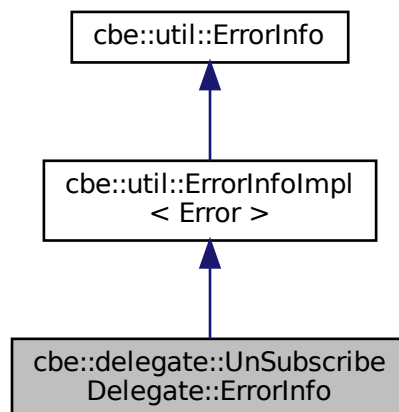
## 11.64 cbe::delegate::UnSubscribeDelegate::ErrorInfo Struct Reference

```
#include <UnSubscribeDelegate.h>
```

Inheritance diagram for cbe::delegate::UnSubscribeDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::UnSubscribeDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.64.1 Detailed Description

Contains all information about a failed UnSubscribe.

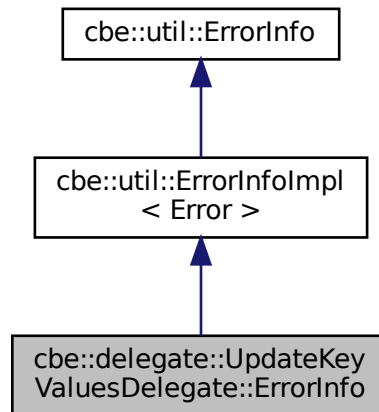
The documentation for this struct was generated from the following file:

- include/cbe/delegate/UnSubscribeDelegate.h

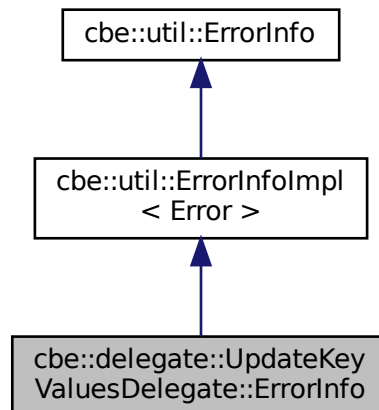
## 11.65 cbe::delegate::UpdateKeyValuesDelegate::ErrorInfo Struct Reference

```
#include <UpdateKeyValuesDelegate.h>
```

Inheritance diagram for cbe::delegate::UpdateKeyValuesDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::UpdateKeyValuesDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.65.1 Detailed Description

Contains all information about a failed UpdateKeyValues.

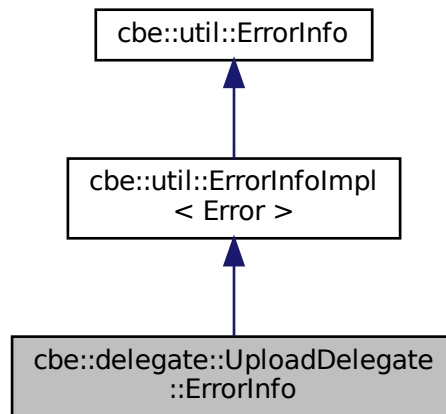
The documentation for this struct was generated from the following file:

- include/cbe/delegate/UpdateKeyValuesDelegate.h

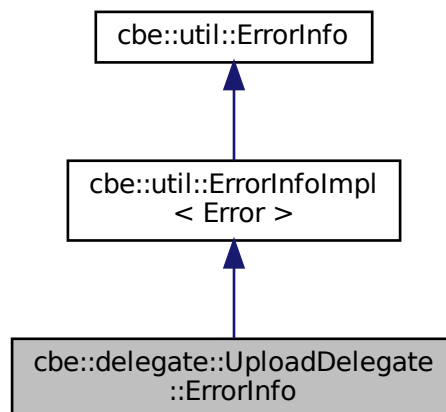
## 11.66 cbe::delegate::UploadDelegate::ErrorInfo Struct Reference

```
#include <UploadDelegate.h>
```

Inheritance diagram for cbe::delegate::UploadDelegate::ErrorInfo:



Collaboration diagram for cbe::delegate::UploadDelegate::ErrorInfo:



### Additional Inherited Members

#### 11.66.1 Detailed Description

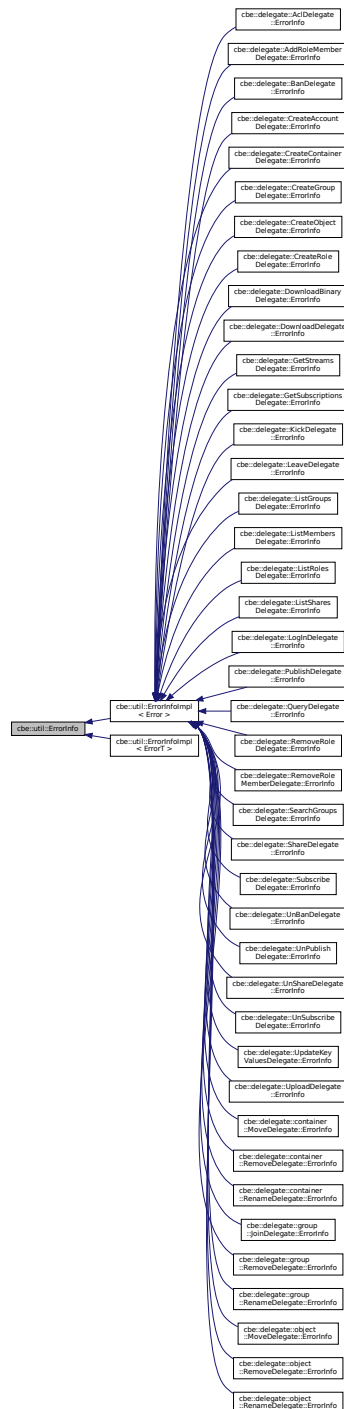
Contains all information about a failed Upload.

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

- include/cbe/delegate/UploadDelegate.h

## 11.67 cbe::util::ErrorInfo Struct Reference

Inheritance diagram for cbe::util::ErrorInfo:



### Public Member Functions

- virtual void **printError** (std::ostream &os) const =0
- **operator bool** () const

## Public Attributes

- `std::string contextStr {}`

## Protected Member Functions

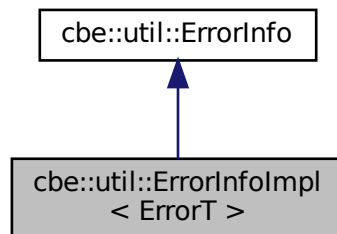
- **ErrorInfo** ([Context](#) &&context)

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

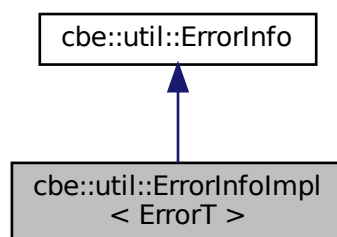
- `include/cbe/util/ErrorInfo.h`

## 11.68 cbe::util::ErrorInfoImpl< ErrorT > Struct Template Reference

Inheritance diagram for `cbe::util::ErrorInfoImpl< ErrorT >`:



Collaboration diagram for `cbe::util::ErrorInfoImpl< ErrorT >`:



## Public Types

- using **Base** = [ErrorInfoImpl](#)< ErrorT >

## Public Member Functions

- **ErrorInfoImpl** ([Context](#) &&context, ErrorT &&error)
- `template<class ErrorT2 >`  
[ErrorInfoImpl](#) & **operator=** ([ErrorInfoImpl](#)< ErrorT2 > &&rh)

- void **printError** (std::ostream &os) const final
- template<class ErrorT2 >  
[ErrorInfoImpl](#)< ErrorT > & **operator=** ([ErrorInfoImpl](#)< ErrorT2 > &&rh)

### Public Attributes

- ErrorT **error** {}

### Additional Inherited Members

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

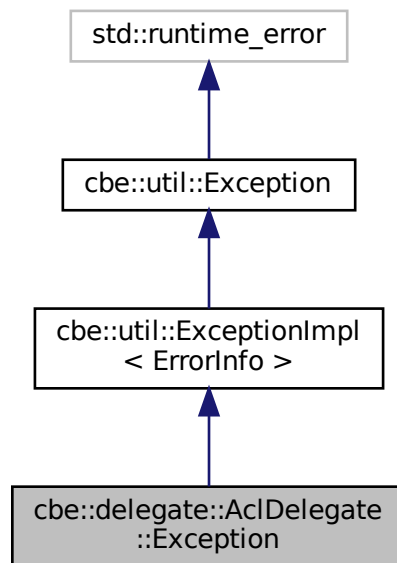
- include/cbe/util/ErrorInfo.h

## 11.69 cbe::delegate::AcIDelegate::Exception Struct Reference

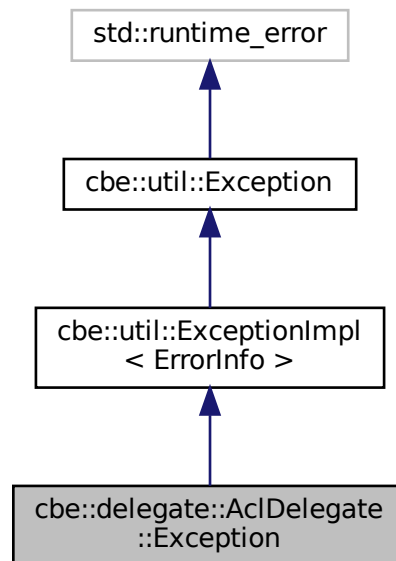
exception thrown by [cbe::Object::getAcI\(\)](#) if the request fails.

```
#include <AcIDelegate.h>
```

Inheritance diagram for cbe::delegate::AcIDelegate::Exception:



Collaboration diagram for `cbe::delegate::AcIDelegate::Exception`:



## Additional Inherited Members

### 11.69.1 Detailed Description

exception thrown by `cbe::Object::getAcI()` if the request fails.

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

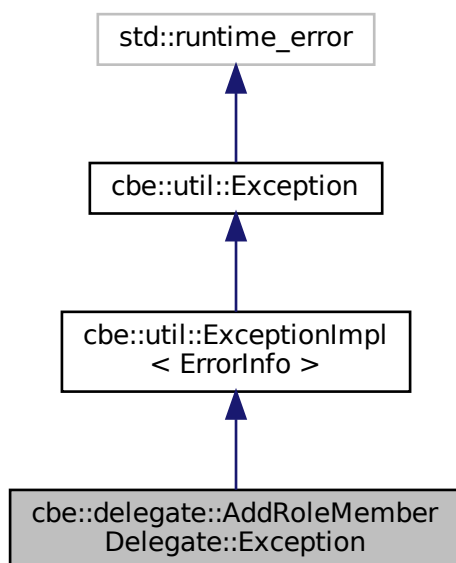
- `include/cbe/delegate/AcIDelegate.h`

## 11.70 cbe::delegate::AddRoleMemberDelegate::Exception Struct Reference

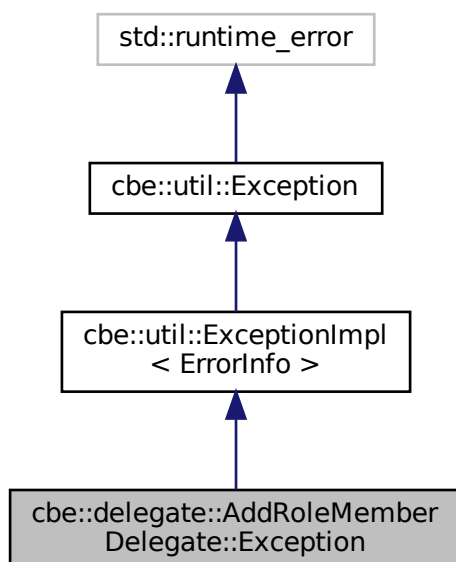
exception thrown by `cbe::Group::AddMember()` if the request fails.

`#include <AddRoleMemberDelegate.h>`

Inheritance diagram for cbe::delegate::AddRoleMemberDelegate::Exception:



Collaboration diagram for cbe::delegate::AddRoleMemberDelegate::Exception:



## Additional Inherited Members

### 11.70.1 Detailed Description

exception thrown by `cbe::Group::AddMember()` if the request fails.  
The documentation for this struct was generated from the following file:

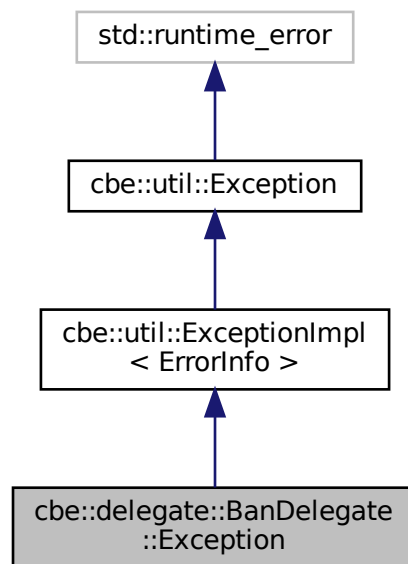
- `include/cbe/delegate/AddRoleMemberDelegate.h`

### 11.71 `cbe::delegate::BanDelegate::Exception` Struct Reference

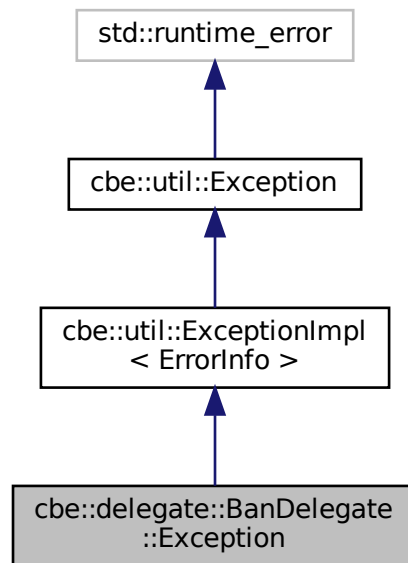
exception thrown by `cbe::Member::ban(std::string)` if the request fails.

```
#include <BanDelegate.h>
```

Inheritance diagram for `cbe::delegate::BanDelegate::Exception`:



Collaboration diagram for cbe::delegate::BanDelegate::Exception:



## Additional Inherited Members

### 11.71.1 Detailed Description

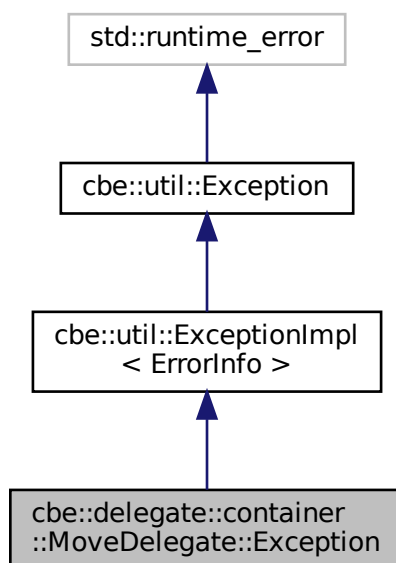
exception thrown by [cbe::Member::ban\(std::string\)](#) if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/BanDelegate.h`

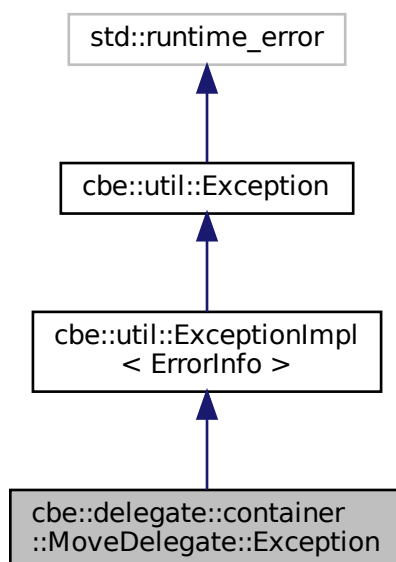
## 11.72 cbe::delegate::container::MoveDelegate::Exception Struct Reference

exception thrown by [cbe::Container::move\(\)](#) if the request fails.  
`#include <MoveDelegate.h>`

Inheritance diagram for `cbe::delegate::container::MoveDelegate::Exception`:



Collaboration diagram for `cbe::delegate::container::MoveDelegate::Exception`:



## Additional Inherited Members

### 11.72.1 Detailed Description

exception thrown by [cbe::Container::move\(\)](#) if the request fails.

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

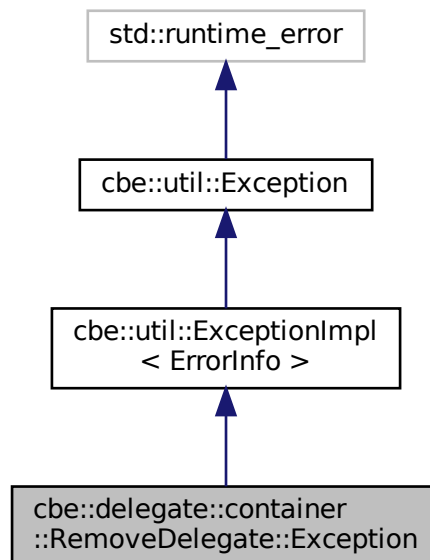
- include/cbe/delegate/container/MoveDelegate.h

## 11.73 cbe::delegate::container::RemoveDelegate::Exception Struct Reference

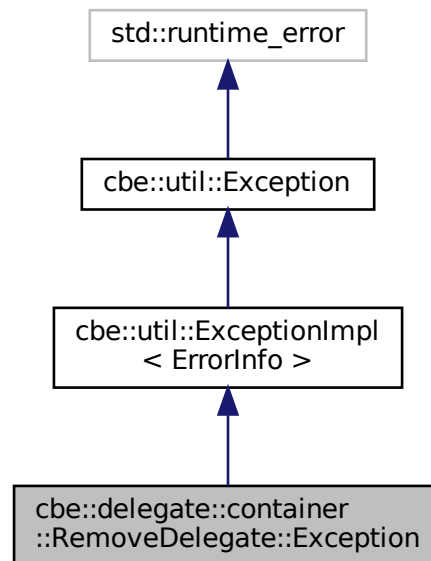
exception thrown by [cbe::Container::remove\(\)](#) if the request fails.

```
#include <RemoveDelegate.h>
```

Inheritance diagram for cbe::delegate::container::RemoveDelegate::Exception:



Collaboration diagram for `cbe::delegate::container::RemoveDelegate::Exception`:



## Additional Inherited Members

### 11.73.1 Detailed Description

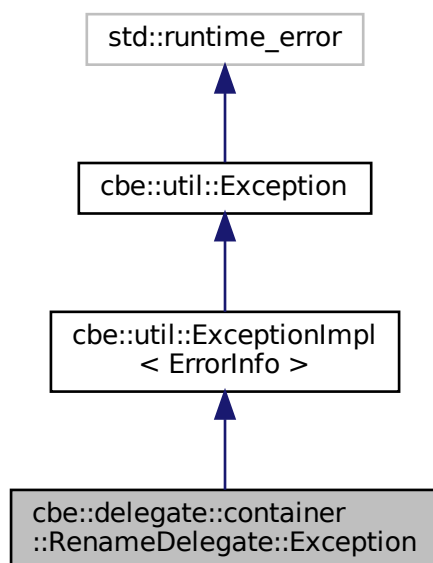
exception thrown by `cbe::Container::remove()` if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/container/RemoveDelegate.h`

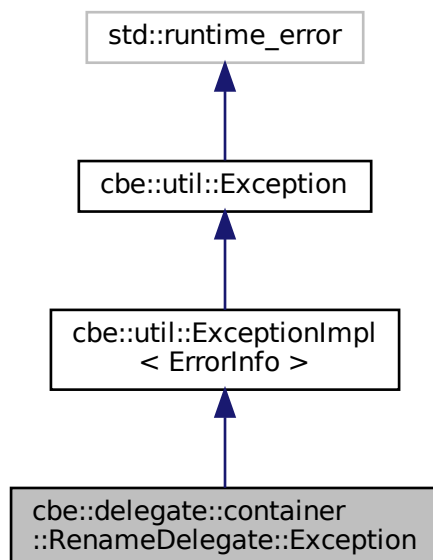
## 11.74 `cbe::delegate::container::RenameDelegate::Exception` Struct Reference

exception thrown by `cbe::Container::rename()` if the request fails.  
`#include <RenameDelegate.h>`

Inheritance diagram for cbe::delegate::container::RenameDelegate::Exception:



Collaboration diagram for cbe::delegate::container::RenameDelegate::Exception:



## Additional Inherited Members

### 11.74.1 Detailed Description

exception thrown by [cbe::Container::rename\(\)](#) if the request fails.

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

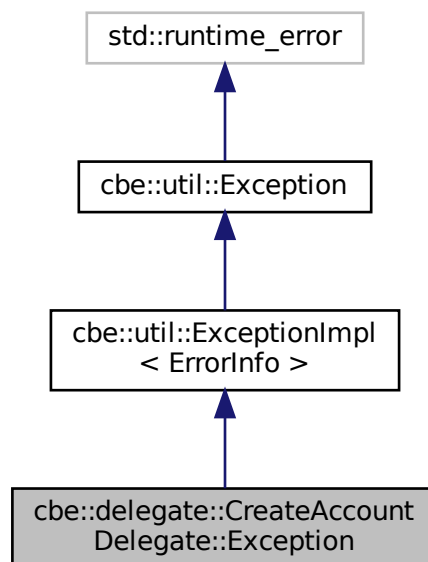
- `include/cbe/delegate/container/RenameDelegate.h`

## 11.75 cbe::delegate::CreateAccountDelegate::Exception Struct Reference

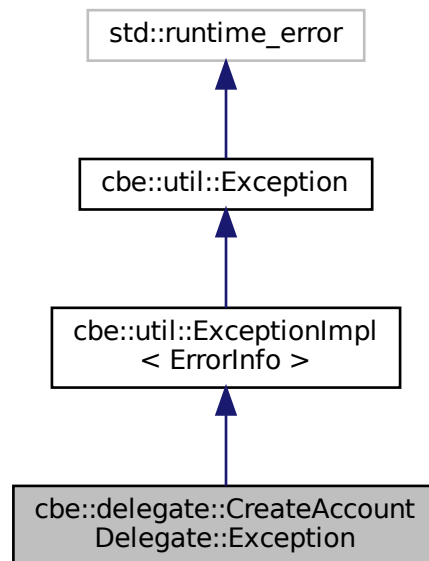
exception thrown by [cbe::CloudBackend::createAccount\(\)](#) if the request fails.

```
#include <CreateAccountDelegate.h>
```

Inheritance diagram for `cbe::delegate::CreateAccountDelegate::Exception`:



Collaboration diagram for cbe::delegate::CreateAccountDelegate::Exception:



## Additional Inherited Members

### 11.75.1 Detailed Description

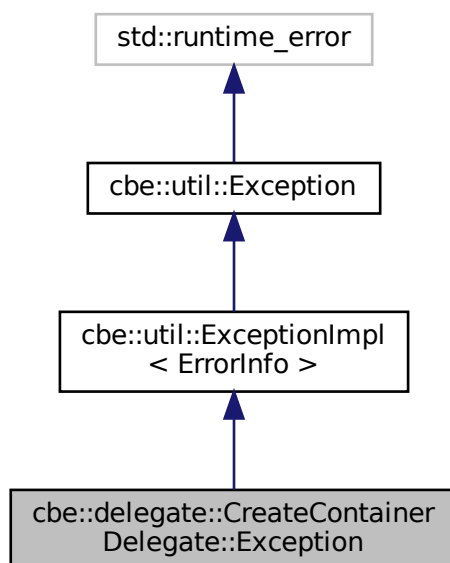
exception thrown by `cbe::CloudBackend::createAccount()` if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/CreateAccountDelegate.h`

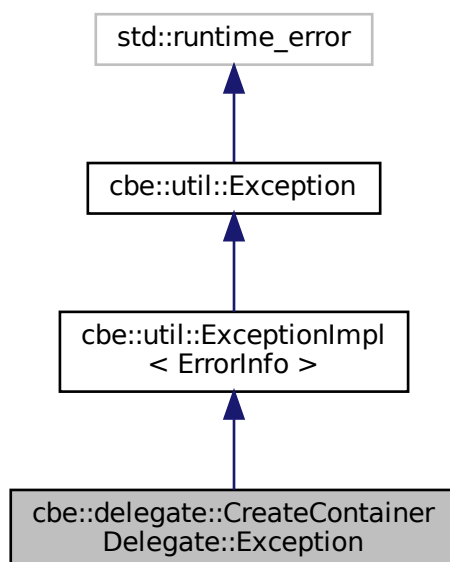
## 11.76 cbe::delegate::CreateContainerDelegate::Exception Struct Reference

exception thrown by `cbe::Container::createContainer()` if the request fails.  
`#include <CreateContainerDelegate.h>`

Inheritance diagram for `cbe::delegate::CreateContainerDelegate::Exception`:



Collaboration diagram for `cbe::delegate::CreateContainerDelegate::Exception`:



## Additional Inherited Members

### 11.76.1 Detailed Description

exception thrown by [cbe::Container::createContainer\(\)](#) if the request fails.

Will change to [cbe::Container::createContainerContainer\(\)](#)

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

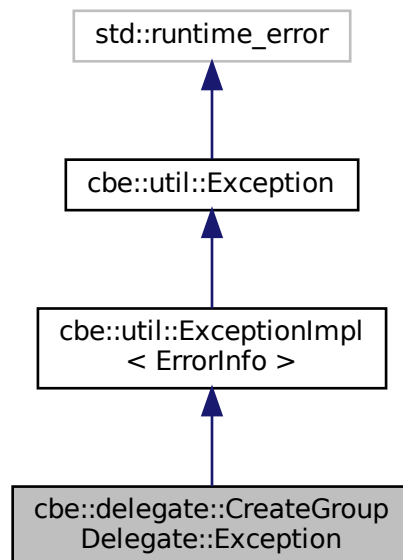
- include/cbe/delegate/CreateContainerDelegate.h

## 11.77 cbe::delegate::CreateGroupDelegate::Exception Struct Reference

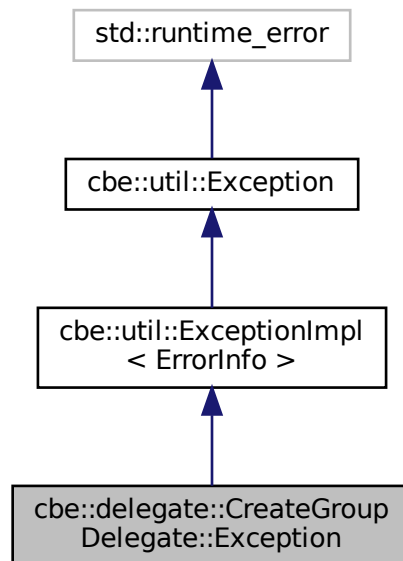
exception thrown by [cbe::Group::createGroup\(\)](#) if the request fails.

```
#include <CreateGroupDelegate.h>
```

Inheritance diagram for cbe::delegate::CreateGroupDelegate::Exception:



Collaboration diagram for `cbe::delegate::CreateGroupDelegate::Exception`:



## Additional Inherited Members

### 11.77.1 Detailed Description

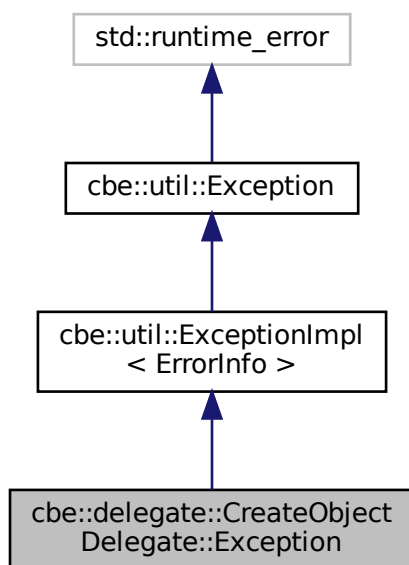
exception thrown by `cbe::Group::createGroup()` if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/CreateGroupDelegate.h`

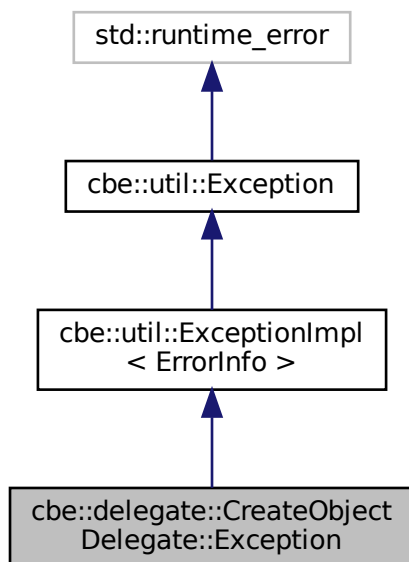
## 11.78 cbe::delegate::CreateObjectDelegate::Exception Struct Reference

exception thrown by `cbe::Container::createObject()` if the request fails.  
`#include <CreateObjectDelegate.h>`

Inheritance diagram for cbe::delegate::CreateObjectDelegate::Exception:



Collaboration diagram for cbe::delegate::CreateObjectDelegate::Exception:



## Additional Inherited Members

### 11.78.1 Detailed Description

exception thrown by `cbe::Container::createObject()` if the request fails.  
The documentation for this struct was generated from the following file:

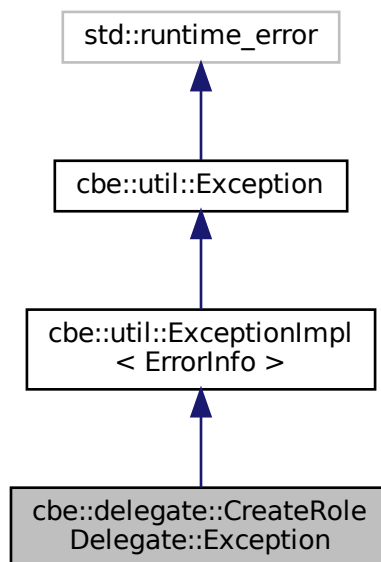
- `include/cbe/delegate/CreateObjectDelegate.h`

## 11.79 `cbe::delegate::CreateRoleDelegate::Exception` Struct Reference

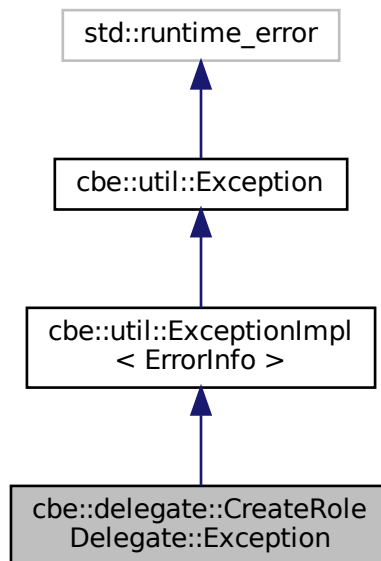
exception thrown by `cbe::Role::createRole()` if the request fails.

```
#include <CreateRoleDelegate.h>
```

Inheritance diagram for `cbe::delegate::CreateRoleDelegate::Exception`:



Collaboration diagram for cbe::delegate::CreateRoleDelegate::Exception:



## Additional Inherited Members

### 11.79.1 Detailed Description

exception thrown by `cbe::Role::createRole()` if the request fails.

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

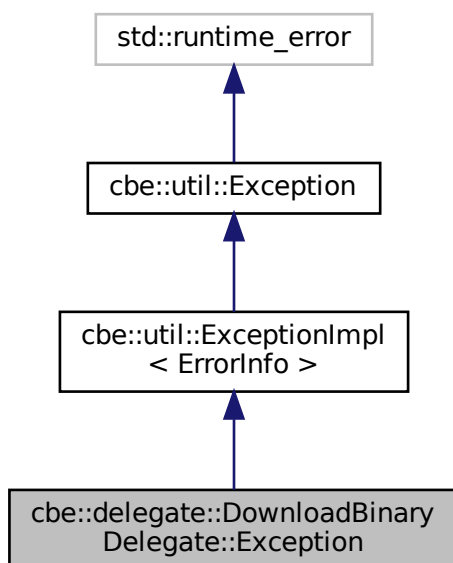
- `include/cbe/delegate/CreateRoleDelegate.h`

## 11.80 cbe::delegate::DownloadBinaryDelegate::Exception Struct Reference

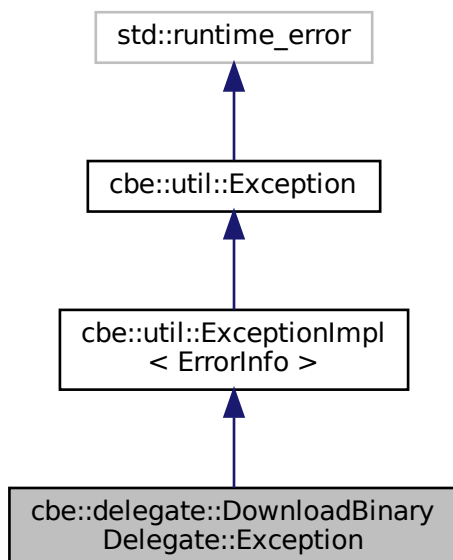
exception thrown by `cbe::Object::download()` if the request fails.

```
#include <DownloadBinaryDelegate.h>
```

Inheritance diagram for cbe::delegate::DownloadBinaryDelegate::Exception:



Collaboration diagram for cbe::delegate::DownloadBinaryDelegate::Exception:



## Additional Inherited Members

### 11.80.1 Detailed Description

exception thrown by [cbe::Object::download\(\)](#) if the request fails.

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

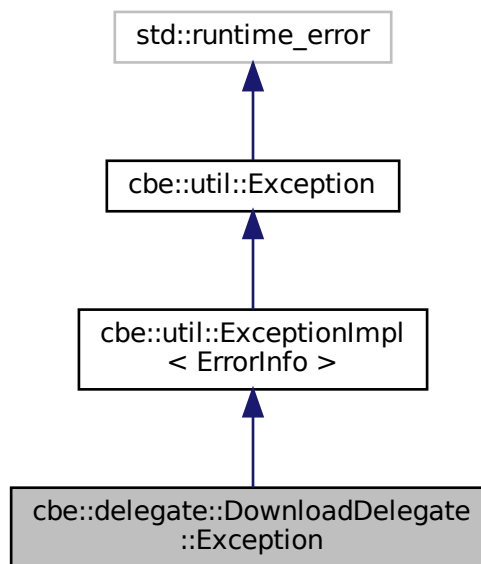
- include/cbe/delegate/DownloadBinaryDelegate.h

## 11.81 cbe::delegate::DownloadDelegate::Exception Struct Reference

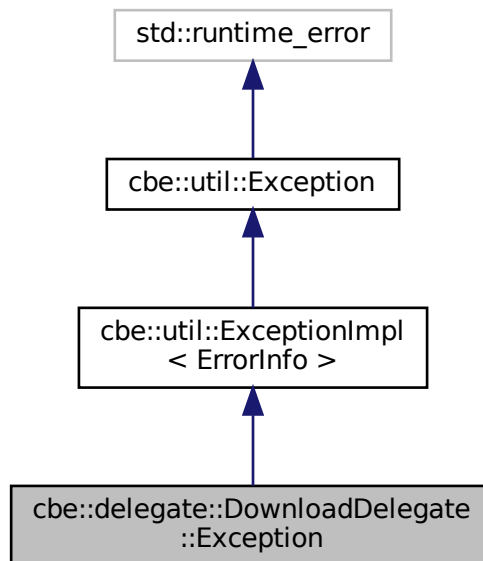
exception thrown by:

```
#include <DownloadDelegate.h>
```

Inheritance diagram for cbe::delegate::DownloadDelegate::Exception:



Collaboration diagram for `cbe::delegate::DownloadDelegate::Exception`:



## Additional Inherited Members

### 11.81.1 Detailed Description

exception thrown by:

- `cbe::Object::download(const std::string&, DownloadDelegatePtr)`
- `cbe::Object::downloadStream(const std::string&, cbe::Stream, DownloadDelegatePtr)`

if the request fails.

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

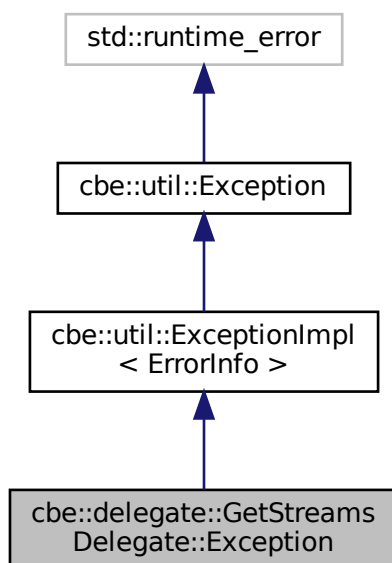
- `include/cbe/delegate/DownloadDelegate.h`

## 11.82 cbe::delegate::GetStreamsDelegate::Exception Struct Reference

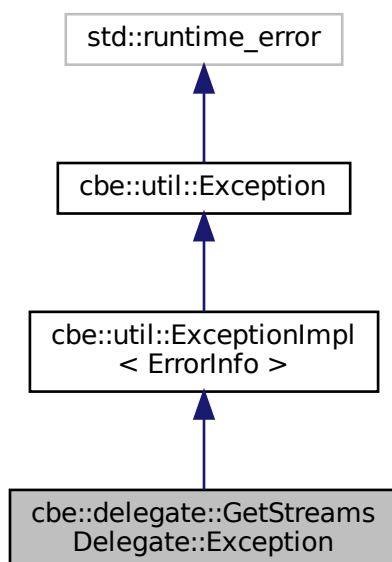
exception thrown by `cbe::Object::getStreams()` if the request fails.

```
#include <GetStreamsDelegate.h>
```

Inheritance diagram for cbe::delegate::GetStreamsDelegate::Exception:



Collaboration diagram for cbe::delegate::GetStreamsDelegate::Exception:



## Additional Inherited Members

### 11.82.1 Detailed Description

exception thrown by [cbe::Object::getStreams\(\)](#) if the request fails.

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

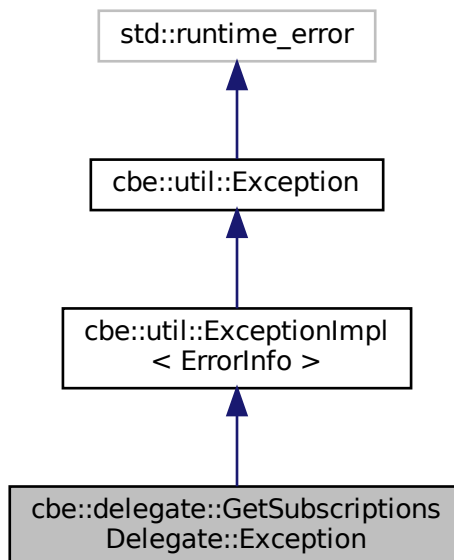
- include/cbe/delegate/GetStreamsDelegate.h

## 11.83 cbe::delegate::GetSubscriptionsDelegate::Exception Struct Reference

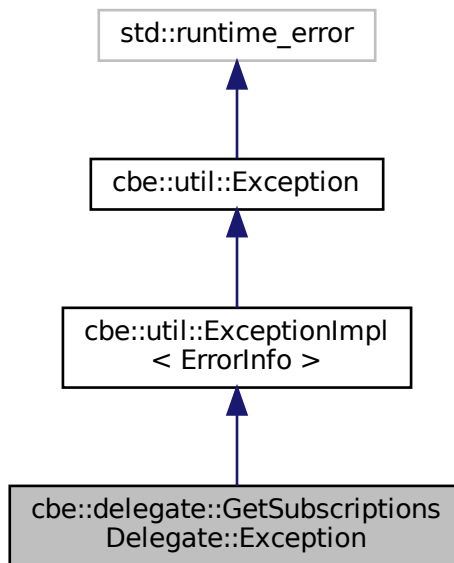
exception thrown by [cbe::SubscribeManager::getSubscriptions\(\)](#) if the request fails.

```
#include <GetSubscriptionsDelegate.h>
```

Inheritance diagram for cbe::delegate::GetSubscriptionsDelegate::Exception:



Collaboration diagram for cbe::delegate::GetSubscriptionsDelegate::Exception:



## Additional Inherited Members

### 11.83.1 Detailed Description

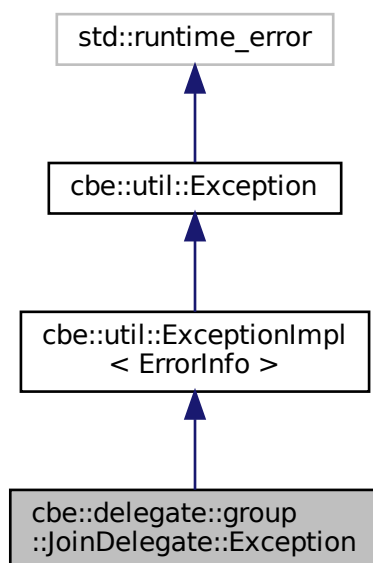
exception thrown by `cbe::SubscribeManager::getSubscriptions()` if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/GetSubscriptionsDelegate.h`

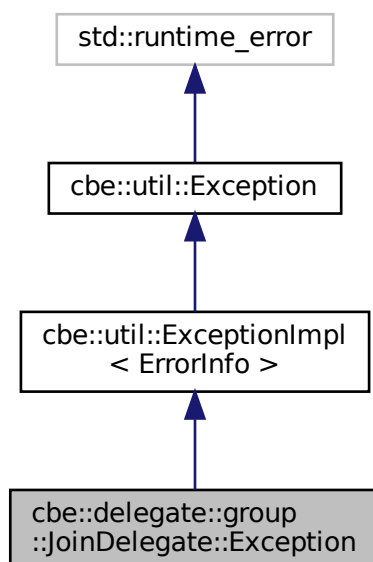
## 11.84 cbe::delegate::group::JoinDelegate::Exception Struct Reference

exception thrown by `cbe::Group::join()` if the request fails.  
`#include <JoinDelegate.h>`

Inheritance diagram for `cbe::delegate::group::JoinDelegate::Exception`:



Collaboration diagram for `cbe::delegate::group::JoinDelegate::Exception`:



## Additional Inherited Members

### 11.84.1 Detailed Description

exception thrown by `cbe::Group::join()` if the request fails.

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

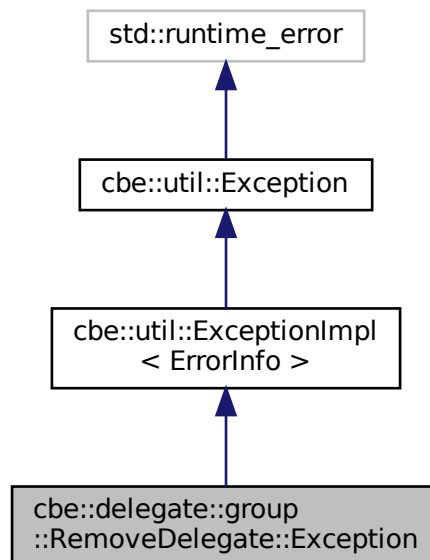
- `include/cbe/delegate/group/JoinDelegate.h`

## 11.85 cbe::delegate::group::RemoveDelegate::Exception Struct Reference

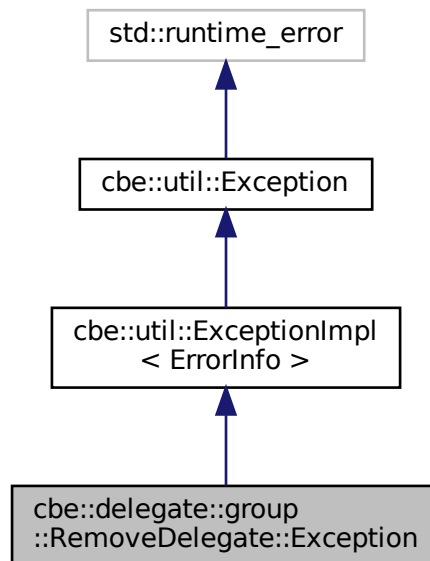
exception thrown by `cbe::Group::remove()` if the request fails.

```
#include <RemoveDelegate.h>
```

Inheritance diagram for `cbe::delegate::group::RemoveDelegate::Exception`:



Collaboration diagram for `cbe::delegate::group::RemoveDelegate::Exception`:



## Additional Inherited Members

### 11.85.1 Detailed Description

exception thrown by `cbe::Group::remove()` if the request fails.

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

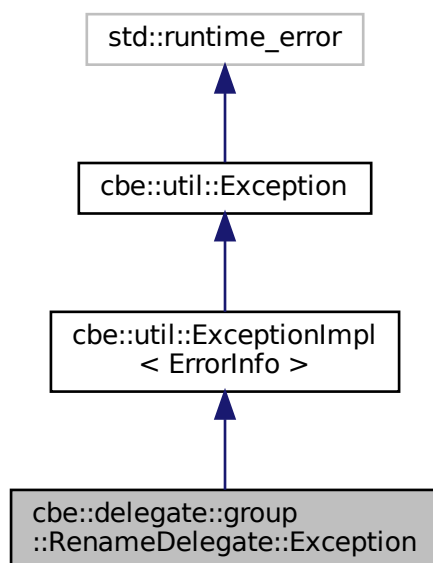
- `include/cbe/delegate/group/RemoveDelegate.h`

## 11.86 `cbe::delegate::group::RenameDelegate::Exception` Struct Reference

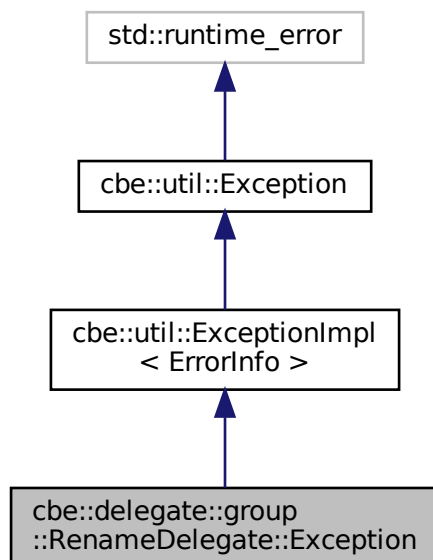
exception thrown by `cbe::Group::rename()` if the request fails.

`#include <RenameDelegate.h>`

Inheritance diagram for cbe::delegate::group::RenameDelegate::Exception:



Collaboration diagram for cbe::delegate::group::RenameDelegate::Exception:



## Additional Inherited Members

### 11.86.1 Detailed Description

exception thrown by [cbe::Group::rename\(\)](#) if the request fails.

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

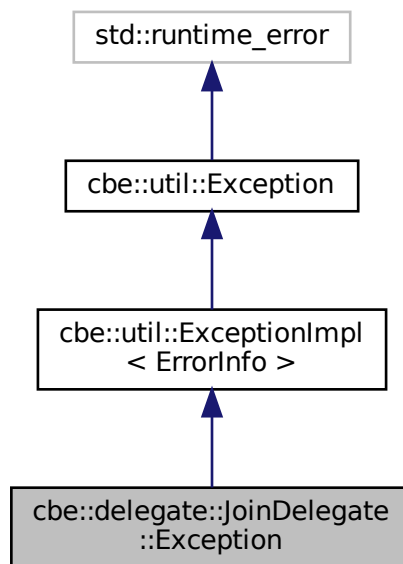
- `include/cbe/delegate/group/RenameDelegate.h`

### 11.87 `cbe::delegate::JoinDelegate::Exception` Struct Reference

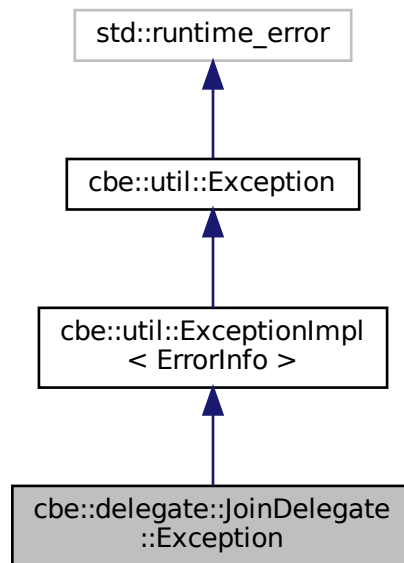
exception thrown by [cbe::QueryChain::join\(\)](#) if the request fails.

```
#include <JoinDelegate.h>
```

Inheritance diagram for `cbe::delegate::JoinDelegate::Exception`:



Collaboration diagram for cbe::delegate::JoinDelegate::Exception:



## Additional Inherited Members

### 11.87.1 Detailed Description

exception thrown by `cbe::QueryChain::join()` if the request fails.

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

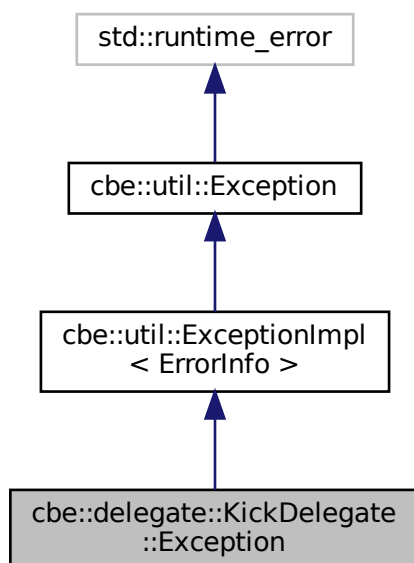
- `include/cbe/delegate/JoinDelegate.h`

## 11.88 cbe::delegate::KickDelegate::Exception Struct Reference

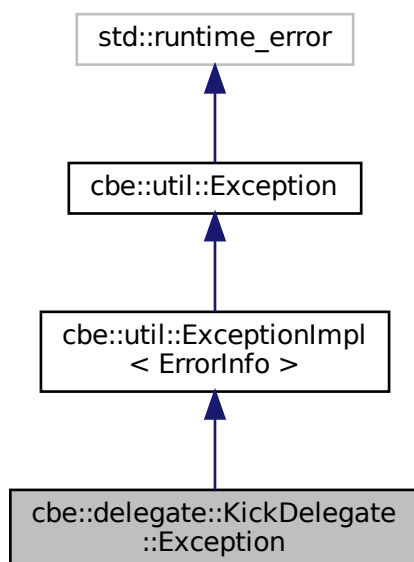
exception thrown by `cbe::Member::kick(std::string)` if the request fails.

```
#include <KickDelegate.h>
```

Inheritance diagram for cbe::delegate::KickDelegate::Exception:



Collaboration diagram for cbe::delegate::KickDelegate::Exception:



## Additional Inherited Members

### 11.88.1 Detailed Description

exception thrown by [cbe::Member::kick\(std::string\)](#) if the request fails.  
The documentation for this struct was generated from the following file:

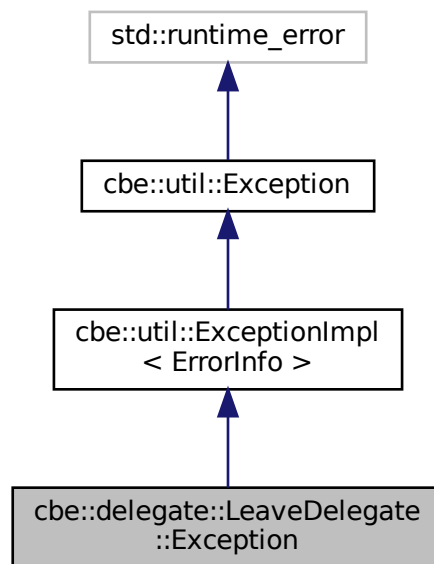
- include/cbe/delegate/KickDelegate.h

## 11.89 cbe::delegate::LeaveDelegate::Exception Struct Reference

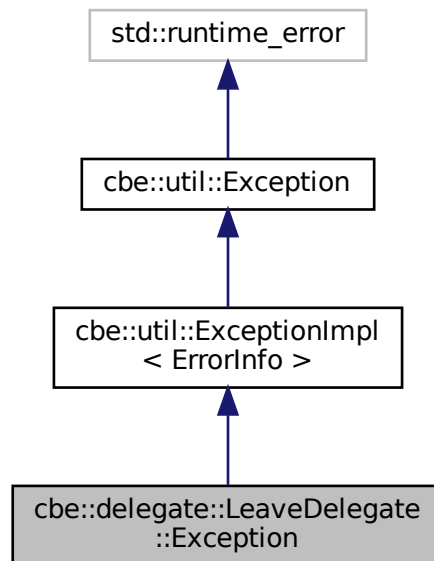
exception thrown by [cbe::Group::leave\(\)](#) if the if the request fails.

```
#include <LeaveDelegate.h>
```

Inheritance diagram for cbe::delegate::LeaveDelegate::Exception:



Collaboration diagram for `cbe::delegate::LeaveDelegate::Exception`:



## Additional Inherited Members

### 11.89.1 Detailed Description

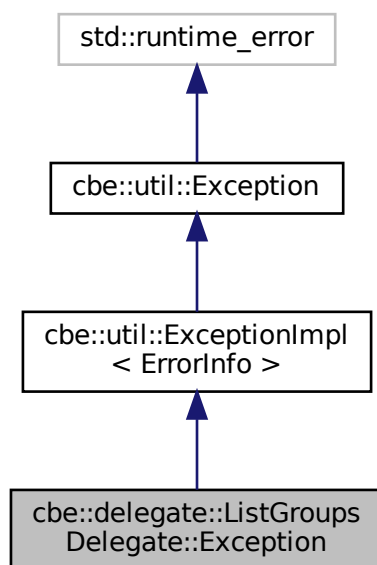
exception thrown by `cbe::Group::leave()` if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/LeaveDelegate.h`

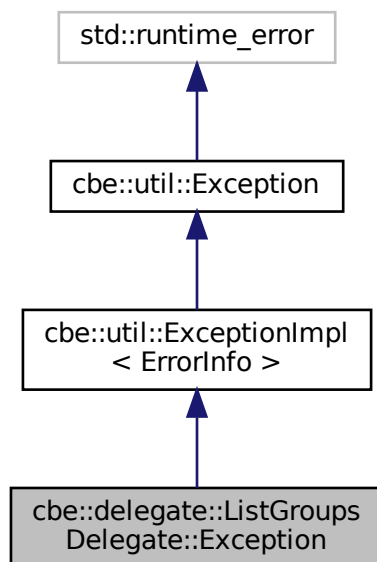
## 11.90 cbe::delegate::ListGroupDelegate::Exception Struct Reference

exception thrown by `cbe::GroupManager::listGroups()` if the request fails.  
`#include <ListGroupDelegate.h>`

Inheritance diagram for cbe::delegate::ListGroupsDelegate::Exception:



Collaboration diagram for cbe::delegate::ListGroupsDelegate::Exception:



## Additional Inherited Members

### 11.90.1 Detailed Description

exception thrown by [cbe::GroupManager::listGroups\(\)](#) if the request fails.  
The documentation for this struct was generated from the following file:

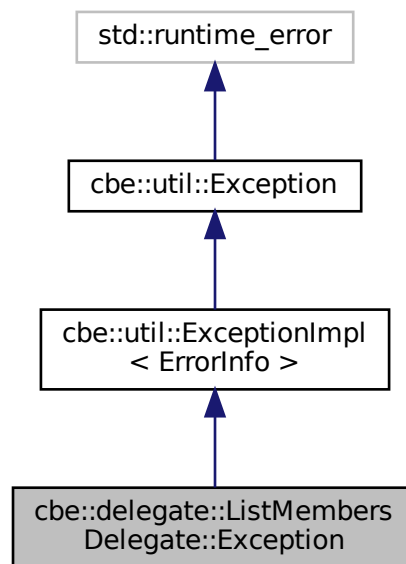
- include/cbe/delegate/ListGroupsDelegate.h

## 11.91 cbe::delegate::ListMembersDelegate::Exception Struct Reference

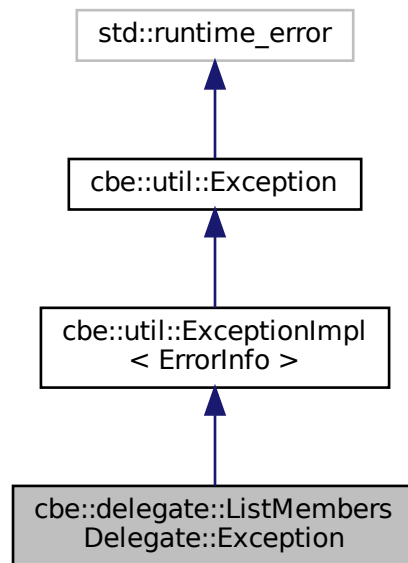
exception thrown by [cbe::Group::listMembers\(\)](#) if the request fails.

```
#include <ListMembersDelegate.h>
```

Inheritance diagram for cbe::delegate::ListMembersDelegate::Exception:



Collaboration diagram for cbe::delegate::ListMembersDelegate::Exception:



## Additional Inherited Members

### 11.91.1 Detailed Description

exception thrown by `cbe::Group::listMembers()` if the request fails.

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

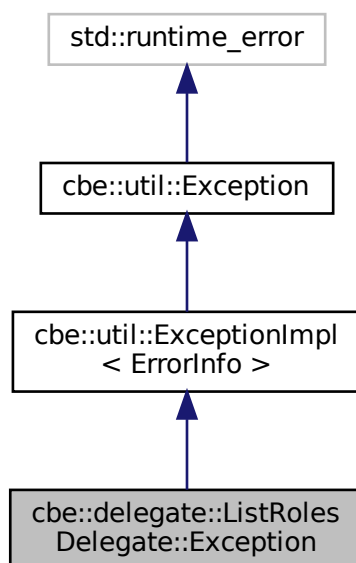
- `include/cbe/delegate/ListMembersDelegate.h`

## 11.92 cbe::delegate::ListRolesDelegate::Exception Struct Reference

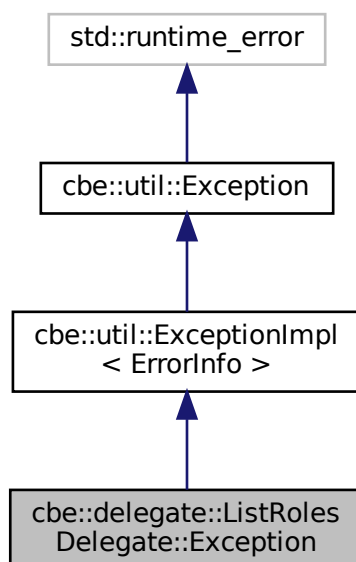
exception thrown by `cbe::Group::ListRoles()` if the request fails.

```
#include <ListRolesDelegate.h>
```

Inheritance diagram for cbe::delegate::ListRolesDelegate::Exception:



Collaboration diagram for cbe::delegate::ListRolesDelegate::Exception:



## Additional Inherited Members

### 11.92.1 Detailed Description

exception thrown by `cbe::Group::ListRoles()` if the request fails.

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

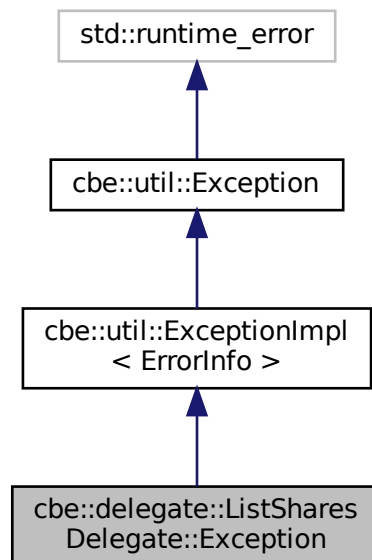
- `include/cbe/delegate/ListRolesDelegate.h`

## 11.93 cbe::delegate::ListSharesDelegate::Exception Struct Reference

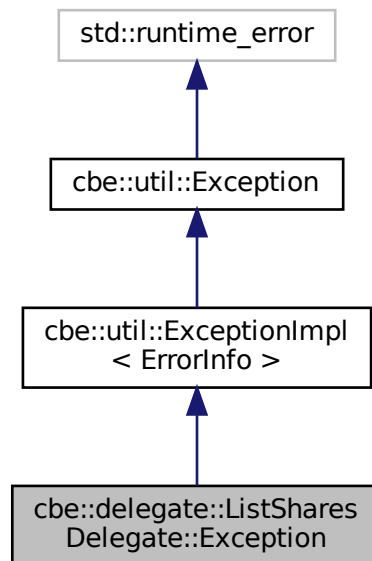
exception thrown by `cbe::ShareManager::listAvailableShares()` or `cbe::ShareManager::listMyShares()` if the request fails.

```
#include <ListSharesDelegate.h>
```

Inheritance diagram for `cbe::delegate::ListSharesDelegate::Exception`:



Collaboration diagram for `cbe::delegate::ListSharesDelegate::Exception`:



## Additional Inherited Members

### 11.93.1 Detailed Description

exception thrown by `cbe::ShareManager::listAvailableShares()` or `cbe::ShareManager::listMyShares()` if the request fails.

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

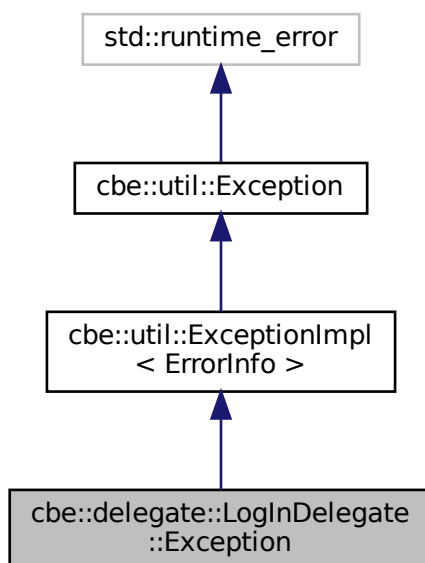
- `include/cbe/delegate/ListSharesDelegate.h`

## 11.94 cbe::delegate::LogInDelegate::Exception Struct Reference

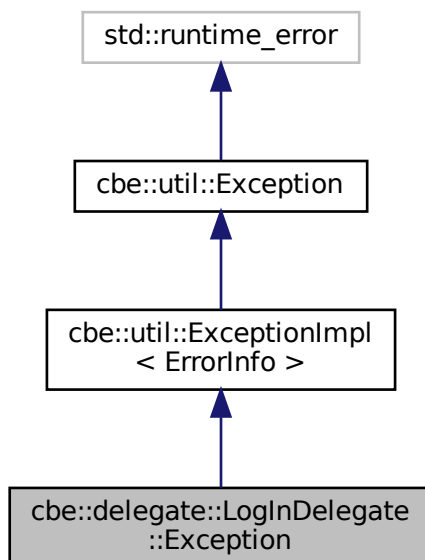
exception thrown by `cbe::CloudBackend::login(const std::string&, const std::string&, const std::string&)` if the request fails.

```
#include <LogInDelegate.h>
```

Inheritance diagram for cbe::delegate::LogInDelegate::Exception:



Collaboration diagram for cbe::delegate::LogInDelegate::Exception:



## Additional Inherited Members

### 11.94.1 Detailed Description

exception thrown by [cbe::CloudBackend::login](#)(const std::string&,const std::string&, const std::string&) if the request fails.

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

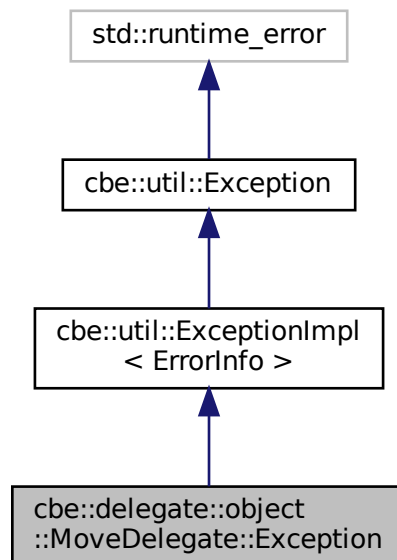
- include/cbe/delegate/LoginDelegate.h

## 11.95 cbe::delegate::object::MoveDelegate::Exception Struct Reference

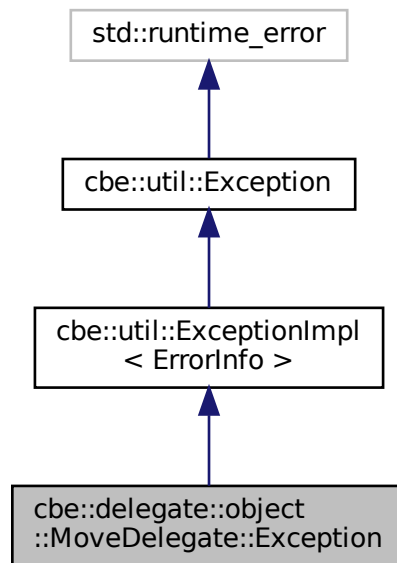
exception thrown by [cbe::Object::move\(\)](#) if the request fails.

```
#include <MoveDelegate.h>
```

Inheritance diagram for cbe::delegate::object::MoveDelegate::Exception:



Collaboration diagram for cbe::delegate::object::MoveDelegate::Exception:



## Additional Inherited Members

### 11.95.1 Detailed Description

exception thrown by `cbe::Object::move()` if the request fails.

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

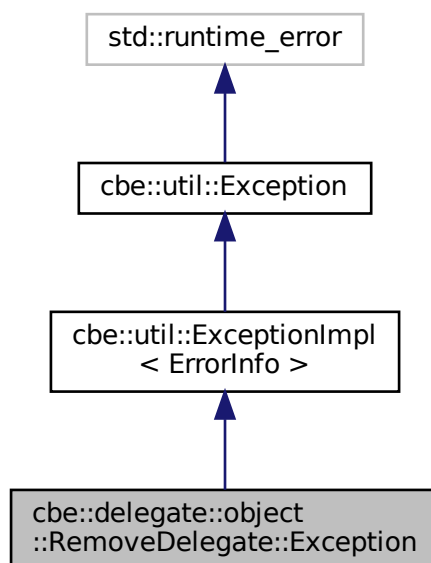
- `include/cbe/delegate/object/MoveDelegate.h`

## 11.96 cbe::delegate::object::RemoveDelegate::Exception Struct Reference

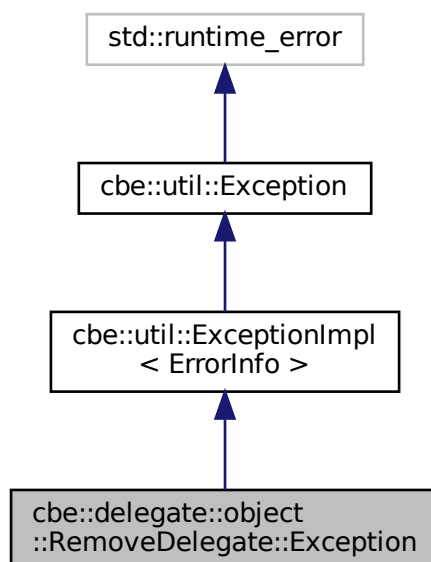
exception thrown by `cbe::Object::remove()` if the request fails.

`#include <RemoveDelegate.h>`

Inheritance diagram for `cbe::delegate::object::RemoveDelegate::Exception`:



Collaboration diagram for `cbe::delegate::object::RemoveDelegate::Exception`:



## Additional Inherited Members

### 11.96.1 Detailed Description

exception thrown by [cbe::Object::remove\(\)](#) if the request fails.

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

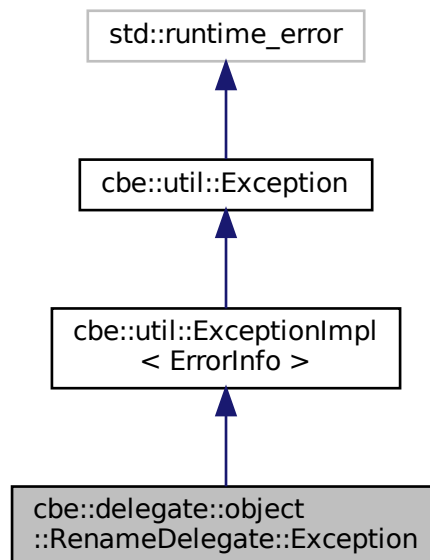
- include/cbe/delegate/object/RemoveDelegate.h

## 11.97 cbe::delegate::object::RenameDelegate::Exception Struct Reference

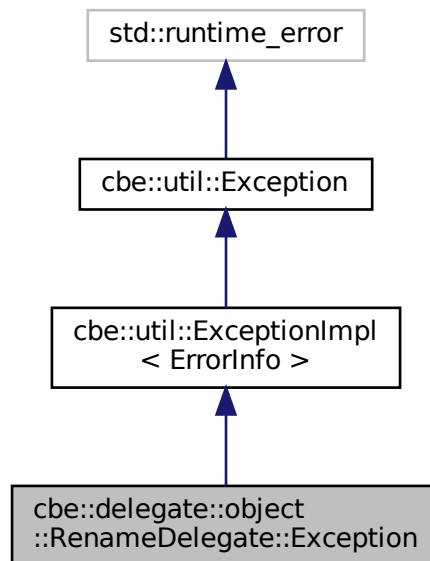
exception thrown by [cbe::Object::rename\(\)](#) if the request fails.

```
#include <RenameDelegate.h>
```

Inheritance diagram for cbe::delegate::object::RenameDelegate::Exception:



Collaboration diagram for `cbe::delegate::object::RenameDelegate::Exception`:



## Additional Inherited Members

### 11.97.1 Detailed Description

exception thrown by `cbe::Object::rename()` if the request fails.

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

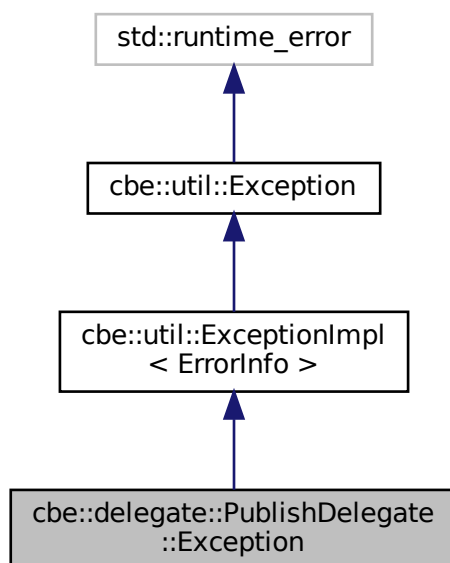
- `include/cbe/delegate/object/RenameDelegate.h`

## 11.98 cbe::delegate::PublishDelegate::Exception Struct Reference

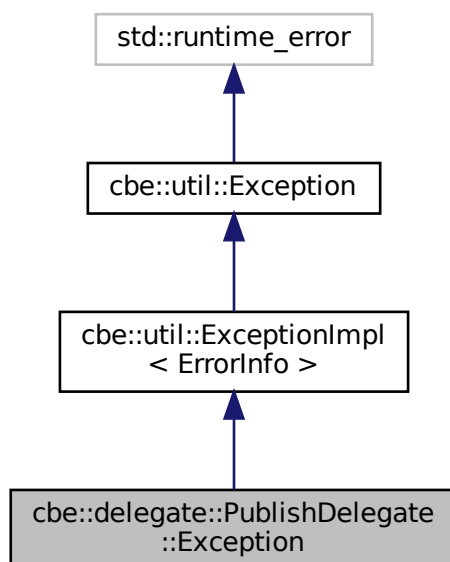
exception thrown by:

```
#include <PublishDelegate.h>
```

Inheritance diagram for cbe::delegate::PublishDelegate::Exception:



Collaboration diagram for cbe::delegate::PublishDelegate::Exception:



## Additional Inherited Members

### 11.98.1 Detailed Description

exception thrown by:

- [cbe::Container::publish\(\)](#)
- [cbe::Object::publish\(\)](#)

if the request fails.

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

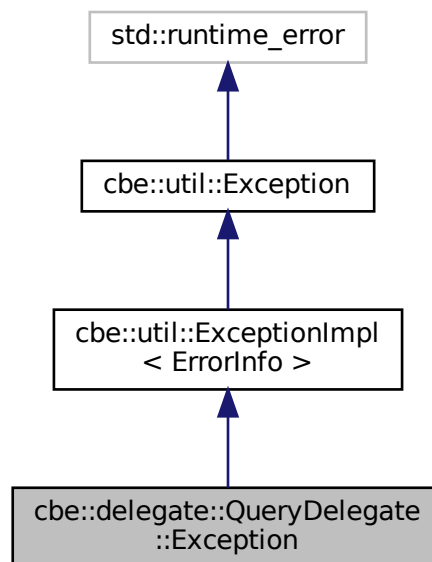
- `include/cbe/delegate/PublishDelegate.h`

## 11.99 cbe::delegate::QueryDelegate::Exception Struct Reference

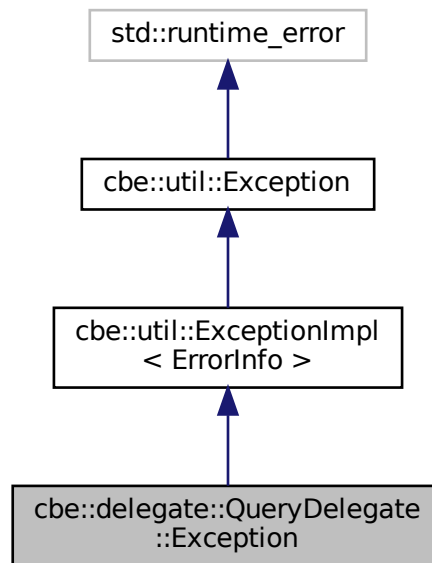
exception thrown by

```
#include <QueryDelegate.h>
```

Inheritance diagram for cbe::delegate::QueryDelegate::Exception:



Collaboration diagram for cbe::delegate::QueryDelegate::Exception:



## Additional Inherited Members

### 11.99.1 Detailed Description

exception thrown by

- [CloudBackend::queryWithPath\(\)](#), and its overloads
- [CloudBackend::query\(\)](#), and its overloads
- [CloudBackend::search\(\)](#), and its overloads
- [Container::queryWithPath\(\)](#)
- [Container::query\(\)](#), and its overloads
- [Container::search\(\)](#), and its overloads

if the request fails.

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

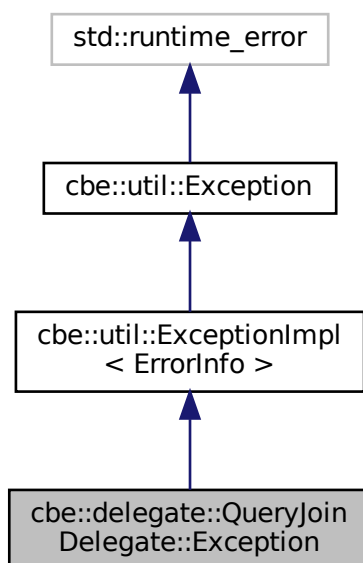
- `include/cbe/delegate/QueryDelegate.h`

## 11.100 cbe::delegate::QueryJoinDelegate::Exception Struct Reference

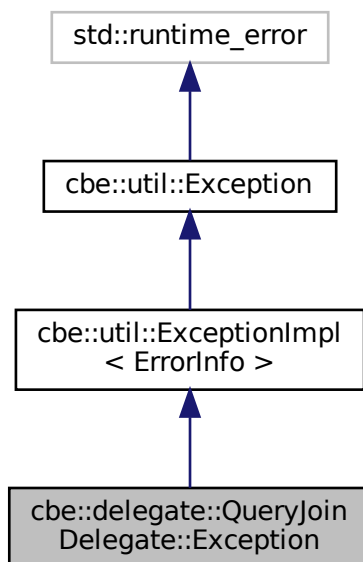
exception thrown by

```
#include <QueryJoinDelegate.h>
```

Inheritance diagram for cbe::delegate::QueryJoinDelegate::Exception:



Collaboration diagram for cbe::delegate::QueryJoinDelegate::Exception:



## Additional Inherited Members

### 11.100.1 Detailed Description

exception thrown by

- [CloudBackend::query\(\)](#), and its overloads
- [Container::query\(\)](#), and its overloads

if a query fails.

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

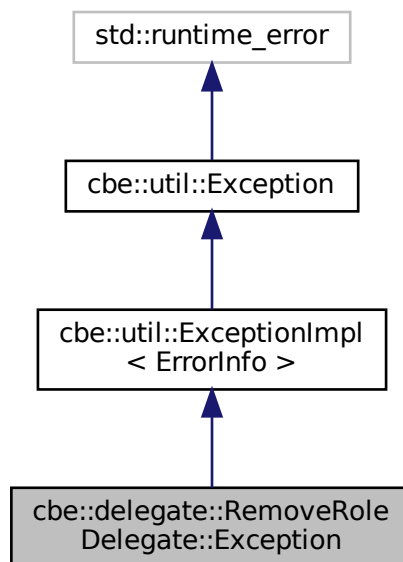
- include/cbe/delegate/QueryJoinDelegate.h

## 11.101 cbe::delegate::RemoveRoleDelegate::Exception Struct Reference

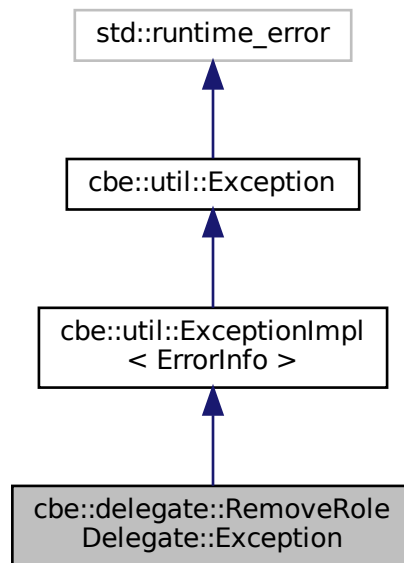
exception thrown by `cbe::Role::removeRole()` if the request fails.

```
#include <RemoveRoleDelegate.h>
```

Inheritance diagram for `cbe::delegate::RemoveRoleDelegate::Exception`:



Collaboration diagram for `cbe::delegate::RemoveRoleDelegate::Exception`:



## Additional Inherited Members

### 11.101.1 Detailed Description

exception thrown by `cbe::Role::removeRole()` if the request fails.

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

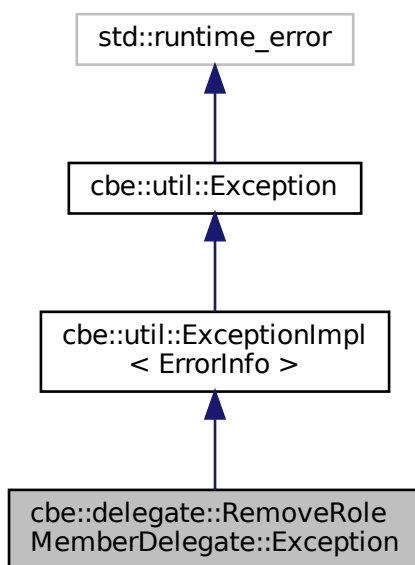
- `include/cbe/delegate/RemoveRoleDelegate.h`

### 11.102 `cbe::delegate::RemoveRoleMemberDelegate::Exception` Struct Reference

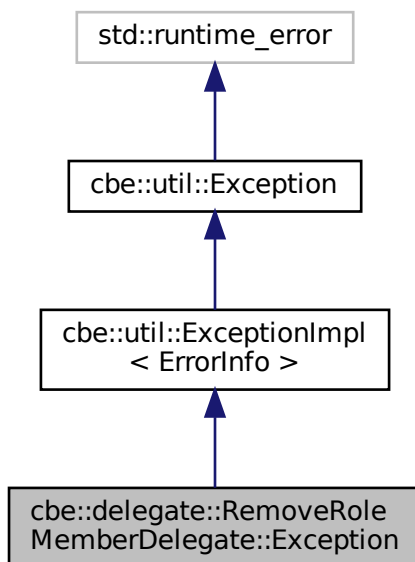
exception thrown by `cbe::Group::RemoveMember()` if the request fails.

`#include <RemoveRoleMemberDelegate.h>`

Inheritance diagram for cbe::delegate::RemoveRoleMemberDelegate::Exception:



Collaboration diagram for cbe::delegate::RemoveRoleMemberDelegate::Exception:



## Additional Inherited Members

### 11.102.1 Detailed Description

exception thrown by `cbe::Group::RemoveMember()` if the request fails.  
The documentation for this struct was generated from the following file:

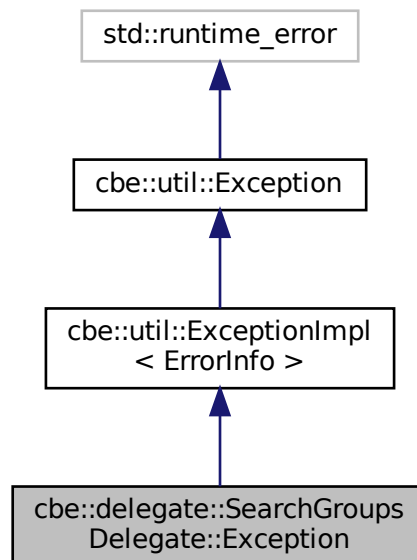
- `include/cbe/delegate/RemoveRoleMemberDelegate.h`

## 11.103 `cbe::delegate::SearchGroupsDelegate::Exception` Struct Reference

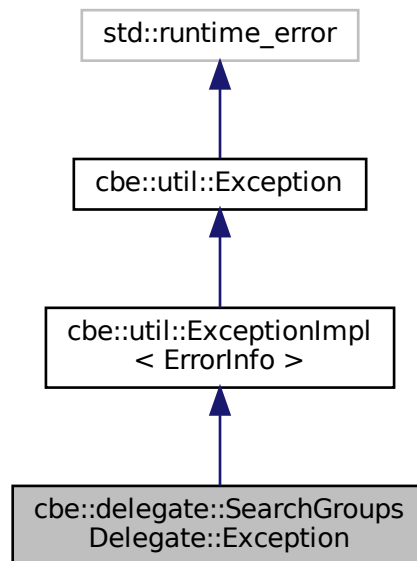
exception thrown by `cbe::GroupManager::searchGroups()` if the request fails.

```
#include <SearchGroupsDelegate.h>
```

Inheritance diagram for `cbe::delegate::SearchGroupsDelegate::Exception`:



Collaboration diagram for cbe::delegate::SearchGroupsDelegate::Exception:



## Additional Inherited Members

### 11.103.1 Detailed Description

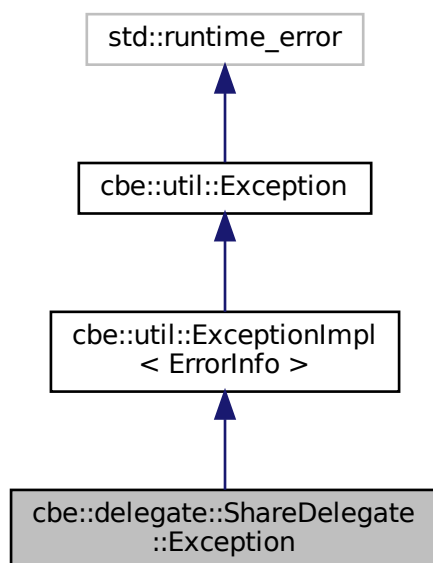
exception thrown by `cbe::GroupManager::searchGroups()` if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/SearchGroupsDelegate.h`

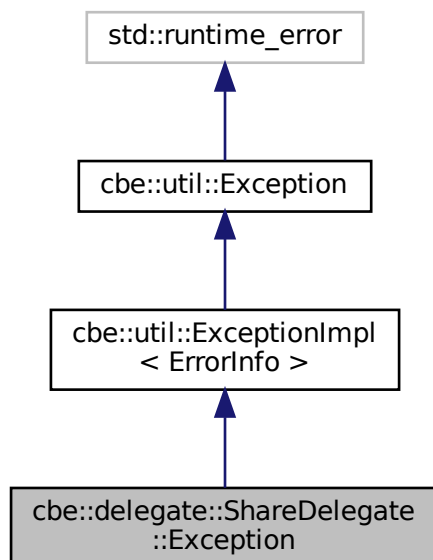
## 11.104 cbe::delegate::ShareDelegate::Exception Struct Reference

exception thrown by  
`#include <ShareDelegate.h>`

Inheritance diagram for cbe::delegate::ShareDelegate::Exception:



Collaboration diagram for cbe::delegate::ShareDelegate::Exception:



## Additional Inherited Members

### 11.104.1 Detailed Description

exception thrown by

- [cbe::Container::share\(cbe::UserId,std::string\)](#)
- [cbe::Object::share\(cbe::UserId,std::string\)](#)

if the request fails.

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

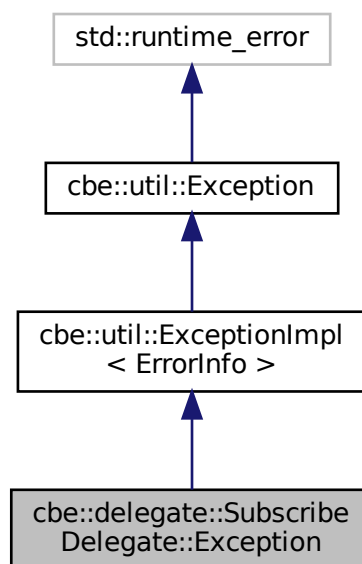
- include/cbe/delegate/ShareDelegate.h

## 11.105 cbe::delegate::SubscribeDelegate::Exception Struct Reference

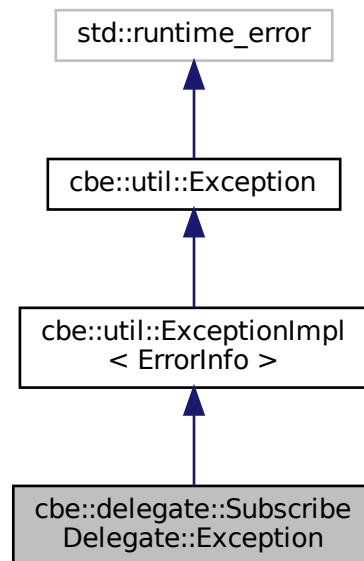
exception thrown by [cbe::SubscribeManager::subscribe\(\)](#) and its overloads if the request fails.

```
#include <SubscribeDelegate.h>
```

Inheritance diagram for cbe::delegate::SubscribeDelegate::Exception:



Collaboration diagram for `cbe::delegate::SubscribeDelegate::Exception`:



## Additional Inherited Members

### 11.105.1 Detailed Description

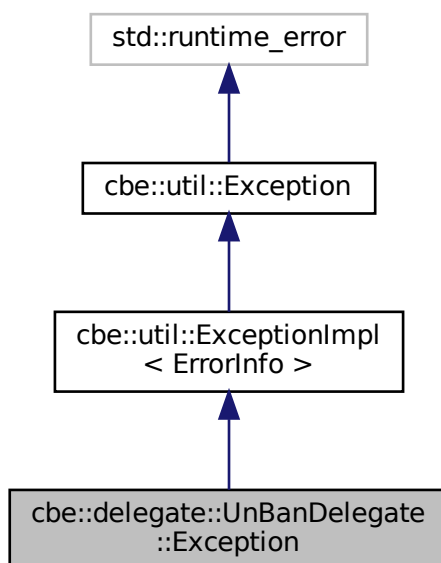
exception thrown by `cbe::SubscribeManager::subscribe()` and its overloads if the request fails.  
The documentation for this struct was generated from the following file:

- `include/cbe/delegate/SubscribeDelegate.h`

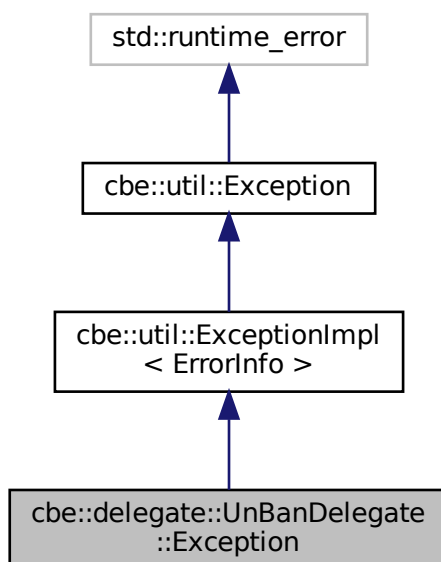
### 11.106 `cbe::delegate::UnBanDelegate::Exception` Struct Reference

exception thrown by `cbe::Member::unBan()` if the request fails.  
`#include <UnBanDelegate.h>`

Inheritance diagram for cbe::delegate::UnBanDelegate::Exception:



Collaboration diagram for cbe::delegate::UnBanDelegate::Exception:



## Additional Inherited Members

### 11.106.1 Detailed Description

exception thrown by [cbe::Member::unBan\(\)](#) if the request fails.

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

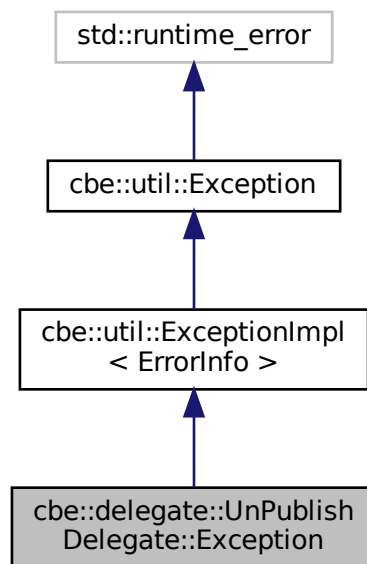
- include/cbe/delegate/UnBanDelegate.h

## 11.107 cbe::delegate::UnPublishDelegate::Exception Struct Reference

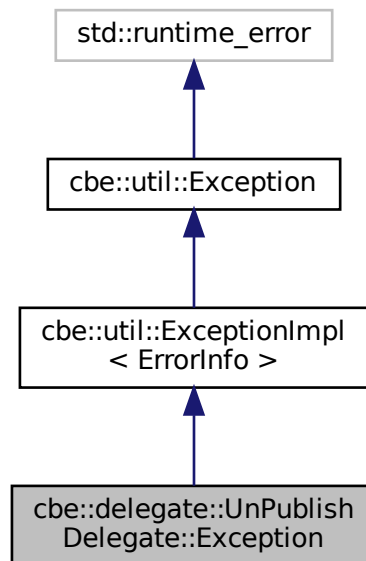
exception thrown by

```
#include <UnPublishDelegate.h>
```

Inheritance diagram for cbe::delegate::UnPublishDelegate::Exception:



Collaboration diagram for cbe::delegate::UnPublishDelegate::Exception:



## Additional Inherited Members

### 11.107.1 Detailed Description

exception thrown by

- [Container::unPublish\(\)](#)
- [Object::unPublish\(\)](#)

if the request fails.

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

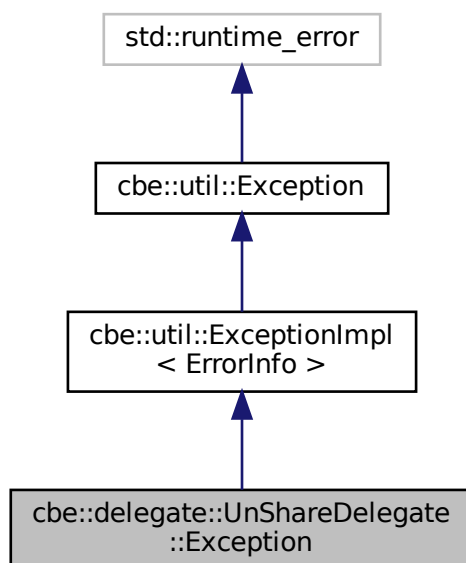
- `include/cbe/delegate/UnPublishDelegate.h`

## 11.108 cbe::delegate::UnShareDelegate::Exception Struct Reference

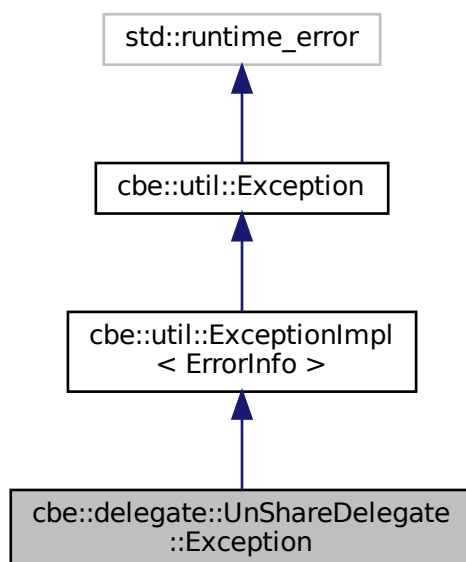
exception thrown by

```
#include <UnShareDelegate.h>
```

Inheritance diagram for cbe::delegate::UnShareDelegate::Exception:



Collaboration diagram for cbe::delegate::UnShareDelegate::Exception:



## Additional Inherited Members

### 11.108.1 Detailed Description

exception thrown by

- [Container::unShare\(\)](#)
- [Object::unShare\(\)](#)

if the request fails.

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

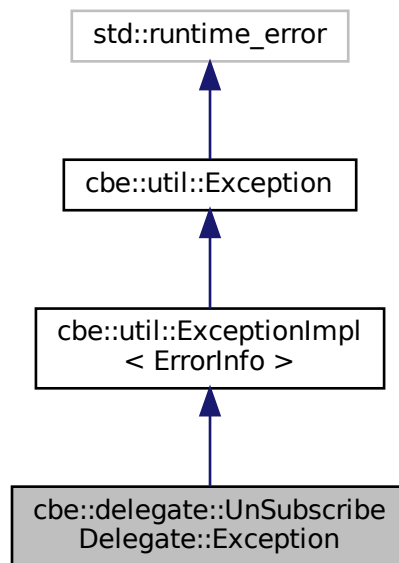
- include/cbe/delegate/UnShareDelegate.h

## 11.109 cbe::delegate::UnSubscribeDelegate::Exception Struct Reference

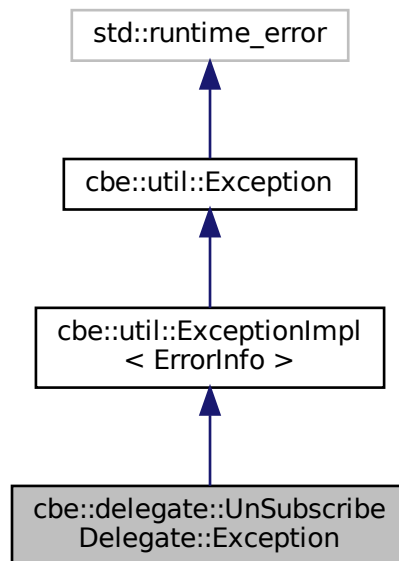
exception thrown by

```
#include <UnSubscribeDelegate.h>
```

Inheritance diagram for cbe::delegate::UnSubscribeDelegate::Exception:



Collaboration diagram for `cbe::delegate::UnSubscribeDelegate::Exception`:



## Additional Inherited Members

### 11.109.1 Detailed Description

exception thrown by

- [Container::unSubscribe\(\)](#)
- [Object::unSubscribe\(\)](#)

if the request fails.

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

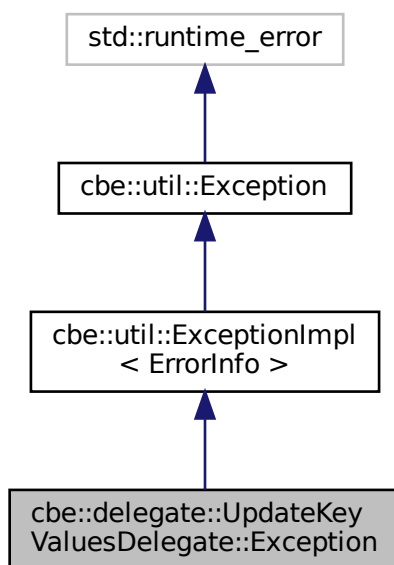
- `include/cbe/delegate/UnSubscribeDelegate.h`

### 11.110 `cbe::delegate::UpdateKeyValuesDelegate::Exception` Struct Reference

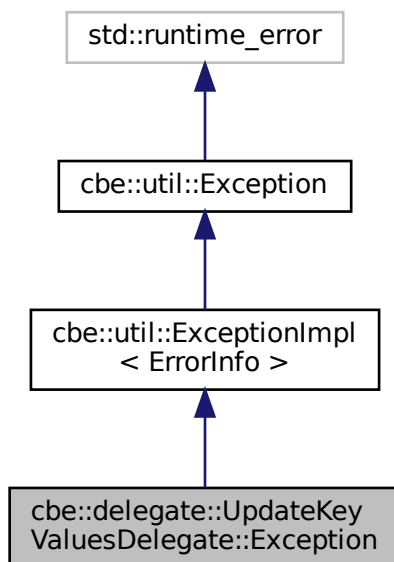
exception thrown by [cbe::Object::updateKeyValues\(\)](#) if the request fails.

```
#include <UpdateKeyValuesDelegate.h>
```

Inheritance diagram for cbe::delegate::UpdateKeyValuesDelegate::Exception:



Collaboration diagram for cbe::delegate::UpdateKeyValuesDelegate::Exception:



## Additional Inherited Members

### 11.110.1 Detailed Description

exception thrown by [cbe::Object::updateKeyValues\(\)](#) if the request fails.  
The documentation for this struct was generated from the following file:

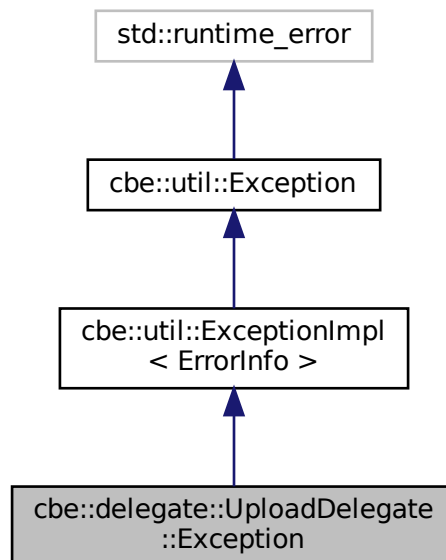
- `include/cbe/delegate/UpdateKeyValuesDelegate.h`

### 11.111 cbe::delegate::UploadDelegate::Exception Struct Reference

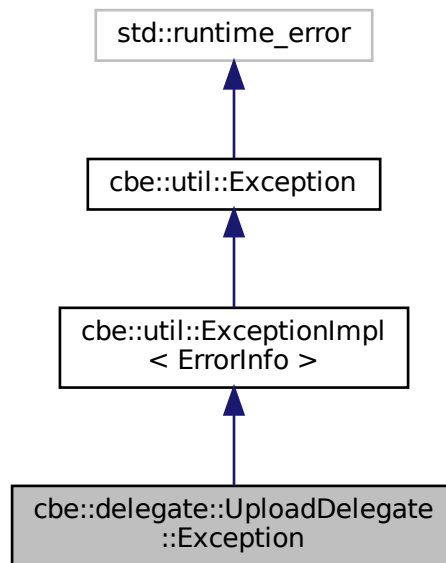
exception thrown by [cbe::Container::upload\(const std::string&,const std::string&\)](#) and its overloads if the request fails.

```
#include <UploadDelegate.h>
```

Inheritance diagram for `cbe::delegate::UploadDelegate::Exception`:



Collaboration diagram for cbe::delegate::UploadDelegate::Exception:



## Additional Inherited Members

### 11.111.1 Detailed Description

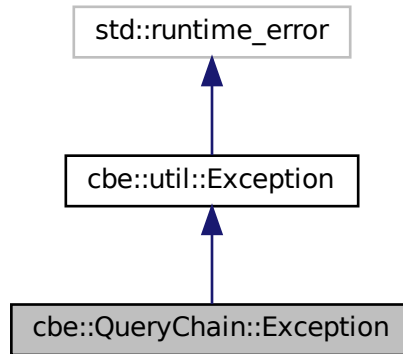
exception thrown by `cbe::Container::upload(const std::string&,const std::string&)` and its overloads if the request fails.

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

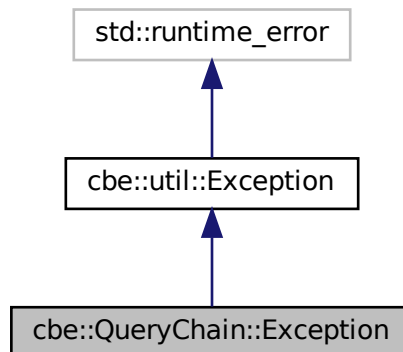
- `include/cbe/delegate/UploadDelegate.h`

### 11.112 cbe::QueryChain::Exception Struct Reference

Inheritance diagram for cbe::QueryChain::Exception:



Collaboration diagram for cbe::QueryChain::Exception:



#### Additional Inherited Members

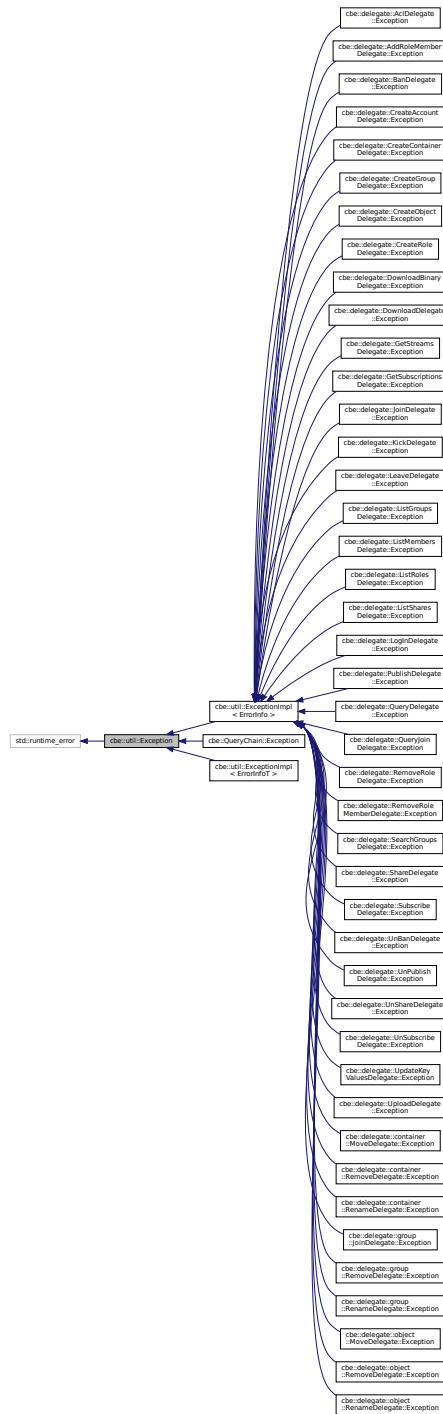
The documentation for this struct was generated from the following file:

- include/cbe/QueryChain.h

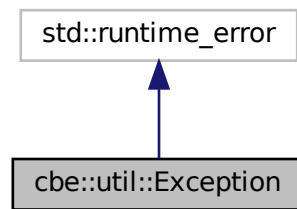
### 11.113 cbe::util::Exception Struct Reference

```
#include <Exception.h>
```

Inheritance diagram for cbe::util::Exception:



Collaboration diagram for `cbe::util::Exception`:



## Public Member Functions

- `template<typename... Ts>`  
**Exception** (Ts &&... args)
- `std::string typeAsString () const`
- `template<class CbeExceptionT >`  
`const CbeExceptionT * derivedCbeException () const`  
*Examines whether current exception is of type `CbeExceptionT`.*

### 11.113.1 Detailed Description

X

### 11.113.2 Member Function Documentation

#### 11.113.2.1 `derivedCbeException()`

```
template<class CbeExceptionT >
const CbeExceptionT* cbe::util::Exception::derivedCbeException () const [inline]
```

Examines whether current exception is of type `CbeExceptionT`.

#### Template Parameters

|                            |                                                                              |
|----------------------------|------------------------------------------------------------------------------|
| <code>CbeExceptionT</code> | Subtype of <code><a href="#">cbe::util::Exception</a></code> to be examined. |
|----------------------------|------------------------------------------------------------------------------|

#### Returns

Pointer to the requested exception, nullptr implies no such subclass type.

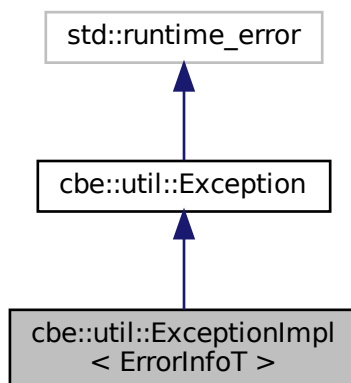
The documentation for this struct was generated from the following file:

- `include/cbe/util/Exception.h`

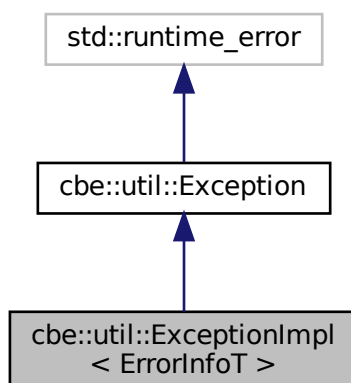
## 11.114 `cbe::util::ExceptionImpl< ErrorInfoT >` Struct Template Reference

```
#include <Exception.h>
```

Inheritance diagram for cbe::util::ExceptionImpl< ErrorInfoT >:



Collaboration diagram for cbe::util::ExceptionImpl< ErrorInfoT >:



## Public Types

- using **Base** = [ExceptionImpl](#)< ErrorInfoT >

## Public Member Functions

- **ExceptionImpl** (ErrorInfoT &&errorInfo)

## Public Attributes

- const ErrorInfoT **errorInfo**

### 11.114.1 Detailed Description

```
template<class ErrorInfoT>
struct cbe::util::ExceptionImpl< ErrorInfoT >
```

XX

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

- include/cbe/util/Exception.h

## 11.115 cbe::Filter Class Reference

Use to select [Item](#) that meets specific criterias when doing a query.

```
#include <Filter.h>
```

### Public Member Functions

- **Filter** (const [Filter](#) &rh)
- **Filter** ([Filter](#) &&rh)
- [Filter](#) & **operator=** (const [Filter](#) &rh)
- [Filter](#) & **operator=** ([Filter](#) &&rh)
- [cbe::ItemType](#) **getDataType** () const  
*Returns the requested data type.*
- std::string **getQuery** () const  
*Returns the query string that was set on the filter.*
- bool **getAscending** () const  
*Returns the settings on how the data was sorted and displayed.*
- bool **getDeleted** () const  
*Determines if deleted objects in the bin are included.*
- bool **getContainerFirst** () const  
*Determines if containers should be sorted before objects.*
- bool **isJoinedResult** () const  
*Determines if the resultset comes from a joined query.*
- std::uint32\_t **getOffset** () const  
*Determines the offset of items to be skipped.*
- std::uint32\_t **getCount** () const  
*Returns the size of the resultset of the query.*
- bool **getByPassCache** () const  
*Returns if skipping the cache was used or not.*
- [cbe::FilterOrder](#) **getItemOrder** () const  
*Returns the order the query was sorted.*
- [cbe::Container](#) **container** ()  
*Returns the parent container that was queried, if it is in the cache. This is after a query was done, not before.*
- [cbe::Filter](#) & **setDataType** ([cbe::ItemType](#) itemType)  
*Set which datatypes to query for.*
- [cbe::Filter](#) & **setQuery** (std::string query)  
*Set the query string.*
- [cbe::Filter](#) & **setAscending** (bool ascending)  
*Sets the order, ascending, in which data should be displayed.*
- [cbe::Filter](#) & **setDeleted** (bool deleted)  
*Whether to include deleted items.*
- [cbe::Filter](#) & **setContainerFirst** (bool containerFirst)  
*If [Container](#) should be displayed before [Object](#).*
- [cbe::Filter](#) & **setOffset** (std::uint32\_t offset)

*Set the offset for paging.*

- [cbe::Filter](#) & [setCount](#) (std::uint32\_t count)

*Set the maximum resultset.*

- [cbe::Filter](#) & [setItemOrder](#) ([cbe::FilterOrder](#) order)

*Set the order of how data will be shown by the enum of FilterOrder.*

- [cbe::Filter](#) & [setByPassCache](#) (bool byPassCache)

*Set to ignore the local SDK cache.*

## Friends

- class **CloudBackend**
- class **Container**
- class **QueryChain**
- class **QueryResult**
- std::ostream & **operator**<< (std::ostream &os, const [Filter](#) &filter)
- CBI::ItemDelegatePtr **delegate::createCbiQueryDelegate** ([delegate::QueryDelegatePtr](#), [cbe::util::Context](#) &&)
- CBI::ItemDelegatePtr **delegate::createCbiQueryDelegate** ([delegate::QueryJoinDelegatePtr](#), [cbe::util::Context](#) &&)

### 11.115.1 Detailed Description

Use to select [Item](#) that meets specific criterias when doing a query.

Filtering on metadata KeyValues can be done with [setQuery\(\)](#) and only to query for [Object](#), not [Container](#).

### 11.115.2 Member Function Documentation

#### 11.115.2.1 getDataType()

```
cbe::ItemType cbe::Filter::getDataType () const
```

Returns the requested data type.

Such as ItemType e.g.,

- [cbe::ItemType::Container](#)
- [cbe::ItemType::Object](#)
- [cbe::ItemType::Group](#)

#### 11.115.2.2 getQuery()

```
std::string cbe::Filter::getQuery () const
```

Returns the query string that was set on the filter.

Query string e.g.,

- name:Abc\*
- age:100
- price<200
- size>40

### 11.115.2.3 getOffset()

```
std::uint32_t cbe::Filter::getOffset () const
```

Determines the offset of items to be skipped.

Returns the offset of items in the beginning of total resultset to be skipped, that was used for the filter in the query.

### 11.115.2.4 getItemOrder()

```
cbe::FilterOrder cbe::Filter::getItemOrder () const
```

Returns the order the query was sorted.

Check enum `FilterOrder` to see options for sorting.

### 11.115.2.5 setDataType()

```
cbe::Filter& cbe::Filter::setDataType (
 cbe::ItemType itemType)
```

Set which datatypes to query for.

#### Parameters

|                 |                                       |
|-----------------|---------------------------------------|
| <i>itemType</i> | as defined in <a href="#">Types.h</a> |
|-----------------|---------------------------------------|

### 11.115.2.6 setQuery()

```
cbe::Filter& cbe::Filter::setQuery (
 std::string query)
```

Set the query string.

E.g., `firstname:*` (would search for all objects with the metadata key labeled `firstname`).

Search string e.g.,

- `name:Abc*`
- `age:100`
- `price<200`
- `size>40`

#### Note

If used with `rootContainer` id as the `dataId` to search in, the whole account will be searched.

#### Parameters

|              |                                                                                                                                                                                                                                                                         |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>query</i> | is a string of key tags or key:value pairs.<br>E.g. <code>Name: Country:Sweden Country:Norway</code><br>This will search for objects with key <code>Name</code> but any value and where key <code>Country</code> is either <code>Sweden</code> or <code>Norway</code> . |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

### 11.115.2.7 setAscending()

```
cbe::Filter& cbe::Filter::setAscending (
 bool ascending)
```

Sets the order, `ascending`, in which data should be displayed.

## Parameters

|                  |                                                                         |
|------------------|-------------------------------------------------------------------------|
| <i>ascending</i> | true = sorted in increasing order<br>false = sorted in decreasing order |
|------------------|-------------------------------------------------------------------------|

**11.115.2.8 setDeleted()**

```
cbe::Filter& cbe::Filter::setDeleted (
 bool deleted)
```

Whether to include deleted items.

## Parameters

|                |                                                     |
|----------------|-----------------------------------------------------|
| <i>deleted</i> | true = include items in the bin<br>default is false |
|----------------|-----------------------------------------------------|

**11.115.2.9 setContainerFirst()**

```
cbe::Filter& cbe::Filter::setContainerFirst (
 bool containerFirst)
```

If [Container](#) should be displayed before [Object](#).

## Parameters

|                       |                                                                                         |
|-----------------------|-----------------------------------------------------------------------------------------|
| <i>containerFirst</i> | true = sort containers at the top of the resultset.<br>default is to not sort for this. |
|-----------------------|-----------------------------------------------------------------------------------------|

**11.115.2.10 setOffset()**

```
cbe::Filter& cbe::Filter::setOffset (
 std::uint32_t offset)
```

Set the offset for paging.

Offset is the item offset where to start your resultset.

E.g.: There is already a query resultset of the first 100 items and to get the rest you put [setOffset\(\)](#) to 100 to get the next resultset from 101-.

## Parameters

|               |                                                             |
|---------------|-------------------------------------------------------------|
| <i>offset</i> | first item start number of the complete resultset to return |
|---------------|-------------------------------------------------------------|

**11.115.2.11 setCount()**

```
cbe::Filter& cbe::Filter::setCount (
 std::uint32_t count)
```

Set the maximum resultset.

Set the maximum number of items you want to get from a container.

E.g.: a container has 50 items but you only want the 10 first in ascending order then set ascending to true and setCount to 10.

## Parameters

|              |                                   |
|--------------|-----------------------------------|
| <i>count</i> | maximum number of items to return |
|--------------|-----------------------------------|

**11.115.2.12 setItemOrder()**

```
cbe::Filter& cbe::Filter::setItemOrder (
 cbe::FilterOrder order)
```

Set the order of how data will be shown by the enum of FilterOrder.  
E.g.: Title first, Relevance, published e.t.c.

## Parameters

|              |                                                     |
|--------------|-----------------------------------------------------|
| <i>order</i> | see the enum FilterOrder in <a href="#">Types.h</a> |
|--------------|-----------------------------------------------------|

**11.115.2.13 setByPassCache()**

```
cbe::Filter& cbe::Filter::setByPassCache (
 bool byPassCache)
```

Set to ignore the local SDK cache.  
Instead allways fetch the resultset from the cloud.

## Parameters

|                    |                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------|
| <i>byPassCache</i> | true = skip the SDK cache and ask the cloud.<br>default is to use the cache when available. |
|--------------------|---------------------------------------------------------------------------------------------|

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

- include/cbe/Filter.h

**11.116 cbe::delegate::GetStreamsDelegate Class Reference**

```
#include <GetStreamsDelegate.h>
```

**Classes**

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::Object::getStreams\(\)](#) if the request fails.*

**Public Types**

- using **Success** = [cbe::Streams](#)
- using **Error** = [delegate::Error](#)

**Public Member Functions**

- virtual void [onGetStreamsSuccess](#) ([cbe::Streams](#) &&streams)=0
- virtual void [onGetStreamsError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.116.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Object::getStreams\(\)](#)  
See [Example of retrieving the Stream attached to a cbe::Object with Object::getStreams\(\)](#)

### 11.116.2 Member Function Documentation

#### 11.116.2.1 onGetStreamsSuccess()

```
virtual void cbe::delegate::GetStreamsDelegate::onGetStreamsSuccess (
 cbe::Streams && streams) [pure virtual]
```

Called upon successful Object::GetStreams.

##### Parameters

|                |                                                                                  |
|----------------|----------------------------------------------------------------------------------|
| <i>streams</i> | The currently loaded <a href="#">stream(s) attached to</a> the requested object. |
|----------------|----------------------------------------------------------------------------------|

#### 11.116.2.2 onGetStreamsError()

```
virtual void cbe::delegate::GetStreamsDelegate::onGetStreamsError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/GetStreamsDelegate.h

## 11.117 cbe::delegate::GetSubscriptionsDelegate Class Reference

```
#include <GetSubscriptionsDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::SubscribeManager::getSubscriptions\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::QueryResult](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onGetSubscriptionsSuccess](#) ([cbe::QueryResult](#) &&object)=0
- virtual void [onGetSubscriptionsError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.117.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::SubscribeManager::getSubscriptions](#)

## 11.117.2 Member Function Documentation

### 11.117.2.1 onGetSubscriptionsSuccess()

```
virtual void cbe::delegate::GetSubscriptionsDelegate::onGetSubscriptionsSuccess (
 cbe::QueryResult && object) [pure virtual]
```

Called upon successful GetSubscriptions.

#### Parameters

|               |                                            |
|---------------|--------------------------------------------|
| <i>object</i> | Instance of object that is being streamed. |
|---------------|--------------------------------------------|

### 11.117.2.2 onGetSubscriptionsError()

```
virtual void cbe::delegate::GetSubscriptionsDelegate::onGetSubscriptionsError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/GetSubscriptionsDelegate.h

## 11.118 cbe::Group Class Reference

A group of members.

```
#include <Group.h>
```

### Public Types

- using **Requests** = std::vector< [cbe::Request](#) >
- using [CreateGroupDelegatePtr](#) = [delegate::CreateGroupDelegatePtr](#)
- using [CreateGroupException](#) = [delegate::CreateGroupDelegate::Exception](#)
- using [CreateGroupError](#) = [delegate::CreateGroupDelegate::ErrorInfo](#)
- using [JoinDelegatePtr](#) = [delegate::group::JoinDelegatePtr](#)
- using [JoinException](#) = [delegate::group::JoinDelegate::Exception](#)
- using [JoinError](#) = [delegate::group::JoinDelegate::ErrorInfo](#)
- using [LeaveDelegatePtr](#) = [delegate::LeaveDelegatePtr](#)
- using [LeaveException](#) = [delegate::LeaveDelegate::Exception](#)
- using [LeaveError](#) = [delegate::LeaveDelegate::ErrorInfo](#)
- using [RemoveDelegatePtr](#) = [delegate::group::RemoveDelegatePtr](#)
- using [RemoveException](#) = [delegate::group::RemoveDelegate::Exception](#)
- using [RemoveError](#) = [delegate::group::RemoveDelegate::ErrorInfo](#)
- using [RenameDelegatePtr](#) = [delegate::group::RenameDelegatePtr](#)
- using [RenameException](#) = [delegate::group::RenameDelegate::Exception](#)
- using [RenameError](#) = [delegate::group::RenameDelegate::ErrorInfo](#)
- using [ListMembersDelegatePtr](#) = [delegate::ListMembersDelegatePtr](#)
- using [ListMembersException](#) = [delegate::ListMembersDelegate::Exception](#)
- using [ListMembersError](#) = [delegate::ListMembersDelegate::ErrorInfo](#)
- using [ListBannedMembersDelegatePtr](#) = [delegate::ListMembersDelegatePtr](#)
- using [ListBannedMembersException](#) = [delegate::ListMembersDelegate::Exception](#)
- using [ListBannedMembersError](#) = [delegate::ListMembersDelegate::ErrorInfo](#)
- using [ListRolesDelegatePtr](#) = [delegate::ListRolesDelegatePtr](#)
- using [ListRolesException](#) = [delegate::ListRolesDelegate::Exception](#)

- using `ListRolesError` = `delegate::ListRolesDelegate::ErrorInfo`
- using `CreateRoleDelegatePtr` = `delegate::CreateRoleDelegatePtr`
- using `CreateRoleException` = `delegate::CreateRoleDelegate::Exception`
- using `CreateRoleError` = `delegate::CreateRoleDelegate::ErrorInfo`
- using `RemoveRoleDelegatePtr` = `delegate::RemoveRoleDelegatePtr`
- using `RemoveRoleException` = `delegate::RemoveRoleDelegate::Exception`
- using `RemoveRoleError` = `delegate::RemoveRoleDelegate::ErrorInfo`

## Public Member Functions

- `std::string name () const`
- `cbe::GroupId id () const`
- `cbe::GroupId parentId () const`
- `cbe::Container groupContainer () const`
- `cbe::Visibility getVisibility () const`
- `bool joined () const`
- Requests `requests () const`
- `cbe::Group createGroup (std::string name, std::string memberAlias, cbe::Visibility visibility, CreateGroupDelegatePtr delegate)`  
*Create a new Group.*
- `cbe::Group createGroup (std::string name, std::string memberAlias, cbe::Visibility visibility)`  
*Synchronous [exception] creates a new group **Synchronous** version of `createGroup(name, memberAlias, delegate, visibility = cbe::Visibility::Public, CreateGroupDelegatePtr)` , and **throws an exception**, `CreateGroupException`, in case of a failed call.*  
*See `createGroup(CreateGroupDelegatePtr)`*
- `cbe::util::Optional< cbe::Group > createGroup (std::string name, std::string memberAlias, cbe::Visibility visibility, CreateGroupError &error)`
- `void join (std::string alias, cbe::Visibility memberVisibility, std::string applicationComment, JoinDelegatePtr delegate)`  
*Synchronous [exception] Ask to join a group. In this first version All members will be Public, meaning visible for other member inside the group. All groups will also be open so all join requests should be accepted directly.*
- `cbe::Group join (std::string alias, cbe::Visibility memberVisibility, std::string applicationComment)`  
*Synchronous [exception] Ask to join a group. **Synchronous** version of `join(std::string, cbe::Visibility, std::string)` , and **throws an exception**, `JoinException`, in case of a failed call.*  
*See `join(JoinDelegatePtr)`*
- `cbe::util::Optional< cbe::Group > join (std::string alias, cbe::Visibility memberVisibility, std::string applicationComment, JoinError &error)`  
*Synchronous [non-throwing] Ask to join a group. **Synchronous** version of `join(alias, memberVisibility, applicationComment, JoinError&,JoinDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*  
*See `join(alias, memberVisibility, applicationComment, JoinError&, JoinDelegatePtr)`*
- `void leave (LeaveDelegatePtr delegate)`  
*Leave group.*
- `delegate::LeaveSuccess leave ()`  
*Synchronous [exception] leaves the specified group. **Synchronous** version of `leave(LeaveDelegatePtr)` , and **throws an exception**, `LeaveException`, in case of a failed call.*  
*See `leave(LeaveDelegatePtr)`*
- `cbe::util::Optional< delegate::LeaveSuccess > leave (LeaveError &error)`  
*Synchronous [non-throwing] leaves the specified group. **Synchronous** version of `leave(LeaveDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*  
*See `leave(LeaveDelegatePtr)`*
- `void remove (RemoveDelegatePtr delegate)`  
*Remove group.*
- `delegate::group::RemoveSuccess remove ()`

*Synchronous [exception]* removes the specified group. **Synchronous** version of [remove\(RemoveDelegatePtr\)](#) , and **throws an exception**, [RemoveException](#), in case of a failed call.

See [remove\(RemoveDelegatePtr\)](#)

- [cbe::util::Optional](#) < [delegate::group::RemoveSuccess](#) > [remove](#) ([RemoveError](#) &error)

*Synchronous [non-throwing]* removes the specified group. **Synchronous** version of [remove\(RemoveDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [remove\(RemoveDelegatePtr\)](#)

- void [rename](#) (std::string newName, [RenameDelegatePtr](#) delegate)

Rename the [Group](#); group id does not change.

- [delegate::group::RenameSuccess](#) [rename](#) (std::string newName)

*Synchronous [exception]* renames a group **Synchronous** version of [rename\(std::string ,RenameDelegatePtr\)](#) , and **throws an exception**, [RenameException](#), in case of a failed call.

See [rename\(RenameDelegatePtr\)](#)

- [cbe::util::Optional](#) < [delegate::group::RenameSuccess](#) > [rename](#) (std::string newName, [RenameError](#) &error)

*Synchronous [non-throwing]* renames a group **Synchronous** version of [rename\(newName,RenameDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [rename\(newName, RenameDelegatePtr\)](#)

- void [listMembers](#) ([ListMembersDelegatePtr](#) delegate)

List all members in the group.

- [delegate::ListMembersDelegate::Members](#) [listMembers](#) ()

*Synchronous [exception]* list members of the specified group. **Synchronous** version of [listMembers\(ListMembersDelegatePtr\)](#) , and **throws an exception**, [ListMembersException](#), in case of a failed call.

See [listMembers\(ListMembersDelegatePtr\)](#)

- [cbe::util::Optional](#) < [delegate::ListMembersDelegate::Members](#) > [listMembers](#) ([ListMembersError](#) &error)

*Synchronous [non-throwing]* list members of the specified group. **Synchronous** version of [listMembers\(ListMembersDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listMembers\(ListMembersDelegatePtr\)](#)

- void [listBannedMembers](#) ([ListBannedMembersDelegatePtr](#) delegate)

Lists all banned former members, or users.

- std::vector< [cbe::Member](#) > [listBannedMembers](#) ()

*Synchronous [exception]* list the banned members of the specified group. **Synchronous** version of [listBannedMembers\(ListMembersDelegatePtr\)](#) , and **throws an exception**, [ListBannedMembersException](#), in case of a failed call.

See [listBannedMembers\(ListMembersDelegatePtr\)](#)

- [cbe::util::Optional](#) < std::vector< [cbe::Member](#) > > [listBannedMembers](#) ([ListBannedMembersError](#) &error)

*Synchronous [non-throwing]* list the banned members of the specified group. **Synchronous** version of [listBannedMembers\(ListMembersDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listBannedMembers\(ListMembersDelegatePtr\)](#)

- void [listRoles](#) ([ListRolesDelegatePtr](#) delegate)

Lists all roles in the group.

- [delegate::ListRolesDelegate::Roles](#) [listRoles](#) ()

*Synchronous [exception]* list roles of a group. **Synchronous** version of [listRoles\(ListRolesDelegatePtr\)](#) , and **throws an exception**, [ListRolesException](#), in case of a failed call.

See [listRoles\(ListRolesDelegatePtr\)](#)

- [cbe::util::Optional](#) < [delegate::ListRolesDelegate::Roles](#) > [listRoles](#) ([ListRolesError](#) &error)

*Synchronous [non-throwing]* list roles of a group. **Synchronous** version of [listRoles\(ListRolesDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listRoles\(ListRolesDelegatePtr\)](#)

- void [createRole](#) (std::string name, [CreateRoleDelegatePtr](#) delegate)

Creates a new role in the group.

- [cbe::Role createRole](#) (std::string name)
 

*Synchronous [exception] creates a new role in the specified group. **Synchronous** version of [createRole\(std::string, CreateRoleDelegatePtr\)](#), and **throws an exception**, [CreateRoleException](#), in case of a failed call.*  
*See [createRole\(CreateRoleDelegatePtr\)](#)*
- [cbe::util::Optional< cbe::Role > createRole](#) (std::string name, [CreateRoleError](#) &error)
 

*Synchronous [non-throwing] creates a new role in the specified group. **Synchronous** version of [createRole\(name, CreateRoleDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*  
*See [createRole\(name, CreateRoleDelegatePtr\)](#)*
- void [removeRole](#) ([RoleId](#) roleId, [RemoveRoleDelegatePtr](#) delegate)
- [cbe::RoleId removeRole](#) ([RoleId](#) roleId)
 

*Synchronous [exception] removes a role in the specified group. **Synchronous** version of [removeRole\(RoleId roleId, RemoveRoleDelegatePtr\)](#), and **throws an exception**, [RemoveRoleException](#), in case of a failed call.*  
*See [removeRole\(RemoveRoleDelegatePtr\)](#)*
- [cbe::util::Optional< cbe::RoleId > removeRole](#) ([RoleId](#) roleId, [RemoveRoleError](#) &error)
 

*Synchronous [non-throwing] removes a role in the specified group. **Synchronous** version of [removeRole\(roleId, RemoveRoleDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*  
*See [removeRole\(roleId, RemoveRoleDelegatePtr\)](#)*
- [Group](#) ([cbe::DefaultCtor](#))
 

*Default constructor.*
- [operator bool](#) () const
 

*Checks if the current instance is real.*

## Friends

- class [GroupManager](#)
- class [GroupQueryResult](#)

## 11.118.1 Detailed Description

A group of members.

## 11.118.2 Member Typedef Documentation

### 11.118.2.1 CreateGroupDelegatePtr

```
using cbe::Group::CreateGroupDelegatePtr = delegate::CreateGroupDelegatePtr
```

Pointer to CreateGroupDelegate that is passed into:

[Group::createGroup](#)(std::string,std::string,CreateGroupDelegatePtr,cbe::Visibility).

### 11.118.2.2 CreateGroupException

```
using cbe::Group::CreateGroupException = delegate::CreateGroupDelegate::Exception
```

See [delegate::group::CreateGroupDelegate::Exception](#)

### 11.118.2.3 CreateGroupError

```
using cbe::Group::CreateGroupError = delegate::CreateGroupDelegate::ErrorInfo
```

Forms the type of the `error` return parameter for the synchronous version of method [createGroup](#)()

See [delegate::CreateGroupDelegate::ErrorInfo](#)

### 11.118.2.4 JoinDelegatePtr

```
using cbe::Group::JoinDelegatePtr = delegate::group::JoinDelegatePtr
```

Pointer to [cbe::delegate::group::JoinDelegate](#) that is passed into asynchronous version of method [join](#)()

#### 11.118.2.5 JoinException

using `cbe::Group::JoinException = delegate::group::JoinDelegate::Exception`  
See `delegate::group::JoinDelegate::Exception`

#### 11.118.2.6 JoinError

using `cbe::Group::JoinError = delegate::group::JoinDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `join()`  
See `delegate::JoinDelegate::ErrorInfo`

#### 11.118.2.7 LeaveDelegatePtr

using `cbe::Group::LeaveDelegatePtr = delegate::LeaveDelegatePtr`  
Pointer to `cbe::delegate::LeaveDelegate` that is passed into asynchronous version of method `leave()`

#### 11.118.2.8 LeaveException

using `cbe::Group::LeaveException = delegate::LeaveDelegate::Exception`  
See `delegate::group::LeaveDelegate::Exception`

#### 11.118.2.9 LeaveError

using `cbe::Group::LeaveError = delegate::LeaveDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `leave()`  
See `delegate::LeaveDelegate::ErrorInfo`

#### 11.118.2.10 RemoveDelegatePtr

using `cbe::Group::RemoveDelegatePtr = delegate::group::RemoveDelegatePtr`  
Pointer to `cbe::delegate::RemoveDelegate` that is passed into asynchronous version of method `remove()`

#### 11.118.2.11 RemoveException

using `cbe::Group::RemoveException = delegate::group::RemoveDelegate::Exception`  
See `delegate::group::RemoveDelegate::Exception`

#### 11.118.2.12 RemoveError

using `cbe::Group::RemoveError = delegate::group::RemoveDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `remove()`  
See `delegate::group::RemoveDelegate::ErrorInfo`

#### 11.118.2.13 RenameDelegatePtr

using `cbe::Group::RenameDelegatePtr = delegate::group::RenameDelegatePtr`  
Pointer to `cbe::delegate::RenameDelegate` that is passed into asynchronous version of method `rename()`

#### 11.118.2.14 RenameException

using `cbe::Group::RenameException = delegate::group::RenameDelegate::Exception`  
See `delegate::group::RenameDelegate::Exception`

#### 11.118.2.15 RenameError

using `cbe::Group::RenameError = delegate::group::RenameDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `rename()`  
See `delegate::group::RenameDelegate::ErrorInfo`

#### 11.118.2.16 ListMembersDelegatePtr

using `cbe::Group::ListMembersDelegatePtr = delegate::ListMembersDelegatePtr`

Pointer to `cbe::delegate::ListMembersDelegate` that is passed into asynchronous version of method `listMembers()`

#### 11.118.2.17 ListMembersException

using `cbe::Group::ListMembersException = delegate::ListMembersDelegate::Exception`

See `delegate::object::ListMembersDelegate::Exception`

#### 11.118.2.18 ListMembersError

using `cbe::Group::ListMembersError = delegate::ListMembersDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `listMembers()`

See `delegate::ListMembersDelegate::ErrorInfo`

#### 11.118.2.19 ListBannedMembersDelegatePtr

using `cbe::Group::ListBannedMembersDelegatePtr = delegate::ListMembersDelegatePtr`

Pointer to `cbe::delegate::ListMembersDelegate` that is passed into asynchronous version of method `listBannedMembers()`

#### 11.118.2.20 ListBannedMembersException

using `cbe::Group::ListBannedMembersException = delegate::ListMembersDelegate::Exception`

See `delegate::object::ListMembersDelegate::Exception`

#### 11.118.2.21 ListBannedMembersError

using `cbe::Group::ListBannedMembersError = delegate::ListMembersDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `listBannedMembers()`

See `delegate::ListMembersDelegate::ErrorInfo`

#### 11.118.2.22 ListRolesDelegatePtr

using `cbe::Group::ListRolesDelegatePtr = delegate::ListRolesDelegatePtr`

Pointer to `cbe::delegate::ListRolesDelegate` that is passed into asynchronous version of method `listRoles()`

#### 11.118.2.23 ListRolesException

using `cbe::Group::ListRolesException = delegate::ListRolesDelegate::Exception`

See `delegate::object::ListRolesDelegate::Exception`

#### 11.118.2.24 ListRolesError

using `cbe::Group::ListRolesError = delegate::ListRolesDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `listRoles()`

See `delegate::ListRolesDelegate::ErrorInfo`

#### 11.118.2.25 CreateRoleDelegatePtr

using `cbe::Group::CreateRoleDelegatePtr = delegate::CreateRoleDelegatePtr`

Pointer to `cbe::delegate::CreateRoleDelegate` that is passed into asynchronous version of method `createRole()`

#### 11.118.2.26 CreateRoleException

using `cbe::Group::CreateRoleException = delegate::CreateRoleDelegate::Exception`

See `delegate::object::CreateRoleDelegate::Exception`

### 11.118.2.27 CreateRoleError

using `cbe::Group::CreateRoleError = delegate::CreateRoleDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `createRole()`

See [delegate::CreateRoleDelegate::ErrorInfo](#)

### 11.118.2.28 RemoveRoleDelegatePtr

using `cbe::Group::RemoveRoleDelegatePtr = delegate::RemoveRoleDelegatePtr`

Pointer to [cbe::delegate::RemoveRoleDelegate](#) that is passed into asynchronous version of method `removeRole()`

### 11.118.2.29 RemoveRoleException

using `cbe::Group::RemoveRoleException = delegate::RemoveRoleDelegate::Exception`

See `delegate::object::RemoveRoleDelegate::Exception`

### 11.118.2.30 RemoveRoleError

using `cbe::Group::RemoveRoleError = delegate::RemoveRoleDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `removeRole()`

See [delegate::RemoveRoleDelegate::ErrorInfo](#)

## 11.118.3 Constructor & Destructor Documentation

### 11.118.3.1 Group()

```
cbe::Group::Group (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the [operator bool\(\)](#) test

## 11.118.4 Member Function Documentation

### 11.118.4.1 name()

```
std::string cbe::Group::name () const
```

Returns the name of the group.

### 11.118.4.2 id()

```
cbe::GroupId cbe::Group::id () const
```

Returns the id number of the group.

### 11.118.4.3 parentId()

```
cbe::GroupId cbe::Group::parentId () const
```

[Group](#)'s Parent id number.

### 11.118.4.4 groupContainer()

```
cbe::Container cbe::Group::groupContainer () const
```

Every group has a drive/container where resources can be shared with the members of the group. Works like a shared container.

**11.118.4.5 getVisibility()**

```
cbe::Visibility cbe::Group::getVisibility () const
```

Visibility of the [Group](#), Public is searchable and Private is not. In this early version you can create Private groups but the ability to invite has been blocked. (work in progress)

See also

[Types.h](#) for [cbe::PublishVisibility](#) enum

**11.118.4.6 joined()**

```
bool cbe::Group::joined () const
```

Searched groups are obtained through the GroupQuery response. This list of groups have non-joined and joined groups. Already joined groups can be found on the `groupManager.groups()` and they are also cached. The joined bool is used to easily sort out groups in the GroupQuery.

**11.118.4.7 requests()**

```
Requests cbe::Group::requests () const
```

Requests to join the group. To retrieve the list call `listJoinRequests(cbe::GroupDelegatePtr delegate)` on the group object.

**11.118.4.8 createGroup() [1/3]**

```
cbe::Group cbe::Group::createGroup (
 std::string name,
 std::string memberAlias,
 cbe::Visibility visibility,
 CreateGroupDelegatePtr delegate)
```

Create a new [Group](#).

Note

Can only be used by Tenant admin/owners to create new groups.

**Parameters**

|                    |                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------|
| <i>name</i>        | of the group                                                                                         |
| <i>memberAlias</i> | another popular name                                                                                 |
| <i>delegate</i>    | Pointer to a <a href="#">delegate::CreateGroupDelegate</a> instance that is implemented by the user. |
| <i>visibility</i>  | default: public                                                                                      |

**Returns**

[cbe::Group](#)

See also

[Types.h](#) for [cbe::visibility](#) enum [cbe::Visibility::PublicGroup](#) or [cbe::Visibility::Private](#)

**11.118.4.9 createGroup() [2/3]**

```
cbe::Group cbe::Group::createGroup (
 std::string name,
```

```
std::string memberAlias,
cbe::Visibility visibility)
```

Synchronous [exception] creates a new group **Synchronous** version of `createGroup(name, memberAlias, delegate, visibility = cbe::Visibility::Public, CreateGroupDelegatePtr)` , and **throws an exception**, [CreateGroupException](#), in case of a failed call.

See `createGroup(CreateGroupDelegatePtr)`

#### Returns

Information about the `createGroup` object — if the call was successful.

See `cbe::delegate::CreateGroupSuccess`

#### Exceptions

|                                      |
|--------------------------------------|
| <a href="#">CreateGroupException</a> |
|--------------------------------------|

#### 11.118.4.10 createGroup() [3/3]

```
cbe::util::Optional<cbe::Group> cbe::Group::createGroup (
 std::string name,
 std::string memberAlias,
 cbe::Visibility visibility,
 CreateGroupError & error)
```

**Synchronous** version of `createGroup(name, memberAlias, delegate, visibility, CreateGroupDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `createGroup(name, memberAlias, delegate, visibility, CreateGroupDelegatePtr)`

#### Parameters

|     |       |                                                                                                                                                                                                                                     |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">CreateGroupError</a> object passed in will be populated with the error information. |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.118.4.11 join() [1/3]

```
void cbe::Group::join (
 std::string alias,
 cbe::Visibility memberVisibility,
 std::string applicationComment,
 JoinDelegatePtr delegate)
```

Synchronous [exception] Ask to join a group. In this first version All members will be Public, meaning visible for other member inside the group. All groups will also be open so all join requests should be accepted directly.

#### Note

This is completely different from the query chain join.

## Parameters

|                           |                                                                                                      |
|---------------------------|------------------------------------------------------------------------------------------------------|
| <i>alias</i>              | name                                                                                                 |
| <i>delegate</i>           | Pointer to a <a href="#">delegate::group::JoinDelegate</a> instance that is implemented by the user. |
| <i>memberVisibility</i>   | default: public                                                                                      |
| <i>applicationComment</i> | message to the owner regarding the application to join                                               |

## 11.118.4.12 join() [2/3]

```
cbe::Group cbe::Group::join (
 std::string alias,
 cbe::Visibility memberVisibility,
 std::string applicationComment)
```

Synchronous [exception] Ask to join a group. **Synchronous** version of [join\(std::string, cbe::Visibility, std::string\)](#) , and **throws an exception**, [JoinException](#), in case of a failed call.

See [join\(JoinDelegatePtr\)](#)

## Returns

Information about the Join — if the call was successful.

See [cbe::delegate::JoinSuccess](#)

## Exceptions

|                               |  |
|-------------------------------|--|
| <a href="#">JoinException</a> |  |
|-------------------------------|--|

## 11.118.4.13 join() [3/3]

```
cbe::util::Optional<cbe::Group> cbe::Group::join (
 std::string alias,
 cbe::Visibility memberVisibility,
 std::string applicationComment,
 JoinError & error)
```

Synchronous [non-throwing] Ask to join a group. **Synchronous** version of [join\(alias, memberVisibility, applicationComment, JoinError&,JoinDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [join\(alias, memberVisibility, applicationComment, JoinError&, JoinDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                              |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">JoinError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

## 11.118.4.14 leave() [1/3]

```
void cbe::Group::leave (
```

```
LeaveDelegatePtr delegate)
```

Leave group.

#### Parameters

|                 |                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::LeaveDelegate</a> instance that is implemented by the user. |
|-----------------|------------------------------------------------------------------------------------------------|

#### 11.118.4.15 leave() [2/3]

```
delegate::LeaveSuccess cbe::Group::leave ()
```

Synchronous [exception] leaves the specified group. **Synchronous** version of [leave\(LeaveDelegatePtr\)](#) , and **throws an exception**, [LeaveException](#), in case of a failed call.

See [leave\(LeaveDelegatePtr\)](#)

#### Returns

Information about the leave — if the call was successful.

See [cbe::delegate::LeaveSuccess](#)

#### Exceptions

|                                |  |
|--------------------------------|--|
| <a href="#">LeaveException</a> |  |
|--------------------------------|--|

#### 11.118.4.16 leave() [3/3]

```
cbe::util::Optional<delegate::LeaveSuccess> cbe::Group::leave (
 LeaveError & error)
```

Synchronous [non-throwing] leaves the specified group. **Synchronous** version of [leave\(LeaveDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [leave\(LeaveDelegatePtr\)](#)

#### Parameters

|            |              |                                                                                                                                                                                                                               |
|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">LeaveError</a> object passed in will be populated with the error information. |
|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.118.4.17 remove() [1/3]

```
void cbe::Group::remove (
 RemoveDelegatePtr delegate)
```

Remove group.

#### Note

This is exclusive for Tenant group owners.

## Parameters

|                 |                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::group::RemoveDelegate</a> instance that is implemented by the user. |
|-----------------|--------------------------------------------------------------------------------------------------------|

**11.118.4.18 remove()** [2/3]

```
delegate::group::RemoveSuccess cbe::Group::remove ()
```

Synchronous [exception] removes the specified group. **Synchronous** version of [remove\(RemoveDelegatePtr\)](#) , and **throws an exception**, [RemoveException](#), in case of a failed call.

See [remove\(RemoveDelegatePtr\)](#)

## Returns

Information about the removed group — if the call was successful.

See [cbe::delegate::group::RemoveSuccess](#)

## Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">RemoveException</a> |  |
|---------------------------------|--|

**11.118.4.19 remove()** [3/3]

```
cbe::util::Optional<delegate::group::RemoveSuccess> cbe::Group::remove (
 RemoveError & error)
```

Synchronous [non-throwing] removes the specified group. **Synchronous** version of [remove\(RemoveDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [remove\(RemoveDelegatePtr\)](#)

## Parameters

|            |              |                                                                                                                                                                                                                                |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">RemoveError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

**11.118.4.20 rename()** [1/3]

```
void cbe::Group::rename (
 std::string newName,
 RenameDelegatePtr delegate)
```

Rename the [Group](#); group id does not change.

## Parameters

|                 |                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------|
| <i>newName</i>  | the new name for the group                                                                             |
| <i>delegate</i> | Pointer to a <a href="#">delegate::group::RenameDelegate</a> instance that is implemented by the user. |

**11.118.4.21 rename()** [2/3]

```
delegate::group::RenameSuccess cbe::Group::rename (
 std::string newName)
```

Synchronous [exception] renames a group **Synchronous** version of `rename(std::string, RenameDelegatePtr)`, and **throws an exception**, `RenameException`, in case of a failed call.

See `rename(RenameDelegatePtr)`

**Returns**

Information about the renamed group — if the call was successful.

See `cbe::delegate::RenameSuccess`

**Exceptions**

|                              |  |
|------------------------------|--|
| <code>RenameException</code> |  |
|------------------------------|--|

**11.118.4.22 rename()** [3/3]

```
cbe::util::Optional<delegate::group::RenameSuccess> cbe::Group::rename (
 std::string newName,
 RenameError & error)
```

Synchronous [non-throwing] renames a group **Synchronous** version of `rename(newName, RenameDelegatePtr)`, and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `rename(newName, RenameDelegatePtr)`

**Parameters**

|     |       |                                                                                                                                                                                                                             |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <code>RenameError</code> object passed in will be populated with the error information. |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

**11.118.4.23 listMembers()** [1/3]

```
void cbe::Group::listMembers (
 ListMembersDelegatePtr delegate)
```

List all members in the group.

The member list is then retrieved via the delegate callback function `cbe::delegate::ListMembersDelegate::onListMembersSuccess()`

**Parameters**

|                       |                                                                                                   |
|-----------------------|---------------------------------------------------------------------------------------------------|
| <code>delegate</code> | Pointer to a <code>delegate::ListMembersDelegate</code> instance that is implemented by the user. |
|-----------------------|---------------------------------------------------------------------------------------------------|

**11.118.4.24 listMembers()** [2/3]

```
delegate::ListMembersDelegate::Members cbe::Group::listMembers ()
```

Synchronous [exception] list members of the specified group. **Synchronous** version of [listMembers\(ListMembersDelegatePtr\)](#), and **throws an exception**, [ListMembersException](#), in case of a failed call.

See [listMembers\(ListMembersDelegatePtr\)](#)

**Returns**

Information about the listMembers object — if the call was successful.

See `cbe::delegate::ListMembersSuccess`

**Exceptions**

|                                      |
|--------------------------------------|
| <a href="#">ListMembersException</a> |
|--------------------------------------|

**11.118.4.25 listMembers()** [3/3]

```
cbe::util::Optional<delegate::ListMembersDelegate::Members> cbe::Group::listMembers (
 ListMembersError & error)
```

Synchronous [non-throwing] list members of the specified group. **Synchronous** version of [listMembers\(ListMembersDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listMembers\(ListMembersDelegatePtr\)](#)

**Parameters**

|                  |                    |                                                                                                                                                                                                                                     |
|------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ListMembersError</a> object passed in will be populated with the error information. |
|------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

**11.118.4.26 listBannedMembers()** [1/3]

```
void cbe::Group::listBannedMembers (
 ListBannedMembersDelegatePtr delegate)
```

Lists all banned former members, or users.

**Parameters**

|                       |                                                                                                      |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <code>delegate</code> | Pointer to a <a href="#">delegate::ListMembersDelegate</a> instance that is implemented by the user. |
|-----------------------|------------------------------------------------------------------------------------------------------|

**11.118.4.27 listBannedMembers()** [2/3]

```
std::vector<cbe::Member> cbe::Group::listBannedMembers ()
```

Synchronous [exception] list the banned members of the specified group. **Synchronous** version of [listBannedMembers\(ListMembersDelegatePtr\)](#), and **throws an exception**, [ListBannedMembersException](#), in case of a failed call.

See [listBannedMembers\(ListMembersDelegatePtr\)](#)

**Returns**

Information about the `listBannedMembers` object — if the call was successful.  
See `cbe::delegate::ListBannedMembersSuccess`

**Exceptions**

|                                            |  |
|--------------------------------------------|--|
| <a href="#">ListBannedMembersException</a> |  |
|--------------------------------------------|--|

**11.118.4.28 listBannedMembers() [3/3]**

```
cbe::util::Optional<std::vector<cbe::Member> > cbe::Group::listBannedMembers (
 ListBannedMembersError & error)
```

Synchronous [non-throwing] list the banned members of the specified group. **Synchronous** version of [listBannedMembers\(ListMembersDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listBannedMembers\(ListMembersDelegatePtr\)](#)

**Parameters**

|                  |                    |                                                                                                                                                                                                                                           |
|------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ListBannedMembersError</a> object passed in will be populated with the error information. |
|------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

**11.118.4.29 listRoles() [1/3]**

```
void cbe::Group::listRoles (
 ListRolesDelegatePtr delegate)
```

Lists all roles in the group.

**Parameters**

|                       |                                                                                                    |
|-----------------------|----------------------------------------------------------------------------------------------------|
| <code>delegate</code> | Pointer to a <a href="#">delegate::ListRolesDelegate</a> instance that is implemented by the user. |
|-----------------------|----------------------------------------------------------------------------------------------------|

**11.118.4.30 listRoles() [2/3]**

```
delegate::ListRolesDelegate::Roles cbe::Group::listRoles ()
```

Synchronous [exception] list roles of a group. **Synchronous** version of [listRoles\(ListRolesDelegatePtr\)](#) , and **throws an exception**, [ListRolesException](#), in case of a failed call.

See [listRoles\(ListRolesDelegatePtr\)](#)

**Returns**

Information about the `ListRoles` object — if the call was successful.  
See `cbe::delegate::ListRolesDelegate::Roles`

## Exceptions

|                                    |  |
|------------------------------------|--|
| <a href="#">ListRolesException</a> |  |
|------------------------------------|--|

**11.118.4.31 listRoles()** [3/3]

```
cbe::util::Optional<delegate::ListRolesDelegate::Roles> cbe::Group::listRoles (
 ListRolesError & error)
```

Synchronous [non-throwing] list roles of a group. **Synchronous** version of [listRoles\(ListRolesDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listRoles\(ListRolesDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                   |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ListRolesError</a> object passed in will be populated with the error information. |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

**11.118.4.32 createRole()** [1/3]

```
void cbe::Group::createRole (
 std::string name,
 CreateRoleDelegatePtr delegate)
```

Creates a new role in the group.

## Parameters

|                 |                                                                                                     |
|-----------------|-----------------------------------------------------------------------------------------------------|
| <i>name</i>     | The name of the role.                                                                               |
| <i>delegate</i> | Pointer to a <a href="#">delegate::CreateRoleDelegate</a> instance that is implemented by the user. |

**11.118.4.33 createRole()** [2/3]

```
cbe::Role cbe::Group::createRole (
 std::string name)
```

Synchronous [exception] creates a new role in the specified group. **Synchronous** version of [createRole\(std::string, CreateRoleDelegatePtr\)](#) , and **throws an exception**, [CreateRoleException](#), in case of a failed call.

See [createRole\(CreateRoleDelegatePtr\)](#)

## Returns

Information about the createRole object — if the call was successful.  
See [cbe::delegate::CreateRoleSuccess](#)

## Exceptions

|                                     |  |
|-------------------------------------|--|
| <a href="#">CreateRoleException</a> |  |
|-------------------------------------|--|

**11.118.4.34 createRole()** [3/3]

```
cbe::util::Optional<cbe::Role> cbe::Group::createRole (
 std::string name,
 CreateRoleError & error)
```

Synchronous [non-throwing] creates a new role in the specified group. **Synchronous** version of createRole(name, CreateRoleDelegatePtr) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See createRole(name, CreateRoleDelegatePtr)

**Parameters**

|     |       |                                                                                                                                                                                                                                    |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">CreateRoleError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

**11.118.4.35 removeRole()** [1/3]

```
void cbe::Group::removeRole (
 RoleId roleId,
 RemoveRoleDelegatePtr delegate)

@brief Removes a role in the group.

@param roleId The id of the role to remove.

@param delegate Pointer to a delegate::RemoveRoleDelegate instance
```

that is implemented by the user.

**11.118.4.36 removeRole()** [2/3]

```
cbe::RoleId cbe::Group::removeRole (
 RoleId roleId)
```

Synchronous [exception] removes a role in the specified group. **Synchronous** version of [removeRole\(RoleId roleId, RemoveRoleDelegatePtr\)](#) , and **throws an exception**, [RemoveRoleException](#), in case of a failed call.

See removeRole(RemoveRoleDelegatePtr)

**Returns**

Information about the removeRole object — if the call was successful.  
See [cbe::delegate::RemoveRoleSuccess](#)

**Exceptions**

|                                     |
|-------------------------------------|
| <a href="#">RemoveRoleException</a> |
|-------------------------------------|

**11.118.4.37 removeRole()** [3/3]

```
cbe::util::Optional<cbe::RoleId> cbe::Group::removeRole (
 RoleId roleId,
 RemoveRoleError & error)
```

Synchronous [non-throwing] removes a role in the specified group. **Synchronous** version of removeRole(roleId, RemoveRoleDelegatePtr) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See removeRole(roleId, RemoveRoleDelegatePtr)

**Parameters**

|     |       |                                                                                                                                                                                                                                    |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">RemoveRoleError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicating a failed call, and the error information is passed out via the `error` out/return parameter.

**11.118.4.38 operator bool()**

```
cbe::Group::operator bool () const [explicit]
```

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [Group\(cbe::DefaultCtor\)](#)

**Returns**

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/Group.h

**11.119 cbe::GroupFilter Class Reference**

To filter when searching a list of [Group](#).

```
#include <GroupFilter.h>
```

**Public Member Functions**

- **GroupFilter** (const [GroupFilter](#) &rh)
- **GroupFilter** ([GroupFilter](#) &&rh)
- [GroupFilter](#) & **operator=** (const [GroupFilter](#) &rh)
- [GroupFilter](#) & **operator=** ([GroupFilter](#) &&rh)
- std::string [getQuery](#) () const
- std::string [getFilter](#) () const
- bool [getAscending](#) () const
- bool [getDeleted](#) () const
- bool [getPublicFirst](#) () const
- uint32\_t [getOffset](#) () const
- uint32\_t [getCount](#) () const
- std::string [getOrder](#) () const

- [cbe::FilterOrder](#) [getGroupOrder](#) ( ) const
- [cbe::GroupFilter](#) & [setQuery](#) (std::string)
- [cbe::GroupFilter](#) & [setFilter](#) (std::string)
- [cbe::GroupFilter](#) & [setAscending](#) (bool)
- [cbe::GroupFilter](#) & [setDeleted](#) (bool)
- [cbe::GroupFilter](#) & [setOffset](#) (uint32\_t)
- [cbe::GroupFilter](#) & [setCount](#) (uint32\_t)

## Friends

- class **Group**
- class **GroupQueryResult**
- class **GroupManager**
- class **delegate::SearchGroupDelegate**
- std::ostream & [operator<<](#) (std::ostream &os, const [GroupFilter](#) &[GroupFilter](#))

### 11.119.1 Detailed Description

To filter when searching a list of [Group](#).

### 11.119.2 Member Function Documentation

#### 11.119.2.1 [getQuery\(\)](#)

```
std::string cbe::GroupFilter::getQuery () const
```

Returns the query string that was set on the filter. E.g., name:Abc\*

#### 11.119.2.2 [getFilter\(\)](#)

```
std::string cbe::GroupFilter::getFilter () const
```

the filter in question

#### 11.119.2.3 [getAscending\(\)](#)

```
bool cbe::GroupFilter::getAscending () const
```

Returns the settings on how the data was sorted and displayed.

#### 11.119.2.4 [getDeleted\(\)](#)

```
bool cbe::GroupFilter::getDeleted () const
```

If deleted objects in the bin are included.

#### 11.119.2.5 [getPublicFirst\(\)](#)

```
bool cbe::GroupFilter::getPublicFirst () const
```

If public groups should be sorted before private.

#### 11.119.2.6 [getOffset\(\)](#)

```
uint32_t cbe::GroupFilter::getOffset () const
```

Returns the offset that was used for the filter in the query.

#### 11.119.2.7 [getCount\(\)](#)

```
uint32_t cbe::GroupFilter::getCount () const
```

Returns the value of the count that was set for the query.

#### 11.119.2.8 getOrder()

`std::string cbe::GroupFilter::getOrder ( ) const`  
get the filterorder as a string.

#### 11.119.2.9 getGroupOrder()

`cbe::FilterOrder cbe::GroupFilter::getGroupOrder ( ) const`  
Returns the order the query was sorted check enum FilterOrder to see options for sorting.

#### 11.119.2.10 setQuery()

`cbe::GroupFilter& cbe::GroupFilter::setQuery (`  
    `std::string )`  
Set the query string, e.g: Name:\* (would search for all objects with the metadata key of Name).

##### Note

If used with rootContainer id as the dataId, search in the whole account will be done.

#### 11.119.2.11 setFilter()

`cbe::GroupFilter& cbe::GroupFilter::setFilter (`  
    `std::string )`  
Not fully tested but should be a regular expression.

#### 11.119.2.12 setAscending()

`cbe::GroupFilter& cbe::GroupFilter::setAscending (`  
    `bool )`  
Sets the Order in which data should be displayed: Acending meaning alfabetical

#### 11.119.2.13 setDeleted()

`cbe::GroupFilter& cbe::GroupFilter::setDeleted (`  
    `bool )`  
If you want to see deleted groups set deleted to true.

#### 11.119.2.14 setOffset()

`cbe::GroupFilter& cbe::GroupFilter::setOffset (`  
    `uint32_t )`  
Set the offset for paging, offset is the item offset where to start your query. i.e: There is already a query of the first 99 items and to get the rest you have setOffset to 100 to get the next set.

#### 11.119.2.15 setCount()

`cbe::GroupFilter& cbe::GroupFilter::setCount (`  
    `uint32_t )`  
Set the Number of items you want to get from a container. So if a container has 50 items but you only want the 10 first in ascending order then set ascending to true and setCount to 10.

### 11.119.3 Friends And Related Function Documentation

### 11.119.3.1 operator<<

```
std::ostream& operator<< (
 std::ostream & os,
 const GroupFilter & GroupFilter) [friend]
```

Set the order of how data will be shown by the enum of FilterOrder ex: Titel first, published e.t.c

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

- include/cbe/GroupFilter.h

## 11.120 cbe::GroupManager Class Reference

For managing the groups.

```
#include <GroupManager.h>
```

### Public Types

- using ListGroupsDelegatePtr = delegate::ListGroupsDelegatePtr
- using ListGroupsException = delegate::ListGroupsDelegate::Exception
- using ListGroupsError = delegate::ListGroupsDelegate::ErrorInfo
- using SearchGroupsDelegatePtr = delegate::SearchGroupsDelegatePtr
- using SearchGroupsException = delegate::SearchGroupsDelegate::Exception
- using SearchGroupsError = delegate::SearchGroupsDelegate::ErrorInfo

### Public Member Functions

- void listGroups (ListGroupsDelegatePtr delegate)
- std::vector< cbe::Group > listGroups ()
 

*Synchronous [exception] **Synchronous** version of `listGroups(ListGroupsDelegatePtr)` , and **throws an exception**, `ListGroupsException`, in case of a failed call.*

*See `listGroups(ListGroupsDelegatePtr)`*
- cbe::util::Optional< std::vector< cbe::Group > > listGroups (ListGroupsError &error)
 

*Synchronous [non-throwing] **Synchronous** version of `listGroups(ListGroupsDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*

*See `listGroups(ListGroupsDelegatePtr)`*
- void searchGroups (cbe::GroupFilter filter, SearchGroupsDelegatePtr delegate, cbe::GroupId parentGroupId=0)
- delegate::SearchGroupsDelegate::Success searchGroups (cbe::GroupFilter filter, cbe::GroupId parentGroupId=0)
 

*Synchronous [exception] **Synchronous** version of `searchGroups(filter, SearchGroupsDelegatePtr, parentGroupId = 0)` , and **throws an exception**, `SearchGroupsException`, in case of a failed call.*

*See `searchGroups(SearchGroupsDelegatePtr)`*
- cbe::util::Optional< delegate::SearchGroupsDelegate::Success > searchGroups (cbe::GroupFilter filter, cbe::GroupId parentGroupId, SearchGroupsError &error)
 

*Synchronous [non-throwing] **Synchronous** version of `searchGroups(filter, parentGroupId, SearchGroupsDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*

*See `searchGroups(filter, parentGroupId, SearchGroupsDelegatePtr)`*
- cbe::GroupId getTenantId () const
- GroupManager (cbe::DefaultCtor)
 

*Default constructor.*
- operator bool () const
 

*Checks if the current instance is real.*

### Friends

- class CloudBackend

### 11.120.1 Detailed Description

For managing the groups.

### 11.120.2 Member Typedef Documentation

#### 11.120.2.1 ListGroupsDelegatePtr

using `cbe::GroupManager::ListGroupsDelegatePtr = delegate::ListGroupsDelegatePtr`

Pointer to `cbe::delegate::ListGroupsDelegate` that is passed into asynchronous version of method `listGroups()`

#### 11.120.2.2 ListGroupsException

using `cbe::GroupManager::ListGroupsException = delegate::ListGroupsDelegate::Exception`

See `delegate::object::ListGroupsDelegate::Exception`

#### 11.120.2.3 ListGroupsError

using `cbe::GroupManager::ListGroupsError = delegate::ListGroupsDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `listGroups()`

See `delegate::ListGroupsDelegate::ErrorInfo`

#### 11.120.2.4 SearchGroupsDelegatePtr

using `cbe::GroupManager::SearchGroupsDelegatePtr = delegate::SearchGroupsDelegatePtr`

Pointer to `SearchGroupsDelegate` that is passed into:

- `cbe::GroupManager::searchGroups()`

#### 11.120.2.5 SearchGroupsException

using `cbe::GroupManager::SearchGroupsException = delegate::SearchGroupsDelegate::Exception`

Pointer to `cbe::delegate::SearchGroupsDelegate` that is passed into asynchronous version of method `searchGroups()` See `delegate::object::SearchGroupsDelegate::Exception`

#### 11.120.2.6 SearchGroupsError

using `cbe::GroupManager::SearchGroupsError = delegate::SearchGroupsDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `searchGroups()`

See `delegate::SearchGroupsDelegate::ErrorInfo`

### 11.120.3 Constructor & Destructor Documentation

#### 11.120.3.1 GroupManager()

```
cbe::GroupManager::GroupManager (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the `operator bool()` test

### 11.120.4 Member Function Documentation

**11.120.4.1 listGroups()** [1/3]

```
void cbe::GroupManager::listGroups (
 ListGroupsDelegatePtr delegate)
```

List all current joined groups.

**Parameters**

|                 |                                                                                                     |
|-----------------|-----------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::ListGroupsDelegate</a> instance that is implemented by the user. |
|-----------------|-----------------------------------------------------------------------------------------------------|

**11.120.4.2 listGroups()** [2/3]

```
std::vector<cbe::Group> cbe::GroupManager::listGroups ()
```

Synchronous [exception] **Synchronous** version of [listGroups\(ListGroupsDelegatePtr\)](#) , and **throws an exception**, [ListGroupsException](#), in case of a failed call.

See [listGroups\(ListGroupsDelegatePtr\)](#)

**Returns**

Information about the listGroups object — if the call was successful.

See [cbe::ListGroupsDelegate::Groups](#)

**Exceptions**

|                                     |  |
|-------------------------------------|--|
| <a href="#">ListGroupsException</a> |  |
|-------------------------------------|--|

**11.120.4.3 listGroups()** [3/3]

```
cbe::util::Optional<std::vector<cbe::Group> > cbe::GroupManager::listGroups (
 ListGroupsError & error)
```

Synchronous [non-throwing] **Synchronous** version of [listGroups\(ListGroupsDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listGroups\(ListGroupsDelegatePtr\)](#)

**Parameters**

|            |              |                                                                                                                                                                                                                                    |
|------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ListGroupsError</a> object passed in will be populated with the error information. |
|------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.120.4.4 searchGroups()** [1/3]

```
void cbe::GroupManager::searchGroups (
 cbe::GroupFilter filter,
 SearchGroupsDelegatePtr delegate,
 cbe::GroupId parentGroupId = 0)
```

Search for open public groups.

## Parameters

|                      |                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <i>filter</i>        | is a group filter to set search criteria for open public groups. Look att class <a href="#">GroupFilter</a> for more information. |
| <i>delegate</i>      | Pointer to a <a href="#">delegate::SearchGroupsDelegate</a> instance that is implemented by the user.                             |
| <i>parentGroupId</i> | is the id of the group to be searched within. If this is not set the tenant id will be used.                                      |

## 11.120.4.5 searchGroups() [2/3]

```
delegate::SearchGroupsDelegate::Success cbe::GroupManager::searchGroups (
 cbe::GroupFilter filter,
 cbe::GroupId parentGroupId = 0)
```

Synchronous [exception] **Synchronous** version of searchGroups(filter, SearchGroupsDelegatePtr, parentGroupId = 0) , and **throws an exception**, [SearchGroupsException](#), in case of a failed call.

See searchGroups(SearchGroupsDelegatePtr)

## Returns

Information about the searchGroups object — if the call was successful.

See cbe::delegate::SearchGroupsDelegate::Success

## Exceptions

|                                       |  |
|---------------------------------------|--|
| <a href="#">SearchGroupsException</a> |  |
|---------------------------------------|--|

## 11.120.4.6 searchGroups() [3/3]

```
cbe::util::Optional<delegate::SearchGroupsDelegate::Success> cbe::GroupManager::searchGroups (
 cbe::GroupFilter filter,
 cbe::GroupId parentGroupId,
 SearchGroupsError & error)
```

Synchronous [non-throwing] **Synchronous** version of searchGroups(filter, parentGroupId, SearchGroupsDelegatePtr) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See searchGroups(filter, parentGroupId, SearchGroupsDelegatePtr)

## Parameters

|            |              |                                                                                                                                                                                                                                      |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SearchGroupsError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

## 11.120.4.7 getTenantId()

```
cbe::GroupId cbe::GroupManager::getTenantId () const
```

Returns the tenant id of the Tenant user group that the user is in.

#### 11.120.4.8 operator bool()

```
cbe::GroupManager::operator bool () const [explicit]
```

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [GroupManager\(cbe::DefaultCtor\)](#)

Returns

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/GroupManager.h

## 11.121 cbe::GroupQueryResult Class Reference

Resultset of data retrieved in a search for [Group](#).

```
#include <GroupQueryResult.h>
```

### Public Types

- using **GroupsSnapshot** = std::vector< [Group](#) >

### Public Member Functions

- **GroupQueryResult** ([cbe::DefaultCtor](#))
- [cbe::GroupFilter](#) filter () const
- GroupsSnapshot [getGroupsSnapshot](#) () const
- uint64\_t [GroupsLoaded](#) () const
- uint64\_t [totalCount](#) () const
- bool [containsGroup](#) (uint64\_t groupId)
- **operator bool** () const

### Friends

- class **GroupManager**
- class **CloudBackend**
- std::ostream & **operator**<< (std::ostream &os, const [GroupQueryResult](#) &groupQuery)

#### 11.121.1 Detailed Description

Resultset of data retrieved in a search for [Group](#).

#### 11.121.2 Member Function Documentation

##### 11.121.2.1 filter()

```
cbe::GroupFilter cbe::GroupQueryResult::filter () const
```

Returns a copy of the filter used for query.

**11.121.2.2 getGroupsSnapshot()**

```
GroupsSnapshot cbe::GroupQueryResult::getGroupsSnapshot () const
```

Returns a copy of a vector containing the groups for the queryResult. The queryResult will update when new data comes in but the copy will not. If iterating, make sure to create a variable for a local copy.

**Returns**

vector<cbe::Group> containing the groups matching the query.

**11.121.2.3 GroupsLoaded()**

```
uint64_t cbe::GroupQueryResult::GroupsLoaded () const
```

Number of groups loaded in the queryResult.

**11.121.2.4 totalCount()**

```
uint64_t cbe::GroupQueryResult::totalCount () const
```

Total number of Groups in the cloud matching the query result. This may be more than loaded.

**11.121.2.5 containsGroup()**

```
bool cbe::GroupQueryResult::containsGroup (
 uint64_t groupId)
```

asking if the group with groupId was loaded in the query.

**Parameters**

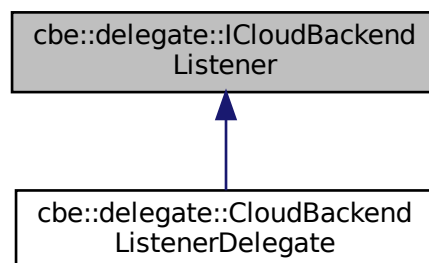
|                |                                 |
|----------------|---------------------------------|
| <i>groupId</i> | the group id number in question |
|----------------|---------------------------------|

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

- include/cbe/GroupQueryResult.h

**11.122 cbe::delegate::ICloudBackendListener Class Reference**

Inheritance diagram for cbe::delegate::ICloudBackendListener:



## Public Member Functions

- virtual void [onRemoteObjectAdded](#) ([cbe::Object](#) &&object)=0
- virtual void [onRemoteObjectMoved](#) ([cbe::Object](#) &&object)=0
- virtual void [onRemoteObjectRemoved](#) ([cbe::ItemId](#) objectId, std::string name)=0
- virtual void [onRemoteObjectRenamed](#) ([cbe::Object](#) &&object)=0
- virtual void [onRemoteContainerAdded](#) ([cbe::Container](#) &&container)=0
- virtual void [onRemoteContainerMoved](#) ([cbe::Container](#) &&container)=0
- virtual void [onRemoteContainerRemoved](#) ([cbe::ItemId](#) containerId, std::string name)=0
- virtual void [onRemoteContainerRenamed](#) ([cbe::Container](#) &&container)=0

### 11.122.1 Member Function Documentation

#### 11.122.1.1 onRemoteObjectAdded()

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteObjectAdded (
 cbe::Object && object) [pure virtual]
```

Called upon successful create of an [object](#).

##### Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>object</i> | Instance of object that is being created. |
|---------------|-------------------------------------------|

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

#### 11.122.1.2 onRemoteObjectMoved()

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteObjectMoved (
 cbe::Object && object) [pure virtual]
```

Called upon successful move of an [object](#).

##### Parameters

|               |                                         |
|---------------|-----------------------------------------|
| <i>object</i> | Instance of object that is being moved. |
|---------------|-----------------------------------------|

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

#### 11.122.1.3 onRemoteObjectRemoved()

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteObjectRemoved (
 cbe::ItemId objectId,
 std::string name) [pure virtual]
```

Called upon successful removal of an [object](#).

##### Parameters

|                 |                                               |
|-----------------|-----------------------------------------------|
| <i>objectId</i> | Instance of the object that is being Removed. |
| <i>name</i>     | Name of the object that is being Removed.     |

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

**11.122.1.4 onRemoteObjectRenamed()**

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteObjectRenamed (
 cbe::Object && object) [pure virtual]
```

Called upon successful renaming of an [object](#).

**Parameters**

|               |                                           |
|---------------|-------------------------------------------|
| <i>object</i> | Instance of object that is being renamed. |
|---------------|-------------------------------------------|

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

**11.122.1.5 onRemoteContainerAdded()**

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteContainerAdded (
 cbe::Container && container) [pure virtual]
```

Called upon successful Create.

**Parameters**

|                  |                                              |
|------------------|----------------------------------------------|
| <i>container</i> | Instance of container that is being created. |
|------------------|----------------------------------------------|

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

**11.122.1.6 onRemoteContainerMoved()**

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteContainerMoved (
 cbe::Container && container) [pure virtual]
```

Called upon successful Move.

**Parameters**

|                  |                                            |
|------------------|--------------------------------------------|
| <i>container</i> | Instance of container that is being moved. |
|------------------|--------------------------------------------|

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

**11.122.1.7 onRemoteContainerRemoved()**

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteContainerRemoved (
 cbe::ItemId containerId,
 std::string name) [pure virtual]
```

Called upon successful Remove.

**Parameters**

|                          |                                                  |
|--------------------------|--------------------------------------------------|
| <i>container↔<br/>Id</i> | Instance of the container that is being Removed. |
| <i>name</i>              | Name of the container that is being removed.     |

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

**11.122.1.8 onRemoteContainerRenamed()**

```
virtual void cbe::delegate::ICloudBackendListener::onRemoteContainerRenamed (
```

```
cbe::Container && container) [pure virtual]
```

Called upon successful Rename.

#### Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <i>container</i> | Instance of container that is being Renamed. |
|------------------|----------------------------------------------|

Implemented in [cbe::delegate::CloudBackendListenerDelegate](#).

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

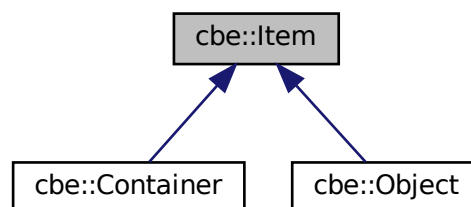
- include/cbe/delegate/ICloudBackendListener.h

## 11.123 cbe::Item Class Reference

A set made up of [Container](#) and [Object](#).

```
#include <Item.h>
```

Inheritance diagram for cbe::Item:



### Public Member Functions

- [cbe::ShareIds](#) getShareIds () const
- [cbe::ShareId](#) getShareFromUserId ([cbe::UserId](#) userId)
- [cbe::UserId](#) getUserFromShareId ([cbe::UserId](#) shareId)
- std::string [aclTag](#) () const
- std::string [description](#) () const
- [cbe::ItemId](#) id () const
- [cbe::ContainerId](#) parentId () const
- [cbe::ContainerId](#) oldParentId () const
- std::string [name](#) () const
- std::string [path](#) () const
- [cbe::UserId](#) ownerId () const
- [cbe::ContainerId](#) driveId () const
- std::string [username](#) () const
- bool [idLoaded](#) () const
- bool [dataLoaded](#) () const
- [cbe::Date](#) created () const
- [cbe::Date](#) updated () const
- [cbe::Date](#) deleted () const
- [cbe::ItemType](#) type () const
- bool [hasPublished](#) () const
- [Publish](#) getPublished () const

- bool [hasSubscribe](#) () const
- [Subscribe](#) [getSubscribe](#) () const
- bool **operator**< (const [cbe::Item](#) &rh) const
- [AclMap](#) [aclMap](#) () const
- [operator](#) bool () const

*Checks if the inherited instance is real.*

## Protected Member Functions

- **Item** (std::shared\_ptr< Impl > pImpl)
- template<class ImplT >  
ImplT & **castImpl** () const

## Friends

- class **CloudBackend**
- class **Container**
- class **Object**
- class **PublishManager**
- class **QueryResult**
- class **SubscribeManager**
- std::ostream & **operator**<< (std::ostream &os, const [Item](#) &item)

### 11.123.1 Detailed Description

A set made up of [Container](#) and [Object](#).

Class for an [Item](#). This class is the base class of Objects and Containers.

### 11.123.2 Member Function Documentation

#### 11.123.2.1 getShareIds()

```
cbe::ShareIds cbe::Item::getShareIds () const
```

a map of share ids for this item

#### 11.123.2.2 getShareFromUserId()

```
cbe::ShareId cbe::Item::getShareFromUserId (
 cbe::UserId userId)
```

Get the shareId by reference of userId

#### 11.123.2.3 getUserFromShareId()

```
cbe::UserId cbe::Item::getUserFromShareId (
 cbe::UserId shareId)
```

Get the userId reference of shareId

#### 11.123.2.4 aclTag()

```
std::string cbe::Item::aclTag () const
```

the ACL of the item

#### 11.123.2.5 description()

```
std::string cbe::Item::description () const
```

A item text description where applicable

#### 11.123.2.6 id()

`cbe::ItemId cbe::Item::id ( ) const`  
Returns an Items id.

#### 11.123.2.7 parentId()

`cbe::ContainerId cbe::Item::parentId ( ) const`  
Returns the id of the Items parent container id.

#### 11.123.2.8 oldParentId()

`cbe::ContainerId cbe::Item::oldParentId ( ) const`  
Returns the id of the Items old parent container id in cases where the item has just been moved.

#### 11.123.2.9 name()

`std::string cbe::Item::name ( ) const`  
Returns the name of the item.

#### 11.123.2.10 path()

`std::string cbe::Item::path ( ) const`  
Returns the path if applicable.

#### 11.123.2.11 ownerId()

`cbe::UserId cbe::Item::ownerId ( ) const`  
Returns the owner id number.

#### 11.123.2.12 driveId()

`cbe::ContainerId cbe::Item::driveId ( ) const`  
Returns which drive number the container resides on.

#### 11.123.2.13 username()

`std::string cbe::Item::username ( ) const`  
Returns the username of the owner if applicable.

#### 11.123.2.14 idLoaded()

`bool cbe::Item::idLoaded ( ) const`  
if the query did get a resultset.

#### 11.123.2.15 dataLoaded()

`bool cbe::Item::dataLoaded ( ) const`  
if data was loaded

#### 11.123.2.16 created()

`cbe::Date cbe::Item::created ( ) const`  
Returns the creation date in Unix epoch time.

#### 11.123.2.17 updated()

`cbe::Date cbe::Item::updated ( ) const`  
Returns the updated date/time in Unix epoch time

### 11.123.2.18 deleted()

```
cbe::Date cbe::Item::deleted () const
```

Returns the deleted date, if applicable (i.e. is in the bin), in Unix epoch time

### 11.123.2.19 type()

```
cbe::ItemType cbe::Item::type () const
```

[Container](#) or [Object](#), see enum in [Types.h](#)

### 11.123.2.20 hasPublished()

```
bool cbe::Item::hasPublished () const
```

Checks if it has published info and if [getPublished\(\)](#) returns a valid object.

### 11.123.2.21 getPublished()

```
Publish cbe::Item::getPublished () const
```

Gets the publish information

See also

[hasPublished\(\)](#)

### 11.123.2.22 hasSubscribe()

```
bool cbe::Item::hasSubscribe () const
```

Checks if it has subscribe info and if [getSubscribe\(\)](#) returns a valid object.

### 11.123.2.23 getSubscribe()

```
Subscribe cbe::Item::getSubscribe () const
```

Gets the subscribe information

See also

[hasSubscribe\(\)](#)

### 11.123.2.24 aclMap()

```
AclMap cbe::Item::aclMap () const
```

Retrieve the map of the permissions the users and groups for this item.

### 11.123.2.25 operator bool()

```
cbe::Item::operator bool () const [explicit]
```

Checks if the inherited instance is real.

An "unreal" instance implies typically a failed event.

Relies on the `Default` constructor [cbe::DefaultCtor](#)

Returns

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- `include/cbe/Item.h`

## 11.124 cbe::delegate::ItemError Class Reference

```
#include <ItemDelegateErrors.h>
```

### Classes

- struct [Error](#)

### Public Member Functions

- virtual void [onItemError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### Friends

- std::ostream & [operator](#)<< (std::ostream &os, const [ItemError::Error](#) &error)

### 11.124.1 Detailed Description

Entity class containing the information passed into method [onItemError\(\)](#) in cbe interface

### 11.124.2 Member Function Documentation

#### 11.124.2.1 onItemError()

```
virtual void cbe::delegate::ItemError::onItemError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called upon failed service calls like:

- move
- remove
- rename

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

- include/cbe/delegate/ItemDelegateErrors.h

## 11.125 cbe::delegate::group::JoinDelegate Class Reference

```
#include <JoinDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Group::join\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::Group](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onJoinSuccess](#) ([cbe::Group](#) &&group)=0
- virtual void [onJoinError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.125.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Group::join](#)

#### Note

This is different from the query chain join.

### 11.125.2 Member Function Documentation

#### 11.125.2.1 onJoinSuccess()

```
virtual void cbe::delegate::group::JoinDelegate::onJoinSuccess (
 cbe::Group && group) [pure virtual]
```

Called upon successful Join.

#### 11.125.2.2 onJoinError()

```
virtual void cbe::delegate::group::JoinDelegate::onJoinError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/group/JoinDelegate.h

## 11.126 cbe::delegate::JoinDelegate Class Reference

```
#include <JoinDelegate.h>
```

### Classes

- struct [Exception](#)  
*exception thrown by [cbe::QueryChain::join\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::QueryResult](#)
- using **JoinError** = [QueryError](#)
- using **Error** = [JoinError](#)
- using [ErrorInfo](#) = [QueryJoinDelegate::ErrorInfo](#)

### Public Member Functions

- virtual void [onJoinSuccess](#) ([cbe::QueryResult](#) &&queryResult)=0
- virtual void [onJoinError](#) ([JoinError](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.126.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::QueryChain::join\(\)](#)

#### Note

This is different from the group join.

## 11.126.2 Member Typedef Documentation

### 11.126.2.1 `ErrorInfo`

using `cbe::delegate::JoinDelegate::ErrorInfo` = `QueryJoinDelegate::ErrorInfo`

Contains all information about a failed join.

## 11.126.3 Member Function Documentation

### 11.126.3.1 `onJoinSuccess()`

```
virtual void cbe::delegate::JoinDelegate::onJoinSuccess (
 cbe::QueryResult && queryResult) [pure virtual]
```

Called upon successful query.

#### Parameters

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <i>queryResult</i> | Instance of a <a href="#">QueryResult</a> containing the result set. |
|--------------------|----------------------------------------------------------------------|

### 11.126.3.2 `onJoinError()`

```
virtual void cbe::delegate::JoinDelegate::onJoinError (
 JoinError && error,
 cbe::util::Context && context) [pure virtual]
```

Called upon a failed join() call.

#### Parameters

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| <i>error</i>   | <a href="#">Error</a> information passed from CloudBackend SDK.                 |
| <i>context</i> | Additional context information about the original service call that has failed. |

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

- `include/cbe/delegate/JoinDelegate.h`

## 11.127 `cbe::delegate::JoinError` Class Reference

```
#include <ItemDelegateErrors.h>
```

### Classes

- class [Error](#)

### Public Member Functions

- virtual void [onJoinError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### Friends

- `std::ostream & operator<< (std::ostream &os, const JoinError::Error &error)`

### 11.127.1 Detailed Description

Entity class containing the information passed into method `onJoinError()` in cbe interface

### 11.127.2 Member Function Documentation

#### 11.127.2.1 onJoinError()

```
virtual void cbe::delegate::JoinError::onJoinError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called upon a failed join() call.

#### Parameters

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| <i>error</i>   | Error information passed from CloudBackend SDK.                                 |
| <i>context</i> | Additional context information about the original service call that has failed. |

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

- include/cbe/delegate/ItemDelegateErrors.h

## 11.128 cbe::delegate::KickDelegate Class Reference

```
#include <KickDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by `cbe::Member::kick(std::string)` if the request fails.*

### Public Types

- using **Success** = [KickSuccess](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onKickSuccess](#) (std::string &&memberName, [cbe::MemberId](#) memberId)=0
- virtual void [onKickError](#) (Error &&error, [cbe::util::Context](#) &&context)=0

### 11.128.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Member::kick](#)

### 11.128.2 Member Function Documentation

### 11.128.2.1 onKickSuccess()

```
virtual void cbe::delegate::KickDelegate::onKickSuccess (
 std::string && memberName,
 cbe::MemberId memberId) [pure virtual]
```

Called upon successful ban.

#### Parameters

|                   |                                  |
|-------------------|----------------------------------|
| <i>memberName</i> | Name of the member being kicked. |
| <i>memberId</i>   | Id of the member being kicked.   |

### 11.128.2.2 onKickError()

```
virtual void cbe::delegate::KickDelegate::onKickError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/KickDelegate.h

## 11.129 cbe::delegate::KickSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::KickDelegate::onKickSuccess](#).

```
#include <KickDelegate.h>
```

### Public Member Functions

- **KickSuccess** ([cbe::DefaultCtor](#))
- **KickSuccess** (std::string &&memberName, [cbe::MemberId](#) memberId)

### Public Attributes

- std::string **memberName** {}
- [cbe::MemberId](#) **memberId** {}

### 11.129.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::KickDelegate::onKickSuccess](#).

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

- include/cbe/delegate/KickDelegate.h

## 11.130 cbe::delegate::LeaveDelegate Class Reference

```
#include <LeaveDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::Group::leave\(\)](#) if the request fails.*

## Public Types

- using **Success** = [LeaveSuccess](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onLeaveSuccess](#) (std::string &&memberName, [cbe::MemberId](#) memberId)=0
- virtual void [onLeaveError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.130.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Group::leave](#)

### 11.130.2 Member Function Documentation

#### 11.130.2.1 onLeaveSuccess()

```
virtual void cbe::delegate::LeaveDelegate::onLeaveSuccess (
 std::string && memberName,
 cbe::MemberId memberId) [pure virtual]
```

Called upon successful leave.

#### 11.130.2.2 onLeaveError()

```
virtual void cbe::delegate::LeaveDelegate::onLeaveError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/LeaveDelegate.h

## 11.131 cbe::delegate::LeaveSuccess Class Reference

### Public Member Functions

- **LeaveSuccess** ([cbe::DefaultCtor](#))
- **LeaveSuccess** (std::string &&memberName, [cbe::MemberId](#) memberId)
- **operator bool** () const  
*Checks if current instance is valid.*

### Public Attributes

- std::string **memberName** {}
- [cbe::MemberId](#) **memberId** {}

### 11.131.1 Member Function Documentation

### 11.131.1.1 operator bool()

```
cbe::delegate::LeaveSuccess::operator bool () const [explicit]
```

Checks if current instance is valid.

#### Returns

`true`: is valid

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

- `include/cbe/delegate/LeaveDelegate.h`

## 11.132 cbe::delegate::ListGroupDelegate Class Reference

```
#include <ListGroupDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::GroupManager::listGroups\(\)](#) if the request fails.*

### Public Types

- using **Groups** = `std::vector< cbe::Group >`
- using **Success** = `Groups`
- using **Error** = `delegate::Error`

### Public Member Functions

- virtual void [onListGroupSuccess](#) (`Groups &&groups`)=0
- virtual void [onListGroupError](#) (`delegate::Error &&error, cbe::util::Context &&context`)=0

### 11.132.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::GroupManager::listGroups](#)

### 11.132.2 Member Function Documentation

#### 11.132.2.1 onListGroupSuccess()

```
virtual void cbe::delegate::ListGroupDelegate::onListGroupSuccess (
 Groups && groups) [pure virtual]
```

Called upon successful listGroup

#### Parameters

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| <i>groups</i> | Ref to vector of <a href="#">cbe::Group</a> holding the joined groups. |
|---------------|------------------------------------------------------------------------|

#### 11.132.2.2 onListGroupError()

```
virtual void cbe::delegate::ListGroupDelegate::onListGroupError (
```

```

 delegate::Error && error,
 cbe::util::Context && context) [pure virtual]

```

Called if an error is encountered.

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

- include/cbe/delegate/ListGroupsDelegate.h

## 11.133 cbe::delegate::ListMembersDelegate Class Reference

```
#include <ListMembersDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Group::listMembers\(\)](#) if the request fails.*

### Public Types

- using **Members** = std::vector< [cbe::Member](#) >
- using **Success** = Members
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onListMembersSuccess](#) (Members &&members)=0
- virtual void [onListMembersError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

#### 11.133.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Group::listMembers](#)

#### 11.133.2 Member Function Documentation

##### 11.133.2.1 onListMembersSuccess()

```

virtual void cbe::delegate::ListMembersDelegate::onListMembersSuccess (
 Members && members) [pure virtual]

```

Called upon successful ListMembers.

##### Parameters

|                |                                                                       |
|----------------|-----------------------------------------------------------------------|
| <i>members</i> | Vector of <a href="#">cbe::Member</a> holding the members of a group. |
|----------------|-----------------------------------------------------------------------|

##### 11.133.2.2 onListMembersError()

```

virtual void cbe::delegate::ListMembersDelegate::onListMembersError (
 Error && error,
 cbe::util::Context && context) [pure virtual]

```

Called if an error is encountered.

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

- `include/cbe/delegate/ListMembersDelegate.h`

## 11.134 cbe::delegate::ListRolesDelegate Class Reference

```
#include <ListRolesDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by `cbe::Group::ListRoles()` if the request fails.*

### Public Types

- using **Roles** = `std::vector< cbe::Role >`
- using **Success** = `Roles`
- using **Error** = `delegate::Error`

### Public Member Functions

- virtual void [onListRolesSuccess](#) (`Roles &&roles`)=0
- virtual void [onListRolesError](#) (`Error &&error`, `cbe::util::Context &&context`)=0

#### 11.134.1 Detailed Description

Delegate class for the asynchronous version of method:

- `cbe::Group::ListRoles`

#### 11.134.2 Member Function Documentation

##### 11.134.2.1 onListRolesSuccess()

```
virtual void cbe::delegate::ListRolesDelegate::onListRolesSuccess (
 Roles && roles) [pure virtual]
```

Called upon successful ListRoles.

##### Parameters

|              |                                                                        |
|--------------|------------------------------------------------------------------------|
| <i>roles</i> | A unordered map with role id as key and role name as value of a group. |
|--------------|------------------------------------------------------------------------|

##### 11.134.2.2 onListRolesError()

```
virtual void cbe::delegate::ListRolesDelegate::onListRolesError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- `include/cbe/delegate/ListRolesDelegate.h`

## 11.135 cbe::delegate::ListSharesDelegate Class Reference

```
#include <ListSharesDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::ShareManager::listAvailableShares\(\)](#) or [cbe::ShareManager::listMyShares\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::QueryResult](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onListSharesSuccess](#) ([cbe::QueryResult](#) &&qResult)=0
- virtual void [onListSharesError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

#### 11.135.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::ShareManager::listMyShares](#)

#### 11.135.2 Member Function Documentation

##### 11.135.2.1 onListSharesSuccess()

```
virtual void cbe::delegate::ListSharesDelegate::onListSharesSuccess (
 cbe::QueryResult && qResult) [pure virtual]
```

Called upon successful Share.

##### Parameters

|                |                                                                         |
|----------------|-------------------------------------------------------------------------|
| <i>qResult</i> | Instance of <a href="#">cbe::QueryResult</a> containing list of shares. |
|----------------|-------------------------------------------------------------------------|

##### 11.135.2.2 onListSharesError()

```
virtual void cbe::delegate::ListSharesDelegate::onListSharesError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/ListSharesDelegate.h

## 11.136 cbe::delegate::LoadError Class Reference

```
#include <ItemDelegateErrors.h>
```

## Classes

- class [Error](#)

## Public Member Functions

- virtual void [onLoadError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

## Friends

- `std::ostream & operator<< (std::ostream &os, const LoadError::Error &error)`

### 11.136.1 Detailed Description

Entity class containing the information passed into method **onLoadError()** in cbe interface

### 11.136.2 Member Function Documentation

#### 11.136.2.1 onLoadError()

```
virtual void cbe::delegate::LoadError::onLoadError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called upon a failed query() or join() call.

#### Parameters

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| <i>error</i>   | <a href="#">Error</a> information passed from CloudBackend SDK.                 |
| <i>context</i> | Additional context information about the original service call that has failed. |

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

- include/cbe/delegate/ItemDelegateErrors.h

## 11.137 cbe::delegate::LoginDelegate Class Reference

```
#include <LoginDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::CloudBackend::login](#)(const std::string&,const std::string&, const std::string&) if the request fails.*

## Public Types

- using **Error** = [cbe::delegate::Error](#)
- using **Success** = [CloudBackend](#)

## Public Member Functions

- virtual void [onLoginSuccess](#) ([CloudBackend](#) &&cloudBackend)=0
- virtual void [onLoginError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.137.1 Detailed Description

Delegate class for the asynchronous version of method:

- `cbe::CloudBackend::login(const std::string&,const std::string&,const std::string&,cbe::delegate::LoginDelegatePtr)`  
See [Example of asynchronous login](#)

### 11.137.2 Member Typedef Documentation

#### 11.137.2.1 Success

using `cbe::delegate::LoginDelegate::Success = CloudBackend`

Type of object delivered on success. I.e., the object delivered through success callback `onLoginSuccess(CloudBackend&&)`.

### 11.137.3 Member Function Documentation

#### 11.137.3.1 onLoginSuccess()

```
virtual void cbe::delegate::LoginDelegate::onLoginSuccess (
 CloudBackend && cloudBackend) [pure virtual]
```

Called upon successful log in.

##### Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <i>cloudBackend</i> | Instance of a <a href="#">CloudBackend</a> holding the session. |
|---------------------|-----------------------------------------------------------------|

#### 11.137.3.2 onLoginError()

```
virtual void cbe::delegate::LoginDelegate::onLoginError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called upon failed log in.

##### Parameters

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| <i>error</i>   | <a href="#">Error</a> information passed from CloudBackend SDK.                 |
| <i>context</i> | Additional context information about the original service call that has failed. |

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

- `include/cbe/delegate/LoginDelegate.h`

## 11.138 cbe::MatchesID< IdT > Class Template Reference

Search lists with ID.

```
#include <Utility.h>
```

### Public Member Functions

- **MatchesID** (const IdT &id)
- `template<class ItemT >`  
`bool operator() (const ItemT &item) const`

### 11.138.1 Detailed Description

```
template<typename IdT>
class cbe::MatchesID< IdT >
```

Search lists with ID.

Example:

```
bool alreadyExists = std::find_if(vector.begin(), vector.end(),
 cbe::MatchesID<uint64_t>(id)) != vector.end();
```

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

- include/cbe/Utility.h

## 11.139 cbe::Member Class Reference

A of member of a [Group](#).

```
#include <Member.h>
```

### Public Types

- using [KickDelegatePtr](#) = [delegate::KickDelegatePtr](#)
- using [KickException](#) = [delegate::KickDelegate::Exception](#)
- using [KickError](#) = [delegate::KickDelegate::ErrorInfo](#)
- using [BanDelegatePtr](#) = [delegate::BanDelegatePtr](#)
- using [BanException](#) = [delegate::BanDelegate::Exception](#)
- using [BanError](#) = [delegate::BanDelegate::ErrorInfo](#)
- using [UnBanDelegatePtr](#) = [delegate::UnBanDelegatePtr](#)
- using [UnBanException](#) = [delegate::UnBanDelegate::Exception](#)
- using [UnBanError](#) = [delegate::UnBanDelegate::ErrorInfo](#)

### Public Member Functions

- void [kick](#) (std::string kickComment, [KickDelegatePtr](#) delegate)
- [delegate::KickSuccess](#) [kick](#) (std::string kickComment)
 

*Synchronous [exception] **Synchronous** version of [kick\(std::string kickComment, KickDelegatePtr\)](#) , and **throws an exception**, [KickException](#), in case of a failed call.*

*See [kick\(KickDelegatePtr\)](#)*
- [cbe::util::Optional](#)< [delegate::KickSuccess](#) > [kick](#) (std::string kickComment, [KickError](#) &error)
 

*Synchronous [non-throwing] **Synchronous** version of [kick\(kickComment, KickDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter *error* is used to provide the error information in connection with a failed call.*

*See [kick\(kickComment, KickDelegatePtr\)](#)*
- void [ban](#) (std::string banComment, [BanDelegatePtr](#) delegate)
- [delegate::BanSuccess](#) [ban](#) (std::string banComment)
 

*Synchronous [exception] **Synchronous** version of [ban\(std::string, BanDelegatePtr\)](#) , and **throws an exception**, [BanException](#), in case of a failed call.*

*See [ban\(BanDelegatePtr\)](#)*
- [cbe::util::Optional](#)< [delegate::BanSuccess](#) > [ban](#) (std::string banComment, [BanError](#) &error)
 

*Synchronous [non-throwing] **Synchronous** version of [ban\(banComment, BanDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter *error* is used to provide the error information in connection with a failed call.*

*See [ban\(banComment, BanDelegatePtr\)](#)*
- void [unBan](#) ([UnBanDelegatePtr](#) delegate)
- [delegate::UnBanSuccess](#) [unBan](#) ()
 

*Synchronous [exception] **Synchronous** version of [unBan\(UnBanDelegatePtr\)](#) , and **throws an exception**, [UnBanException](#), in case of a failed call.*

*See [unBan\(UnBanDelegatePtr\)](#)*
- [cbe::util::Optional](#)< [delegate::UnBanSuccess](#) > [unBan](#) ([UnBanError](#) &error)

Synchronous [non-throwing] **Synchronous** version of [unBan\(UnBanDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [unBan\(UnBanDelegatePtr\)](#)

- [cbe::MemberId memberId](#) () const
- [std::string name](#) () const
- [cbe::Visibility visibility](#) () const
- [cbe::GroupId groupId](#) () const
- [cbe::MemberBanInfo getMemberBanInfo](#) ()

Gets the *member* ban info map.

- [Member](#) ([cbe::DefaultCtor](#))

Default constructor.

- [operator bool](#) () const

Checks if the current instance is real.

## Friends

- class **Group**
- class **Role**

### 11.139.1 Detailed Description

A of member of a [Group](#).

### 11.139.2 Member Typedef Documentation

#### 11.139.2.1 KickDelegatePtr

```
using cbe::Member::KickDelegatePtr = delegate::KickDelegatePtr
```

Pointer to KickDelegate that is passed into:

- [cbe::Member::kick](#)(std::string,KickDelegatePtr)

#### 11.139.2.2 KickException

```
using cbe::Member::KickException = delegate::KickDelegate::Exception
```

Pointer to [cbe::delegate::KickDelegate](#) that is passed into asynchronous version of method `kick()` See [delegate↔::object::KickDelegate::Exception](#)

#### 11.139.2.3 KickError

```
using cbe::Member::KickError = delegate::KickDelegate::ErrorInfo
```

Forms the type of the `error` return parameter for the synchronous version of method `kick()`

See [delegate::KickDelegate::ErrorInfo](#)

#### 11.139.2.4 BanDelegatePtr

```
using cbe::Member::BanDelegatePtr = delegate::BanDelegatePtr
```

Pointer to BanDelegate that is passed into:

- [cbe::Member::ban](#)(std::string,BanDelegatePtr)

### 11.139.2.5 BanException

using `cbe::Member::BanException = delegate::BanDelegate::Exception`

Pointer to `cbe::delegate::BanDelegate` that is passed into asynchronous version of method `ban()` See `delegate::object::BanDelegate::Exception`

### 11.139.2.6 BanError

using `cbe::Member::BanError = delegate::BanDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `ban()`

See `delegate::BanDelegate::ErrorInfo`

### 11.139.2.7 UnBanDelegatePtr

using `cbe::Member::UnBanDelegatePtr = delegate::UnBanDelegatePtr`

Pointer to `UnBanDelegate` that is passed into:

- `cbe::Member::unBan(UnBanDelegatePtr)`

### 11.139.2.8 UnBanException

using `cbe::Member::UnBanException = delegate::UnBanDelegate::Exception`

Pointer to `cbe::delegate::UnBanDelegate` that is passed into asynchronous version of method `unBan()` See `delegate::object::UnBanDelegate::Exception`

### 11.139.2.9 UnBanError

using `cbe::Member::UnBanError = delegate::UnBanDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `unBan()`

See `delegate::UnBanDelegate::ErrorInfo`

## 11.139.3 Constructor & Destructor Documentation

### 11.139.3.1 Member()

```
cbe::Member::Member (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the `operator bool()` test

## 11.139.4 Member Function Documentation

### 11.139.4.1 kick() [1/3]

```
void cbe::Member::kick (
 std::string kickComment,
 KickDelegatePtr delegate)
```

Kick out a member from a group

#### Parameters

|                    |                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <i>kickComment</i> | free text describing the reason for this decision                                                                                           |
| <i>delegate</i>    | a shared pointer to the class in which you implement <code>cbe::delegate::KickDelegate</code> to receive the callback on server completion. |

**11.139.4.2 kick()** [2/3]

```
delegate::KickSuccess cbe::Member::kick (
 std::string kickComment)
```

Synchronous [exception] **Synchronous** version of `kick(std::string kickComment, KickDelegatePtr)` , and **throws an exception**, `KickException`, in case of a failed call.

See `kick(KickDelegatePtr)`

**Returns**

Information about the kick object — if the call was successful.

See `cbe::delegate::KickSuccess`

**Exceptions**

|                            |  |
|----------------------------|--|
| <code>KickException</code> |  |
|----------------------------|--|

**11.139.4.3 kick()** [3/3]

```
cbe::util::Optional<delegate::KickSuccess> cbe::Member::kick (
 std::string kickComment,
 KickError & error)
```

Synchronous [non-throwing] **Synchronous** version of `kick(kickComment, KickDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `kick(kickComment, KickDelegatePtr)`

**Parameters**

|                  |                    |                                                                                                                                                                                                                           |
|------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <code>KickError</code> object passed in will be populated with the error information. |
|------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.139.4.4 ban()** [1/3]

```
void cbe::Member::ban (
 std::string banComment,
 BanDelegatePtr delegate)
```

Ban or evict a member from a group

**Parameters**

|                         |                                                                                                                                                     |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>banComment</code> | Free text describing the reason for this decision                                                                                                   |
| <code>delegate</code>   | Shared pointer to the class in which you implement <code>cbe::delegate::BanDelegate</code> to receive the callback upon completion of this request. |

**11.139.4.5 ban()** [2/3]

```
delegate::BanSuccess cbe::Member::ban (
 std::string banComment)
```

Synchronous [exception] **Synchronous** version of [ban\(std::string,BanDelegatePtr\)](#) , and **throws an exception**, [BanException](#), in case of a failed call.

See [ban\(BanDelegatePtr\)](#)

**Returns**

Information about the ban object — if the call was successful.

See [cbe::delegate::BanSuccess](#)

**Exceptions**

|                              |  |
|------------------------------|--|
| <a href="#">BanException</a> |  |
|------------------------------|--|

**11.139.4.6 ban()** [3/3]

```
cbe::util::Optional<delegate::BanSuccess> cbe::Member::ban (
 std::string banComment,
 BanError & error)
```

Synchronous [non-throwing] **Synchronous** version of [ban\(banComment,BanDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [ban\(banComment, BanDelegatePtr\)](#)

**Parameters**

|     |       |                                                                                                                                                                                                                             |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">BanError</a> object passed in will be populated with the error information. |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.139.4.7 unBan()** [1/3]

```
void cbe::Member::unBan (
 UnBanDelegatePtr delegate)
```

Revoke a ban of a member from a group

**Parameters**

|          |                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| delegate | a shared pointer to the class in which you implement <a href="#">cbe::delegate::UnBanDelegate</a> to receive the callback on server completion. |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|

**11.139.4.8 unBan()** [2/3]

```
delegate::UnBanSuccess cbe::Member::unBan ()
```

Synchronous [exception] **Synchronous** version of [unBan\(UnBanDelegatePtr\)](#) , and **throws an exception**, [UnBanException](#), in case of a failed call.

See [unBan\(UnBanDelegatePtr\)](#)

#### Returns

Information about the unBan object — if the call was successful.

See [cbe::delegate::UnBanSuccess](#)

#### Exceptions

|                                |
|--------------------------------|
| <a href="#">UnBanException</a> |
|--------------------------------|

#### 11.139.4.9 unBan() [3/3]

```
cbe::util::Optional<delegate::UnBanSuccess> cbe::Member::unBan (
 UnBanError & error)
```

Synchronous [non-throwing] **Synchronous** version of [unBan\(UnBanDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [unBan\(UnBanDelegatePtr\)](#)

#### Parameters

|     |       |                                                                                                                                                                                                                               |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UnBanError</a> object passed in will be populated with the error information. |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.139.4.10 memberId()

```
cbe::MemberId cbe::Member::memberId () const
```

the member id number

#### 11.139.4.11 name()

```
std::string cbe::Member::name () const
```

Returns the member name.

#### 11.139.4.12 visibility()

```
cbe::Visibility cbe::Member::visibility () const
```

see the enum [Visibility](#) in [Types.h](#)

#### 11.139.4.13 groupId()

```
cbe::GroupId cbe::Member::groupId () const
```

the group id number

#### 11.139.4.14 operator bool()

```
cbe::Member::operator bool () const [explicit]
```

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [Member\(cbe::DefaultCtor\)](#)

##### Returns

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/Member.h

### 11.140 cbe::delegate::container::MoveDelegate Class Reference

```
#include <MoveDelegate.h>
```

#### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Container::move\(\)](#) if the request fails.*

#### Public Types

- using **Success** = [cbe::Container](#)
- using **Error** = [delegate::Error](#)

#### Public Member Functions

- virtual void [onMoveSuccess](#) ([cbe::Container](#) &&container)=0
- virtual void [onMoveError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

#### 11.140.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Container::move](#)

#### 11.140.2 Member Function Documentation

##### 11.140.2.1 onMoveSuccess()

```
virtual void cbe::delegate::container::MoveDelegate::onMoveSuccess (
 cbe::Container && container) [pure virtual]
```

Called upon successful Move.

##### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <i>container</i> | Instance of container that is being moved. |
|------------------|--------------------------------------------|

### 11.140.2.2 onMoveError()

```
virtual void cbe::delegate::container::MoveDelegate::onMoveError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/container/MoveDelegate.h

## 11.141 cbe::delegate::object::MoveDelegate Class Reference

```
#include <MoveDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Object::move\(\)](#) if the request fails.*

### Public Types

- using **Success** = [cbe::Object](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onMoveSuccess](#) ([cbe::Object](#) &&object)=0
- virtual void [onMoveError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.141.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Object::move](#)

### 11.141.2 Member Function Documentation

#### 11.141.2.1 onMoveSuccess()

```
virtual void cbe::delegate::object::MoveDelegate::onMoveSuccess (
 cbe::Object && object) [pure virtual]
```

Called upon successful Move.

#### Parameters

|               |                                         |
|---------------|-----------------------------------------|
| <i>object</i> | Instance of object that is being moved. |
|---------------|-----------------------------------------|

#### 11.141.2.2 onMoveError()

```
virtual void cbe::delegate::object::MoveDelegate::onMoveError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

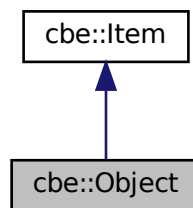
- include/cbe/delegate/object/MoveDelegate.h

## 11.142 cbe::Object Class Reference

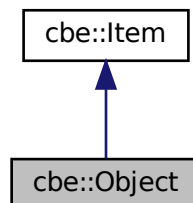
Holder of a set of data, can represent a table row.

```
#include <Object.h>
```

Inheritance diagram for cbe::Object:



Collaboration diagram for cbe::Object:



## Public Types

- using [MoveDelegatePtr](#) = [delegate::object::MoveDelegatePtr](#)
- using [MoveException](#) = [delegate::object::MoveDelegate::Exception](#)
- using [MoveError](#) = [delegate::object::MoveDelegate::ErrorInfo](#)
- using [RenameDelegatePtr](#) = [delegate::object::RenameDelegatePtr](#)
- using [RenameException](#) = [delegate::object::RenameDelegate::Exception](#)
- using [RenameError](#) = [delegate::object::RenameDelegate::ErrorInfo](#)
- using [RemoveDelegatePtr](#) = [delegate::object::RemoveDelegatePtr](#)
- using [RemoveException](#) = [delegate::object::RemoveDelegate::Exception](#)
- using [RemoveError](#) = [delegate::object::RemoveDelegate::ErrorInfo](#)
- using [DownloadDelegatePtr](#) = [delegate::DownloadDelegatePtr](#)
- using [DownloadException](#) = [delegate::DownloadDelegate::Exception](#)
- using [DownloadError](#) = [delegate::DownloadDelegate::ErrorInfo](#)
- using [DownloadBinaryDelegatePtr](#) = [delegate::DownloadBinaryDelegatePtr](#)

- using `DownloadBinaryException` = `delegate::DownloadBinaryDelegate::Exception`
- using `DownloadBinaryError` = `delegate::DownloadBinaryDelegate::ErrorInfo`
- using `UploadDelegatePtr` = `delegate::UploadDelegatePtr`
- using `UploadException` = `delegate::UploadDelegate::Exception`
- using `UploadError` = `delegate::UploadDelegate::ErrorInfo`
- using `UpdateKeyValuesDelegatePtr` = `delegate::UpdateKeyValuesDelegatePtr`
- using `UpdateKeyValuesException` = `delegate::UpdateKeyValuesDelegate::Exception`
- using `UpdateKeyValuesError` = `delegate::UpdateKeyValuesDelegate::ErrorInfo`
- using `GetStreamsDelegatePtr` = `delegate::GetStreamsDelegatePtr`
- using `Streams` = `cbe::Streams`

*Collection of `Stream` objects.*

- using `GetStreamsException` = `delegate::GetStreamsDelegate::Exception`
- using `GetStreamsError` = `delegate::GetStreamsDelegate::ErrorInfo`
- using `GetAclDelegatePtr` = `delegate::AclDelegatePtr`
- using `GetAclException` = `delegate::AclDelegate::Exception`
- using `GetAclError` = `delegate::AclDelegate::ErrorInfo`
- using `SetAclDelegatePtr` = `delegate::AclDelegatePtr`
- using `SetAclException` = `delegate::AclDelegate::Exception`
- using `SetAclError` = `delegate::AclDelegate::ErrorInfo`
- using `ShareDelegatePtr` = `delegate::ShareDelegatePtr`
- using `ShareException` = `delegate::ShareDelegate::Exception`
- using `ShareError` = `delegate::ShareDelegate::ErrorInfo`
- using `UnShareDelegatePtr` = `delegate::UnShareDelegatePtr`
- using `UnShareException` = `delegate::UnShareDelegate::Exception`
- using `UnShareError` = `delegate::UnShareDelegate::ErrorInfo`
- using `PublishDelegatePtr` = `delegate::PublishDelegatePtr`
- using `PublishException` = `delegate::PublishDelegate::Exception`
- using `PublishError` = `delegate::PublishDelegate::ErrorInfo`
- using `UnPublishDelegatePtr` = `delegate::UnPublishDelegatePtr`
- using `UnPublishException` = `delegate::UnPublishDelegate::Exception`
- using `UnPublishError` = `delegate::UnPublishDelegate::ErrorInfo`
- using `UnSubscribeDelegatePtr` = `delegate::UnSubscribeDelegatePtr`
- using `UnSubscribeException` = `delegate::UnSubscribeDelegate::Exception`
- using `UnSubscribeError` = `delegate::UnSubscribeDelegate::ErrorInfo`

## Public Member Functions

- void `move` (`cbe::ContainerId` dstId, `MoveDelegatePtr` delegate)  
*Relocates an object to a different container.*
- `cbe::Object` `move` (`cbe::ContainerId` dstId)
- `cbe::util::Optional< cbe::Object >` `move` (`cbe::ContainerId` dstId, `MoveError` &error)
- void `rename` (const std::string &name, `RenameDelegatePtr` delegate)  
*Rename object.*
- `cbe::Object` `rename` (const std::string &name)
- `cbe::util::Optional< cbe::Object >` `rename` (const std::string &name, `RenameError` &error)
- void `remove` (`RemoveDelegatePtr` delegate)  
*Remove the object from cloud and locally.*
- `delegate::object::RemoveSuccess` `remove` ()
- `cbe::util::Optional< delegate::object::RemoveSuccess >` `remove` (`RemoveError` &error)
- void `download` (const std::string &path, `DownloadDelegatePtr` delegate)  
*Download the data of current object to the the local file system.*
- `delegate::DownloadSuccess` `download` (const std::string &path, `delegate::ProgressEventFn` &&progress↵  
EventFn)
- `delegate::DownloadSuccess` `download` (const std::string &path)

- [cbe::util::Optional](#)< [delegate::DownloadSuccess](#) > [download](#) (const std::string &path, [delegate::ProgressEventFn](#) &&progressEventFn, [DownloadError](#) &error)
- [cbe::util::Optional](#)< [delegate::DownloadSuccess](#) > [download](#) (const std::string &path, [DownloadError](#) &error)
- void [download](#) (std::size\_t &&sizeLimit, [DownloadBinaryDelegatePtr](#) delegate)
 

*Download the binary data associated with current object.*
- [delegate::DownloadBinarySuccess](#) [download](#) (std::size\_t &&sizeLimit, [delegate::ProgressEventFn](#) &&progressEventFn)
- [delegate::DownloadBinarySuccess](#) [download](#) (std::size\_t &&sizeLimit)
- [cbe::util::Optional](#)< [delegate::DownloadBinarySuccess](#) > [download](#) (std::size\_t &&sizeLimit, [delegate::ProgressEventFn](#) &&progressEventFn, [DownloadBinaryError](#) &error)
- [cbe::util::Optional](#)< [delegate::DownloadBinarySuccess](#) > [download](#) (std::size\_t &&sizeLimit, [DownloadBinaryError](#) &error)
- void [downloadStream](#) (const std::string &path, [cbe::Stream](#) stream, [DownloadDelegatePtr](#) delegate)
- [delegate::DownloadSuccess](#) [downloadStream](#) (const std::string &path, [cbe::Stream](#) stream, [delegate::ProgressEventFn](#) &&progressEventFn)
- [delegate::DownloadSuccess](#) [downloadStream](#) (const std::string &path, [cbe::Stream](#) stream)
- [cbe::util::Optional](#)< [delegate::DownloadSuccess](#) > [downloadStream](#) (const std::string &path, [cbe::Stream](#) stream, [delegate::ProgressEventFn](#) &&progressEventFn, [DownloadError](#) &error)
- [cbe::util::Optional](#)< [delegate::DownloadSuccess](#) > [downloadStream](#) (const std::string &path, [cbe::Stream](#) stream, [DownloadError](#) &error)
- void [uploadStream](#) (const std::string &filePath, [cbe::StreamId](#) streamId, [UploadDelegatePtr](#) delegate)
 

*Upload a file for adding a new or replacing existing stream attached to this object.*
- [cbe::Object](#) [uploadStream](#) (const std::string &filePath, [cbe::StreamId](#) streamId, [delegate::ProgressEventFn](#) &&progressEventFn)
- [cbe::Object](#) [uploadStream](#) (const std::string &filePath, [cbe::StreamId](#) streamId)
- [cbe::util::Optional](#)< [cbe::Object](#) > [uploadStream](#) (const std::string &filePath, [cbe::StreamId](#) streamId, [delegate::ProgressEventFn](#) &&progressEventFn, [UploadError](#) &error)
- [cbe::util::Optional](#)< [cbe::Object](#) > [uploadStream](#) (const std::string &filePath, [cbe::StreamId](#) streamId, [UploadError](#) &error)
- void [updateKeyValues](#) ([KeyValues](#) keyValues, [UpdateKeyValuesDelegatePtr](#) delegate)
 

*Adds key/value pair data to the object.*
- void [updateKeyValues](#) ([UpdateKeyValuesDelegatePtr](#) delegate)
 

*Deletes all key/value pairs of data to the object.*
- [cbe::Object](#) [updateKeyValues](#) ([KeyValues](#) keyValues)
 

*Synchronous [exception] Adds key/value pair data to the object.*
- [cbe::Object](#) [updateKeyValues](#) ()
 

*Synchronous [exception] Deletes all key/value pairs of data to the object.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [updateKeyValues](#) ([KeyValues](#) keyValues, [UpdateKeyValuesError](#) &error)
 

*Synchronous [non-throwing] Adds key/value pair data to the object.*
- [cbe::util::Optional](#)< [cbe::Object](#) > [updateKeyValues](#) ([UpdateKeyValuesError](#) &error)
 

*Synchronous [non-throwing] Deletes all key/value pairs of data to the object.*
- void [getStreams](#) ([GetStreamsDelegatePtr](#) delegate)
 

*Downloads the streams meta data associated with current object to the SDK's cache.*
- [Streams](#) [getStreams](#) ()
- [cbe::util::Optional](#)< [Streams](#) > [getStreams](#) ([GetStreamsError](#) &error)
- std::string [getMimeType](#) () const
 

*Returns the mime type of the object.*
- uint64\_t [length](#) () const
 

*Returns the binary length/size in bytes of current object.*
- [cbe::object\\_t](#) [getObjectType](#) () const
 

*Returns the [Object](#) type.*
- [cbe::KeyValues](#) [keyValues](#) ()
 

*Returns all the key/values for current object as a map.*

- void `getAcl` (`GetAclDelegatePtr` delegate)  
Returns the Access Control List for current *Object*.
- `cbe::AclMap` `getAcl` ()
- `cbe::util::Optional` < `cbe::AclMap` > `getAcl` (`GetAclError` &error)
- void `setAcl` (`cbe::AclMap` `aclMap`, `SetAclDelegatePtr` delegate)  
Sets the Access Control List for current object.
- `cbe::AclMap` `setAcl` (`cbe::AclMap` `aclMap`)
- `cbe::util::Optional` < `cbe::AclMap` > `setAcl` (`cbe::AclMap` `aclMap`, `SetAclError` &error)
- void `share` (`cbe::UserId` `toUserGroup`, `std::string` `description`, `ShareDelegatePtr` delegate)  
Share current object to a user.
- `delegate::ShareSuccess` `share` (`cbe::UserId` `toUserGroup`, `std::string` `description`)
- `cbe::util::Optional` < `delegate::ShareSuccess` > `share` (`cbe::UserId` `toUserGroup`, `std::string` `description`, `ShareError` &error)
- void `unShare` (`cbe::ShareId` `shareId`, `UnShareDelegatePtr` delegate)  
Unshare the object to a specific shareId created when sharing.
- `delegate::UnShareSuccess` `unShare` (`cbe::ShareId` `shareId`)
- `cbe::util::Optional` < `delegate::UnShareSuccess` > `unShare` (`cbe::ShareId` `shareId`, `UnShareError` &error)
- void `publish` (`cbe::PublishAccess` `security`, `cbe::PublishVisibility` `privacy`, `std::string` `description`, `std::string` `password`, `PublishDelegatePtr` delegate)  
Publishes current object to any user.
- `delegate::PublishSuccess` `publish` (`cbe::PublishAccess` `security`, `cbe::PublishVisibility` `privacy`, `std::string` `description`, `std::string` `password`)
- `cbe::util::Optional` < `delegate::PublishSuccess` > `publish` (`cbe::PublishAccess` `security`, `cbe::PublishVisibility` `privacy`, `std::string` `description`, `std::string` `password`, `PublishError` &error)
- void `unPublish` (`UnPublishDelegatePtr` delegate)  
UnPublishes current object.
- `delegate::UnPublishSuccess` `unPublish` ()
- `cbe::util::Optional` < `delegate::UnPublishSuccess` > `unPublish` (`UnPublishError` &error)
- void `unsubscribe` (`UnSubscribeDelegatePtr` delegate)  
UnSubscribes from this object.
- `delegate::UnSubscribeSuccess` `unsubscribe` ()
- `cbe::util::Optional` < `delegate::UnSubscribeSuccess` > `unsubscribe` (`UnSubscribeError` &error)
- `std::string` `url` ()  
URL to current object.
- **Object** (`cbe::DefaultCtor`)

## Friends

- class **CloudBackend**
- class **Container**

## Additional Inherited Members

### 11.142.1 Detailed Description

Holder of a set of data, can represent a table row.

#### Note

The object name may not contain characters < & : /

Any key name must start with a letter or \_

The following key names are reserved and should not be used: category, content, id, link and date

Key names are case sensitive, hence variations with uppercase are permitted.

### 11.142.2 Member Typedef Documentation

#### 11.142.2.1 MoveDelegatePtr

using `cbe::Object::MoveDelegatePtr = delegate::object::MoveDelegatePtr`

Pointer to `cbe::delegate::object::MoveDelegate` that is passed into asynchronous version of method `move()`

#### 11.142.2.2 MoveException

using `cbe::Object::MoveException = delegate::object::MoveDelegate::Exception`

See `delegate::object::MoveDelegate::Exception`

#### 11.142.2.3 MoveError

using `cbe::Object::MoveError = delegate::object::MoveDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `move()`

See `delegate::object::MoveDelegate::ErrorInfo`

#### 11.142.2.4 RenameDelegatePtr

using `cbe::Object::RenameDelegatePtr = delegate::object::RenameDelegatePtr`

Pointer to `cbe::delegate::object::RenameDelegate` that is passed into asynchronous version of method `rename()`

#### 11.142.2.5 RenameException

using `cbe::Object::RenameException = delegate::object::RenameDelegate::Exception`

See `delegate::object::RenameDelegate::Exception`

#### 11.142.2.6 RenameError

using `cbe::Object::RenameError = delegate::object::RenameDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `rename()`

See `delegate::object::RenameDelegate::ErrorInfo`

#### 11.142.2.7 RemoveDelegatePtr

using `cbe::Object::RemoveDelegatePtr = delegate::object::RemoveDelegatePtr`

Pointer to `cbe::delegate::object::RemoveDelegate` that is passed into asynchronous version of method `remove()`

#### 11.142.2.8 RemoveException

using `cbe::Object::RemoveException = delegate::object::RemoveDelegate::Exception`

See `delegate::object::RemoveDelegate::Exception`

#### 11.142.2.9 RemoveError

using `cbe::Object::RemoveError = delegate::object::RemoveDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `remove()`

See `delegate::object::RemoveDelegate::ErrorInfo`

#### 11.142.2.10 DownloadDelegatePtr

using `cbe::Object::DownloadDelegatePtr = delegate::DownloadDelegatePtr`

Pointer to `cbe::delegate::DownloadDelegate` that is passed into asynchronous version of methods:

- `download(const std::string&, DownloadDelegatePtr)`
- `downloadStream(const std::string&, cbe::Stream, DownloadDelegatePtr)`

#### 11.142.2.11 DownloadException

using `cbe::Object::DownloadException` = `delegate::DownloadDelegate::Exception`

The exception type thrown by the synchronous version of methods:

- `download(const std::string&, delegate::ProgressEventFn&&)`
- `download(const std::string&)`
- `downloadStream(const std::string&, cbe::Stream, delegate::ProgressEventFn&&)`
- `downloadStream(const std::string&, cbe::Stream)`

See `delegate::DownloadDelegate::Exception`

#### 11.142.2.12 DownloadError

using `cbe::Object::DownloadError` = `delegate::DownloadDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of methods:

- `download(const std::string&, delegate::ProgressEventFn&&, DownloadError&)`
- `download(const std::string&, DownloadError&)`
- `downloadStream(const std::string&, cbe::Stream, delegate::ProgressEventFn&&, DownloadError&)`
- `downloadStream(const std::string&, cbe::Stream, DownloadError&)`

See `delegate::DownloadDelegate::ErrorInfo`

#### 11.142.2.13 DownloadBinaryDelegatePtr

using `cbe::Object::DownloadBinaryDelegatePtr` = `delegate::DownloadBinaryDelegatePtr`

Pointer to `cbe::delegate::DownloadBinaryDelegate` that is passed into asynchronous version of method `download()`

#### 11.142.2.14 DownloadBinaryException

using `cbe::Object::DownloadBinaryException` = `delegate::DownloadBinaryDelegate::Exception`

See `delegate::DownloadBinaryDelegate::Exception`

#### 11.142.2.15 DownloadBinaryError

using `cbe::Object::DownloadBinaryError` = `delegate::DownloadBinaryDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of methods:

- `download(delegate::ProgressEventFn&&, DownloadBinaryError&)`
- `download(DownloadBinaryError&)`

See `delegate::DownloadBinaryDelegate::ErrorInfo`

#### 11.142.2.16 UploadDelegatePtr

using `cbe::Object::UploadDelegatePtr` = `delegate::UploadDelegatePtr`

Pointer to `cbe::delegate::UploadDelegate` that is passed into asynchronous version of method `uploadStream()`

#### 11.142.2.17 UploadException

using `cbe::Object::UploadException` = `delegate::UploadDelegate::Exception`

See `delegate::object::UploadDelegate::Exception`

**11.142.2.18 UploadError**

using `cbe::Object::UploadError = delegate::UploadDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of methods:

- `uploadStream(const std::string&,cbe::StreamId,delegate::ProgressEventFn&&,UploadError&)`
- `uploadStream(const std::string&,cbe::StreamId,UploadError&)`

See `delegate::UploadDelegate::ErrorInfo`

**11.142.2.19 UpdateKeyValuesDelegatePtr**

using `cbe::Object::UpdateKeyValuesDelegatePtr = delegate::UpdateKeyValuesDelegatePtr`

Pointer to `cbe::delegate::UpdateKeyValuesDelegate` that is passed into asynchronous version of method `updateKeyValues()`

**11.142.2.20 UpdateKeyValuesException**

using `cbe::Object::UpdateKeyValuesException = delegate::UpdateKeyValuesDelegate::Exception`

See `delegate::object::UpdateKeyValuesDelegate::Exception`

**11.142.2.21 UpdateKeyValuesError**

using `cbe::Object::UpdateKeyValuesError = delegate::UpdateKeyValuesDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `updateKeyValues()`

See `delegate::UpdateKeyValuesDelegate::ErrorInfo`

**11.142.2.22 GetStreamsDelegatePtr**

using `cbe::Object::GetStreamsDelegatePtr = delegate::GetStreamsDelegatePtr`

Pointer to `cbe::delegate::GetStreamsDelegate` that is passed into asynchronous version of method `getStream()`

**11.142.2.23 Streams**

using `cbe::Object::Streams = cbe::Streams`

Collection of `Stream` objects.

See `cbe::Streams`

**11.142.2.24 GetStreamsException**

using `cbe::Object::GetStreamsException = delegate::GetStreamsDelegate::Exception`

See `delegate::GetStreamsDelegate::Exception`

**11.142.2.25 GetStreamsError**

using `cbe::Object::GetStreamsError = delegate::GetStreamsDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `getStreams(GetStreamsError&)`

See `delegate::GetStreamsDelegate::ErrorInfo`

**11.142.2.26 GetAclDelegatePtr**

using `cbe::Object::GetAclDelegatePtr = delegate::AclDelegatePtr`

Pointer to `cbe::delegate::AclDelegate` that is passed into asynchronous version of method `getAcl()`

**11.142.2.27 GetAclException**

using `cbe::Object::GetAclException = delegate::AclDelegate::Exception`

See `delegate::object::AclDelegate::Exception`

#### 11.142.2.28 GetAclError

using `cbe::Object::GetAclError = delegate::AclDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `getAcl()`

See `delegate::AclDelegate::ErrorInfo`

#### 11.142.2.29 SetAclDelegatePtr

using `cbe::Object::SetAclDelegatePtr = delegate::AclDelegatePtr`

Pointer to `cbe::delegate::AclDelegate` that is passed into asynchronous version of method `setAcl()`

#### 11.142.2.30 SetAclException

using `cbe::Object::SetAclException = delegate::AclDelegate::Exception`

See `delegate::object::AclDelegate::Exception`

#### 11.142.2.31 SetAclError

using `cbe::Object::SetAclError = delegate::AclDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `setAcl()`

See `delegate::AclDelegate::ErrorInfo`

#### 11.142.2.32 ShareDelegatePtr

using `cbe::Object::ShareDelegatePtr = delegate::ShareDelegatePtr`

Pointer to `cbe::delegate::ShareDelegate` that is passed into asynchronous version of method `share()`

#### 11.142.2.33 ShareException

using `cbe::Object::ShareException = delegate::ShareDelegate::Exception`

See `delegate::object::ShareDelegate::Exception`

#### 11.142.2.34 ShareError

using `cbe::Object::ShareError = delegate::ShareDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `share()`

See `delegate::ShareDelegate::ErrorInfo`

#### 11.142.2.35 UnShareDelegatePtr

using `cbe::Object::UnShareDelegatePtr = delegate::UnShareDelegatePtr`

Pointer to `cbe::delegate::UnShareDelegate` that is passed into asynchronous version of method `unShare()`

#### 11.142.2.36 UnShareException

using `cbe::Object::UnShareException = delegate::UnShareDelegate::Exception`

See `delegate::object::UnShareDelegate::Exception`

#### 11.142.2.37 UnShareError

using `cbe::Object::UnShareError = delegate::UnShareDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `unShare()`

See `delegate::UnShareDelegate::ErrorInfo`

#### 11.142.2.38 PublishDelegatePtr

using `cbe::Object::PublishDelegatePtr = delegate::PublishDelegatePtr`

Pointer to `cbe::delegate::PublishDelegate` that is passed into asynchronous version of method `publish()`

### 11.142.2.39 PublishException

using `cbe::Object::PublishException = delegate::PublishDelegate::Exception`  
 See `delegate::object::PublishDelegate::Exception`

### 11.142.2.40 PublishError

using `cbe::Object::PublishError = delegate::PublishDelegate::ErrorInfo`  
 Forms the type of the `error` return parameter for the synchronous version of method `publish()`  
 See `delegate::PublishDelegate::ErrorInfo`

### 11.142.2.41 UnPublishDelegatePtr

using `cbe::Object::UnPublishDelegatePtr = delegate::UnPublishDelegatePtr`  
 Pointer to `cbe::delegate::UnPublishDelegate` that is passed into asynchronous version of method `unPublish()`

### 11.142.2.42 UnPublishException

using `cbe::Object::UnPublishException = delegate::UnPublishDelegate::Exception`  
 See `delegate::object::UnPublishDelegate::Exception`

### 11.142.2.43 UnPublishError

using `cbe::Object::UnPublishError = delegate::UnPublishDelegate::ErrorInfo`  
 Forms the type of the `error` return parameter for the synchronous version of method `unPublish()`  
 See `delegate::UnPublishDelegate::ErrorInfo`

### 11.142.2.44 UnSubscribeDelegatePtr

using `cbe::Object::UnSubscribeDelegatePtr = delegate::UnSubscribeDelegatePtr`  
 Pointer to `cbe::delegate::UnSubscribeDelegate` that is passed into asynchronous version of method `unSubscribe()`

### 11.142.2.45 UnSubscribeException

using `cbe::Object::UnSubscribeException = delegate::UnSubscribeDelegate::Exception`  
 See `delegate::object::UnSubscribeDelegate::Exception`

### 11.142.2.46 UnSubscribeError

using `cbe::Object::UnSubscribeError = delegate::UnSubscribeDelegate::ErrorInfo`  
 Forms the type of the `error` return parameter for the synchronous version of method `unSubscribe()`  
 See `delegate::UnSubscribeDelegate::ErrorInfo`

## 11.142.3 Member Function Documentation

### 11.142.3.1 move() [1/3]

```
void cbe::Object::move (
 cbe::ContainerId dstId,
 MoveDelegatePtr delegate)
```

Relocates an object to a different container.

**Asynchronous** version of this service function.

#### Parameters

|                 |                                                                                                    |
|-----------------|----------------------------------------------------------------------------------------------------|
| <i>dstId</i>    | Id of the destination container.                                                                   |
| <i>delegate</i> | Pointer to a <code>delegate::object::MoveDelegate</code> instance that is implemented by the user. |

**11.142.3.2 move()** [2/3]

```
cbe::Object cbe::Object::move (
 cbe::ContainerId dstId)
```

**Synchronous** version of [move\(cbe::ContainerId,MoveDelegatePtr\)](#) , and **throws an exception**, [MoveException](#), in case of a failed call.

See [move\(cbe::ContainerId,MoveDelegatePtr\)](#)

**Returns**

The moved object — if the call was successful.

**Exceptions**

|                               |  |
|-------------------------------|--|
| <a href="#">MoveException</a> |  |
|-------------------------------|--|

**11.142.3.3 move()** [3/3]

```
cbe::util::Optional<cbe::Object> cbe::Object::move (
 cbe::ContainerId dstId,
 MoveError & error)
```

**Synchronous** version of [move\(cbe::ContainerId,MoveDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [move\(cbe::ContainerId,MoveDelegatePtr\)](#)

**Parameters**

|     |       |                                                                                                                                                                                                                              |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">MoveError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.142.3.4 rename()** [1/3]

```
void cbe::Object::rename (
 const std::string & name,
 RenameDelegatePtr delegate)
```

Rename object.

**Asynchronous** version of this service function.

**Parameters**

|                 |                                                                                                         |
|-----------------|---------------------------------------------------------------------------------------------------------|
| <i>name</i>     | string name of the object.                                                                              |
| <i>delegate</i> | Pointer to a <a href="#">delegate::object::RenameDelegate</a> instance that is implemented by the user. |

**11.142.3.5 rename()** [2/3]

```
cbe::Object cbe::Object::rename (
 const std::string & name)
```

**Synchronous** version of [rename\(const std::string&,RenameDelegatePtr\)](#) , and **throws an exception**, [RenameException](#), in case of a failed call.

See [rename\(const std::string&,RenameDelegatePtr\)](#)

**Returns**

The renamed object — if the call was successful.

**Exceptions**

|                                 |  |
|---------------------------------|--|
| <a href="#">RenameException</a> |  |
|---------------------------------|--|

**11.142.3.6 rename()** [3/3]

```
cbe::util::Optional<cbe::Object> cbe::Object::rename (
 const std::string & name,
 RenameError & error)
```

**Synchronous** version of [rename\(const std::string&,RenameDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [rename\(const std::string&,RenameDelegatePtr\)](#)

**Parameters**

|     |       |                                                                                                                                                                                                                                |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">RenameError</a> object passed in will be populated with the error information. |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.142.3.7 remove()** [1/3]

```
void cbe::Object::remove (
 RemoveDelegatePtr delegate)
```

Remove the object from cloud and locally.

**Asynchronous** version of this service function.

**Parameters**

|                 |                                                                                                         |
|-----------------|---------------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::object::RemoveDelegate</a> instance that is implemented by the user. |
|-----------------|---------------------------------------------------------------------------------------------------------|

**11.142.3.8 remove()** [2/3]

```
delegate::object::RemoveSuccess cbe::Object::remove ()
```

**Synchronous** version of [remove\(RemoveDelegatePtr\)](#) , and **throws an exception**, [RemoveException](#), in case of a failed call.

See [remove\(RemoveDelegatePtr\)](#)

## Returns

Information about the removed object — if the call was successful.

See [RemoveSuccess](#)

## Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">RemoveException</a> |  |
|---------------------------------|--|

## 11.142.3.9 remove() [3/3]

```
cbe::util::Optional<delegate::object::RemoveSuccess> cbe::Object::remove (
 RemoveError & error)
```

**Synchronous** version of [remove\(RemoveDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [remove\(RemoveDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">RemoveError</a> object passed in will be populated with the error information. |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

## 11.142.3.10 download() [1/10]

```
void cbe::Object::download (
 const std::string & path,
 DownloadDelegatePtr delegate)
```

Download the data of current object to the the local file system.

**Asynchronous** version of this service function.

The data will be contained in file, named after the name of current object (method [name\(\)](#)), to the location given by parameter `path`.

## Parameters

|                 |                                                                                                                   |
|-----------------|-------------------------------------------------------------------------------------------------------------------|
| <i>path</i>     | Folder location, on the local file system, of the file to be downloaded. This string must end with a slash ("/"). |
| <i>delegate</i> | Pointer to a <a href="#">delegate::DownloadDelegate</a> instance that is implemented by the user.                 |

## 11.142.3.11 download() [2/10]

```
delegate::DownloadSuccess cbe::Object::download (
 const std::string & path,
 delegate::ProgressEventFn && progressEventFn)
```

**Synchronous** version of [download\(const std::string&,DownloadDelegatePtr\)](#), and **throws an exception**, [DownloadException](#), in case of a failed call.

See [download\(const std::string&,DownloadDelegatePtr\)](#)

#### Parameters

|                        |                                                                                                                                                                                                                                                                                                            |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>progressEventFn</i> | <p>Callback function that is called for each memory chunk downloaded. The callback function will be executed the calling thread's context. Also see <a href="#">cbe::delegate::ProgressEventFn</a>. For the other parameters, see <a href="#">download(const std::string&amp;,DownloadDelegatePtr)</a></p> |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Information about the download object — if the call was successful.  
See [DownloadSuccess](#)

#### Note

To use this method, header file [cbe/delegate/DownloadSuccess.h](#) must be included

#### Exceptions

|                                   |
|-----------------------------------|
| <a href="#">DownloadException</a> |
|-----------------------------------|

#### 11.142.3.12 download() [3/10]

```
delegate::DownloadSuccess cbe::Object::download (
 const std::string & path)
```

Same as [download\(const std::string&,delegate::ProgressEventFn&&\)](#), but without the parameter, `progressEventFn`.

#### 11.142.3.13 download() [4/10]

```
cbe::util::Optional<delegate::DownloadSuccess> cbe::Object::download (
 const std::string & path,
 delegate::ProgressEventFn && progressEventFn,
 DownloadError & error)
```

Similar to synchronous method [download\(const std::string&,delegate::ProgressEventFn&&\)](#), but **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [download\(const std::string&,delegate::ProgressEventFn&&\)](#)

#### Parameters

|            |              |                                                                                                                                                                                                                                                                                                                                                         |
|------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | <p>Return parameter containing the error information in case of a failed call. An empty return value will indicate failure, and the <a href="#">DownloadError</a> object passed in will be populated with the error information. For the other parameters, see <a href="#">download(const std::string&amp;,delegate::ProgressEventFn&amp;&amp;)</a></p> |
|------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.142.3.14 download()** [5/10]

```
cbe::util::Optional<delegate::DownloadSuccess> cbe::Object::download (
 const std::string & path,
 DownloadError & error)
```

Same as [download\(const std::string&,delegate::ProgressEventFn&&,DownloadError&\)](#) , but without the parameter, `progressEventFn`.

**11.142.3.15 download()** [6/10]

```
void cbe::Object::download (
 std::size_t && sizeLimit,
 DownloadBinaryDelegatePtr delegate)
```

Download the binary data associated with current object.

**Asynchronous** version of this service function.

The data, delivered as a BLOB (Binary Large Object), via parameter `data` in the the callback method [cbe::delegate::DownloadBinaryDelegate::onDownloadBinarySuccess\(\)](#).

**Parameters**

|                  |                                                                                                                                  |
|------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <i>delegate</i>  | Pointer to a <a href="#">delegate::DownloadBinaryDelegate</a> instance that is implemented by the user.                          |
| <i>sizeLimit</i> | Blocks anything larger than the size limit the user inputs. Prevents accidental downloads of too large objects on to the device. |

**11.142.3.16 download()** [7/10]

```
delegate::DownloadBinarySuccess cbe::Object::download (
 std::size_t && sizeLimit,
 delegate::ProgressEventFn && progressEventFn)
```

**Synchronous** version of [download\(DownloadBinaryDelegatePtr\)](#) , and **throws an exception**, [DownloadBinaryException](#), in case of a failed call.

See [download\(DownloadBinaryDelegatePtr\)](#)

**Parameters**

|                        |                                                                                                                                                                                 |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>progressEventFn</i> | See <a href="#">download(const std::string&amp;,delegate::ProgressEventFn&amp;&amp;)</a> .<br>For the other parameters, see <a href="#">download(DownloadBinaryDelegatePtr)</a> |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Information about the downloaded object — if the call was successful.

See [DownloadBinarySuccess](#)

**Note**

To use this method, header file  
[cbe/delegate/DownloadBinarySuccess.h](#)  
must be included

**Exceptions**

|                                         |  |
|-----------------------------------------|--|
| <a href="#">DownloadBinaryException</a> |  |
|-----------------------------------------|--|

**11.142.3.17 download()** [8/10]

```
delegate::DownloadBinarySuccess cbe::Object::download (
 std::size_t && sizeLimit)
```

Same as `download(delegate::ProgressEventFn&&)` , but without the parameter, `progressEventFn`.

**11.142.3.18 download()** [9/10]

```
cbe::util::Optional<delegate::DownloadBinarySuccess> cbe::Object::download (
 std::size_t && sizeLimit,
 delegate::ProgressEventFn && progressEventFn,
 DownloadBinaryError & error)
```

Similar to synchronous method `download(delegate::ProgressEventFn&&)` , but **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `download(delegate::ProgressEventFn&&)`

**Parameters**

|                  |                    |                                                                                                                                                                                                                                                                                                                              |
|------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">DownloadError</a> object passed in will be populated with the error information.<br>For the other parameters, see <code>download(delegate::ProgressEventFn&amp;&amp;)</code> |
|------------------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.142.3.19 download()** [10/10]

```
cbe::util::Optional<delegate::DownloadBinarySuccess> cbe::Object::download (
 std::size_t && sizeLimit,
 DownloadBinaryError & error)
```

Same as `download(delegate::ProgressEventFn&&,DownloadBinaryError&)` , but without the parameter, `progressEventFn`.

**11.142.3.20 downloadStream()** [1/5]

```
void cbe::Object::downloadStream (
 const std::string & path,
 cbe::Stream stream,
 DownloadDelegatePtr delegate)
```

Download a stream of an [Object](#) to local filesystem.

**Asynchronous** version of this service function.

**Parameters**

|                       |                                                                                                                   |
|-----------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>path</code>     | Folder location, on the local file system, of the file to be downloaded. This string must end with a slash ("/"). |
| <code>stream</code>   | Get which stream you want by first calling <code>getStream()</code> and then choose which one to download.        |
| <code>delegate</code> | Pointer to a <a href="#">delegate::DownloadDelegate</a> instance that is implemented by the user.                 |

## Example

**Async** downloading the data attached to a `cbe::Object` via a `Stream` with `Object::downloadStream()`  
 Continuation of: [Async uploading data to a cbe::Object via a Stream with Object::uploadStream\(\)](#)

```

~~~
#include "cbe/delegate/DownloadDelegate.h"
~~~
struct MyDownloadDelegate : cbe::delegate::DownloadDelegate {
 std::mutex mutex{};
 std::condition_variable conditionVariable{};
 bool called{};
 Success success{};
 void onDownloadSuccess(cbe::Object&& object,
 std::string path) final {
 {
 std::lock_guard<std::mutex> lock{mutex};
 success = Success{std::move(object), std::move(path)};
 called = true;
 }
 conditionVariable.notify_one();
 }
 void onDownloadError(cbe::delegate::TransferError&& transferError,
 cbe::util::Context&& context) final {
 {
 std::lock_guard<std::mutex> lock{mutex};
 success = Success{};
 called = true;
 }
 conditionVariable.notify_one();
 }
 cbe::Object waitForRsp() {
 std::unique_lock<std::mutex> lock{mutex};
 conditionVariable.wait(lock, [this]{ return called; });
 // Reset called flag, so current delegate instance can be reused
 called = false;
 // If object is still default constructed, this implies a failed
 // cbe::Object::downloadStream() operation
 return success.object;
 }
}; // struct MyDownloadDelegate
~~~
std::shared_ptr<MyDownloadDelegate> myDownloadDelegate =
    std::make_shared<MyDownloadDelegate>();
constexpr const char* const downloadPath = "/tmp/download/";
for (const auto& stream : streams2) {
    const auto path = std::string{downloadPath} +
        std::to_string(stream.streamId);
    object.downloadStream(path, stream, myDownloadDelegate);
    if (!myDownloadDelegate->waitForRsp()) {
        // Failure, bail out
        ~~~
 }
}

```

## 11.142.3.21 downloadStream() [2/5]

```

delegate::DownloadSuccess cbe::Object::downloadStream (
 const std::string & path,
 cbe::Stream stream,
 delegate::ProgressEventFn && progressEventFn)

```

**Synchronous** version of `downloadStream(const std::string&,cbe::Stream,DownloadDelegatePtr)` , and **throws an exception**, `DownloadException`, in case of a failed call.

See `downloadStream(DownloadDelegatePtr)`

## Parameters

|                              |                                                                                                                                                                           |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>progressEventFn</code> | See <code>download(const std::string&amp;,delegate::ProgressEventFn&amp;&amp;)</code> .<br>For the other parameters, see <code>downloadStream(DownloadDelegatePtr)</code> |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Information about the downloaded stream — if the call was successful.

See [DownloadSuccess](#)

## Exceptions

|                                   |
|-----------------------------------|
| <a href="#">DownloadException</a> |
|-----------------------------------|

## Example

**Sync** [exception] downloading the data attached to a [cbe::Object](#) via a [Stream](#) with **Object::downloadStream()**

Continuation of: [Sync](#) [exception] uploading data to a [cbe::Object](#) via a [Stream](#) with [Object::uploadStream\(\)](#)

```

~~~
constexpr const char* const downloadPath = "/tmp/download/";
for (const auto& stream : streams2) {
    const auto path = std::string{downloadPath} + std::to_string(stream.streamId);
    try
    {
        auto result = object.downloadStream(path, stream);
    }
    catch (const cbe::Object::DownloadStreamException& e)
    {
        std::cout << "Error!" << std::endl << e.what() << std::endl;
        ~~~
 }
}
~~~

```

## 11.142.3.22 downloadStream() [3/5]

```

delegate::DownloadSuccess cbe::Object::downloadStream (
    const std::string & path,
    cbe::Stream stream )

```

Same as [downloadStream\(const std::string&,cbe::Stream,delegate::ProgressEventFn&&\)](#) , but without the parameter, [progressEventFn](#).

## 11.142.3.23 downloadStream() [4/5]

```

cbe::util::Optional<delegate::DownloadSuccess> cbe::Object::downloadStream (
    const std::string & path,
    cbe::Stream stream,
    delegate::ProgressEventFn && progressEventFn,
    DownloadError & error )

```

Similar to synchronous method [downloadStream\(const std::string&,cbe::Stream,DownloadDelegatePtr\)](#) , but **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [downloadStream\(const std::string&,cbe::Stream,DownloadDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                                                                                                                                          |
|-----|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">DownloadError</a> object passed in will be populated with the error information.<br>For the other parameters, see <a href="#">downloadStream(const std::string&amp;,cbe::Stream,DownloadDelegatePtr)</a> |
|-----|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**Example**

**Sync** [non-throwing] downloading the data attached to a `cbe::Object` via a `Stream` with `Object::downloadStream()`

Continuation of: [Sync \[non-throwing\] uploading data to a cbe::Object via a Stream with Object::uploadStream\(\)](#)

```

~~~
constexpr const char* const downloadPath = "/tmp/download/";
for (const auto& stream : streams2) {
 const auto path = std::string(downloadPath) + std::to_string(stream.streamId);
 cbe::Object::DownloadStreamError downloadStreamError;
 auto result = *object.downloadStream(path, stream, downloadStreamError);
 if (downloadStreamError) {
 std::cout << "Error! " << downloadStreamError << std::endl;
        ~~~
    }
}

```

**11.142.3.24 downloadStream() [5/5]**

```

cbe::util::Optional<delegate::DownloadSuccess> cbe::Object::downloadStream (
    const std::string & path,
    cbe::Stream stream,
    DownloadError & error )

```

Same as `downloadStream(const std::string&,cbe::Stream,delegate::ProgressEventFn&&,DownloadError&)` , but without the parameter, `progressEventFn`.

**11.142.3.25 uploadStream() [1/5]**

```

void cbe::Object::uploadStream (
    const std::string & filePath,
    cbe::StreamId streamId,
    UploadDelegatePtr delegate )

```

Upload a file for adding a new or replacing existing stream attached to this object.

**Asynchronous** version of this service function.

Requires that method `getStreams(GetStreamsDelegatePtr)` is called to identify all streams associated with current object.

**Parameters**

|                 |                                                                                              |
|-----------------|----------------------------------------------------------------------------------------------|
| <i>filePath</i> | Fully qualified file name. I.e., the path, relative or absolute, including file name.        |
| <i>streamId</i> | If the stream id already exists, it will be overwritten.                                     |
| <i>delegate</i> | Pointer to a <code>delegate::UploadDelegate</code> instance that is implemented by the user. |

**Example**

**Async** uploading data to a `cbe::Object` via a `Stream` with `Object::uploadStream()`

Continuation of: [Async retrieving the Streams attached to a cbe::Object with the Object::getStreams\(\)](#)

```

~~~
#include "cbe/delegate/UploadDelegate.h"
~~~
#include <algorithm> // std::max_element
#include <vector>    // std::cbegin, std::cend
~~~
struct MyUploadDelegate : cbe::delegate::UploadDelegate {

```

```

std::mutex mutex{};
std::condition_variable conditionVariable{};
bool called{};
// Default construct, further see comment for object member in
// MyUploadDelegate above
cbe::Object object{ cbe::DefaultCtor{} };
void onUploadSuccess(cbe::Object&& object) final {
 {
 std::lock_guard<std::mutex> lock{mutex};
 this->object = std::move(object);
 called = true;
 }
 conditionVariable.notify_one();
}
void onUploadError(cbe::delegate::TransferError&& transferError,
 cbe::util::Context&& context) final {
 {
 std::lock_guard<std::mutex> lock{mutex};
 object = cbe::Object{ cbe::DefaultCtor{} };
 called = true;
 }
 conditionVariable.notify_one();
}
cbe::Object waitForRsp() {
 std::unique_lock<std::mutex> lock{mutex};
 conditionVariable.wait(lock, [this]{ return called; });
 // Reset called flag, so current delegate instance can be reused
 called = false;
 // If object is still default constructed, this implies a failed
 // cbe::Object::uploadStream() operation
 return object;
}
}; // struct MyUploadDelegate
~~~
constexpr const char* const myFile2Name = "myStream2";
const std::string      qualFile2Name = std::string(uploadPath) + myFile2Name;

std::ofstream ofs2(qualFile2Name);
ofs2 << "Line 21\n" << "Line 22\n" << "Line 23\n" << "Line 24\n"
    << std::flush;
ofs2.close();
const auto nextStreamId =
    std::max_element(std::cbegin(streams1), std::cend(streams1),
        [](const cbe::Stream& stream1,
           const cbe::Stream& stream2) {
            return stream1.streamId < stream2.streamId;
        })->streamId + 1;
std::shared_ptr<MyUploadDelegate> myUploadDelegate =
    std::make_shared<MyUploadDelegate>();
object.uploadStream(qualFile2Name, nextStreamId, myUploadDelegate);
// Use of the bool type conversion operator to detect success or failure
// as above
if (!myUploadDelegate->waitForRsp()) {
    // Failure, bail out
    ~~~
}
object.getStreams(getStreamsDelegate);
auto streamsOpt2 = getStreamsDelegate->waitForRsp();
if (!streamsOpt2) {
 // Failure, bail out
    ~~~
}
const auto streams2 = std::move(*streamsOpt2);
~~~

```

Continues at: [Async downloading the data attached to a cbe::Object via a Stream with Object::downloadStream\(\)](#)

### 11.142.3.26 uploadStream() [2/5]

```

cbe::Object cbe::Object::uploadStream (
 const std::string & filePath,
 cbe::StreamId streamId,
 delegate::ProgressEventFn && progressEventFn)

```

**Synchronous** version of `uploadStream(const std::string&,cbe::StreamId,UploadDelegatePtr)` , and **throws an exception**, `UploadException`, in case of a failed call.

See `uploadStream(const std::string&,cbe::StreamId,UploadDelegatePtr)`

## Parameters

|                        |                                                                                                                                                                                                                  |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>progressEventFn</i> | See <a href="#">download(const std::string&amp;,delegate::ProgressEventFn&amp;&amp;)</a> .<br>For the other parameters, see <a href="#">uploadStream(const std::string&amp;,cbe::StreamId,UploadDelegatePtr)</a> |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Current object that was uploaded — if the call was successful.  
See [cbe::Object](#)

## Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">UploadException</a> |  |
|---------------------------------|--|

## Example

**Sync** [exception] uploading data to a [cbe::Object](#) via a [Stream](#) with **Object::uploadStream()**  
Continuation of: [Sync](#) [exception] retrieving the Streams attached to a [cbe::Object](#) with the [Object::getStreams\(\)](#)

```

~~~
#include "cbe/delegate/UploadDelegate.h"
~~~
#include <algorithm> // std::max_element
#include <vector> // std::cbegin, std::cend
~~~
constexpr const char* const myFile2Name = "myStream2";
const std::string        qualFile2Name = std::string(uploadPath) + myFile2Name;

std::ofstream ofs2{qualFile2Name};
ofs2 << "Line 21\n" << "Line 22\n" << "Line 23\n" << "Line 24\n"
    << std::flush;
ofs2.close();
const auto nextStreamId =
    std::max_element(std::cbegin(streams1), std::cend(streams1),
        [](const cbe::Stream& stream1,
            const cbe::Stream& stream2) {
            return stream1.streamId < stream2.streamId;
        })->streamId + 1;

try
{
    object.uploadStream(qualFile2Name, nextStreamId);
    const auto streams2 = object.getStreams(getStreamsDelegate);
}
catch (const cbe::Object::UploadStreamException& e)
{
    std::cout << "Error!" << std::endl << e.what() << std::endl;
    ~~~
}
catch (const cbe::Object::GetStreamException& e)
{
 std::cout << "Error!" << std::endl << e.what() << std::endl;
    ~~~
}
~~~

```

Continues at: [Sync](#) [exception] downloading the data attached to a [cbe::Object](#) via a [Stream](#) with [Object::downloadStream\(\)](#)

## 11.142.3.27 uploadStream() [3/5]

```

cbe::Object cbe::Object::uploadStream (
 const std::string & filePath,
 cbe::StreamId streamId)

```

Same as [uploadStream\(const std::string&,cbe::StreamId,delegate::ProgressEventFn&&\)](#) , but without the parameter, [progressEventFn](#).

**11.142.3.28 uploadStream() [4/5]**

```
cbe::util::Optional<cbe::Object> cbe::Object::uploadStream (
 const std::string & filePath,
 cbe::StreamId streamId,
 delegate::ProgressEventFn && progressEventFn,
 UploadError & error)
```

Similar to synchronous method [uploadStream\(const std::string&,cbe::StreamId,delegate::ProgressEventFn&&\)](#) , but **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [uploadStream\(const std::string&,cbe::StreamId,UploadDelegatePtr\)](#)

**Parameters**

|     |       |                                                                                                                                                                                                                                                                                                                                                                          |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">DownloadError</a> object passed in will be populated with the error information.<br>For the other parameters, see <a href="#">uploadStream(const std::string&amp;,cbe::StreamId,delegate::ProgressEventFn&amp;&amp;)</a> |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**Example**

**Sync** [non-throwing] uploading data to a [cbe::Object](#) via a [Stream](#) with [Object::uploadStream\(\)](#)

Continuation of: [Sync non-throwing of retrieving the Streams attached to a cbe::Object with the Object::getStreams\(\)](#)

```
#include <algorithm> // std::max_element
#include <vector> // std::cbegin, std::cend
~~~
constexpr const char* const myFile2Name = "myStream2";
const std::string      qualFile2Name = std::string(uploadPath) + myFile2Name;

std::ofstream ofs2(qualFile2Name);
ofs2 << "Line 21\n" << "Line 22\n" << "Line 23\n" << "Line 24\n"
    << std::flush;
ofs2.close();
const auto nextStreamId =
    std::max_element(std::cbegin(streams1), std::cend(streams1),
        [](const cbe::Stream& stream1,
            const cbe::Stream& stream2) {
            return stream1.streamId < stream2.streamId;
        })->streamId + 1;
cbe::Object::UploadStreamError uploadStreamError;
object.uploadStream(qualFile2Name, nextStreamId, uploadStreamError);
if (uploadStreamError) {
    std::cout << "Error! " << uploadStreamError << std::endl;
    ~~~
}
cbe::Object::GetStreamsError getStreamsError;
const auto streams2 = *object.getStreams(getStreamsError);
if (getStreamsError) {
 std::cout << "Error! " << getStreamsError << std::endl;
    ~~~
}
~~~
```

Continues at: [Sync \[non-throwing\] downloading the data attached to a cbe::Object via a Stream with Object::downloadStream\(\)](#)

**11.142.3.29 uploadStream() [5/5]**

```
cbe::util::Optional<cbe::Object> cbe::Object::uploadStream (
 const std::string & filePath,
```

```
cbe::StreamId streamId,
UploadError & error)
```

Same as [uploadStream\(const std::string&,cbe::StreamId,delegate::ProgressEventFn&&,UploadError&\)](#) , but without the parameter, `progressEventFn`.

### 11.142.3.30 updateKeyValues() [1/6]

```
void cbe::Object::updateKeyValues (
 KeyValues keyValues,
 UpdateKeyValuesDelegatePtr delegate)
```

Adds key/value pair data to the object.

**Asynchronous** version of this service function.

#### Note

All existing key will be overwritten, new created.

Any key name must start with a letter or `_`

The following key names are reserved and should not be used: category, content, id, link and date

Key names are case sensitive, hence variations with uppercase are permitted.

#### Parameters

|                  |                                                                                                          |
|------------------|----------------------------------------------------------------------------------------------------------|
| <i>keyValues</i> | Map of key/value pairs (metadata).                                                                       |
| <i>delegate</i>  | Pointer to a <a href="#">delegate::UpdateKeyValuesDelegate</a> instance that is implemented by the user. |

### 11.142.3.31 updateKeyValues() [2/6]

```
void cbe::Object::updateKeyValues (
 UpdateKeyValuesDelegatePtr delegate)
```

Deletes all key/value pairs of data to the object.

Same as [updateKeyValues\(KeyValues,UpdateKeyValuesDelegatePtr\)](#), but without the `keyValues` parameter.

#### Note

Any and all existing key/values will be erased.

### 11.142.3.32 updateKeyValues() [3/6]

```
cbe::Object cbe::Object::updateKeyValues (
 KeyValues keyValues)
```

Synchronous [exception] Adds key/value pair data to the object.

**Synchronous** version of [updateKeyValues\(KeyValues,UpdateKeyValuesDelegatePtr\)](#) , and **throws an exception**, [UpdateKeyValuesException](#), in case of a failed call.

See [updateKeyValues\(UpdateKeyValuesDelegatePtr\)](#)

#### Returns

Current object that was uploaded — if the call was successful.

See [cbe::Object](#)

#### Exceptions

|                                          |  |
|------------------------------------------|--|
| <a href="#">UpdateKeyValuesException</a> |  |
|------------------------------------------|--|

**11.142.3.33 updateKeyValues()** [4/6]

```
cbe::Object cbe::Object::updateKeyValues ()
```

Synchronous [exception] Deletes all key/value pairs of data to the object.

Same as [updateKeyValues\(KeyValues\)](#), but without the `keyValues` parameter.

**11.142.3.34 updateKeyValues()** [5/6]

```
cbe::util::Optional<cbe::Object> cbe::Object::updateKeyValues (
 KeyValues keyValues,
 UpdateKeyValuesError & error)
```

Synchronous [non-throwing] Adds key/value pair data to the object.

**Synchronous** version of [updateKeyValues\(KeyValues,UpdateKeyValuesDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [updateKeyValues\(KeyValues, UpdateKeyValuesDelegatePtr\)](#)

**Parameters**

|                  |                    |                                                                                                                                                                                                                                         |
|------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UpdateKeyValuesError</a> object passed in will be populated with the error information. |
|------------------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.142.3.35 updateKeyValues()** [6/6]

```
cbe::util::Optional<cbe::Object> cbe::Object::updateKeyValues (
 UpdateKeyValuesError & error)
```

Synchronous [non-throwing] Deletes all key/value pairs of data to the object.

Same as [updateKeyValues\(KeyValues,UpdateKeyValuesError&\)](#), but without the `keyValues` parameter.

**11.142.3.36 getStreams()** [1/3]

```
void cbe::Object::getStreams (
 GetStreamsDelegatePtr delegate)
```

Downloads the streams meta data associated with current object to the SDK's cache.

**Asynchronous** version of this service function.

The meta data is delivered as [cbe::Streams](#) via the delegate callback method [cbe::delegate::GetStreamsDelegate::onGetStreamsSuccess](#).

Further, the actual stream data are retrieved through method [downloadStream\(const std::string&,cbe::Stream,DownloadDelegatePtr\)](#).

**Note**

This method must be re-called if you upload more streams, see [uploadStream\(\)](#)

**Parameters**

|                       |                                                                                                     |
|-----------------------|-----------------------------------------------------------------------------------------------------|
| <code>delegate</code> | Pointer to a <a href="#">delegate::GetStreamsDelegate</a> instance that is implemented by the user. |
|-----------------------|-----------------------------------------------------------------------------------------------------|

## Example

**Async** retrieving the Streams attached to a [cbe::Object](#) with the [Object::getStreams\(\)](#)  
Continuation of: [Async creation of a cbe::Object by using Container::upload\(\)](#)

```

~~~
#include "cbe/delegate/GetStreamsDelegate.h"
#include "cbe/util/Optional.h"
~~~
struct MyGetStreamsDelegate : cbe::delegate::GetStreamsDelegate {
 std::mutex mutex{};
 std::condition_variable conditionVariable{};
 bool called{};
 // Represents the result stream meta data as a cbe::Optional. Where an empty
 // Optional implies an error
 cbe::util::Optional<cbe::Streams> streams{};
 void onGetStreamsSuccess(cbe::Streams&& streams) final {
 {
 std::lock_guard<std::mutex> lock{mutex};
 this->streams = std::move(streams);
 called = true;
 }
 conditionVariable.notify_one();
 }
 void onGetStreamsError(cbe::delegate::Error&& error,
 cbe::util::Context&& context) final {
 {
 std::lock_guard<std::mutex> lock{mutex};
 object = cbe::Object{ cbe::DefaultCtor{} };
 called = true;
 }
 conditionVariable.notify_one();
 }
 cbe::util::Optional<cbe::Streams> waitForRsp() {
 std::unique_lock<std::mutex> lock{mutex};
 conditionVariable.wait(lock, [this]{ return called; });
 // Reset called flag, so current delegate instance can be reused
 called = false;
 // If streams is still default constructed, this implies a failed
 // cbe::Object::getStreams() operation
 return std::move(streams);
 }
}; // struct MyGetStreamsDelegate
~~~
std::shared_ptr<MyGetStreamsDelegate> getStreamsDelegate =
    std::make_shared<MyGetStreamsDelegate>();
object.getStreams(getStreamsDelegate);
// Simply make use of the Optional bool type conversion operator on the return
// value from the waitForRsp() method to determine the outcome of the
// getStreams() invocation above
cbe::util::Optional<cbe::Streams> streamsOpt1 =
    getStreamsDelegate->waitForRsp();
if (!streamsOpt1) {
    // Failure, bail out
    ~~~
}
// Now, streamsOpt1 contains the streams meta data, so move it to variable
// streams1
const cbe::Object::Streams streams1 = std::move(*streamsOpt1);
~~~

```

Continues at: [Async uploading data to a cbe::Object via a Stream with Object::uploadStream\(\)](#)

## 11.142.3.37 getStreams() [2/3]

[Streams](#) [cbe::Object::getStreams](#) ( )

**Synchronous** version of [getStreams\(GetStreamsDelegatePtr\)](#) , and **throws an exception**, [#GetStreamException](#), in case of a failed call.

See [getStreams\(GetStreamsDelegatePtr\)](#)

## Returns

The streams attached to current object — if the call was successful.

## Exceptions

|                                     |  |
|-------------------------------------|--|
| <a href="#">#GetStreamException</a> |  |
|-------------------------------------|--|

**Example**

**Sync** [exception] retrieving the Streams attached to a [cbe::Object](#) with the **Object::getStreams()**  
 Continuation of: [Sync \[exception\] creation of a cbe::Object by using Container::upload\(\)](#)

```

~~~
#include "cbe/util/Optional.h"
~~~
try
{
    auto streams = object.getStreams();
}
catch (const cbe::Object::GetStreamsException& e)
{
    std::cout << "Error!" << std::endl << e.what() << std::endl;
    ~~~
}
~~~

```

Continues at: [Sync \[exception\] uploading data to a cbe::Object via a Stream with Object::uploadStream\(\)](#)

**11.142.3.38 getStreams() [3/3]**

```

cbe::util::Optional<Streams> cbe::Object::getStreams (
    GetStreamsError & error )

```

**Synchronous** version of [getStreams\(GetStreamsDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.  
 See [getStreams\(GetStreamsDelegatePtr\)](#)

**Parameters**

|     |       |                                                                                                                                                                                                                                    |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">GetStreamsError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**Example**

**Sync** [non-throwing] retrieving the Streams attached to a [cbe::Object](#) with the **Object::getStreams()**  
 Continuation of: [Sync \[non-throwing\] creation of a cbe::Object by using Container::upload\(\)](#)

```

~~~
cbe::Object::GetStreamsError getStreamsError;
auto streams = *object.getStreams(getStreamsError);
if (getStreamsError) {
 std::cout << "Error! " << getStreamsError << std::endl;
    ~~~
}
~~~

```

Continues at: [Sync \[non-throwing\] uploading data to a cbe::Object via a Stream with Object::uploadStream\(\)](#)

**11.142.3.39 getMimeType()**

```
std::string cbe::Object::getMimeType () const
```

Returns the mime type of the object.

E.g., application/pdf, audio/wav, image/jpg, text/xml, video/mp4 etc.

**11.142.3.40 getObjectType()**

```
cbe::object_t cbe::Object::getObjectType () const
```

Returns the [Object](#) type.  
See [cbe::ObjectType](#).

#### 11.142.3.41 `getAcl()` [1/3]

```
void cbe::Object::getAcl (
 GetAclDelegatePtr delegate)
```

Returns the Access Control List for current [Object](#).  
**Asynchronous** version of this service function.

##### Parameters

|                          |                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------|
| <a href="#">delegate</a> | Pointer to a <a href="#">delegate::AclDelegate</a> instance that is implemented by the user. |
|--------------------------|----------------------------------------------------------------------------------------------|

#### 11.142.3.42 `getAcl()` [2/3]

```
cbe::AclMap cbe::Object::getAcl ()
```

**Synchronous** version of [getAcl\(GetAclDelegatePtr\)](#) , and **throws an exception**, [GetAclException](#), in case of a failed call.

See [getAcl\(GetAclDelegatePtr\)](#)

##### Returns

Information about the object from which the ACL is fetched — if the call was successful.  
See [AclSuccess](#)

##### Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">GetAclException</a> |  |
|---------------------------------|--|

#### 11.142.3.43 `getAcl()` [3/3]

```
cbe::util::Optional<cbe::AclMap> cbe::Object::getAcl (
 GetAclError & error)
```

**Synchronous** version of [getAcl\(GetAclDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [getAcl\(GetAclDelegatePtr\)](#)

##### Parameters

|                  |                    |                                                                                                                                                                                                                                |
|------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">GetAclError</a> object passed in will be populated with the error information. |
|------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

##### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.142.3.44 `setAcl()` [1/3]

```
void cbe::Object::setAcl (
```

```

cbe::AclMap aclMap,
SetAclDelegatePtr delegate)

```

Sets the Access Control List for current object.

**Asynchronous** version of this service function.

#### Parameters

|                 |                                                                                              |
|-----------------|----------------------------------------------------------------------------------------------|
| <i>aclMap</i>   | The desired <a href="#">permission</a> for current object.                                   |
| <i>delegate</i> | Pointer to a <a href="#">delegate::AclDelegate</a> instance that is implemented by the user. |

#### 11.142.3.45 setAcl() [2/3]

```

cbe::AclMap cbe::Object::setAcl (
 cbe::AclMap aclMap)

```

**Synchronous** version of [setAcl\(cbe::AclMap,SetAclDelegatePtr\)](#) , and **throws an exception**, [SetAclException](#), in case of a failed call.

See [setAcl\(cbe::AclMap,SetAclDelegatePtr\)](#)

#### Returns

Information about the object from which the ACL has been set — if the call was successful.

See [AclSuccess](#)

#### Exceptions

|                                 |  |
|---------------------------------|--|
| <a href="#">SetAclException</a> |  |
|---------------------------------|--|

#### 11.142.3.46 setAcl() [3/3]

```

cbe::util::Optional<cbe::AclMap> cbe::Object::setAcl (
 cbe::AclMap aclMap,
 SetAclError & error)

```

**Synchronous** version of [setAcl\(cbe::AclMap,SetAclDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [setAcl\(cbe::AclMap,SetAclDelegatePtr\)](#)

#### Parameters

|            |              |                                                                                                                                                                                                                                |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SetAclError</a> object passed in will be populated with the error information. |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.142.3.47 share() [1/3]

```

void cbe::Object::share (
 cbe::UserId toUserGroup,
 std::string description,
 ShareDelegatePtr delegate)

```

Share current object to a user.

**Asynchronous** version of this service function.

Notifies the user that a share has occurred so that the user can check what permissions they have been given. Sharing gives read permissions as of right now but might change in the future.

#### Parameters

|                    |                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------|
| <i>toUserGroup</i> | Takes a user id or group id (lastly named is for the future) and share to.                        |
| <i>description</i> | Names the specific share between you and the user/group.                                          |
| <i>delegate</i>    | Pointer to a <a href="#">delegate::ShareDelegatePtr</a> instance that is implemented by the user. |

#### 11.142.3.48 share() [2/3]

```
delegate::ShareSuccess cbe::Object::share (
 cbe::UserId toUserGroup,
 std::string description)
```

**Synchronous** version of [share\(cbe::UserId,std::string,ShareDelegatePtr\)](#) , and **throws an exception**, [ShareException](#), in case of a failed call.

See [share\(ShareDelegatePtr\)](#)

#### Returns

Information about the id of the shared object — if the call was successful.

See [ShareSuccess](#)

#### Exceptions

|                                |  |
|--------------------------------|--|
| <a href="#">ShareException</a> |  |
|--------------------------------|--|

#### 11.142.3.49 share() [3/3]

```
cbe::util::Optional<delegate::ShareSuccess> cbe::Object::share (
 cbe::UserId toUserGroup,
 std::string description,
 ShareError & error)
```

**Synchronous** version of [share\(cbe::UserId,std::string,ShareDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [share\(cbe::UserId,std::string,ShareDelegatePtr\)](#)

#### Parameters

|            |              |                                                                                                                                                                                                                               |
|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ShareError</a> object passed in will be populated with the error information. |
|------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.142.3.50 unShare() [1/3]

```
void cbe::Object::unShare (
```

```

 cbe::ShareId shareId,
 UnShareDelegatePtr delegate)

```

Unshare the object to a specific shareId created when sharing.

**Asynchronous** version of this service function.

Each share is unique between user/group and the one sharing. This is represented with a unique share id.

#### Parameters

|                 |                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------|
| <i>shareId</i>  | The unique id for a share between the owner and other user/group.                                |
| <i>delegate</i> | Pointer to a <a href="#">delegate::UnShareDelegate</a> instance that is implemented by the user. |

#### 11.142.3.51 unShare() [2/3]

```

delegate::UnShareSuccess cbe::Object::unShare (
 cbe::ShareId shareId)

```

**Synchronous** version of [unShare\(cbe::ShareId,UnShareDelegatePtr\)](#) , and **throws an exception**, [UnShareException](#), in case of a failed call.

See [unShare\(cbe::ShareId,UnShareDelegatePtr\)](#)

#### Returns

Information about the unshared object — if the call was successful.

See [UnShareSuccess](#)

#### Exceptions

|                                  |  |
|----------------------------------|--|
| <a href="#">UnShareException</a> |  |
|----------------------------------|--|

#### 11.142.3.52 unShare() [3/3]

```

cbe::util::Optional<delegate::UnShareSuccess> cbe::Object::unShare (
 cbe::ShareId shareId,
 UnShareError & error)

```

**Synchronous** version of [unShare\(cbe::ShareId,UnShareDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [unShare\(cbe::ShareId,UnShareDelegatePtr\)](#)

#### Parameters

|            |              |                                                                                                                                                                                                                                 |
|------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>out</i> | <i>error</i> | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UnShareError</a> object passed in will be populated with the error information. |
|------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

#### 11.142.3.53 publish() [1/3]

```

void cbe::Object::publish (
 cbe::PublishAccess security,

```

```

 cbe::PublishVisibility privacy,
 std::string description,
 std::string password,
 PublishDelegatePtr delegate)

```

Publishes current object to any user.

**Asynchronous** version of this service function.

Can be revoked with [unPublish\(\)](#)

#### Parameters

|                    |                                                                                                  |
|--------------------|--------------------------------------------------------------------------------------------------|
| <i>security</i>    | A <a href="#">cbe::PublishAccess</a> enum                                                        |
| <i>privacy</i>     | A <a href="#">cbe::WebShareVisibility</a> enum                                                   |
| <i>description</i> | Free text                                                                                        |
| <i>password</i>    | Password                                                                                         |
| <i>delegate</i>    | Pointer to a <a href="#">delegate::PublishDelegate</a> instance that is implemented by the user. |

#### 11.142.3.54 publish() [2/3]

```

delegate::PublishSuccess cbe::Object::publish (
 cbe::PublishAccess security,
 cbe::PublishVisibility privacy,
 std::string description,
 std::string password)

```

**Synchronous** version of [publish\(cbe::PublishAccess,cbe::PublishVisibility,std::string,std::string,PublishDelegatePtr\)](#), and **throws an exception**, [PublishException](#), in case of a failed call.

See [publish\(cbe::PublishAccess,cbe::PublishVisibility,std::string,std::string,PublishDelegatePtr\)](#)

#### Returns

Information about the published object — if the call was successful.

See [PublishSuccess](#)

#### Note

To use this method, header file  
[cbe/delegate/PublishSuccess.h](#)  
 must be included

#### Exceptions

|                                  |  |
|----------------------------------|--|
| <a href="#">PublishException</a> |  |
|----------------------------------|--|

#### 11.142.3.55 publish() [3/3]

```

cbe::util::Optional<delegate::PublishSuccess> cbe::Object::publish (
 cbe::PublishAccess security,
 cbe::PublishVisibility privacy,
 std::string description,
 std::string password,
 PublishError & error)

```

**Synchronous** version of [publish\(cbe::PublishAccess,cbe::PublishVisibility,std::string,std::string,PublishDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [publish\(cbe::PublishAccess,cbe::PublishVisibility,std::string,std::string,PublishDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                 |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">PublishError</a> object passed in will be populated with the error information. |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

## Note

To use this method, header file  
[cbe/delegate/PublishSuccess.h](#)  
must be included

**11.142.3.56 unPublish()** [1/3]

```
void cbe::Object::unPublish (
 UnPublishDelegatePtr delegate)
```

UnPublishes current object.

**Asynchronous** version of this service function.

Revokes previous [publish\(\)](#).

## Parameters

|                 |                                                                                                    |
|-----------------|----------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::UnPublishDelegate</a> instance that is implemented by the user. |
|-----------------|----------------------------------------------------------------------------------------------------|

**11.142.3.57 unPublish()** [2/3]

```
delegate::UnPublishSuccess cbe::Object::unPublish ()
```

**Synchronous** version of [unPublish\(UnPublishDelegatePtr\)](#) , and **throws an exception**, [UnPublishException](#), in case of a failed call.

See [unPublish\(UnPublishDelegatePtr\)](#)

## Returns

Information about the unpublished object — if the call was successful.

See [UnPublishSuccess](#)

## Exceptions

|                                    |  |
|------------------------------------|--|
| <a href="#">UnPublishException</a> |  |
|------------------------------------|--|

**11.142.3.58 unPublish()** [3/3]

```
cbe::util::Optional<delegate::UnPublishSuccess> cbe::Object::unPublish (
 UnPublishError & error)
```

**Synchronous** version of [unPublish\(UnPublishDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [unPublish\( UnPublishDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                   |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UnPublishError</a> object passed in will be populated with the error information. |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.142.3.59 unsubscribe()** [1/3]

```
void cbe::Object::unsubscribe (
 UnSubscribeDelegatePtr delegate)
```

UnSubscribes from this object.

**Asynchronous** version of this service function.

Revokes the subscription previously established with [cbe::SubscribeManager::subscribe\(\)](#)

## Parameters

|          |                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------|
| delegate | Pointer to a <a href="#">delegate::UnSubscribeDelegate</a> instance that is implemented by the user. |
|----------|------------------------------------------------------------------------------------------------------|

**11.142.3.60 unsubscribe()** [2/3]

```
delegate::UnSubscribeSuccess cbe::Object::unsubscribe ()
```

**Synchronous** version of [unsubscribe\(UnSubscribeDelegatePtr\)](#), and **throws an exception**, [UnSubscribeException](#), in case of a failed call.

See [unsubscribe\(UnSubscribeDelegatePtr\)](#)

## Returns

Information about the unsubscribed object — if the call was successful.

See [UnSubscribeSuccess](#)

## Exceptions

|                                      |  |
|--------------------------------------|--|
| <a href="#">UnSubscribeException</a> |  |
|--------------------------------------|--|

**11.142.3.61 unsubscribe()** [3/3]

```
cbe::util::Optional<delegate::UnSubscribeSuccess> cbe::Object::unsubscribe (
 UnSubscribeError & error)
```

**Synchronous** version of [unsubscribe\(UnSubscribeDelegatePtr\)](#), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [unsubscribe\(UnSubscribeDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                     |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">UnSubscribeError</a> object passed in will be populated with the error information. |
|-----|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

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

- `include/cbe/Object.h`

## 11.143 `cbe::util::Optional< T >` Class Template Reference

Class template `Optional` manages an optional contained value - i.e., a value that is either present or absent.

`#include <Optional.h>`

### Classes

- class `BadAccess`

*Thrown by `value()` method when accessing an object that does not contain a value.*

### Public Member Functions

- `Optional ()=default`  
*Creates an empty `Optional`, i.e., no value present.*
- `Optional (const T &value)`  
*Copy value ctor.*
- `Optional (T &&value) noexcept(std::is_nothrow_move_constructible< T >::value)`  
*Move value ctor.*
- `Optional (const Optional &rh)`  
*Copy ctor.*
- `Optional (Optional &&rh) noexcept(std::is_nothrow_move_constructible< T >::value)`  
*Move ctor.*
- `Optional & operator= (const Optional &rh)`  
*Copy assignment operator.*
- `Optional & operator= (Optional &&rh) noexcept`  
*Move assignment operator.*
- `T & value ()`  
*Value access non-const method.*
- `T & operator* ()`
- `T * operator-> ()`
- `const T & value () const`  
*Value access const method.*
- `const T & operator* () const`
- `const T * operator-> () const`
- `bool hasValue () const noexcept`  
*Checks whether a value is present.*
- `operator bool () const noexcept`  
*Checks whether a value is present.*
- `void reset ()`  
*Invalidates possible contained value. The contained value will be destructed.*
- `void swap (Optional &other) noexcept`  
*Swaps the contents with those of other.*
- `template<typename... ArgTs>`  
`void emplace (ArgTs &&... args)`  
*Constructs the contained value in-place. If `*this` already contains a value before the call, the contained value is destroyed by calling its destructor.*

### 11.143.1 Detailed Description

```
template<typename T>
class cbe::util::Optional< T >
```

Class template [Optional](#) manages an optional contained value - i.e., a value that is either present or absent.

#### Template Parameters

|          |                             |
|----------|-----------------------------|
| <i>T</i> | Type of the contained value |
|----------|-----------------------------|

### 11.143.2 Member Function Documentation

#### 11.143.2.1 operator bool()

```
template<typename T >
cbe::util::Optional< T >::operator bool () const [inline], [explicit], [noexcept]
```

Checks whether a value is present.

```
cbe::util::Optional<int> inquireInt(const char* prompt) {
 while (true) {
 std::cout << prompt << ": ";
 std::string answer;
 std::cin >> answer;
 try {
 return std::stoi(answer); // I.e., return cbe::util::Optional<int>{std::stoi(answer)}
 }
 catch (...) {
 // Invalid input
 return {}; // I.e., return cbe::util::Optional<int>{};
 }
 }
}
~~~
auto n = inquireInt("Please, give an integer value");
if (n) {
    std::cout << "Thanks for the number" << *n << "!\n";
} else {
    std::cout << "Not a number!\n";
}
~~~
```

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

- include/cbe/util/Optional.h

## 11.144 cbe::Publish Class Reference

Managing a published [Item](#).

```
#include <Publish.h>
```

### Public Member Functions

- void [setTitle](#) (const std::string &title)  
*Set the Title of this publication.*
- void [setSecurity](#) ([cbe::PublishAccess](#) security)  
*Set the Security of this publication.*
- void [setPrivacy](#) ([cbe::PublishVisibility](#) privacy)  
*Set the Privacy of this publication.*
- void [setPassword](#) (const std::string &password)  
*Set the Password of this publication.*
- void [setDescription](#) (const std::string &description)  
*Set the Description of this publication.*

- `std::string getTitle () const`
- `cbe::PublishAccess getSecurity () const`
- `cbe::PublishVisibility getPrivacy () const`
- `std::string getPassword () const`
- `std::string getDescription () const`
- `cbe::PublishId getPublishId () const`
- `Publish (cbe::DefaultCtor)`  
*Default constructor.*
- `operator bool () const`  
*Checks if the current instance is real.*

## Friends

- class `CloudBackend`
- class `Item`

### 11.144.1 Detailed Description

Managing a published `Item`.

### 11.144.2 Constructor & Destructor Documentation

#### 11.144.2.1 Publish()

```
cbe::Publish::Publish (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the `operator bool()` test

### 11.144.3 Member Function Documentation

#### 11.144.3.1 setTitle()

```
void cbe::Publish::setTitle (
 const std::string & title)
```

Set the Title of this publication.

##### Parameters

|              |      |
|--------------|------|
| <i>title</i> | name |
|--------------|------|

#### 11.144.3.2 setSecurity()

```
void cbe::Publish::setSecurity (
 cbe::PublishAccess security)
```

Set the Security of this publication.

##### Parameters

|                 |                                              |
|-----------------|----------------------------------------------|
| <i>security</i> | of type <code>cbe::PublishAccess</code> enum |
|-----------------|----------------------------------------------|

### 11.144.3.3 setPrivacy()

```
void cbe::Publish::setPrivacy (
 cbe::PublishVisibility privacy)
```

Set the Privacy of this publication.

#### Parameters

|                |                                                     |
|----------------|-----------------------------------------------------|
| <i>privacy</i> | of type <a href="#">cbe::PublishVisibility</a> enum |
|----------------|-----------------------------------------------------|

### 11.144.3.4 setPassword()

```
void cbe::Publish::setPassword (
 const std::string & password)
```

Set the Password of this publication.

#### Parameters

|                 |      |
|-----------------|------|
| <i>password</i> | text |
|-----------------|------|

### 11.144.3.5 setDescription()

```
void cbe::Publish::setDescription (
 const std::string & description)
```

Set the Description of this publication.

#### Parameters

|                    |           |
|--------------------|-----------|
| <i>description</i> | free text |
|--------------------|-----------|

### 11.144.3.6 getTitle()

```
std::string cbe::Publish::getTitle () const
```

Gets the title

### 11.144.3.7 getSecurity()

```
cbe::PublishAccess cbe::Publish::getSecurity () const
```

Gets the security setting, see [cbe::PublishAccess](#) enum

### 11.144.3.8 getPrivacy()

```
cbe::PublishVisibility cbe::Publish::getPrivacy () const
```

Gets the privacy setting, see [cbe::PublishVisibility](#) enum

### 11.144.3.9 getPassword()

```
std::string cbe::Publish::getPassword () const
```

Checks if a password has been set.

**Returns**

"true" : if a password has been set  
 "" : empty string if not.

**11.144.3.10 getDescription()**

```
std::string cbe::Publish::getDescription () const
```

Gets the description

**11.144.3.11 getPublishId()**

```
cbe::PublishId cbe::Publish::getPublishId () const
```

Gets the publish id number

**11.144.3.12 operator bool()**

```
cbe::Publish::operator bool () const [explicit]
```

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [Publish\(cbe::DefaultCtor\)](#)

**Returns**

true : is real  
 false : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/Publish.h

**11.145 cbe::delegate::PublishDelegate Class Reference**

```
#include <PublishDelegate.h>
```

**Classes**

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by:*

**Public Types**

- using **Success** = [PublishSuccess](#)
- using **Error** = [delegate::Error](#)

**Public Member Functions**

- virtual void [onPublishSuccess](#) (cbe::Items &&items)=0
- virtual void [onPublishError](#) (Error &&error, cbe::util::Context &&context)=0

**11.145.1 Detailed Description**

Delegate class for the asynchronous version of methods:

- [cbe::Container::publish](#)
- [cbe::Object::publish](#)

## 11.145.2 Member Function Documentation

### 11.145.2.1 onPublishSuccess()

```
virtual void cbe::delegate::PublishDelegate::onPublishSuccess (
 cbe::Items && items) [pure virtual]
```

Called upon successful publish.

#### Parameters

|              |                                             |
|--------------|---------------------------------------------|
| <i>items</i> | Instance of items that are being published. |
|--------------|---------------------------------------------|

### 11.145.2.2 onPublishError()

```
virtual void cbe::delegate::PublishDelegate::onPublishError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/PublishDelegate.h

## 11.146 cbe::PublishManager Class Reference

Managing the list of web shares.

```
#include <PublishManager.h>
```

### Public Member Functions

- [cbe::Items getPublished](#) () const  
*Lists publications that are shared with other [userids](#).*
- [PublishManager](#) ([cbe::DefaultCtor](#))  
*Default constructor.*
- [operator bool](#) () const  
*Checks if the current instance is real.*

### Friends

- class [CloudBackend](#)

### 11.146.1 Detailed Description

Managing the list of web shares.

### 11.146.2 Constructor & Destructor Documentation

#### 11.146.2.1 PublishManager()

```
cbe::PublishManager::PublishManager (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the [operator bool\(\)](#) test

### 11.146.3 Member Function Documentation

#### 11.146.3.1 getPublished()

`cbe::Items` `cbe::PublishManager::getPublished ( ) const`

Lists publications that are shared with other `userid`s.

Listing is done independently of where in the actual `container/subcontainer` tree the `items` are located.

(Publications are also known as web shares.)

##### Returns

The publications in terms of `items`.

#### 11.146.3.2 operator bool()

`cbe::PublishManager::operator bool ( ) const` `[explicit]`

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the `Default` constructor `PublishManager(cbe::DefaultCtor)`

##### Returns

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- `include/cbe/PublishManager.h`

## 11.147 cbe::delegate::PublishSuccess Class Reference

Convenience type that bundles the parameter passed to method `cbe::delegate::ShareDelegate::onShareSuccess`.

`#include <PublishSuccess.h>`

### Public Member Functions

- `PublishSuccess (cbe::DefaultCtor)`
- `PublishSuccess (cbe::Items &&items)`
- `operator bool ( ) const`

*Checks if current instance is valid.*

### Public Attributes

- `cbe::Items items {}`

#### 11.147.1 Detailed Description

Convenience type that bundles the parameter passed to method `cbe::delegate::ShareDelegate::onShareSuccess`.

#### 11.147.2 Member Function Documentation

### 11.147.2.1 operator bool()

```
cbe::delegate::PublishSuccess::operator bool () const [explicit]
```

Checks if current instance is valid.

Returns

`true`: is valid

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

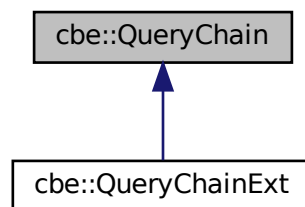
- include/cbe/delegate/PublishSuccess.h

## 11.148 cbe::QueryChain Class Reference

To do a search for [Object](#) combining more than one [Container](#) table.

```
#include <QueryChain.h>
```

Inheritance diagram for cbe::QueryChain:



### Classes

- struct [Exception](#)

### Public Types

- using `JoinDelegatePtr` = [delegate::JoinDelegatePtr](#)

### Public Member Functions

- [QueryChain](#) `join` ([Container](#) containerToQuery, std::string key1, std::string key2, [JoinDelegatePtr](#) joinDelegate)
- [QueryChain](#) `join` ([Container](#) containerToQuery, std::string key1, std::string key2, [Filter](#) constraints, [JoinDelegatePtr](#) joinDelegate)
- [QueryChain](#) `join` ([Container](#) containerToQuery, std::string key1, std::string key2, [Container](#) containerForResults, [JoinDelegatePtr](#) joinDelegate)
- [QueryChain](#) `join` ([Container](#) containerToQuery, std::string key1, std::string key2, [Filter](#) constraints, [Container](#) containerForResults, [JoinDelegatePtr](#) joinDelegate)
- [QueryResult](#) `getQueryResult` () const
- [QueryChain](#) ([cbe::DefaultCtor](#))
- `operator bool` () const

### Friends

- class `CloudBackend`
- class `Container`
- class `QueryChainSync`

### 11.148.1 Detailed Description

To do a search for [Object](#) combining more than one [Container](#) table.

[Async] [QueryChain](#) is used when a query() is carried out with a solely [QueryDelegate](#).

Anyway, if a subsequent [join\(\)](#) will be used, an additional [JoinDelegate](#) is required in the [join\(\)](#)-call.

### 11.148.2 Member Function Documentation

#### 11.148.2.1 [join\(\)](#) [1/4]

```
QueryChain cbe::QueryChain::join (
 Container containerToQuery,
 std::string key1,
 std::string key2,
 JoinDelegatePtr joinDelegate)
```

Performs a join request in conjunction with a query, or previous join, to get related items from another container.

**Asynchronous** version of this service function.

##### Parameters

|                         |                                                                                                                                                                                                                                                                        |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>containerToQuery</i> | The container of which the join shall be done with.                                                                                                                                                                                                                    |
| <i>key1</i>             | The key from the previous query on which to perform the join.                                                                                                                                                                                                          |
| <i>key2</i>             | The key on objects in <i>containerToQuery</i> on which to perform the join                                                                                                                                                                                             |
| <i>joinDelegate</i>     | Pointer to a <a href="#">JoinDelegate</a> instance, implemented by the user, that receives the response of this query request.<br>I.e., either of the JoinDelegate callback functions <a href="#">onJoinSuccess()</a> or <a href="#">onJoinError()</a> will be called. |

#### 11.148.2.2 [join\(\)](#) [2/4]

```
QueryChain cbe::QueryChain::join (
 Container containerToQuery,
 std::string key1,
 std::string key2,
 Filter constraints,
 JoinDelegatePtr joinDelegate)
```

Same as [join\(Container,std::string,std::string,delegate::JoinDelegatePtr\)](#), but with an additional [Filter](#) parameter *constraints*.

##### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <i>constraints</i> | A filter that provides any constraints for the query. |
|--------------------|-------------------------------------------------------|

#### 11.148.2.3 [join\(\)](#) [3/4]

```
QueryChain cbe::QueryChain::join (
 Container containerToQuery,
 std::string key1,
 std::string key2,
 Container containerForResults,
 JoinDelegatePtr joinDelegate)
```

Same as [join\(Container,std::string,std::string,delegate::JoinDelegatePtr\)](#), but with an additional [Container](#) parame-

ter containerForResults .

#### Parameters

|                            |                                                                                                              |
|----------------------------|--------------------------------------------------------------------------------------------------------------|
| <i>containerForResults</i> | Can be the <a href="#">Container</a> from the previous query if desired items should be from that container. |
|----------------------------|--------------------------------------------------------------------------------------------------------------|

#### 11.148.2.4 join() [4/4]

```
QueryChain cbe::QueryChain::join (
 Container containerToQuery,
 std::string key1,
 std::string key2,
 Filter constraints,
 Container containerForResults,
 JoinDelegatePtr joinDelegate)
```

Same as [join\(Container,std::string,std::string,Container,delegate::JoinDelegatePtr\)](#), but with an additional [Filter](#) parameter *constraints* .

#### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <i>constraints</i> | A filter that provides any constraints for the query. |
|--------------------|-------------------------------------------------------|

#### 11.148.2.5 getQueryResult()

```
QueryResult cbe::QueryChain::getQueryResult () const
```

Returns the [QueryResult](#) provided to the delegate after the the last query() or [join\(\)](#) call. If this method is called before the delegate call, an empty [QueryResult](#) is returned.

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

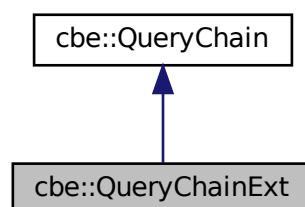
- include/cbe/QueryChain.h

## 11.149 cbe::QueryChainExt Class Reference

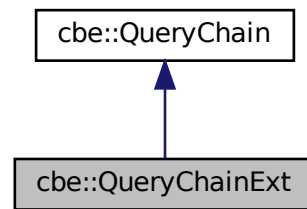
Extension of class [QueryChain](#).

```
#include <QueryChain.h>
```

Inheritance diagram for cbe::QueryChainExt:



Collaboration diagram for `cbe::QueryChainExt`:



## Public Member Functions

- `QueryChainExt join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2)
- `QueryChainExt join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2, [Filter](#) constraints)
- `QueryChainExt join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2, [Container](#) containerForResults)
- `QueryChainExt join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2, [Filter](#) constraints, [Container](#) containerForResults)
- `QueryChain join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2, [JoinDelegatePtr](#) joinDelegate)
- `QueryChain join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2, [Filter](#) constraints, [JoinDelegatePtr](#) joinDelegate)
- `QueryChain join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2, [Container](#) containerForResults, [JoinDelegatePtr](#) joinDelegate)
- `QueryChain join` ([Container](#) containerToQuery, `std::string` key1, `std::string` key2, [Filter](#) constraints, [Container](#) containerForResults, [JoinDelegatePtr](#) joinDelegate)

## Friends

- class `CloudBackend`
- class `Container`

## Additional Inherited Members

### 11.149.1 Detailed Description

Extension of class [QueryChain](#).

[Async] `QueryChainExt` is used when the query is carried out with subsequent intended join in mind.

I.e., we have passed a [QueryJoinDelegate](#) to the previous call and that will then be reused in the trailing `join()`-call.

It is an extension of class `QueryChain` offering `join()` methods that do not require a new delegate.

Instead, these will use the delegate passed as argument in the previous invocation in the call chain — i.e.:

- `CloudBackend::query(ContainerId, delegate::QueryJoinDelegatePtr)` and overloads.
- `Container::query(delegate::QueryJoinDelegatePtr)` and overloads.
- `QueryChain::join(Container, std::string, std::string, JoinDelegatePtr)` and overloads.

### 11.149.2 Member Function Documentation

**11.149.2.1 join()** [1/8]

```
QueryChainExt cbe::QueryChainExt::join (
 Container containerToQuery,
 std::string key1,
 std::string key2)
```

Performs a join request in line with function [join\(\)](#) in class [QueryChain](#).

If a call to this join function is chained with a previous call to a query, only such a query functions accepting a [QueryJoinDelegate](#) can be used as:

- [CloudBackend::query\(ContainerId,delegate::QueryJoinDelegatePtr\)](#)
- [CloudBackend::query\(ContainerId,Filter,delegate::QueryJoinDelegatePtr\)](#)
- [Container::query\(delegate::QueryJoinDelegatePtr\)](#)
- [Container::query\(Filter,delegate::QueryJoinDelegatePtr\)](#)

**Asynchronous** version of this service function.

See also

[QueryChain::join\(Container,std::string,std::string,delegate::JoinDelegatePtr\)](#)

**11.149.2.2 join()** [2/8]

```
QueryChainExt cbe::QueryChainExt::join (
 Container containerToQuery,
 std::string key1,
 std::string key2,
 Filter constraints)
```

See also

[QueryChain::join\(Container,std::string,std::string,Filter,delegate::JoinDelegatePtr\)](#)

**11.149.2.3 join()** [3/8]

```
QueryChainExt cbe::QueryChainExt::join (
 Container containerToQuery,
 std::string key1,
 std::string key2,
 Container containerForResults)
```

See also

[QueryChain::join\(Container,std::string,std::string,Container,delegate::JoinDelegatePtr\)](#)

**11.149.2.4 join()** [4/8]

```
QueryChainExt cbe::QueryChainExt::join (
 Container containerToQuery,
 std::string key1,
 std::string key2,
 Filter constraints,
 Container containerForResults)
```

See also

[QueryChain::join\(Container,std::string,std::string,Filter,Container,delegate::JoinDelegatePtr\)](#)

**11.149.2.5 join()** [5/8]

`QueryChain cbe::QueryChain::join`

Performs a join request in conjunction with a query, or previous join, to get related items from another container.

**Asynchronous** version of this service function.

**Parameters**

|                         |                                                                                                                                                                                                                                                                        |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>containerToQuery</i> | The container of which the join shall be done with.                                                                                                                                                                                                                    |
| <i>key1</i>             | The key from the previous query on which to perform the join.                                                                                                                                                                                                          |
| <i>key2</i>             | The key on objects in <code>containerToQuery</code> on which to perform the join                                                                                                                                                                                       |
| <i>joinDelegate</i>     | Pointer to a <a href="#">JoinDelegate</a> instance, implemented by the user, that receives the response of this query request.<br>I.e., either of the JoinDelegate callback functions <a href="#">onJoinSuccess()</a> or <a href="#">onJoinError()</a> will be called. |

**11.149.2.6 join()** [6/8]

`QueryChain cbe::QueryChain::join`

Same as [join\(Container, std::string, std::string, delegate::JoinDelegatePtr\)](#), but with an additional [Filter](#) parameter `constraints`.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <i>constraints</i> | A filter that provides any constraints for the query. |
|--------------------|-------------------------------------------------------|

**11.149.2.7 join()** [7/8]

`QueryChain cbe::QueryChain::join`

Same as [join\(Container, std::string, std::string, delegate::JoinDelegatePtr\)](#), but with an additional [Container](#) parameter `containerForResults`.

**Parameters**

|                            |                                                                                                              |
|----------------------------|--------------------------------------------------------------------------------------------------------------|
| <i>containerForResults</i> | Can be the <a href="#">Container</a> from the previous query if desired items should be from that container. |
|----------------------------|--------------------------------------------------------------------------------------------------------------|

**11.149.2.8 join()** [8/8]

`QueryChain cbe::QueryChain::join`

Same as [join\(Container, std::string, std::string, Container, delegate::JoinDelegatePtr\)](#), but with an additional [Filter](#) parameter `constraints`.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <i>constraints</i> | A filter that provides any constraints for the query. |
|--------------------|-------------------------------------------------------|

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

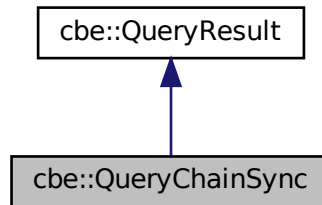
- `include/cbe/QueryChain.h`

## 11.150 cbe::QueryChainSync Class Reference

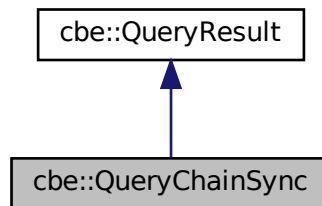
Synchronous version of querychain.

```
#include <QueryChainSync.h>
```

Inheritance diagram for cbe::QueryChainSync:



Collaboration diagram for cbe::QueryChainSync:



### Public Types

- using `JoinException` = `delegate::JoinDelegate::Exception`

### Public Member Functions

- `QueryChainSync join` (`Container` containerToQuery, `std::string` key1, `std::string` key2)
- `QueryChainSync join` (`Container` containerToQuery, `std::string` key1, `std::string` key2, `Filter` constraints)
- `QueryChainSync join` (`Container` containerToQuery, `std::string` key1, `std::string` key2, `Container` container↔  
ForResults)
- `QueryChainSync join` (`Container` containerToQuery, `std::string` key1, `std::string` key2, `Filter` constraints,  
`Container` containerForResults)
- `QueryResult getResult` () const

### Friends

- `template<class ImplT, typename... QueryAsyncArgsTs>`  
`QueryChainSync querySync` (`ImplT` &impl, `delegate::QueryJoinDelegate::ErrorInfo` \*queryJoinError, const  
`char` \*fnName, `QueryAsyncArgsTs` &&... queryAsyncArgs)

### 11.150.1 Detailed Description

Synchronous version of querychain.

Provides the possibility to chain synchronous [join\(\)](#) calls after either a [query\(\)](#) or [join\(\)](#) in a synchronous call.

### 11.150.2 Member Typedef Documentation

#### 11.150.2.1 JoinException

```
using cbe::QueryChainSync::JoinException = delegate::JoinDelegate::Exception
```

See also

[delegate::JoinDelegate::Exception](#)

### 11.150.3 Member Function Documentation

#### 11.150.3.1 join() [1/4]

```
QueryChainSync cbe::QueryChainSync::join (
 Container containerToQuery,
 std::string key1,
 std::string key2)
```

**Synchronous** version of [QueryChain::query\(\)](#), and throws an exception, [JoinException](#), in case of a failed join.

See also

[QueryChain::join\(Container,std::string,std::string,delegate::JoinDelegatePtr\)](#)

#### Parameters

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <i>containerToQuery</i> | The container the join shall be done with.                                 |
| <i>key1</i>             | The key from the previous query on which to perform the join.              |
| <i>key2</i>             | The key on objects in <i>containerToQuery</i> on which to perform the join |

#### Exceptions

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">JoinException</a> | <p>Possible only if current instance of <a href="#">QueryChainSync</a> has been retrieved by a call to any of the following methods:</p> <ul style="list-style-type: none"> <li>• <a href="#">CloudBackend::query(ContainerId)</a></li> <li>• <a href="#">CloudBackend::query(ContainerId,Filter)</a></li> <li>• <a href="#">Container::query()</a></li> <li>• <a href="#">Container::query(Filter)</a></li> </ul> |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### 11.150.3.2 join() [2/4]

```
QueryChainSync cbe::QueryChainSync::join (
 Container containerToQuery,
 std::string key1,
```

```
std::string key2,
Filter constraints)
```

Same as [join\(Container,std::string,std::string\)](#), but with an additional [Filter](#) parameter `constraints`.

See also

[join\(Container,std::string,std::string\)](#)

#### Parameters

|                    |                                                                       |
|--------------------|-----------------------------------------------------------------------|
| <i>constraints</i> | A <a href="#">Filter</a> that provides any constraints for the query. |
|--------------------|-----------------------------------------------------------------------|

### 11.150.3.3 join() [3/4]

```
QueryChainSync cbe::QueryChainSync::join (
Container containerToQuery,
std::string key1,
std::string key2,
Container containerForResults)
```

Same as [join\(Container,std::string,std::string\)](#), but with an additional [Container](#) parameter `containerForResults`.

See also

[join\(Container,std::string,std::string\)](#)

#### Parameters

|                            |                                                                                                                        |
|----------------------------|------------------------------------------------------------------------------------------------------------------------|
| <i>containerForResults</i> | The <a href="#">Container</a> from the previous query, if desired <a href="#">items</a> originate from that container. |
|----------------------------|------------------------------------------------------------------------------------------------------------------------|

### 11.150.3.4 join() [4/4]

```
QueryChainSync cbe::QueryChainSync::join (
Container containerToQuery,
std::string key1,
std::string key2,
Filter constraints,
Container containerForResults)
```

Same as [join\(Container,std::string,std::string,Container\)](#), but with an additional [Filter](#) parameter `constraints`.

See also

[join\(Container,std::string,std::string,Container\)](#)

#### Parameters

|                    |                                                                       |
|--------------------|-----------------------------------------------------------------------|
| <i>constraints</i> | A <a href="#">Filter</a> that provides any constraints for the query. |
|--------------------|-----------------------------------------------------------------------|

### 11.150.3.5 getQueryResult()

```
QueryResult cbe::QueryChainSync::getQueryResult () const
```

Returns the [QueryResult](#) provided to the delegate after the the last `query()` or `join()` call.

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

- include/cbe/QueryChainSync.h

## 11.151 cbe::delegate::QueryDelegate Class Reference

```
#include <QueryDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by*

### Public Types

- using **Success** = [cbe::QueryResult](#)
- using **Error** = [QueryError](#)

### Public Member Functions

- virtual void [onQuerySuccess](#) ([cbe::QueryResult](#) &&queryResult)=0
- virtual void [onQueryError](#) ([cbe::delegate::QueryError](#) &&error, [cbe::util::Context](#) &&context)=0

#### 11.151.1 Detailed Description

Delegate class for the asynchronous version of methods:

- [CloudBackend::queryWithPath\(\)](#), and its overloads
- [CloudBackend::query\(\)](#), and its overloads
- [CloudBackend::search\(\)](#), and its overloads
- [Container::queryWithPath\(\)](#)
- [Container::query\(\)](#), and its overloads
- [Container::search\(\)](#), and its overloads

### Example

```
#include "cbe/delegate/QueryDelegate.h"
~~~
#include <condition_variable>
#include <mutex>
~~~
class MyQueryDelegate : public cbe::delegate::QueryDelegate {
 std::mutex mutex{};
 std::condition_variable conditionVariable{};
 // Indicates operation completed - success or failure
 bool called = false;
public:
 cbe::QueryResult queryResult(cbe::DefaultCtor{});
 ErrorInfo errorInfo{};
private:
 // Called upon successful query.
 void onQuerySuccess(cbe::QueryResult&& queryResult) final {
 {
 std::lock_guard<std::mutex> lock(mutex);
 // Change state of queryResult member to indicate success
 this->queryResult = std::move(queryResult);
 // Indicate operation completed, member queryResult indicates success
 called = true;
 // If delegate is reused, clear possibly error state
 errorInfo = ErrorInfo{};
 }
 conditionVariable.notify_one();
 } // onQuerySuccess()
}
```

```

// Called upon a failed query() or join() call.
void onQueryError(cbe::delegate::QueryError&& error,
 cbe::util::Context&& context) final {
{
 std::lock_guard<std::mutex> lock{mutex};
 // Activate errorInfo member to indicate no-success
 errorInfo = ErrorInfo{std::move(context), std::move(error)};
 // Indicate operation completed, member object indicates failure
 called = true;
 // If delegate is reused, clear possibly success state
 queryResult = QueryResult{cbe::DefaultCtor{}};
}
conditionVariable.notify_one();
} // onQueryError()
public:
void waitForRsp() {
 std::unique_lock<std::mutex> lock{mutex};
 conditionVariable.wait(lock, [this] { return called; });
 // Reset called flag, so current delegate instance can be reused
 called = false;
} // waitForRsp()
}; // class MyQueryDelegate

```

Usage of the class above, see [Example of asynchronous query](#)

## 11.151.2 Member Function Documentation

### 11.151.2.1 onQuerySuccess()

```

virtual void cbe::delegate::QueryDelegate::onQuerySuccess (
 cbe::QueryResult && queryResult) [pure virtual]

```

Called upon successful query.

#### Parameters

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <i>queryResult</i> | Instance of a <a href="#">QueryResult</a> containing the result set. |
|--------------------|----------------------------------------------------------------------|

### 11.151.2.2 onQueryError()

```

virtual void cbe::delegate::QueryDelegate::onQueryError (
 cbe::delegate::QueryError && error,
 cbe::util::Context && context) [pure virtual]

```

Called upon a failed query() or join() call.

#### Parameters

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| <i>error</i>   | <a href="#">Error</a> information passed from CloudBackend SDK.                 |
| <i>context</i> | Additional context information about the original service call that has failed. |

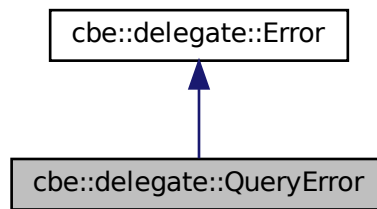
The documentation for this class was generated from the following file:

- include/cbe/delegate/QueryDelegate.h

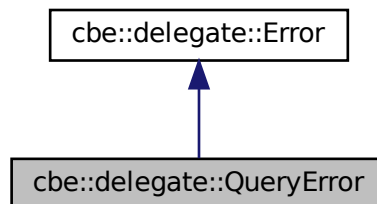
## 11.152 cbe::delegate::QueryError Class Reference

```
#include <QueryError.h>
```

Inheritance diagram for `cbe::delegate::QueryError`:



Collaboration diagram for `cbe::delegate::QueryError`:



## Public Member Functions

- **QueryError** ([ErrorCode](#) `errorCode`, `std::string` &&`reason`, `std::string` &&`message`)
- **QueryError** ([Filter](#) &&`filter`, [ErrorCode](#) `errorCode`, `std::string` &&`reason`, `std::string` &&`message`)
- `bool` **hasFilter** () const
- `const` [Filter](#) & **getFilter** () const

## Friends

- `std::ostream` & **operator**<< (`std::ostream` &`os`, const [QueryError](#) &`error`)

## Additional Inherited Members

### 11.152.1 Detailed Description

Contains error information delivered from CloudBackend SDK i connection with a failed query, or join  
The documentation for this class was generated from the following file:

- `include/cbe/delegate/QueryError.h`

## 11.153 cbe::delegate::QueryJoinDelegate Class Reference

```
#include <QueryJoinDelegate.h>
```

## Classes

- struct [Exception](#)  
*exception thrown by*

## Public Types

- using **Success** = [cbe::QueryResult](#)
- using **QueryJoinError** = [QueryError](#)
- using **Error** = [QueryJoinError](#)
- using [ErrorInfo](#) = [QueryDelegate::ErrorInfo](#)

## Public Member Functions

- virtual void [onQueryJoinSuccess](#) ([cbe::QueryResult](#) &&queryResult)=0
- virtual void [onQueryJoinError](#) ([QueryJoinError](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.153.1 Detailed Description

Delegate callback interface for asynchronous methods:

- [CloudBackend::query\(ContainerId,delegate::QueryJoinDelegatePtr\)](#) and overloads.
- [Container::query\(delegate::QueryJoinDelegatePtr\)](#) and overloads.

### 11.153.2 Member Typedef Documentation

#### 11.153.2.1 ErrorInfo

using [cbe::delegate::QueryJoinDelegate::ErrorInfo](#) = [QueryDelegate::ErrorInfo](#)  
Contains all information about a failed query.

### 11.153.3 Member Function Documentation

#### 11.153.3.1 onQueryJoinSuccess()

```
virtual void cbe::delegate::QueryJoinDelegate::onQueryJoinSuccess (
 cbe::QueryResult && queryResult) [pure virtual]
```

Called upon successful query.

##### Parameters

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <i>queryResult</i> | Instance of a <a href="#">QueryResult</a> containing the result set. |
|--------------------|----------------------------------------------------------------------|

#### 11.153.3.2 onQueryJoinError()

```
virtual void cbe::delegate::QueryJoinDelegate::onQueryJoinError (
 QueryJoinError && error,
 cbe::util::Context && context) [pure virtual]
```

Called upon a failed query() or join() call.

## Parameters

## Parameters

|                |                                                                                 |
|----------------|---------------------------------------------------------------------------------|
| <i>error</i>   | <a href="#">Error</a> information passed from CloudBackend SDK.                 |
| <i>context</i> | Additional context information about the original service call that has failed. |

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

- include/cbe/delegate/QueryJoinDelegate.h

## 11.154 cbe::delegate::QueryLoaded Struct Reference

### Public Member Functions

- virtual void [onQuerySuccess](#) ([cbe::QueryResult](#) &&queryResult)=0

### 11.154.1 Member Function Documentation

#### 11.154.1.1 onQuerySuccess()

```
virtual void cbe::delegate::QueryLoaded::onQuerySuccess (
 cbe::QueryResult && queryResult) [pure virtual]
```

Called upon successful query.

## Parameters

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <i>queryResult</i> | Instance of a <a href="#">QueryResult</a> containing the result set. |
|--------------------|----------------------------------------------------------------------|

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

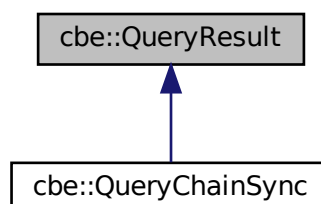
- include/cbe/delegate/QueryLoaded.h

## 11.155 cbe::QueryResult Class Reference

resultset of data retrieved.

```
#include <QueryResult.h>
```

Inheritance diagram for cbe::QueryResult:



## Public Types

- using [ItemsSnapshot](#) = std::vector< [Item](#) >  
A vector of items to hold a snapshot.

## Public Member Functions

- [Filter](#) [filter](#) () const
- [ItemsSnapshot](#) [getItemsSnapshot](#) () const  
Returns a copy of a vector containing the items for this [QueryResult](#).
- std::uint64\_t [itemsLoaded](#) () const
- std::uint64\_t [totalCount](#) () const
- std::uint64\_t [objectsLoaded](#) () const
- std::uint64\_t [containersLoaded](#) () const
- bool [containsItem](#) (ItemId itemId) const
- [QueryResult](#) ([cbe::DefaultCtor](#))  
Default constructor.
- [operator bool](#) () const  
Checks if the current instance is real.

## Friends

- class **CloudBackend**
- class **Container**
- class **QueryChain**
- class **QueryChainSync**
- class **ShareManager**
- class **SubscribeManager**
- std::ostream & [operator<<](#) (std::ostream &os, const [QueryResult](#) &queryResult)
- CBI::ItemDelegatePtr [delegate::createCbiQueryDelegate](#) ([delegate::QueryDelegatePtr](#), [cbe::util::Context](#) &&)
- CBI::ItemDelegatePtr [delegate::createCbiQueryDelegate](#) ([delegate::QueryJoinDelegatePtr](#), [cbe::util::Context](#) &&)
- CBI::ItemDelegatePtr [delegate::createCbiJoinDelegate](#) ([delegate::JoinDelegatePtr](#), [cbe::util::Context](#) &&)

### 11.155.1 Detailed Description

resultset of data retrieved.

### 11.155.2 Constructor & Destructor Documentation

#### 11.155.2.1 QueryResult()

```
cbe::QueryResult::QueryResult (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the [operator bool\(\)](#) test

### 11.155.3 Member Function Documentation

#### 11.155.3.1 filter()

```
Filter cbe::QueryResult::filter () const
```

Returns a copy of the filter used for query.

### 11.155.3.2 getItemsSnapshot()

`ItemsSnapshot cbe::QueryResult::getItemsSnapshot ( ) const`

Returns a copy of a vector containing the items for this [QueryResult](#).

The [QueryResult](#) will be updated when new data comes in, but the returned copy will not.

#### Note

If iterating, make sure to create a variable for a local copy.

#### Returns

The items matching the query.

### 11.155.3.3 itemsLoaded()

`std::uint64_t cbe::QueryResult::itemsLoaded ( ) const`

Number of `items` loaded in current [QueryResult](#).

### 11.155.3.4 totalCount()

`std::uint64_t cbe::QueryResult::totalCount ( ) const`

total number of items in the cloud matching the query result. This may be more than loaded.

### 11.155.3.5 objectsLoaded()

`std::uint64_t cbe::QueryResult::objectsLoaded ( ) const`

Returns number of `objects` loaded in to the query result.

### 11.155.3.6 containersLoaded()

`std::uint64_t cbe::QueryResult::containersLoaded ( ) const`

Returns number of `containers` loaded in to the query result.

### 11.155.3.7 containsItem()

`bool cbe::QueryResult::containsItem (   
 ItemId itemId ) const`

Checks if the [Item](#) with `id` is in the result-set.

#### Parameters

|                            |                                 |
|----------------------------|---------------------------------|
| <i>item</i> ↔<br><i>Id</i> | id number of the item asked for |
|----------------------------|---------------------------------|

### 11.155.3.8 operator bool()

`cbe::QueryResult::operator bool ( ) const [explicit]`

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [QueryResult\(cbe::DefaultCtor\)](#)

#### Returns

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/QueryResult.h

## 11.156 cbe::delegate::container::RemoveDelegate Class Reference

```
#include <RemoveDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Container::remove\(\)](#) if the request fails.*

### Public Types

- using **Success** = [RemoveSuccess](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onRemoveSuccess](#) ([cbe::ItemId](#) containerId, std::string name)=0
- virtual void [onRemoveError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

#### 11.156.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Container::remove](#)

#### 11.156.2 Member Function Documentation

##### 11.156.2.1 onRemoveSuccess()

```
virtual void cbe::delegate::container::RemoveDelegate::onRemoveSuccess (
 cbe::ItemId containerId,
 std::string name) [pure virtual]
```

Called upon successful Remove.

##### Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <i>containerId</i> | Id of the <a href="#">cbe::Container</a> being removed. |
| <i>name</i>        | Name of container that is being removed.                |

##### 11.156.2.2 onRemoveError()

```
virtual void cbe::delegate::container::RemoveDelegate::onRemoveError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/container/RemoveDelegate.h

## 11.157 cbe::delegate::group::RemoveDelegate Class Reference

```
#include <RemoveDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Group::remove\(\)](#) if the request fails.*

### Public Types

- using **Success** = [RemoveSuccess](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onRemoveSuccess](#) ([cbe::GroupId](#) groupId, std::string &&groupName)=0
- virtual void [onRemoveError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

#### 11.157.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Group::remove](#)

#### 11.157.2 Member Function Documentation

##### 11.157.2.1 onRemoveSuccess()

```
virtual void cbe::delegate::group::RemoveDelegate::onRemoveSuccess (
 cbe::GroupId groupId,
 std::string && groupName) [pure virtual]
```

Called upon successful remove.

##### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <i>groupId</i>   | Id of the <a href="#">cbe::Group</a> being removed. |
| <i>groupName</i> | Name of the group being removed.                    |

##### 11.157.2.2 onRemoveError()

```
virtual void cbe::delegate::group::RemoveDelegate::onRemoveError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/group/RemoveDelegate.h

## 11.158 cbe::delegate::object::RemoveDelegate Class Reference

```
#include <RemoveDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::Object::remove\(\)](#) if the request fails.*

## Public Types

- using **Success** = [RemoveSuccess](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onRemoveSuccess](#) ([cbe::ItemId](#) objectId, std::string name)=0
- virtual void [onRemoveError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.158.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Object::remove](#)

### 11.158.2 Member Function Documentation

#### 11.158.2.1 onRemoveSuccess()

```
virtual void cbe::delegate::object::RemoveDelegate::onRemoveSuccess (
 cbe::ItemId objectId,
 std::string name) [pure virtual]
```

Called upon successful object remove.

##### Parameters

|                 |                               |
|-----------------|-------------------------------|
| <i>objectId</i> | Id of object being removed.   |
| <i>name</i>     | Name of object being removed. |

#### 11.158.2.2 onRemoveError()

```
virtual void cbe::delegate::object::RemoveDelegate::onRemoveError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/object/RemoveDelegate.h

## 11.159 cbe::delegate::RemoveRoleDelegate Class Reference

```
#include <RemoveRoleDelegate.h>
```

## Classes

- struct [ErrorInfo](#)

- struct [Exception](#)

*exception thrown by `cbe::Role::removeRole()` if the request fails.*

## Public Types

- using **Success** = [cbe::RoleId](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onRemoveRoleSuccess](#) ([cbe::RoleId](#) &&roleId)=0
- virtual void [onRemoveRoleError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.159.1 Detailed Description

Delegate class for the asynchronous version of method:

- `cbe::Role::removeRole`

### 11.159.2 Member Function Documentation

#### 11.159.2.1 onRemoveRoleSuccess()

```
virtual void cbe::delegate::RemoveRoleDelegate::onRemoveRoleSuccess (
 cbe::RoleId && roleId) [pure virtual]
```

Called upon successful removal of the [Role](#).

#### Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <a href="#">Role</a> | the role id of the removed role. |
|----------------------|----------------------------------|

#### 11.159.2.2 onRemoveRoleError()

```
virtual void cbe::delegate::RemoveRoleDelegate::onRemoveRoleError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- `include/cbe/delegate/RemoveRoleDelegate.h`

## 11.160 cbe::delegate::RemoveRoleMemberDelegate Class Reference

```
#include <RemoveRoleMemberDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by `cbe::Group::RemoveMember()` if the request fails.*

## Public Types

- using **Success** = [MemberId](#)

- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onRemoveRoleMemberSuccess](#) ([MemberId](#) memberId)=0
- virtual void [onRemoveRoleMemberError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.160.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Group::RemoveMember](#)

### 11.160.2 Member Function Documentation

#### 11.160.2.1 onRemoveRoleMemberSuccess()

```
virtual void cbe::delegate::RemoveRoleMemberDelegate::onRemoveRoleMemberSuccess (
 MemberId memberId) [pure virtual]
```

Called upon successful RemoveMember.

#### 11.160.2.2 onRemoveRoleMemberError()

```
virtual void cbe::delegate::RemoveRoleMemberDelegate::onRemoveRoleMemberError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/RemoveRoleMemberDelegate.h

## 11.161 cbe::delegate::container::RemoveSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::container::RemoveDelegate::onRemoveSuccess](#).

```
#include <RemoveDelegate.h>
```

## Public Member Functions

- **RemoveSuccess** ([cbe::DefaultCtor](#))
- **RemoveSuccess** ([cbe::ItemId](#) containerId, std::string name)

## Public Attributes

- [cbe::ItemId](#) containerId {}
- std::string name {}

### 11.161.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::container::RemoveDelegate::onRemoveSuccess](#).

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

- include/cbe/delegate/container/RemoveDelegate.h

## 11.162 cbe::delegate::group::RemoveSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::group::RemoveDelegate::onRemoveSuccess](#).

```
#include <RemoveDelegate.h>
```

## Public Member Functions

- **RemoveSuccess** ([cbe::DefaultCtor](#))
- **RemoveSuccess** ([cbe::GroupId](#) groupId, std::string &&groupName)

## Public Attributes

- [cbe::GroupId](#) **groupId** {}
- std::string **groupName** {}

### 11.162.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::group::RemoveDelegate::onRemoveSuccess](#). The documentation for this class was generated from the following file:

- include/cbe/delegate/group/RemoveDelegate.h

## 11.163 cbe::delegate::object::RemoveSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::object::onRemoveSuccess](#).  
`#include <RemoveDelegate.h>`

## Public Member Functions

- **RemoveSuccess** ([cbe::DefaultCtor](#))
- **RemoveSuccess** ([cbe::ItemId](#) objectId, std::string name)
- **operator bool** () const

*Checks if current instance is valid.*

## Public Attributes

- [cbe::ItemId](#) **objectId** {}
- std::string **name** {}

### 11.163.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::object::onRemoveSuccess](#).

### 11.163.2 Member Function Documentation

#### 11.163.2.1 operator bool()

```
cbe::delegate::object::RemoveSuccess::operator bool () const [explicit]
```

Checks if current instance is valid.

#### Returns

`true`: is valid

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

- include/cbe/delegate/object/RemoveDelegate.h

## 11.164 cbe::delegate::container::RenameDelegate Class Reference

```
#include <RenameDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::Container::rename\(\)](#) if the request fails.*

## Public Types

- using **Success** = [cbe::Container](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onRenameSuccess](#) ([cbe::Container](#) &&container)=0
- virtual void [onRenameError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.164.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Container::rename](#)

### 11.164.2 Member Function Documentation

#### 11.164.2.1 onRenameSuccess()

```
virtual void cbe::delegate::container::RenameDelegate::onRenameSuccess (
 cbe::Container && container) [pure virtual]
```

Called upon successful Rename.

##### Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <i>container</i> | Instance of container that is being Renamed. |
|------------------|----------------------------------------------|

#### 11.164.2.2 onRenameError()

```
virtual void cbe::delegate::container::RenameDelegate::onRenameError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/container/RenameDelegate.h

## 11.165 cbe::delegate::group::RenameDelegate Class Reference

```
#include <RenameDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::Group::rename\(\)](#) if the request fails.*

## Public Types

- using **Success** = [RenameSuccess](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onRenameSuccess](#) ([cbe::Group](#) &&group, std::string &&newName)=0
- virtual void [onRenameError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.165.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Group::rename](#)

### 11.165.2 Member Function Documentation

#### 11.165.2.1 onRenameSuccess()

```
virtual void cbe::delegate::group::RenameDelegate::onRenameSuccess (
 cbe::Group && group,
 std::string && newName) [pure virtual]
```

Called upon successful rename.

#### 11.165.2.2 onRenameError()

```
virtual void cbe::delegate::group::RenameDelegate::onRenameError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/group/RenameDelegate.h

## 11.166 cbe::delegate::object::RenameDelegate Class Reference

```
#include <RenameDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)
  - exception thrown by [cbe::Object::rename\(\)](#) if the request fails.*

## Public Types

- using **Success** = [cbe::Object](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onRenameSuccess](#) ([cbe::Object](#) &&object)=0
- virtual void [onRenameError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.166.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Object::rename](#)

### 11.166.2 Member Function Documentation

#### 11.166.2.1 onRenameSuccess()

```
virtual void cbe::delegate::object::RenameDelegate::onRenameSuccess (
 cbe::Object && object) [pure virtual]
```

Called upon successful Rename.

##### Parameters

|               |                                            |
|---------------|--------------------------------------------|
| <i>object</i> | Instance of object that are being renamed. |
|---------------|--------------------------------------------|

#### 11.166.2.2 onRenameError()

```
virtual void cbe::delegate::object::RenameDelegate::onRenameError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

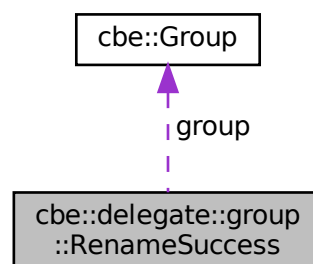
- include/cbe/delegate/object/RenameDelegate.h

## 11.167 cbe::delegate::group::RenameSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::group::RenameDelegate::onRenameSuccess](#).

```
#include <RenameSuccess.h>
```

Collaboration diagram for cbe::delegate::group::RenameSuccess:



### Public Member Functions

- **RenameSuccess** ([cbe::DefaultCtor](#))
- **RenameSuccess** ([cbe::Group](#) &&group, std::string &&newName)

- `operator bool ()` const  
Checks if current instance is valid.

## Public Attributes

- `cbe::Group` **group** {`cbe::DefaultCtor`{}}
- `std::string` **newName** {}

### 11.167.1 Detailed Description

Convenience type that bundles all parameters passed to method `cbe::delegate::group::RenameDelegate::onRenameSuccess`.

### 11.167.2 Member Function Documentation

#### 11.167.2.1 `operator bool()`

`cbe::delegate::group::RenameSuccess::operator bool ( ) const [explicit]`  
Checks if current instance is valid.

#### Returns

`true`: is valid

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

- `include/cbe/delegate/group/RenameSuccess.h`

## 11.168 `cbe::Request` Struct Reference

```
#include <Member.h>
```

### Public Member Functions

- **Request** (`cbe::UserId` `userId`, `std::string` `alias`, `std::string` `applicationComment`, `cbe::Date` `date`)

### Public Attributes

- `cbe::UserId` **userId** {}
- `std::string` **alias** {}
- `std::string` **applicationComment** {}
- `cbe::Date` **date** {}

#### 11.168.1 Detailed Description

Contains information on a request for a `Member` to join a group.  
The documentation for this struct was generated from the following file:

- `include/cbe/Member.h`

## 11.169 `cbe::Role` Class Reference

User role information.  
`#include <Role.h>`

## Public Types

- using [ListMembersDelegatePtr](#) = [delegate::ListMembersDelegatePtr](#)
- using [ListMembersException](#) = [delegate::ListMembersDelegate::Exception](#)
- using [ListMembersError](#) = [delegate::ListMembersDelegate::ErrorInfo](#)
- using [AddRoleMemberDelegatePtr](#) = [delegate::AddRoleMemberDelegatePtr](#)
- using [AddRoleMemberException](#) = [delegate::AddRoleMemberDelegate::Exception](#)
- using [AddRoleMemberError](#) = [delegate::AddRoleMemberDelegate::ErrorInfo](#)
- using [RemoveRoleMemberDelegatePtr](#) = [delegate::RemoveRoleMemberDelegatePtr](#)
- using [RemoveRoleMemberException](#) = [delegate::RemoveRoleMemberDelegate::Exception](#)
- using [RemoveRoleMemberError](#) = [delegate::RemoveRoleMemberDelegate::ErrorInfo](#)

## Public Member Functions

- `std::string name () const`
- `cbe::RoleId id () const`
- `cbe::GroupId groupId () const`
- `void listMembers (ListMembersDelegatePtr delegate)`  
*Lists the members of this role,.*
- `std::vector< cbe::Member > listMembers ()`  
*Synchronous [exception] **Synchronous** version of `listMembers(ListMembersDelegatePtr)` , and **throws an exception**, `ListMembersException`, in case of a failed call.  
See `listMembers(ListMembersDelegatePtr)`*
- `cbe::util::Optional< std::vector< cbe::Member > > listMembers (ListMembersError &error)`  
*Synchronous [non-throwing] **Synchronous** version of `listMembers(ListMembersDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.  
See `listMembers(ListMembersDelegatePtr)`*
- `void addRoleMember (MemberId memberId, AddRoleMemberDelegatePtr delegate)`  
*Adds a member to this role.*
- `MemberId addRoleMember (MemberId memberId)`  
*Synchronous [exception] **Synchronous** version of `AddRoleMember(MemberId, AddRoleMemberDelegatePtr)` , and **throws an exception**, `AddRoleMemberException`, in case of a failed call.  
See `AddRoleMember(AddRoleMemberDelegatePtr)`*
- `cbe::util::Optional< MemberId > addRoleMember (MemberId memberId, AddRoleMemberError &error)`  
*Synchronous [non-throwing] **Synchronous** version of `AddRoleMember(memberId, AddRoleMemberDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.  
See `AddRoleMember(memberId, AddRoleMemberDelegatePtr)`*
- `void removeRoleMember (MemberId memberId, RemoveRoleMemberDelegatePtr delegate)`  
*Removes a member to this role.*
- `MemberId removeRoleMember (MemberId memberId)`  
*Synchronous [exception] **Synchronous** version of `RemoveRoleMember(MemberId memberId, RemoveRoleMemberDelegatePtr)` , and **throws an exception**, `RemoveRoleMemberException`, in case of a failed call.  
See `RemoveRoleMember(RemoveRoleMemberDelegatePtr)`*
- `cbe::util::Optional< MemberId > removeRoleMember (MemberId memberId, RemoveRoleMemberError &error)`  
*Synchronous [non-throwing] **Synchronous** version of `RemoveRoleMember(memberId, RemoveRoleMemberDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.  
See `RemoveRoleMember(memberId, RemoveRoleMemberDelegatePtr)`*
- `Role (cbe::DefaultCtor)`  
*Default constructor.*
- `operator bool () const`  
*Checks if the current instance is real.*

## Friends

- class **Group**

### 11.169.1 Detailed Description

User role information.

### 11.169.2 Member Typedef Documentation

#### 11.169.2.1 ListMembersDelegatePtr

using `cbe::Role::ListMembersDelegatePtr = delegate::ListMembersDelegatePtr`

Pointer to `cbe::delegate::ListMembersDelegate` that is passed into asynchronous version of method `listMembers()`

#### 11.169.2.2 ListMembersException

using `cbe::Role::ListMembersException = delegate::ListMembersDelegate::Exception`

Pointer to `cbe::delegate::ListMembersDelegate` that is passed into asynchronous version of method `listMembers()`

See `delegate::object::ListMembersDelegate::Exception`

#### 11.169.2.3 ListMembersError

using `cbe::Role::ListMembersError = delegate::ListMembersDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `listMembers()`

See `delegate::ListMembersDelegate::ErrorInfo`

#### 11.169.2.4 AddRoleMemberDelegatePtr

using `cbe::Role::AddRoleMemberDelegatePtr = delegate::AddRoleMemberDelegatePtr`

Pointer to `AddRoleMemberDelegate` that is passed into:

- `cbe::Group::AddRoleMember(AddRoleMemberDelegatePtr).`

#### 11.169.2.5 AddRoleMemberException

using `cbe::Role::AddRoleMemberException = delegate::AddRoleMemberDelegate::Exception`

Pointer to `cbe::delegate::AddRoleMemberDelegate` that is passed into asynchronous version of method `AddRoleMember()` See `delegate::object::AddRoleMemberDelegate::Exception`

#### 11.169.2.6 AddRoleMemberError

using `cbe::Role::AddRoleMemberError = delegate::AddRoleMemberDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `AddRoleMember()`

See `delegate::AddRoleMemberDelegate::ErrorInfo`

#### 11.169.2.7 RemoveRoleMemberDelegatePtr

using `cbe::Role::RemoveRoleMemberDelegatePtr = delegate::RemoveRoleMemberDelegatePtr`

Pointer to `RemoveRoleMember` that is passed into:

`Group::RemoveRoleMember(std::string, RemoveRoleMemberDelegatePtr).`

#### 11.169.2.8 RemoveRoleMemberException

using `cbe::Role::RemoveRoleMemberException = delegate::RemoveRoleMemberDelegate::Exception`

Pointer to `cbe::delegate::RemoveRoleMember` that is passed into asynchronous version of method `RemoveRoleMember()` See `delegate::object::RemoveRoleMember::Exception`

### 11.169.2.9 RemoveRoleMemberError

using `cbe::Role::RemoveRoleMemberError` = `delegate::RemoveRoleMemberDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `RemoveRoleMember()`

See `delegate::RemoveRoleMember::ErrorInfo`

## 11.169.3 Constructor & Destructor Documentation

### 11.169.3.1 Role()

```
cbe::Role::Role (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the `operator bool()` test

## 11.169.4 Member Function Documentation

### 11.169.4.1 name()

```
std::string cbe::Role::name () const
```

Name of the role.

### 11.169.4.2 id()

```
cbe::RoleId cbe::Role::id () const
```

Id of the role.

### 11.169.4.3 groupId()

```
cbe::GroupId cbe::Role::groupId () const
```

Id of the group the role belongs to.

### 11.169.4.4 listMembers() [1/3]

```
void cbe::Role::listMembers (
 ListMembersDelegatePtr delegate)
```

Lists the members of this role,.

#### Parameters

|                 |                                                                                                   |
|-----------------|---------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <code>delegate::ListMembersDelegate</code> instance that is implemented by the user. |
|-----------------|---------------------------------------------------------------------------------------------------|

### 11.169.4.5 listMembers() [2/3]

```
std::vector<cbe::Member> cbe::Role::listMembers ()
```

Synchronous [exception] **Synchronous** version of `listMembers(ListMembersDelegatePtr)` , and **throws an exception**, `ListMembersException`, in case of a failed call.

See `listMembers(ListMembersDelegatePtr)`

#### Returns

Information about the `listMembers` object — if the call was successful.

See `cbe::delegate::ListMembersSuccess`

## Exceptions

|                                      |
|--------------------------------------|
| <a href="#">ListMembersException</a> |
|--------------------------------------|

**11.169.4.6 listMembers()** [3/3]

```
cbe::util::Optional<std::vector<cbe::Member> > cbe::Role::listMembers (
 ListMembersError & error)
```

Synchronous [non-throwing] **Synchronous** version of [listMembers\(ListMembersDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listMembers\(ListMembersDelegatePtr\)](#)

## Parameters

|                  |                    |                                                                                                                                                                                                                                  |
|------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>out</code> | <code>error</code> | Return parameter containing the error information in case of a failed call. An empty return value will indicate failure, and the <a href="#">ListMembersError</a> object passed in will be populated with the error information. |
|------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.169.4.7 addRoleMember()** [1/3]

```
void cbe::Role::addRoleMember (
 MemberId memberId,
 AddRoleMemberDelegatePtr delegate)
```

Adds a member to this role.

## Parameters

|                       |                                                                                                        |
|-----------------------|--------------------------------------------------------------------------------------------------------|
| <code>memberId</code> | The id of the member to add to the role.                                                               |
| <code>delegate</code> | Pointer to a <a href="#">delegate::AddRoleMemberDelegate</a> instance that is implemented by the user. |

**11.169.4.8 addRoleMember()** [2/3]

```
MemberId cbe::Role::addRoleMember (
 MemberId memberId)
```

Synchronous [exception] **Synchronous** version of `AddRoleMember(MemberId, AddRoleMemberDelegatePtr)` , and **throws an exception**, [AddRoleMemberException](#), in case of a failed call.

See `AddRoleMember(AddRoleMemberDelegatePtr)`

## Returns

Information about the `AddRoleMember` object — if the call was successful.

See `cbe::delegate::AddRoleMemberDelegate::Success`

## Exceptions

|                                        |
|----------------------------------------|
| <a href="#">AddRoleMemberException</a> |
|----------------------------------------|

**11.169.4.9 addRoleMember()** [3/3]

```
cbe::util::Optional<MemberId> cbe::Role::addRoleMember (
 MemberId memberId,
 AddRoleMemberError & error)
```

Synchronous [non-throwing] **Synchronous** version of AddRoleMember(memberId, AddRoleMemberDelegatePtr) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See AddRoleMember(memberId, AddRoleMemberDelegatePtr)

**Parameters**

|     |       |                                                                                                                                                                                                                                       |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">AddRoleMemberError</a> object passed in will be populated with the error information. |
|-----|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.169.4.10 removeRoleMember()** [1/3]

```
void cbe::Role::removeRoleMember (
 MemberId memberId,
 RemoveRoleMemberDelegatePtr delegate)
```

Removes a member to this role.

**Parameters**

|                 |                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------|
| <i>memberId</i> | The id of the member to remove from the role.                                                  |
| <i>delegate</i> | Pointer to a <code>delegate::RemoveRoleMember</code> instance that is implemented by the user. |

**11.169.4.11 removeRoleMember()** [2/3]

```
MemberId cbe::Role::removeRoleMember (
 MemberId memberId)
```

Synchronous [exception] **Synchronous** version of RemoveRoleMember(MemberId memberId, RemoveRoleMemberDelegatePtr) , and **throws an exception**, [RemoveRoleMemberException](#), in case of a failed call.

See RemoveRoleMember(RemoveRoleMemberDelegatePtr)

**Returns**

Information about the RemoveRoleMember object — if the call was successful.  
See `cbe::delegate::RemoveRoleMember::Success`

**Exceptions**

|                                           |  |
|-------------------------------------------|--|
| <a href="#">RemoveRoleMemberException</a> |  |
|-------------------------------------------|--|

**11.169.4.12 removeRoleMember()** [3/3]

```
cbe::util::Optional<MemberId> cbe::Role::removeRoleMember (
 MemberId memberId,
 RemoveRoleMemberError & error)
```

Synchronous [non-throwing] **Synchronous** version of RemoveRoleMember(memberId, RemoveRoleMemberDelegatePtr), and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See RemoveRoleMember(memberId, RemoveRoleMemberDelegatePtr)

**Parameters**

|     |       |                                                                                                                                                                                                                                          |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">RemoveRoleMemberError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.169.4.13 operator bool()**

```
cbe::Role::operator bool () const [explicit]
```

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [Role\(cbe::DefaultCtor\)](#)

**Returns**

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/Role.h

**11.170 cbe::delegate::SearchGroupsDelegate Class Reference**

```
#include <SearchGroupsDelegate.h>
```

**Classes**

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::GroupManager::searchGroups\(\)](#) if the request fails.*

**Public Types**

- using **Success** = [cbe::GroupQueryResult](#)
- using **Error** = [delegate::Error](#)

**Public Member Functions**

- virtual void [onSearchGroupsSuccess](#) ([cbe::GroupQueryResult](#) &&queryResult)=0
- virtual void [onSearchGroupsError](#) ([delegate::Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.170.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::GroupManager::searchGroups\(\)](#)

### 11.170.2 Member Function Documentation

#### 11.170.2.1 onSearchGroupsSuccess()

```
virtual void cbe::delegate::SearchGroupsDelegate::onSearchGroupsSuccess (
 cbe::GroupQueryResult && queryResult) [pure virtual]
```

Called upon successful search.

##### Parameters

|                    |                                                                                          |
|--------------------|------------------------------------------------------------------------------------------|
| <i>queryResult</i> | Ref. instance of <a href="#">cbe::GroupQueryResult</a> holding the result of the search. |
|--------------------|------------------------------------------------------------------------------------------|

#### 11.170.2.2 onSearchGroupsError()

```
virtual void cbe::delegate::SearchGroupsDelegate::onSearchGroupsError (
 delegate::Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/SearchGroupsDelegate.h

## 11.171 cbe::ShareData Struct Reference

```
#include <Utility.h>
```

### Public Member Functions

- `template<class ShareDataT >`  
**ShareData** (ShareDataT &&rh)
- **ShareData** (uint64\_t id, bool isUserId)

### Public Attributes

- uint64\_t **id**
- bool **isUserId**

#### 11.171.1 Detailed Description

This is the data used from shares(shareId) to keep track of user/group-id relating to a shareId  
The documentation for this struct was generated from the following file:

- include/cbe/Utility.h

## 11.172 cbe::delegate::ShareDelegate Class Reference

```
#include <ShareDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by*

## Public Types

- using **Success** = [ShareSuccess](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onShareSuccess](#) ([cbe::ShareId](#) shareId)=0
- virtual void [onShareError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.172.1 Detailed Description

Delegate class for the asynchronous version of methods:

- [cbe::Container::share](#)
- [cbe::Object::share](#)

### 11.172.2 Member Function Documentation

#### 11.172.2.1 onShareSuccess()

```
virtual void cbe::delegate::ShareDelegate::onShareSuccess (
 cbe::ShareId shareId) [pure virtual]
```

Called upon successful share.

#### Parameters

|                |                                                                                                                       |
|----------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>shareId</i> | Id of the share.<br>Can be used when unsharing.<br>See <a href="#">Asynchronous version of cbe::Object::unshare()</a> |
|----------------|-----------------------------------------------------------------------------------------------------------------------|

#### 11.172.2.2 onShareError()

```
virtual void cbe::delegate::ShareDelegate::onShareError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/ShareDelegate.h

## 11.173 cbe::ShareManager Class Reference

Managing Shares.

```
#include <ShareManager.h>
```

## Public Types

- using [ListSharesDelegatePtr](#) = [delegate::ListSharesDelegatePtr](#)
- using [ListSharesException](#) = [delegate::ListSharesDelegate::Exception](#)
- using [ListSharesError](#) = [delegate::ListSharesDelegate::ErrorInfo](#)
- using [ListMySharesException](#) = [delegate::ListSharesDelegate::Exception](#)
- using [ListMySharesError](#) = [delegate::ListSharesDelegate::ErrorInfo](#)

## Public Member Functions

- void [listAvailableShares](#) ([ListSharesDelegatePtr](#) delegate)  
*Lists the shares exposed by other users to current user. This will give specific share information.*
- [cbe::QueryResult listAvailableShares](#) ()  
*Synchronous [exception] **Synchronous** version of [listAvailableShares\(ListSharesDelegatePtr\)](#) , and **throws an exception**, [ListSharesException](#), in case of a failed call.*  
*See [listAvailableShares\(ListSharesDelegatePtr\)](#)*
- [cbe::util::Optional< cbe::QueryResult > listAvailableShares](#) ([ListSharesError](#) &error)  
*Synchronous [non-throwing] **Synchronous** version of [listAvailableShares\(ListSharesDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*  
*See [listAvailableShares\(ListSharesDelegatePtr\)](#)*
- void [listMyShares](#) ([ListSharesDelegatePtr](#) delegate)  
*Lists the shares exposed by current user. This will give specific share information.*
- [delegate::ListSharesDelegate::Success listMyShares](#) ()  
*Synchronous [exception] **Synchronous** version of [listMyShares\(ListSharesDelegatePtr\)](#) , and **throws an exception**, [ListMySharesException](#), in case of a failed call.*  
*See [listMyShares\(ListSharesDelegatePtr\)](#)*
- [cbe::util::Optional< delegate::ListSharesDelegate::Success > listMyShares](#) ([ListSharesError](#) &error)  
*Synchronous [non-throwing] **Synchronous** version of [listMyShares\(ListSharesDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.*  
*See [listMyShares\(ListSharesDelegatePtr\)](#)*
- [ShareManager](#) ([cbe::DefaultCtor](#))  
*Default constructor.*
- [operator bool](#) () const  
*Checks if the current instance is real.*

## Friends

- class [CloudBackend](#)

### 11.173.1 Detailed Description

Managing Shares.

This class represents a list of Shares.

### 11.173.2 Member Typedef Documentation

#### 11.173.2.1 ListSharesDelegatePtr

using [cbe::ShareManager::ListSharesDelegatePtr](#) = [delegate::ListSharesDelegatePtr](#)

Pointer to ListSharesDelegate that is passed into:

- [ShareManager::listAvailableShares\(ListSharesDelegatePtr\)](#)
- [ShareManager::listMyShares\(ListSharesDelegatePtr\)](#)

### 11.173.2.2 ListSharesException

using `cbe::ShareManager::ListSharesException = delegate::ListSharesDelegate::Exception`

Pointer to `cbe::delegate::ListSharesDelegate` that is passed into asynchronous version of method `listAvailableShares()`

See `delegate::object::ListSharesDelegate::Exception`

### 11.173.2.3 ListSharesError

using `cbe::ShareManager::ListSharesError = delegate::ListSharesDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `listAvailableShares()`

See `delegate::ListSharesDelegate::ErrorInfo`

### 11.173.2.4 ListMySharesException

using `cbe::ShareManager::ListMySharesException = delegate::ListSharesDelegate::Exception`

Pointer to `cbe::delegate::ListSharesDelegate` that is passed into asynchronous version of method `listMyShares()`

See `delegate::object::ListSharesDelegate::Exception`

### 11.173.2.5 ListMySharesError

using `cbe::ShareManager::ListMySharesError = delegate::ListSharesDelegate::ErrorInfo`

Forms the type of the `error` return parameter for the synchronous version of method `listMyShares()`

See `delegate::ListSharesDelegate::ErrorInfo`

## 11.173.3 Constructor & Destructor Documentation

### 11.173.3.1 ShareManager()

```
cbe::ShareManager::ShareManager (
 cbe::DefaultCtor)
```

Default constructor.

Construct a new object with the `DefaultCtor` to enable the `operator bool()` test

## 11.173.4 Member Function Documentation

### 11.173.4.1 listAvailableShares() [1/3]

```
void cbe::ShareManager::listAvailableShares (
 ListSharesDelegatePtr delegate)
```

Lists the shares exposed by other users to current user. This will give specific share information.

#### Parameters

|                 |                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <code>delegate::ListSharesDelegate</code> instance that is implemented by the user. |
|-----------------|--------------------------------------------------------------------------------------------------|

### 11.173.4.2 listAvailableShares() [2/3]

```
cbe::QueryResult cbe::ShareManager::listAvailableShares ()
```

Synchronous [exception] **Synchronous** version of `listAvailableShares(ListSharesDelegatePtr)`, and **throws an exception**, `ListSharesException`, in case of a failed call.

See `listAvailableShares(ListSharesDelegatePtr)`

## Returns

Information about the listAvailableShares object — if the call was successful.  
See cbe::delegate::ListSharesDelegate::Success

## Exceptions

|                                     |  |
|-------------------------------------|--|
| <a href="#">ListSharesException</a> |  |
|-------------------------------------|--|

## 11.173.4.3 listAvailableShares() [3/3]

```
cbe::util::Optional<cbe::QueryResult> cbe::ShareManager::listAvailableShares (
 ListSharesError & error)
```

Synchronous [non-throwing] **Synchronous** version of [listAvailableShares\(ListSharesDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listAvailableShares\(ListSharesDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                    |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ListSharesError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

## 11.173.4.4 listMyShares() [1/3]

```
void cbe::ShareManager::listMyShares (
 ListSharesDelegatePtr delegate)
```

Lists the shares exposed by current user. This will give specific share information.

## Parameters

|                 |                                                                                                     |
|-----------------|-----------------------------------------------------------------------------------------------------|
| <i>delegate</i> | Pointer to a <a href="#">delegate::ListSharesDelegate</a> instance that is implemented by the user. |
|-----------------|-----------------------------------------------------------------------------------------------------|

## 11.173.4.5 listMyShares() [2/3]

```
delegate::ListSharesDelegate::Success cbe::ShareManager::listMyShares ()
```

Synchronous [exception] **Synchronous** version of [listMyShares\(ListSharesDelegatePtr\)](#) , and **throws an exception**, [ListMySharesException](#), in case of a failed call.

See [listMyShares\(ListSharesDelegatePtr\)](#)

## Returns

Information about the listMyShares object — if the call was successful.  
See cbe::delegate::ListSharesDelegate::Success

## Exceptions

|                                       |  |
|---------------------------------------|--|
| <a href="#">ListMySharesException</a> |  |
|---------------------------------------|--|

## 11.173.4.6 listMyShares() [3/3]

```
cbe::util::Optional<delegate::ListSharesDelegate::Success> cbe::ShareManager::listMyShares (
 ListSharesError & error)
```

Synchronous [non-throwing] **Synchronous** version of [listMyShares\(ListSharesDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See [listMyShares\(ListSharesDelegatePtr\)](#)

## Parameters

|     |       |                                                                                                                                                                                                                                      |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">ListMySharesError</a> object passed in will be populated with the error information. |
|-----|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

## 11.173.4.7 operator bool()

```
cbe::ShareManager::operator bool () const [explicit]
```

Checks if the current instance is real.

An "unreal" instance implies typically a failed event.

Relies on the Default constructor [ShareManager\(cbe::DefaultCtor\)](#)

## Returns

`true` : is real

`false` : unreal; got nullptr; if current instance is unbound/undefined. I.e., if it is only default constructed.

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

- include/cbe/ShareManager.h

## 11.174 cbe::delegate::ShareSuccess Class Reference

Convenience type that bundles the parameter passed to method [cbe::delegate::ShareDelegate::onShareSuccess](#).  

```
#include <ShareDelegate.h>
```

## Public Member Functions

- **ShareSuccess** ([cbe::DefaultCtor](#))
- **ShareSuccess** ([cbe::ShareId](#) shareId)
- **operator bool** () const

*Checks if current instance is valid.*

## Public Attributes

- [cbe::ShareId](#) shareId = 0

### 11.174.1 Detailed Description

Convenience type that bundles the parameter passed to method [cbe::delegate::ShareDelegate::onShareSuccess](#).

### 11.174.2 Member Function Documentation

#### 11.174.2.1 operator bool()

```
cbe::delegate::ShareSuccess::operator bool () const [explicit]
```

Checks if current instance is valid.

Returns

true: is valid

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

- include/cbe/delegate/ShareDelegate.h

## 11.175 cbe::Stream Class Reference

A data file attached to [Object](#).

```
#include <Stream.h>
```

### Public Attributes

- [cbe::StreamId](#) `streamId` {}
- `std::size_t` `length` {}

### Friends

- class `Object`

### 11.175.1 Detailed Description

A data file attached to [Object](#).

An object kan have zero, one or many streams.

### 11.175.2 Member Data Documentation

#### 11.175.2.1 streamId

```
cbe::StreamId cbe::Stream::streamId {}
```

the id number of the stream

#### 11.175.2.2 length

```
std::size_t cbe::Stream::length {}
```

the size in Bytes

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

- include/cbe/Stream.h

## 11.176 cbe::Subscribe Class Reference

Managing a subscribed [Item](#).

```
#include <Subscribe.h>
```

## Public Member Functions

- [cbe::Date](#) [getDate](#) () const
- std::string [getTitle](#) () const
- std::string [getDescription](#) () const
- std::string [getPassword](#) () const
- [cbe::PublishAccess](#) [getSecurity](#) () const
- [cbe::PublishVisibility](#) [getPrivacy](#) () const
- [cbe::Subscribeld](#) [getSubscribeld](#) () const
- [cbe::PublishId](#) [getPublishId](#) () const
- [cbe::UserId](#) [getOwner](#) () const
- void [unSubscribe](#) ()
- **Subscribe** ([cbe::DefaultCtor](#))
- **operator bool** () const

## Friends

- class **CloudBackend**
- class **Item**

### 11.176.1 Detailed Description

Managing a subscribed [Item](#).

To inspect the settings of a subscription.

### 11.176.2 Member Function Documentation

#### 11.176.2.1 getDate()

[cbe::Date](#) [cbe::Subscribe::getDate](#) ( ) const

Gets the date as unix epoch number

#### 11.176.2.2 getTitle()

std::string [cbe::Subscribe::getTitle](#) ( ) const

Gets the title

#### 11.176.2.3 getDescription()

std::string [cbe::Subscribe::getDescription](#) ( ) const

Gets the description

#### 11.176.2.4 getPassword()

std::string [cbe::Subscribe::getPassword](#) ( ) const

Checks if a password has been set.

##### Returns

"true" : if a password has been set

"" : empty string if not.

#### 11.176.2.5 getSecurity()

[cbe::PublishAccess](#) [cbe::Subscribe::getSecurity](#) ( ) const

Gets the security see [cbe::PublishAccess](#) enum

**11.176.2.6 getPrivacy()**

`cbe::PublishVisibility` `cbe::Subscribe::getPrivacy ( ) const`  
 Gets the privacy see `cbe::PublishVisibility` enum

**11.176.2.7 getSubscribeId()**

`cbe::SubscribeId` `cbe::Subscribe::getSubscribeId ( ) const`  
 Gets the subscribe id number

**11.176.2.8 getPublishId()**

`cbe::PublishId` `cbe::Subscribe::getPublishId ( ) const`  
 Gets the publish id number

**11.176.2.9 getOwner()**

`cbe::UserId` `cbe::Subscribe::getOwner ( ) const`  
 Gets the owner id number

**11.176.2.10 unSubscribe()**

`void cbe::Subscribe::unSubscribe ( )`  
 unSubscribe to this subscription.  
 The documentation for this class was generated from the following file:

- `include/cbe/Subscribe.h`

**11.177 cbe::delegate::SubscribeDelegate Class Reference**

`#include <SubscribeDelegate.h>`

**Classes**

- struct `ErrorInfo`
- struct `Exception`

*exception thrown by `cbe::SubscribeManager::subscribe()` and its overloads if the request fails.*

**Public Types**

- using `Success` = `cbe::Items`
- using `Error` = `delegate::Error`

**Public Member Functions**

- virtual void `onSubscribeSuccess` (`cbe::Items` &&success)=0
- virtual void `onSubscribeError` (`Error` &&error, `cbe::util::Context` &&context)=0

**11.177.1 Detailed Description**

Delegate class for the asynchronous version of method:

- `cbe::SubscribeManager::subscribe`

**11.177.2 Member Function Documentation**

### 11.177.2.1 onSubscribeSuccess()

```
virtual void cbe::delegate::SubscribeDelegate::onSubscribeSuccess (
 cbe::Items && success) [pure virtual]
```

Called upon successful subscribe.

#### Parameters

|                |                                              |
|----------------|----------------------------------------------|
| <i>success</i> | Instance of Items that are being subscribed. |
|----------------|----------------------------------------------|

### 11.177.2.2 onSubscribeError()

```
virtual void cbe::delegate::SubscribeDelegate::onSubscribeError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/SubscribeDelegate.h

## 11.178 cbe::SubscribeManager Class Reference

For managing subscriptions.

```
#include <SubscribeManager.h>
```

### Public Types

- using [GetSubscriptionsDelegatePtr](#) = [delegate::GetSubscriptionsDelegatePtr](#)
- using [GetSubscriptionsException](#) = [delegate::GetSubscriptionsDelegate::Exception](#)
- using [GetSubscriptionsError](#) = [delegate::GetSubscriptionsDelegate::ErrorInfo](#)
- using [SubscribeDelegatePtr](#) = [delegate::SubscribeDelegatePtr](#)
- using [SubscribeException](#) = [delegate::SubscribeDelegate::Exception](#)
- using [SubscribeError](#) = [delegate::SubscribeDelegate::ErrorInfo](#)

### Public Member Functions

- void [getSubscriptions](#) ([GetSubscriptionsDelegatePtr](#) subscribeDelegate)  
*List your current subscriptions.*
- [delegate::GetSubscriptionsDelegate::Success](#) [getSubscriptions](#) ()  
*Synchronous [exception] **Synchronous** version of [getSubscriptions\(GetSubscriptionsDelegatePtr\)](#) , and **throws an exception**, [GetSubscriptionsException](#), in case of a failed call.*  
*See [getSubscriptions\(GetSubscriptionsDelegatePtr\)](#)*
- [cbe::util::Optional](#)< [delegate::GetSubscriptionsDelegate::Success](#) > [getSubscriptions](#) ([GetSubscriptionsError](#) &error)  
*Synchronous [non-throwing] **Synchronous** version of [getSubscriptions\(GetSubscriptionsDelegatePtr\)](#) , and **throws no exception** on error, instead the out/return parameter *error* is used to provide the error information in connection with a failed call.*  
*See [getSubscriptions\(GetSubscriptionsDelegatePtr\)](#)*
- void [subscribe](#) ([cbe::UserId](#) sharingUserId, std::string sharingUserName, [cbe::PublishId](#) publishId, std::string publishName, std::string password, std::string subscribeName, [SubscribeDelegatePtr](#) subscribeDelegate)  
*Subscribes to a [Publish](#) using all parameters.*
- [delegate::SubscribeDelegate::Success](#) [subscribe](#) ([cbe::UserId](#) sharingUserId, std::string sharingUserName, [cbe::PublishId](#) publishId, std::string publishName, std::string password, std::string subscribeName)

*Synchronous [exception]* **Synchronous** version of `subscribe(cbe::UserId, std::string, cbe::PublishId, std::string, std::string, std::string, SubscribeDelegatePtr)`, and **throws an exception**, `SubscribeException`, in case of a failed call.  
See `subscribe(SubscribeDelegatePtr)`

- `cbe::util::Optional< delegate::SubscribeDelegate::Success >` `subscribe(cbe::UserId sharingUserId, std::string sharingUserName, cbe::PublishId publishId, std::string publishName, std::string password, std::string subscribeName, SubscribeError &error)`

*Synchronous [non-throwing]* **Synchronous** version of `subscribe(sharingUserId, sharingUserName, publishId, publishName, password, subscribeName, SubscribeDelegatePtr)`, and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.  
See `subscribe(sharingUserId, sharingUserName, publishId, publishName, password, subscribeName, SubscribeDelegatePtr)`

- `void subscribe(std::string sharingUserName, std::string publishName, SubscribeDelegatePtr subscribeDelegate)`

Subscribes to a `Publish` using names.

- `void subscribe(cbe::UserId sharingUserId, cbe::PublishId publishId, std::string subscribeName, SubscribeDelegatePtr subscribeDelegate)`

Subscribes to a `Publish` using ids.

- `SubscribeManager(cbe::DefaultCtor)`
- `operator bool()` const

## Friends

- class `CloudBackend`

## 11.178.1 Detailed Description

For managing subscriptions.

## 11.178.2 Member Typedef Documentation

### 11.178.2.1 GetSubscriptionsDelegatePtr

using `cbe::SubscribeManager::GetSubscriptionsDelegatePtr = delegate::GetSubscriptionsDelegatePtr`  
Pointer to `GetSubscriptionsDelegate` that is passed into asynchronous version of method:

- `getSubscriptions()`

### 11.178.2.2 GetSubscriptionsException

using `cbe::SubscribeManager::GetSubscriptionsException = delegate::GetSubscriptionsDelegate::Exception`  
Pointer to `cbe::delegate::GetSubscriptionsDelegate` that is passed into asynchronous version of method `getSubscriptions()` See `delegate::object::GetSubscriptionsDelegate::Exception`

### 11.178.2.3 GetSubscriptionsError

using `cbe::SubscribeManager::GetSubscriptionsError = delegate::GetSubscriptionsDelegate::ErrorInfo`  
Forms the type of the `error` return parameter for the synchronous version of method `getSubscriptions()`  
See `delegate::GetSubscriptionsDelegate::ErrorInfo`

### 11.178.2.4 SubscribeDelegatePtr

using `cbe::SubscribeManager::SubscribeDelegatePtr = delegate::SubscribeDelegatePtr`  
Pointer to `SubscribeDelegate` that is passed into asynchronous version of method:

- `subscribe()`

### 11.178.2.5 SubscribeException

using `cbe::SubscribeManager::SubscribeException = delegate::SubscribeDelegate::Exception`  
 Pointer to `cbe::delegate::SubscribeDelegate` that is passed into asynchronous version of method `subscribe()` See `delegate::object::SubscribeDelegate::Exception`

### 11.178.2.6 SubscribeError

using `cbe::SubscribeManager::SubscribeError = delegate::SubscribeDelegate::ErrorInfo`  
 Forms the type of the `error` return parameter for the synchronous version of method `subscribe(sharingUserId, sharingUserName, publishId, publishName, password, subscribeName, SubscribeError&) "subscribe()"`  
 See `delegate::SubscribeDelegate::ErrorInfo`

## 11.178.3 Member Function Documentation

### 11.178.3.1 getSubscriptions() [1/3]

```
void cbe::SubscribeManager::getSubscriptions (
 GetSubscriptionsDelegatePtr subscribeDelegate)
```

List your current subscriptions.

Listing is done independently of where in the actual directory tree the files are located.

#### Parameters

|                                |                                                                                                        |
|--------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>subscribeDelegate</code> | Pointer to a <code>delegate::GetSubscriptionsDelegate</code> instance that is implemented by the user. |
|--------------------------------|--------------------------------------------------------------------------------------------------------|

### 11.178.3.2 getSubscriptions() [2/3]

`delegate::GetSubscriptionsDelegate::Success` `cbe::SubscribeManager::getSubscriptions ( )`  
 Synchronous [exception] **Synchronous** version of `getSubscriptions(GetSubscriptionsDelegatePtr)` , and **throws an exception**, `GetSubscriptionsException`, in case of a failed call.  
 See `getSubscriptions(GetSubscriptionsDelegatePtr)`

#### Returns

Information about the `getSubscriptions` object — if the call was successful.  
 See `cbe::delegate::GetSubscriptionsDelegate::Success`

#### Exceptions

|                                        |  |
|----------------------------------------|--|
| <code>GetSubscriptionsException</code> |  |
|----------------------------------------|--|

### 11.178.3.3 getSubscriptions() [3/3]

```
cbe::util::Optional<delegate::GetSubscriptionsDelegate::Success> cbe::SubscribeManager::get←
Subscriptions (
 GetSubscriptionsError & error)
```

Synchronous [non-throwing] **Synchronous** version of `getSubscriptions(GetSubscriptionsDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call.

See `getSubscriptions(GetSubscriptionsDelegatePtr)`

## Parameters

|     |       |                                                                                                                                                                                                                                          |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">GetSubscriptionsError</a> object passed in will be populated with the error information. |
|-----|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

## 11.178.3.4 subscribe() [1/5]

```
void cbe::SubscribeManager::subscribe (
 cbe::UserId sharingUserId,
 std::string sharingUserName,
 cbe::PublishId publishId,
 std::string publishName,
 std::string password,
 std::string subscribeName,
 SubscribeDelegatePtr subscribeDelegate)
```

Subscribes to a [Publish](#) using all parameters.

Subscribe to a [Publish](#) that was issued by some other user (or yourself).

A [Publish](#) must not require a password, if required it should be provided by the publisher.

The retrieved subscription may be revoked with [cbe::Object::unsubscribe\(\)](#) or [cbe::Container::unsubscribe\(\)](#).

Subscribes to a publish shared by some other user (or yourself).

## Parameters

|                          |                                                                                        |
|--------------------------|----------------------------------------------------------------------------------------|
| <i>sharingUserId</i>     | User id of the owner of the publish or 0 if sharingUserName should be used instead     |
| <i>sharingUserName</i>   | User name of the owner of the publish or empty if sharingUserId should be used instead |
| <i>publishId</i>         | <a href="#">Publish</a> id or 0 if publishName should be used instead                  |
| <i>publishName</i>       | <a href="#">Publish</a> name or empty if publishId should be used instead              |
| <i>password</i>          | Required password or an empty string if no password                                    |
| <i>subscribeName</i>     | Name of created favorite (usually same as the publish)                                 |
| <i>subscribeDelegate</i> | Gets notified of the result                                                            |

## 11.178.3.5 subscribe() [2/5]

```
delegate::SubscribeDelegate::Success cbe::SubscribeManager::subscribe (
 cbe::UserId sharingUserId,
 std::string sharingUserName,
 cbe::PublishId publishId,
 std::string publishName,
 std::string password,
 std::string subscribeName)
```

Synchronous [exception] **Synchronous** version of `subscribe(cbe::UserId, std::string, cbe::PublishId, std::string, std::string, std::string, SubscribeDelegatePtr)`, and **throws an exception**, [SubscribeException](#), in case of a failed call.

See `subscribe(SubscribeDelegatePtr)`

**Returns**

Information about the subscribe object — if the call was successful.  
See `cbe::delegate::SubscribeDelegate::Success`

**Exceptions**

|                                    |
|------------------------------------|
| <a href="#">SubscribeException</a> |
|------------------------------------|

**11.178.3.6 subscribe() [3/5]**

```
cbe::util::Optional<delegate::SubscribeDelegate::Success> cbe::SubscribeManager::subscribe (
 cbe::UserId sharingUserId,
 std::string sharingUserName,
 cbe::PublishId publishId,
 std::string publishName,
 std::string password,
 std::string subscribeName,
 SubscribeError & error)
```

Synchronous [non-throwing] **Synchronous** version of `subscribe(sharingUserId, sharingUserName, publishId, publishName, password, subscribeName, SubscribeDelegatePtr)` , and **throws no exception** on error, instead the out/return parameter `error` is used to provide the error information in connection with a failed call. See `subscribe(sharingUserId, sharingUserName, publishId, publishName, password, subscribeName, SubscribeDelegatePtr)`

**Parameters**

|     |       |                                                                                                                                                                                                                                   |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| out | error | Return parameter containing the error information in case of a failed call.<br>An empty return value will indicate failure, and the <a href="#">SubscribeError</a> object passed in will be populated with the error information. |
|-----|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

Empty — i.e., **false** — indicates a failed call, and the error information is passed out via the `error` out/return parameter.

**11.178.3.7 subscribe() [4/5]**

```
void cbe::SubscribeManager::subscribe (
 std::string sharingUserName,
 std::string publishName,
 SubscribeDelegatePtr subscribeDelegate) [inline]
```

Subscribes to a [Publish](#) using names.  
Overload of [subscribe\(\)](#)

**Parameters**

|                          |                                                                              |
|--------------------------|------------------------------------------------------------------------------|
| <i>sharingUserName</i>   | User name of the owner of the publish                                        |
| <i>publishName</i>       | <a href="#">Publish</a> name that will also be the name of created favorite. |
| <i>subscribeDelegate</i> | Gets notified of the result                                                  |

**11.178.3.8 subscribe()** [5/5]

```
void cbe::SubscribeManager::subscribe (
 cbe::UserId sharingUserId,
 cbe::PublishId publishId,
 std::string subscribeName,
 SubscribeDelegatePtr subscribeDelegate) [inline]
```

Subscribes to a [Publish](#) using ids.

Overload of [subscribe\(\)](#)

**Parameters**

|                          |                                                         |
|--------------------------|---------------------------------------------------------|
| <i>sharingUserId</i>     | User id of the owner of the publish                     |
| <i>publishId</i>         | publish id (publish id)                                 |
| <i>subscribeName</i>     | Name of created subscribe (usually same as the publish) |
| <i>subscribeDelegate</i> | Gets notified of the result                             |

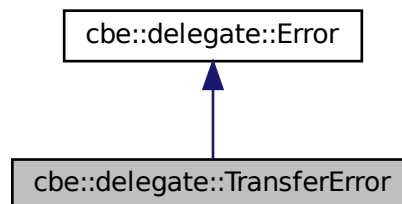
The documentation for this class was generated from the following file:

- include/cbe/SubscribeManager.h

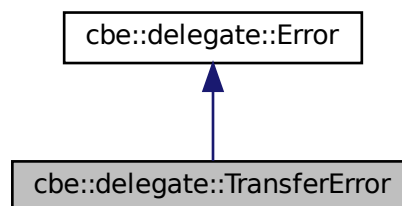
**11.179 cbe::delegate::TransferError Class Reference**

```
#include <TransferError.h>
```

Inheritance diagram for cbe::delegate::TransferError:



Collaboration diagram for cbe::delegate::TransferError:



## Public Member Functions

- **TransferError** ([Error](#) &&error, std::string name, [cbe::ObjectId](#) objectId, [cbe::ContainerId](#) parentId)

## Public Attributes

- std::string **name** {}
- [cbe::ObjectId](#) **objectId** {}
- [cbe::ContainerId](#) **parentId** {}

## Friends

- std::ostream & **operator**<< (std::ostream &os, const [TransferError](#) &transferError)

### 11.179.1 Detailed Description

Contains error information delivered from CloudBackend SDK in connection with a failed transfer, i.e., upload/download.

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

- include/cbe/delegate/TransferError.h

## 11.180 cbe::delegate::UnBanDelegate Class Reference

```
#include <UnBanDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Member::unBan\(\)](#) if the request fails.*

## Public Types

- using **Success** = [UnBanSuccess](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onUnBanSuccess](#) (std::string &&memberName, [cbe::MemberId](#) memberId)=0
- virtual void [onUnBanError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.180.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Member::unBan](#)

### 11.180.2 Member Function Documentation

#### 11.180.2.1 onUnBanSuccess()

```
virtual void cbe::delegate::UnBanDelegate::onUnBanSuccess (
 std::string && memberName,
 cbe::MemberId memberId) [pure virtual]
```

Called upon successful unBan.

## Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <i>memberName</i> | Name of the member being unbanned. |
| <i>memberId</i>   | Id of the member being unbanned.   |

## 11.180.2.2 onUnBanError()

```
virtual void cbe::delegate::UnBanDelegate::onUnBanError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/UnBanDelegate.h

## 11.181 cbe::delegate::UnBanSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::UnBanDelegate::onUnBanSuccess](#).

```
#include <UnBanDelegate.h>
```

## Public Member Functions

- **UnBanSuccess** ([cbe::DefaultCtor](#))
- **UnBanSuccess** (std::string &&memberName, [cbe::MemberId](#) memberId)

## Public Attributes

- std::string **memberName** {}
- [cbe::MemberId](#) **memberId** {}

## 11.181.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::UnBanDelegate::onUnBanSuccess](#).

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

- include/cbe/delegate/UnBanDelegate.h

## 11.182 cbe::delegate::UnPublishDelegate Class Reference

```
#include <UnPublishDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by*

## Public Types

- using **Success** = [UnPublishSuccess](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onUnPublishSuccess](#) ([cbe::PublishId](#) publishId, [cbe::ItemId](#) itemId)=0
- virtual void [onUnPublishError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.182.1 Detailed Description

Delegate class for the asynchronous version of methods:

- [cbe::Container::unPublish](#)
- [cbe::Object::unPublish](#)

### 11.182.2 Member Function Documentation

#### 11.182.2.1 onUnPublishSuccess()

```
virtual void cbe::delegate::UnPublishDelegate::onUnPublishSuccess (
 cbe::PublishId publishId,
 cbe::ItemId itemId) [pure virtual]
```

Called upon successful unPublish.

##### Parameters

|                  |                                                                                                            |
|------------------|------------------------------------------------------------------------------------------------------------|
| <i>publishId</i> | Id of the publictoion being published.                                                                     |
| <i>itemId</i>    | Id of the item - <a href="#">cbe::Object</a> or <a href="#">cbe::Container</a> - that has been unPublished |

#### 11.182.2.2 onUnPublishError()

```
virtual void cbe::delegate::UnPublishDelegate::onUnPublishError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error was encountered.

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

- include/cbe/delegate/UnPublishDelegate.h

## 11.183 cbe::delegate::UnPublishSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::UnPublishDelegate::onUnPublishError](#).

```
#include <UnPublishDelegate.h>
```

### Public Member Functions

- **UnPublishSuccess** ([cbe::DefaultCtor](#))
- **UnPublishSuccess** ([cbe::PublishId](#) publishId, [cbe::ItemId](#) itemId)
- [operator bool](#) () const

*Checks if current instance is valid.*

### Public Attributes

- [cbe::PublishId](#) publishId {}
- [cbe::ItemId](#) itemId {}

#### 11.183.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::UnPublishDelegate::onUnPublishError](#).

## 11.183.2 Member Function Documentation

### 11.183.2.1 operator bool()

`cbe::delegate::UnPublishSuccess::operator bool ( ) const [explicit]`  
Checks if current instance is valid.

#### Returns

`true`: is valid

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

- `include/cbe/delegate/UnPublishDelegate.h`

## 11.184 cbe::delegate::UnShareDelegate Class Reference

```
#include <UnShareDelegate.h>
```

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by*

### Public Types

- using **Success** = [UnShareSuccess](#)
- using **Error** = [delegate::Error](#)

### Public Member Functions

- virtual void [onUnShareSuccess](#) (std::string &&message)=0
- virtual void [onUnShareError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.184.1 Detailed Description

Delegate class for the asynchronous version of methods:

- [cbe::Container::unShare\(\)](#)
- [cbe::Object::unShare\(\)](#)

## 11.184.2 Member Function Documentation

### 11.184.2.1 onUnShareSuccess()

```
virtual void cbe::delegate::UnShareDelegate::onUnShareSuccess (
 std::string && message) [pure virtual]
```

Called upon successful UnShare.

#### Parameters

|                |                         |
|----------------|-------------------------|
| <i>message</i> | Message of the Unshare. |
|----------------|-------------------------|

### 11.184.2.2 onUnShareError()

```
virtual void cbe::delegate::UnShareDelegate::onUnShareError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/UnShareDelegate.h

## 11.185 cbe::delegate::UnShareSuccess Class Reference

Convenience type that bundles the parameter passed to method [cbe::delegate::ShareDelegate::onShareSuccess](#).  
`#include <UnShareDelegate.h>`

### Public Member Functions

- **UnShareSuccess** ([cbe::DefaultCtor](#))
- **UnShareSuccess** (std::string &&message)
- [operator bool](#) () const

*Checks if current instance is valid.*

### Public Attributes

- std::string **message** {}

### 11.185.1 Detailed Description

Convenience type that bundles the parameter passed to method [cbe::delegate::ShareDelegate::onShareSuccess](#).

### 11.185.2 Member Function Documentation

#### 11.185.2.1 operator bool()

```
cbe::delegate::UnShareSuccess::operator bool () const [explicit]
```

Checks if current instance is valid.

#### Returns

`true`: is valid

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

- include/cbe/delegate/UnShareDelegate.h

## 11.186 cbe::delegate::UnSubscribeDelegate Class Reference

`#include <UnSubscribeDelegate.h>`

### Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by*

## Public Types

- using **Success** = [UnSubscribeSuccess](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onUnSubscribeSuccess](#) ([cbe::PublishId](#) publishId, [cbe::ItemId](#) itemId)=0
- virtual void [onUnSubscribeError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.186.1 Detailed Description

Delegate class for the asynchronous version of methods:

- [cbe::Container::unSubscribe](#)
- [cbe::Object::unSubscribe](#)

### 11.186.2 Member Function Documentation

#### 11.186.2.1 onUnSubscribeSuccess()

```
virtual void cbe::delegate::UnSubscribeDelegate::onUnSubscribeSuccess (
 cbe::PublishId publishId,
 cbe::ItemId itemId) [pure virtual]
```

Called upon successful UnSubscribe.

##### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <i>publishId</i> | The PublishId of the UnSubscribe.              |
| <i>itemId</i>    | Instance of ItemId that is being UnSubscribed. |

#### 11.186.2.2 onUnSubscribeError()

```
virtual void cbe::delegate::UnSubscribeDelegate::onUnSubscribeError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/UnSubscribeDelegate.h

## 11.187 cbe::delegate::UnSubscribeSuccess Class Reference

Convenience type that bundles all parameters passed to method [cbe::delegate::UnSubscribeDelegate::onUnSubscribeSuccess](#).  
 #include <UnSubscribeDelegate.h>

## Public Member Functions

- **UnSubscribeSuccess** ([cbe::PublishId](#) publishId, [cbe::ItemId](#) itemId)
  - [operator bool](#) () const
- Checks if current instance is valid.*

## Public Attributes

- [cbe::PublishId](#) publishId {}
- [cbe::ItemId](#) itemId {}

### 11.187.1 Detailed Description

Convenience type that bundles all parameters passed to method [cbe::delegate::UnSubscribeDelegate::onUnSubscribeSuccess](#).

### 11.187.2 Member Function Documentation

#### 11.187.2.1 operator bool()

```
cbe::delegate::UnSubscribeSuccess::operator bool () const [explicit]
```

Checks if current instance is valid.

Returns

true: is valid

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

- include/cbe/delegate/UnSubscribeDelegate.h

## 11.188 cbe::delegate::UpdateKeyValuesDelegate Class Reference

```
#include <UpdateKeyValuesDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)  
*exception thrown by [cbe::Object::updateKeyValues\(\)](#) if the request fails.*

## Public Types

- using **Success** = [cbe::Object](#)
- using **Error** = [delegate::Error](#)

## Public Member Functions

- virtual void [onUpdateKeyValuesSuccess](#) ([cbe::Object](#) &&object)=0
- virtual void [onUpdateKeyValuesError](#) ([Error](#) &&error, [cbe::util::Context](#) &&context)=0

### 11.188.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Object::updateKeyValues\(\)](#)

### 11.188.2 Member Function Documentation

#### 11.188.2.1 onUpdateKeyValuesSuccess()

```
virtual void cbe::delegate::UpdateKeyValuesDelegate::onUpdateKeyValuesSuccess (
 cbe::Object && object) [pure virtual]
```

Called upon successful UpdateKeyValues.

## Parameters

|               |                                                            |
|---------------|------------------------------------------------------------|
| <i>object</i> | Instance of object that is getting its Key/Values updated. |
|---------------|------------------------------------------------------------|

## 11.188.2.2 onUpdateKeyValuesError()

```
virtual void cbe::delegate::UpdateKeyValuesDelegate::onUpdateKeyValuesError (
 Error && error,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

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

- include/cbe/delegate/UpdateKeyValuesDelegate.h

## 11.189 cbe::delegate::UploadDelegate Class Reference

```
#include <UploadDelegate.h>
```

## Classes

- struct [ErrorInfo](#)
- struct [Exception](#)

*exception thrown by [cbe::Container::upload\(const std::string&,const std::string&\)](#) and its overloads if the request fails.*

## Public Types

- using **Success** = [cbe::Object](#)
- using **Error** = [TransferError](#)

## Public Member Functions

- virtual void [onUploadSuccess](#) ([cbe::Object](#) &&object)=0
- virtual void [onUploadError](#) ([cbe::delegate::TransferError](#) &&transferError, [cbe::util::Context](#) &&context)=0
- virtual void [onChunkSent](#) ([cbe::Object](#) &&object, std::uint64\_t sent, std::uint64\_t total)

## 11.189.1 Detailed Description

Delegate class for the asynchronous version of method:

- [cbe::Container::upload\(const std::string&,UploadDelegatePtr\)](#)  
See [Async example of creation of a cbe::Object by using Container::upload\(\)](#)
- [cbe::Container::upload\(const std::string&,const std::string&,UploadDelegatePtr\)](#)
- [cbe::Object::uploadStream\(const std::string&,cbe::StreamId,UploadDelegatePtr\)](#)

## 11.189.2 Member Function Documentation

## 11.189.2.1 onUploadSuccess()

```
virtual void cbe::delegate::UploadDelegate::onUploadSuccess (
 cbe::Object && object) [pure virtual]
```

Called upon successful Upload.

## Parameters

|               |                                            |
|---------------|--------------------------------------------|
| <i>object</i> | Instance of object that is being uploaded. |
|---------------|--------------------------------------------|

**11.189.2.2 onUploadError()**

```
virtual void cbe::delegate::UploadDelegate::onUploadError (
 cbe::delegate::TransferError && transferError,
 cbe::util::Context && context) [pure virtual]
```

Called if an error is encountered.

**11.189.2.3 onChunkSent()**

```
virtual void cbe::delegate::UploadDelegate::onChunkSent (
 cbe::Object && object,
 std::uint64_t sent,
 std::uint64_t total) [virtual]
```

Called when a chunk of data has been sent.

## Parameters

|               |                                                        |
|---------------|--------------------------------------------------------|
| <i>object</i> | <a href="#">Object</a> associated with current upload. |
| <i>sent</i>   | Size, in number of bytes, of the sent chunk.           |
| <i>total</i>  | Total number of bytes sent so far.                     |

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

- include/cbe/delegate/UploadDelegate.h

# Index

- Account
  - cbe::Account, [58](#)
- account
  - cbe::CloudBackend, [81](#)
- AccountStatus
  - cbe, [43](#)
- AclDelegatePtr
  - cbe::delegate, [48](#)
- AclGroupId
  - cbe, [42](#)
- AclMap
  - cbe, [42](#)
- aclMap
  - cbe::Item, [285](#)
- AclScope
  - cbe, [43](#)
- aclTag
  - cbe::Item, [283](#)
- addListener
  - cbe::CloudBackend, [73](#)
- addRoleMember
  - cbe::Role, [374](#), [375](#)
- AddRoleMemberDelegatePtr
  - cbe::delegate, [49](#)
  - cbe::Role, [372](#)
- AddRoleMemberError
  - cbe::Role, [372](#)
- AddRoleMemberException
  - cbe::Role, [372](#)
- ban
  - cbe::Member, [301](#), [302](#)
- BanDelegatePtr
  - cbe::delegate, [49](#)
  - cbe::Member, [299](#)
- BanError
  - cbe::Member, [300](#)
- BanException
  - cbe::Member, [299](#)
- castContainer
  - cbe::CloudBackend, [81](#)
- castObject
  - cbe::CloudBackend, [81](#)
- cbe, [39](#)
  - AccountStatus, [43](#)
  - AclGroupId, [42](#)
  - AclMap, [42](#)
  - AclScope, [43](#)
  - Date, [42](#)
  - DefaultCtor, [43](#)
  - ErrorCode, [42](#)
  - FilterOrder, [44](#)
  - ItemType, [44](#)
  - KeyValues, [42](#)
  - MemberBanInfo, [43](#)
  - ObjectType, [44](#)
  - Permissions, [44](#)
  - PublishAccess, [45](#)
  - PublishVisibility, [45](#)
  - Visibility, [46](#)
- cbe::Account, [57](#)
  - Account, [58](#)
  - client, [58](#)
  - databases, [59](#)
  - firstName, [58](#)
  - libContainerId, [59](#)
  - operator bool, [59](#)
  - password, [58](#)
  - rootContainer, [58](#)
  - rootDatabase, [59](#)
  - source, [58](#)
  - tenantContainerId, [58](#)
  - username, [58](#)
- cbe::CloudBackend, [64](#)
  - account, [81](#)
  - addListener, [73](#)
  - castContainer, [81](#)
  - castObject, [81](#)
  - clearCache, [81](#)
  - CloudBackend, [69](#)
  - CloudBackendListenerDelegatePtr, [68](#)
  - createAccount, [72](#), [73](#)
  - CreateAccountDelegatePtr, [67](#)
  - CreateAccountError, [67](#)
  - CreateAccountException, [67](#)
  - groupManager, [81](#)
  - logIn, [69](#), [71](#)
  - LogInDelegatePtr, [67](#)
  - LogInError, [67](#)
  - LogInException, [67](#)
  - operator bool, [82](#)
  - publishManager, [82](#)
  - query, [74–77](#)
  - QueryDelegatePtr, [68](#)
  - QueryError, [68](#)
  - QueryException, [68](#)
  - QueryJoinDelegatePtr, [68](#)
  - QueryJoinError, [68](#)

- queryWithPath, 78
- removeListener, 74
- search, 78–80
- SearchError, 69
- SearchException, 69
- shareManager, 82
- subscribeManager, 82
- terminate, 82
- version, 81
- cbe::Container, 85
  - createContainer, 95, 96
  - CreateContainerDelegatePtr, 90
  - CreateContainerError, 91
  - CreateContainerException, 90
  - createObject, 99, 100
  - CreateObjectDelegatePtr, 91
  - CreateObjectError, 92
  - CreateObjectException, 92
  - getAcl, 113, 114
  - GetAclDelegatePtr, 94
  - GetAclError, 94
  - GetAclException, 94
  - move, 96, 97
  - MoveDelegatePtr, 91
  - MoveError, 91
  - MoveException, 91
  - publish, 116, 117
  - PublishDelegatePtr, 95
  - PublishError, 95
  - PublishException, 95
  - query, 107–109
  - QueryDelegatePtr, 92
  - QueryError, 93
  - QueryException, 93
  - QueryJoinDelegatePtr, 93
  - QueryJoinError, 93
  - queryWithPath, 109, 110
  - queryWithPathError, 93
  - queryWithPathException, 93
  - remove, 98, 99
  - RemoveDelegatePtr, 91
  - RemoveError, 91
  - RemoveException, 91
  - rename, 97, 98
  - RenameDelegatePtr, 91
  - RenameError, 91
  - RenameException, 91
  - search, 110–112
  - SearchError, 93
  - SearchException, 93
  - setAcl, 112, 113
  - SetAclDelegatePtr, 93
  - SetAclError, 94
  - SetAclException, 94
  - share, 114, 115
  - ShareDelegatePtr, 94
  - ShareError, 94
  - ShareException, 94
  - unPublish, 118
  - UnPublishDelegatePtr, 95
  - UnPublishError, 95
  - UnPublishException, 95
  - unShare, 115, 116
  - UnShareDelegatePtr, 94
  - UnShareError, 94
  - UnShareException, 94
  - unsubscribe, 119
  - UnSubscribeDelegatePtr, 95
  - UnSubscribeError, 95
  - UnSubscribeException, 95
  - upload, 101–107
  - UploadDelegatePtr, 92
  - UploadError, 92
  - UploadException, 92
- cbe::Database, 125
  - containerId, 126
  - created, 126
  - databaseId, 126
  - name, 125
  - ownerId, 126
  - readLocked, 126
  - rootContainer, 127
  - writeLocked, 126
- cbe::delegate, 46
  - AclDelegatePtr, 48
  - AddRoleMemberDelegatePtr, 49
  - BanDelegatePtr, 49
  - CloudBackendListenerDelegatePtr, 49
  - CreateAccountDelegatePtr, 49
  - CreateContainerDelegatePtr, 49
  - CreateGroupDelegatePtr, 49
  - CreateObjectDelegatePtr, 49
  - CreateRoleDelegatePtr, 49
  - DownloadBinaryDelegatePtr, 50
  - DownloadDelegatePtr, 50
  - GetStreamsDelegatePtr, 50
  - GetSubscriptionsDelegatePtr, 50
  - ICloudBackendListenerPtr, 50
  - JoinDelegatePtr, 50
  - KickDelegatePtr, 50
  - LeaveDelegatePtr, 50
  - ListGroupsDelegatePtr, 51
  - ListMembersDelegatePtr, 51
  - ListRolesDelegatePtr, 51
  - ListSharesDelegatePtr, 51
  - LoginDelegatePtr, 51
  - ProgressEventFn, 51
  - PublishDelegatePtr, 51
  - QueryDelegatePtr, 51
  - QueryJoinDelegatePtr, 52
  - RemoveRoleDelegatePtr, 52
  - RemoveRoleMemberDelegatePtr, 52
  - SearchGroupsDelegatePtr, 52
  - ShareDelegatePtr, 52
  - SubscribeDelegatePtr, 52
  - UnBanDelegatePtr, 52

- UnPublishDelegatePtr, 53
- UnShareDelegatePtr, 53
- UnSubscribeDelegatePtr, 53
- UpdateKeyValuesDelegatePtr, 53
- UploadDelegatePtr, 53
- cbe::delegate::AclDelegate, 59
  - onAclError, 60
  - onAclSuccess, 60
- cbe::delegate::AclDelegate::ErrorInfo, 135
- cbe::delegate::AclDelegate::Exception, 179
- cbe::delegate::AddRoleMemberDelegate, 60
  - onAddRoleMemberError, 61
  - onAddRoleMemberSuccess, 61
- cbe::delegate::AddRoleMemberDelegate::ErrorInfo, 137
- cbe::delegate::AddRoleMemberDelegate::Exception, 180
- cbe::delegate::BanDelegate, 62
  - onBanError, 63
  - onBanSuccess, 62
- cbe::delegate::BanDelegate::ErrorInfo, 138
- cbe::delegate::BanDelegate::Exception, 182
- cbe::delegate::BanSuccess, 63
- cbe::delegate::ChunkTransferred, 63
- cbe::delegate::CloudBackendListenerDelegate, 83
  - onRemoteContainerAdded, 84
  - onRemoteContainerMoved, 85
  - onRemoteContainerRemoved, 85
  - onRemoteContainerRenamed, 85
  - onRemoteObjectAdded, 83
  - onRemoteObjectMoved, 84
  - onRemoteObjectRemoved, 84
  - onRemoteObjectRenamed, 84
- cbe::delegate::container, 53
  - MoveDelegatePtr, 54
  - RemoveDelegatePtr, 54
  - RenameDelegatePtr, 54
- cbe::delegate::container::MoveDelegate, 304
  - onMoveError, 304
  - onMoveSuccess, 304
- cbe::delegate::container::MoveDelegate::ErrorInfo, 139
- cbe::delegate::container::MoveDelegate::Exception, 183
- cbe::delegate::container::RemoveDelegate, 361
  - onRemoveError, 361
  - onRemoveSuccess, 361
- cbe::delegate::container::RemoveDelegate::ErrorInfo, 140
- cbe::delegate::container::RemoveDelegate::Exception, 185
- cbe::delegate::container::RemoveSuccess, 365
- cbe::delegate::container::RenameDelegate, 366
  - onRenameError, 367
  - onRenameSuccess, 367
- cbe::delegate::container::RenameDelegate::ErrorInfo, 141
- cbe::delegate::container::RenameDelegate::Exception, 186
- cbe::delegate::CreateAccountDelegate, 120
  - onCreateAccountError, 121
  - onCreateAccountSuccess, 121
- cbe::delegate::CreateAccountDelegate::ErrorInfo, 142
- cbe::delegate::CreateAccountDelegate::Exception, 188
- cbe::delegate::CreateContainerDelegate, 121
  - onCreateContainerError, 122
  - onCreateContainerSuccess, 122
- cbe::delegate::CreateContainerDelegate::ErrorInfo, 143
- cbe::delegate::CreateContainerDelegate::Exception, 189
- cbe::delegate::CreateGroupDelegate, 122
  - onCreateGroupError, 123
  - onCreateGroupSuccess, 122
- cbe::delegate::CreateGroupDelegate::ErrorInfo, 144
- cbe::delegate::CreateGroupDelegate::Exception, 191
- cbe::delegate::CreateObjectDelegate, 123
  - onCreateObjectError, 124
  - onCreateObjectSuccess, 123
- cbe::delegate::CreateObjectDelegate::ErrorInfo, 145
- cbe::delegate::CreateObjectDelegate::Exception, 192
- cbe::delegate::CreateRoleDelegate, 124
  - onCreateRoleError, 124
  - onCreateRoleSuccess, 124
- cbe::delegate::CreateRoleDelegate::ErrorInfo, 146
- cbe::delegate::CreateRoleDelegate::Exception, 194
- cbe::delegate::DownloadBinaryDelegate, 127
  - onChunkReceived, 128
  - onDownloadBinaryError, 128
  - onDownloadBinarySuccess, 127
- cbe::delegate::DownloadBinaryDelegate::ErrorInfo, 147
- cbe::delegate::DownloadBinaryDelegate::Exception, 195
- cbe::delegate::DownloadBinarySuccess, 128
  - operator bool, 129
- cbe::delegate::DownloadDelegate, 129
  - onChunkReceived, 130
  - onDownloadError, 130
  - onDownloadSuccess, 130
- cbe::delegate::DownloadDelegate::ErrorInfo, 148
- cbe::delegate::DownloadDelegate::Exception, 197
- cbe::delegate::DownloadSuccess, 131
  - operator bool, 131
- cbe::delegate::Error, 132
  - errorCode, 133
  - message, 133
  - operator bool, 132
  - operator<<, 133
  - reason, 133
- cbe::delegate::GetStreamsDelegate, 252
  - onGetStreamsError, 253
  - onGetStreamsSuccess, 253
- cbe::delegate::GetStreamsDelegate::ErrorInfo, 149
- cbe::delegate::GetStreamsDelegate::Exception, 198
- cbe::delegate::GetSubscriptionsDelegate, 253
  - onGetSubscriptionsError, 254
  - onGetSubscriptionsSuccess, 254
- cbe::delegate::GetSubscriptionsDelegate::ErrorInfo, 150

cbe::delegate::GetSubscriptionsDelegate::Exception, 200  
 cbe::delegate::group, 54  
     JoinDelegatePtr, 55  
     RemoveDelegatePtr, 55  
     RenameDelegatePtr, 55  
 cbe::delegate::group::JoinDelegate, 286  
     onJoinError, 287  
     onJoinSuccess, 287  
 cbe::delegate::group::JoinDelegate::ErrorInfo, 151  
 cbe::delegate::group::JoinDelegate::Exception, 201  
 cbe::delegate::group::RemoveDelegate, 362  
     onRemoveError, 362  
     onRemoveSuccess, 362  
 cbe::delegate::group::RemoveDelegate::ErrorInfo, 152  
 cbe::delegate::group::RemoveDelegate::Exception, 203  
 cbe::delegate::group::RemoveSuccess, 365  
 cbe::delegate::group::RenameDelegate, 367  
     onRenameError, 368  
     onRenameSuccess, 368  
 cbe::delegate::group::RenameDelegate::ErrorInfo, 153  
 cbe::delegate::group::RenameDelegate::Exception, 204  
 cbe::delegate::group::RenameSuccess, 369  
     operator bool, 370  
 cbe::delegate::ICloudBackendListener, 279  
     onRemoteContainerAdded, 281  
     onRemoteContainerMoved, 281  
     onRemoteContainerRemoved, 281  
     onRemoteContainerRenamed, 281  
     onRemoteObjectAdded, 280  
     onRemoteObjectMoved, 280  
     onRemoteObjectRemoved, 280  
     onRemoteObjectRenamed, 280  
 cbe::delegate::ItemError, 286  
     onItemError, 286  
 cbe::delegate::ItemError::Error, 134  
 cbe::delegate::JoinDelegate, 287  
     ErrorInfo, 288  
     onJoinError, 288  
     onJoinSuccess, 288  
 cbe::delegate::JoinDelegate::Exception, 206  
 cbe::delegate::JoinError, 288  
     onJoinError, 289  
 cbe::delegate::JoinError::Error, 134  
 cbe::delegate::KickDelegate, 289  
     onKickError, 290  
     onKickSuccess, 289  
 cbe::delegate::KickDelegate::ErrorInfo, 154  
 cbe::delegate::KickDelegate::Exception, 207  
 cbe::delegate::KickSuccess, 290  
 cbe::delegate::LeaveDelegate, 290  
     onLeaveError, 291  
     onLeaveSuccess, 291  
 cbe::delegate::LeaveDelegate::ErrorInfo, 155  
 cbe::delegate::LeaveDelegate::Exception, 209  
 cbe::delegate::LeaveSuccess, 291  
     operator bool, 291  
 cbe::delegate::ListGroupDelegate, 292  
     onListGroupError, 292  
     onListGroupSuccess, 292  
 cbe::delegate::ListGroupDelegate::ErrorInfo, 156  
 cbe::delegate::ListGroupDelegate::Exception, 210  
 cbe::delegate::ListMembersDelegate, 293  
     onListMembersError, 293  
     onListMembersSuccess, 293  
 cbe::delegate::ListMembersDelegate::ErrorInfo, 157  
 cbe::delegate::ListMembersDelegate::Exception, 212  
 cbe::delegate::ListRolesDelegate, 294  
     onListRolesError, 294  
     onListRolesSuccess, 294  
 cbe::delegate::ListRolesDelegate::ErrorInfo, 158  
 cbe::delegate::ListRolesDelegate::Exception, 213  
 cbe::delegate::ListSharesDelegate, 295  
     onListSharesError, 295  
     onListSharesSuccess, 295  
 cbe::delegate::ListSharesDelegate::ErrorInfo, 159  
 cbe::delegate::ListSharesDelegate::Exception, 215  
 cbe::delegate::LoadError, 295  
     onLoadError, 296  
 cbe::delegate::LoadError::Error, 135  
     errorCode, 135  
 cbe::delegate::LogInDelegate, 296  
     onLogInError, 297  
     onLogInSuccess, 297  
     Success, 297  
 cbe::delegate::LogInDelegate::ErrorInfo, 160  
 cbe::delegate::LogInDelegate::Exception, 216  
 cbe::delegate::object, 55  
     MoveDelegatePtr, 56  
     RemoveDelegatePtr, 56  
     RenameDelegatePtr, 56  
 cbe::delegate::object::MoveDelegate, 305  
     onMoveError, 305  
     onMoveSuccess, 305  
 cbe::delegate::object::MoveDelegate::ErrorInfo, 161  
 cbe::delegate::object::MoveDelegate::Exception, 218  
 cbe::delegate::object::RemoveDelegate, 362  
     onRemoveError, 363  
     onRemoveSuccess, 363  
 cbe::delegate::object::RemoveDelegate::ErrorInfo, 162  
 cbe::delegate::object::RemoveDelegate::Exception, 219  
 cbe::delegate::object::RemoveSuccess, 366  
     operator bool, 366  
 cbe::delegate::object::RenameDelegate, 368  
     onRenameError, 369  
     onRenameSuccess, 369  
 cbe::delegate::object::RenameDelegate::ErrorInfo, 163  
 cbe::delegate::object::RenameDelegate::Exception, 221  
 cbe::delegate::PublishDelegate, 342  
     onPublishError, 343  
     onPublishSuccess, 343  
 cbe::delegate::PublishDelegate::ErrorInfo, 164  
 cbe::delegate::PublishDelegate::Exception, 222  
 cbe::delegate::PublishSuccess, 344  
     operator bool, 344  
 cbe::delegate::QueryDelegate, 354

- onQueryError, 355
- onQuerySuccess, 355
- cbe::delegate::QueryDelegate::ErrorInfo, 165
- cbe::delegate::QueryDelegate::Exception, 224
- cbe::delegate::QueryError, 355
- cbe::delegate::QueryJoinDelegate, 356
  - ErrorInfo, 357
  - onQueryJoinError, 357
  - onQueryJoinSuccess, 357
- cbe::delegate::QueryJoinDelegate::Exception, 225
- cbe::delegate::QueryLoaded, 358
  - onQuerySuccess, 358
- cbe::delegate::RemoveRoleDelegate, 363
  - onRemoveRoleError, 364
  - onRemoveRoleSuccess, 364
- cbe::delegate::RemoveRoleDelegate::ErrorInfo, 166
- cbe::delegate::RemoveRoleDelegate::Exception, 227
- cbe::delegate::RemoveRoleMemberDelegate, 364
  - onRemoveRoleMemberError, 365
  - onRemoveRoleMemberSuccess, 365
- cbe::delegate::RemoveRoleMemberDelegate::ErrorInfo, 167
- cbe::delegate::RemoveRoleMemberDelegate::Exception, 228
- cbe::delegate::SearchGroupsDelegate, 376
  - onSearchGroupsError, 377
  - onSearchGroupsSuccess, 377
- cbe::delegate::SearchGroupsDelegate::ErrorInfo, 168
- cbe::delegate::SearchGroupsDelegate::Exception, 230
- cbe::delegate::ShareDelegate, 377
  - onShareError, 378
  - onShareSuccess, 378
- cbe::delegate::ShareDelegate::ErrorInfo, 169
- cbe::delegate::ShareDelegate::Exception, 231
- cbe::delegate::ShareSuccess, 382
  - operator bool, 383
- cbe::delegate::SubscribeDelegate, 385
  - onSubscribeError, 386
  - onSubscribeSuccess, 385
- cbe::delegate::SubscribeDelegate::ErrorInfo, 170
- cbe::delegate::SubscribeDelegate::Exception, 233
- cbe::delegate::TransferError, 391
- cbe::delegate::UnBanDelegate, 392
  - onUnBanError, 393
  - onUnBanSuccess, 392
- cbe::delegate::UnBanDelegate::ErrorInfo, 171
- cbe::delegate::UnBanDelegate::Exception, 234
- cbe::delegate::UnBanSuccess, 393
- cbe::delegate::UnPublishDelegate, 393
  - onUnPublishError, 394
  - onUnPublishSuccess, 394
- cbe::delegate::UnPublishDelegate::ErrorInfo, 172
- cbe::delegate::UnPublishDelegate::Exception, 236
- cbe::delegate::UnPublishSuccess, 394
  - operator bool, 395
- cbe::delegate::UnShareDelegate, 395
  - onUnShareError, 395
  - onUnShareSuccess, 395
- cbe::delegate::UnShareDelegate::ErrorInfo, 173
- cbe::delegate::UnShareDelegate::Exception, 237
- cbe::delegate::UnShareSuccess, 396
  - operator bool, 396
- cbe::delegate::UnSubscribeDelegate, 396
  - onUnSubscribeError, 397
  - onUnSubscribeSuccess, 397
- cbe::delegate::UnSubscribeDelegate::ErrorInfo, 174
- cbe::delegate::UnSubscribeDelegate::Exception, 239
- cbe::delegate::UnSubscribeSuccess, 397
  - operator bool, 398
- cbe::delegate::UpdateKeyValuesDelegate, 398
  - onUpdateKeyValuesError, 399
  - onUpdateKeyValuesSuccess, 398
- cbe::delegate::UpdateKeyValuesDelegate::ErrorInfo, 175
- cbe::delegate::UpdateKeyValuesDelegate::Exception, 240
- cbe::delegate::UploadDelegate, 399
  - onChunkSent, 400
  - onUploadError, 400
  - onUploadSuccess, 399
- cbe::delegate::UploadDelegate::ErrorInfo, 176
- cbe::delegate::UploadDelegate::Exception, 242
- cbe::Filter, 248
  - getDataType, 249
  - getItemOrder, 250
  - getOffset, 249
  - getQuery, 249
  - setAscending, 250
  - setByPassCache, 252
  - setContainerFirst, 251
  - setCount, 251
  - setDataType, 250
  - setDeleted, 251
  - setItemOrder, 252
  - setOffset, 251
  - setQuery, 250
- cbe::Group, 254
  - createGroup, 261, 262
  - CreateGroupDelegatePtr, 257
  - CreateGroupError, 257
  - CreateGroupException, 257
  - createRole, 269, 270
  - CreateRoleDelegatePtr, 259
  - CreateRoleError, 259
  - CreateRoleException, 259
  - getVisibility, 260
  - Group, 260
  - groupContainer, 260
  - id, 260
  - join, 262, 263
  - JoinDelegatePtr, 257
  - joined, 261
  - JoinError, 258
  - JoinException, 257
  - leave, 263, 264
  - LeaveDelegatePtr, 258

- LeaveError, 258
- LeaveException, 258
- listBannedMembers, 267, 268
- ListBannedMembersDelegatePtr, 259
- ListBannedMembersError, 259
- ListBannedMembersException, 259
- listMembers, 266, 267
- ListMembersDelegatePtr, 258
- ListMembersError, 259
- ListMembersException, 259
- listRoles, 268, 269
- ListRolesDelegatePtr, 259
- ListRolesError, 259
- ListRolesException, 259
- name, 260
- operator bool, 271
- parentId, 260
- remove, 264, 265
- RemoveDelegatePtr, 258
- RemoveError, 258
- RemoveException, 258
- removeRole, 270
- RemoveRoleDelegatePtr, 260
- RemoveRoleError, 260
- RemoveRoleException, 260
- rename, 265, 266
- RenameDelegatePtr, 258
- RenameError, 258
- RenameException, 258
- requests, 261
- cbe::GroupFilter, 271
  - getAscending, 272
  - getCount, 272
  - getDeleted, 272
  - getFilter, 272
  - getGroupOrder, 273
  - getOffset, 272
  - getOrder, 272
  - getPublicFirst, 272
  - getQuery, 272
  - operator<<, 273
  - setAscending, 273
  - setCount, 273
  - setDeleted, 273
  - setFilter, 273
  - setOffset, 273
  - setQuery, 273
- cbe::GroupManager, 274
  - getTenantId, 277
  - GroupManager, 275
  - listGroups, 275, 276
  - ListGroupsDelegatePtr, 275
  - ListGroupsError, 275
  - ListGroupsException, 275
  - operator bool, 277
  - searchGroups, 276, 277
  - SearchGroupsDelegatePtr, 275
  - SearchGroupsError, 275
  - SearchGroupsException, 275
- cbe::GroupQueryResult, 278
  - containsGroup, 279
  - filter, 278
  - getGroupsSnapshot, 278
  - GroupsLoaded, 279
  - totalCount, 279
- cbe::Item, 282
  - aclMap, 285
  - aclTag, 283
  - created, 284
  - dataLoaded, 284
  - deleted, 284
  - description, 283
  - driveId, 284
  - getPublished, 285
  - getShareFromUserId, 283
  - getShareIds, 283
  - getSubscribe, 285
  - getUserFromShareId, 283
  - hasPublished, 285
  - hasSubscribe, 285
  - id, 283
  - idLoaded, 284
  - name, 284
  - oldParentId, 284
  - operator bool, 285
  - ownerId, 284
  - parentId, 284
  - path, 284
  - type, 285
  - updated, 284
  - username, 284
- cbe::MatchesID< IdT >, 297
- cbe::Member, 298
  - ban, 301, 302
  - BanDelegatePtr, 299
  - BanError, 300
  - BanException, 299
  - groupId, 303
  - kick, 300, 301
  - KickDelegatePtr, 299
  - KickError, 299
  - KickException, 299
  - Member, 300
  - memberId, 303
  - name, 303
  - operator bool, 303
  - unBan, 302, 303
  - UnBanDelegatePtr, 300
  - UnBanError, 300
  - UnBanException, 300
  - visibility, 303
- cbe::Object, 306
  - download, 317–320
  - DownloadBinaryDelegatePtr, 311
  - DownloadBinaryError, 311
  - DownloadBinaryException, 311

- DownloadDelegatePtr, 310
- DownloadError, 311
- DownloadException, 310
- downloadStream, 320–323
- getAcl, 331
- GetAclDelegatePtr, 312
- GetAclError, 312
- GetAclException, 312
- getMimeType, 330
- getObjectType, 330
- getStreams, 328–330
- GetStreamsDelegatePtr, 312
- GetStreamsError, 312
- GetStreamsException, 312
- move, 314, 315
- MoveDelegatePtr, 309
- MoveError, 310
- MoveException, 310
- publish, 334, 335
- PublishDelegatePtr, 313
- PublishError, 314
- PublishException, 313
- remove, 316, 317
- RemoveDelegatePtr, 310
- RemoveError, 310
- RemoveException, 310
- rename, 315, 316
- RenameDelegatePtr, 310
- RenameError, 310
- RenameException, 310
- setAcl, 331, 332
- SetAclDelegatePtr, 313
- SetAclError, 313
- SetAclException, 313
- share, 332, 333
- ShareDelegatePtr, 313
- ShareError, 313
- ShareException, 313
- Streams, 312
- unPublish, 336
- UnPublishDelegatePtr, 314
- UnPublishError, 314
- UnPublishException, 314
- unShare, 333, 334
- UnShareDelegatePtr, 313
- UnShareError, 313
- UnShareException, 313
- unsubscribe, 337
- UnSubscribeDelegatePtr, 314
- UnSubscribeError, 314
- UnSubscribeException, 314
- updateKeyValues, 327, 328
- UpdateKeyValuesDelegatePtr, 312
- UpdateKeyValuesError, 312
- UpdateKeyValuesException, 312
- UploadDelegatePtr, 311
- UploadError, 311
- UploadException, 311
- uploadStream, 323–326
- cbe::Publish, 339
  - getDescription, 342
  - getPassword, 341
  - getPrivacy, 341
  - getPublishId, 342
  - getSecurity, 341
  - getTitle, 341
  - operator bool, 342
  - Publish, 340
  - setDescription, 341
  - setPassword, 341
  - setPrivacy, 341
  - setSecurity, 340
  - setTitle, 340
- cbe::PublishManager, 343
  - getPublished, 344
  - operator bool, 344
  - PublishManager, 343
- cbe::QueryChain, 345
  - getQueryResult, 347
  - join, 346, 347
- cbe::QueryChain::Exception, 244
- cbe::QueryChainExt, 347
  - join, 348–350
- cbe::QueryChainSync, 351
  - getQueryResult, 353
  - join, 352, 353
  - JoinException, 352
- cbe::QueryResult, 358
  - containersLoaded, 360
  - containsItem, 360
  - filter, 359
  - getItemsSnapshot, 359
  - itemsLoaded, 360
  - objectsLoaded, 360
  - operator bool, 360
  - QueryResult, 359
  - totalCount, 360
- cbe::Request, 370
- cbe::Role, 370
  - addRoleMember, 374, 375
  - AddRoleMemberDelegatePtr, 372
  - AddRoleMemberError, 372
  - AddRoleMemberException, 372
  - groupId, 373
  - id, 373
  - listMembers, 373, 374
  - ListMembersDelegatePtr, 372
  - ListMembersError, 372
  - ListMembersException, 372
  - name, 373
  - operator bool, 376
  - removeRoleMember, 375
  - RemoveRoleMemberDelegatePtr, 372
  - RemoveRoleMemberError, 372
  - RemoveRoleMemberException, 372
  - Role, 373

- cbe::ShareData, 377
- cbe::ShareManager, 378
  - listAvailableShares, 380, 381
  - listMyShares, 381, 382
  - ListMySharesError, 380
  - ListMySharesException, 380
  - ListSharesDelegatePtr, 379
  - ListSharesError, 380
  - ListSharesException, 379
  - operator bool, 382
  - ShareManager, 380
- cbe::Stream, 383
  - length, 383
  - streamId, 383
- cbe::Subscribe, 383
  - getDate, 384
  - getDescription, 384
  - getOwner, 385
  - getPassword, 384
  - getPrivacy, 384
  - getPublishId, 385
  - getSecurity, 384
  - getSubscribed, 385
  - getTitle, 384
  - unsubscribe, 385
- cbe::SubscribeManager, 386
  - getSubscriptions, 388
  - GetSubscriptionsDelegatePtr, 387
  - GetSubscriptionsError, 387
  - GetSubscriptionsException, 387
  - subscribe, 389, 390
  - SubscribeDelegatePtr, 387
  - SubscribeError, 388
  - SubscribeException, 387
- cbe::util, 56
- cbe::util::Context, 120
- cbe::util::ErrorInfo, 177
- cbe::util::ErrorInfoImpl< ErrorT >, 178
- cbe::util::Exception, 244
  - derivedCbeException, 246
- cbe::util::ExceptionImpl< ErrorInfoT >, 246
- cbe::util::Optional< T >, 338
  - operator bool, 339
- cbe::util::Optional< T >::BadAccess, 61
- clearCache
  - cbe::CloudBackend, 81
- client
  - cbe::Account, 58
- CloudBackend
  - cbe::CloudBackend, 69
- CloudBackendListenerDelegatePtr
  - cbe::CloudBackend, 68
  - cbe::delegate, 49
- containerId
  - cbe::Database, 126
- containersLoaded
  - cbe::QueryResult, 360
- containsGroup
  - cbe::GroupQueryResult, 279
- containsItem
  - cbe::QueryResult, 360
- createAccount
  - cbe::CloudBackend, 72, 73
- CreateAccountDelegatePtr
  - cbe::CloudBackend, 67
  - cbe::delegate, 49
- CreateAccountError
  - cbe::CloudBackend, 67
- CreateAccountException
  - cbe::CloudBackend, 67
- createContainer
  - cbe::Container, 95, 96
- CreateContainerDelegatePtr
  - cbe::Container, 90
  - cbe::delegate, 49
- CreateContainerError
  - cbe::Container, 91
- CreateContainerException
  - cbe::Container, 90
- created
  - cbe::Database, 126
  - cbe::Item, 284
- createGroup
  - cbe::Group, 261, 262
- CreateGroupDelegatePtr
  - cbe::delegate, 49
  - cbe::Group, 257
- CreateGroupError
  - cbe::Group, 257
- CreateGroupException
  - cbe::Group, 257
- createObject
  - cbe::Container, 99, 100
- CreateObjectDelegatePtr
  - cbe::Container, 91
  - cbe::delegate, 49
- CreateObjectError
  - cbe::Container, 92
- CreateObjectException
  - cbe::Container, 92
- createRole
  - cbe::Group, 269, 270
- CreateRoleDelegatePtr
  - cbe::delegate, 49
  - cbe::Group, 259
- CreateRoleError
  - cbe::Group, 259
- CreateRoleException
  - cbe::Group, 259
- databaseId
  - cbe::Database, 126
- databases
  - cbe::Account, 59
- dataLoaded
  - cbe::Item, 284
- Date

- cbe, [42](#)
- DefaultCtor
  - cbe, [43](#)
- deleted
  - cbe::Item, [284](#)
- derivedCbeException
  - cbe::util::Exception, [246](#)
- description
  - cbe::Item, [283](#)
- download
  - cbe::Object, [317–320](#)
- DownloadBinaryDelegatePtr
  - cbe::delegate, [50](#)
  - cbe::Object, [311](#)
- DownloadBinaryError
  - cbe::Object, [311](#)
- DownloadBinaryException
  - cbe::Object, [311](#)
- DownloadDelegatePtr
  - cbe::delegate, [50](#)
  - cbe::Object, [310](#)
- DownloadError
  - cbe::Object, [311](#)
- DownloadException
  - cbe::Object, [310](#)
- downloadStream
  - cbe::Object, [320–323](#)
- driveld
  - cbe::Item, [284](#)
- ErrorCode
  - cbe, [42](#)
- errorCode
  - cbe::delegate::Error, [133](#)
  - cbe::delegate::LoadError::Error, [135](#)
- ErrorInfo
  - cbe::delegate::JoinDelegate, [288](#)
  - cbe::delegate::QueryJoinDelegate, [357](#)
- filter
  - cbe::GroupQueryResult, [278](#)
  - cbe::QueryResult, [359](#)
- FilterOrder
  - cbe, [44](#)
- firstName
  - cbe::Account, [58](#)
- getAcl
  - cbe::Container, [113, 114](#)
  - cbe::Object, [331](#)
- GetAclDelegatePtr
  - cbe::Container, [94](#)
  - cbe::Object, [312](#)
- GetAclError
  - cbe::Container, [94](#)
  - cbe::Object, [312](#)
- GetAclException
  - cbe::Container, [94](#)
  - cbe::Object, [312](#)
- getAscending
  - cbe::GroupFilter, [272](#)
- getCount
  - cbe::GroupFilter, [272](#)
- getDataType
  - cbe::Filter, [249](#)
- getDate
  - cbe::Subscribe, [384](#)
- getDeleted
  - cbe::GroupFilter, [272](#)
- getDescription
  - cbe::Publish, [342](#)
  - cbe::Subscribe, [384](#)
- getFilter
  - cbe::GroupFilter, [272](#)
- getGroupOrder
  - cbe::GroupFilter, [273](#)
- getGroupsSnapshot
  - cbe::GroupQueryResult, [278](#)
- getItemOrder
  - cbe::Filter, [250](#)
- getItemsSnapshot
  - cbe::QueryResult, [359](#)
- getMimeType
  - cbe::Object, [330](#)
- getObjectType
  - cbe::Object, [330](#)
- getOffset
  - cbe::Filter, [249](#)
  - cbe::GroupFilter, [272](#)
- getOrder
  - cbe::GroupFilter, [272](#)
- getOwner
  - cbe::Subscribe, [385](#)
- getPassword
  - cbe::Publish, [341](#)
  - cbe::Subscribe, [384](#)
- getPrivacy
  - cbe::Publish, [341](#)
  - cbe::Subscribe, [384](#)
- getPublicFirst
  - cbe::GroupFilter, [272](#)
- getPublished
  - cbe::Item, [285](#)
  - cbe::PublishManager, [344](#)
- getPublishId
  - cbe::Publish, [342](#)
  - cbe::Subscribe, [385](#)
- getQuery
  - cbe::Filter, [249](#)
  - cbe::GroupFilter, [272](#)
- getQueryResult
  - cbe::QueryChain, [347](#)
  - cbe::QueryChainSync, [353](#)
- getSecurity
  - cbe::Publish, [341](#)
  - cbe::Subscribe, [384](#)
- getShareFromUserId

- cbe::Item, 283
- getShareIds
  - cbe::Item, 283
- getStreams
  - cbe::Object, 328–330
- GetStreamsDelegatePtr
  - cbe::delegate, 50
  - cbe::Object, 312
- GetStreamsError
  - cbe::Object, 312
- GetStreamsException
  - cbe::Object, 312
- getSubscribe
  - cbe::Item, 285
- getSubscribed
  - cbe::Subscribe, 385
- getSubscriptions
  - cbe::SubscribeManager, 388
- GetSubscriptionsDelegatePtr
  - cbe::delegate, 50
  - cbe::SubscribeManager, 387
- GetSubscriptionsError
  - cbe::SubscribeManager, 387
- GetSubscriptionsException
  - cbe::SubscribeManager, 387
- getTenantId
  - cbe::GroupManager, 277
- getTitle
  - cbe::Publish, 341
  - cbe::Subscribe, 384
- getUserFromShareId
  - cbe::Item, 283
- getVisibility
  - cbe::Group, 260
- Group
  - cbe::Group, 260
- groupContainer
  - cbe::Group, 260
- groupId
  - cbe::Member, 303
  - cbe::Role, 373
- GroupManager
  - cbe::GroupManager, 275
- groupManager
  - cbe::CloudBackend, 81
- GroupsLoaded
  - cbe::GroupQueryResult, 279
- hasPublished
  - cbe::Item, 285
- hasSubscribe
  - cbe::Item, 285
- ICloudBackendListenerPtr
  - cbe::delegate, 50
- id
  - cbe::Group, 260
  - cbe::Item, 283
  - cbe::Role, 373
- idLoaded
  - cbe::Item, 284
- itemsLoaded
  - cbe::QueryResult, 360
- ItemType
  - cbe, 44
- join
  - cbe::Group, 262, 263
  - cbe::QueryChain, 346, 347
  - cbe::QueryChainExt, 348–350
  - cbe::QueryChainSync, 352, 353
- JoinDelegatePtr
  - cbe::delegate, 50
  - cbe::delegate::group, 55
  - cbe::Group, 257
- joined
  - cbe::Group, 261
- JoinError
  - cbe::Group, 258
- JoinException
  - cbe::Group, 257
  - cbe::QueryChainSync, 352
- KeyValues
  - cbe, 42
- kick
  - cbe::Member, 300, 301
- KickDelegatePtr
  - cbe::delegate, 50
  - cbe::Member, 299
- KickError
  - cbe::Member, 299
- KickException
  - cbe::Member, 299
- leave
  - cbe::Group, 263, 264
- LeaveDelegatePtr
  - cbe::delegate, 50
  - cbe::Group, 258
- LeaveError
  - cbe::Group, 258
- LeaveException
  - cbe::Group, 258
- length
  - cbe::Stream, 383
- libContainerId
  - cbe::Account, 59
- listAvailableShares
  - cbe::ShareManager, 380, 381
- listBannedMembers
  - cbe::Group, 267, 268
- ListBannedMembersDelegatePtr
  - cbe::Group, 259
- ListBannedMembersError
  - cbe::Group, 259
- ListBannedMembersException
  - cbe::Group, 259

- listGroups
  - cbe::GroupManager, 275, 276
- ListGroupsDelegatePtr
  - cbe::delegate, 51
  - cbe::GroupManager, 275
- ListGroupsError
  - cbe::GroupManager, 275
- ListGroupsException
  - cbe::GroupManager, 275
- listMembers
  - cbe::Group, 266, 267
  - cbe::Role, 373, 374
- ListMembersDelegatePtr
  - cbe::delegate, 51
  - cbe::Group, 258
  - cbe::Role, 372
- ListMembersError
  - cbe::Group, 259
  - cbe::Role, 372
- ListMembersException
  - cbe::Group, 259
  - cbe::Role, 372
- listMyShares
  - cbe::ShareManager, 381, 382
- ListMySharesError
  - cbe::ShareManager, 380
- ListMySharesException
  - cbe::ShareManager, 380
- listRoles
  - cbe::Group, 268, 269
- ListRolesDelegatePtr
  - cbe::delegate, 51
  - cbe::Group, 259
- ListRolesError
  - cbe::Group, 259
- ListRolesException
  - cbe::Group, 259
- ListSharesDelegatePtr
  - cbe::delegate, 51
  - cbe::ShareManager, 379
- ListSharesError
  - cbe::ShareManager, 380
- ListSharesException
  - cbe::ShareManager, 379
- logIn
  - cbe::CloudBackend, 69, 71
- LogInDelegatePtr
  - cbe::CloudBackend, 67
  - cbe::delegate, 51
- LogInError
  - cbe::CloudBackend, 67
- LogInException
  - cbe::CloudBackend, 67
- Member
  - cbe::Member, 300
- MemberBanInfo
  - cbe, 43
- memberId
  - cbe::Member, 303
- message
  - cbe::delegate::Error, 133
- move
  - cbe::Container, 96, 97
  - cbe::Object, 314, 315
- MoveDelegatePtr
  - cbe::Container, 91
  - cbe::delegate::container, 54
  - cbe::delegate::object, 56
  - cbe::Object, 309
- MoveError
  - cbe::Container, 91
  - cbe::Object, 310
- MoveException
  - cbe::Container, 91
  - cbe::Object, 310
- name
  - cbe::Database, 125
  - cbe::Group, 260
  - cbe::Item, 284
  - cbe::Member, 303
  - cbe::Role, 373
- objectsLoaded
  - cbe::QueryResult, 360
- ObjectType
  - cbe, 44
- oldParentId
  - cbe::Item, 284
- onAclError
  - cbe::delegate::AclDelegate, 60
- onAclSuccess
  - cbe::delegate::AclDelegate, 60
- onAddRoleMemberError
  - cbe::delegate::AddRoleMemberDelegate, 61
- onAddRoleMemberSuccess
  - cbe::delegate::AddRoleMemberDelegate, 61
- onBanError
  - cbe::delegate::BanDelegate, 63
- onBanSuccess
  - cbe::delegate::BanDelegate, 62
- onChunkReceived
  - cbe::delegate::DownloadBinaryDelegate, 128
  - cbe::delegate::DownloadDelegate, 130
- onChunkSent
  - cbe::delegate::UploadDelegate, 400
- onCreateAccountError
  - cbe::delegate::CreateAccountDelegate, 121
- onCreateAccountSuccess
  - cbe::delegate::CreateAccountDelegate, 121
- onCreateContainerError
  - cbe::delegate::CreateContainerDelegate, 122
- onCreateContainerSuccess
  - cbe::delegate::CreateContainerDelegate, 122
- onCreateGroupError
  - cbe::delegate::CreateGroupDelegate, 123
- onCreateGroupSuccess

- cbe::delegate::CreateGroupDelegate, 122
- onCreateObjectError
  - cbe::delegate::CreateObjectDelegate, 124
- onCreateObjectSuccess
  - cbe::delegate::CreateObjectDelegate, 123
- onCreateRoleError
  - cbe::delegate::CreateRoleDelegate, 124
- onCreateRoleSuccess
  - cbe::delegate::CreateRoleDelegate, 124
- onDownloadBinaryError
  - cbe::delegate::DownloadBinaryDelegate, 128
- onDownloadBinarySuccess
  - cbe::delegate::DownloadBinaryDelegate, 127
- onDownloadError
  - cbe::delegate::DownloadDelegate, 130
- onDownloadSuccess
  - cbe::delegate::DownloadDelegate, 130
- onGetStreamsError
  - cbe::delegate::GetStreamsDelegate, 253
- onGetStreamsSuccess
  - cbe::delegate::GetStreamsDelegate, 253
- onGetSubscriptionsError
  - cbe::delegate::GetSubscriptionsDelegate, 254
- onGetSubscriptionsSuccess
  - cbe::delegate::GetSubscriptionsDelegate, 254
- onItemError
  - cbe::delegate::ItemError, 286
- onJoinError
  - cbe::delegate::group::JoinDelegate, 287
  - cbe::delegate::JoinDelegate, 288
  - cbe::delegate::JoinError, 289
- onJoinSuccess
  - cbe::delegate::group::JoinDelegate, 287
  - cbe::delegate::JoinDelegate, 288
- onKickError
  - cbe::delegate::KickDelegate, 290
- onKickSuccess
  - cbe::delegate::KickDelegate, 289
- onLeaveError
  - cbe::delegate::LeaveDelegate, 291
- onLeaveSuccess
  - cbe::delegate::LeaveDelegate, 291
- onListGroupError
  - cbe::delegate::ListGroupDelegate, 292
- onListGroupSuccess
  - cbe::delegate::ListGroupDelegate, 292
- onListMembersError
  - cbe::delegate::ListMembersDelegate, 293
- onListMembersSuccess
  - cbe::delegate::ListMembersDelegate, 293
- onListRolesError
  - cbe::delegate::ListRolesDelegate, 294
- onListRolesSuccess
  - cbe::delegate::ListRolesDelegate, 294
- onListSharesError
  - cbe::delegate::ListSharesDelegate, 295
- onListSharesSuccess
  - cbe::delegate::ListSharesDelegate, 295
- onLoadError
  - cbe::delegate::LoadError, 296
- onLoginError
  - cbe::delegate::LoginDelegate, 297
- onLoginSuccess
  - cbe::delegate::LoginDelegate, 297
- onMoveError
  - cbe::delegate::container::MoveDelegate, 304
  - cbe::delegate::object::MoveDelegate, 305
- onMoveSuccess
  - cbe::delegate::container::MoveDelegate, 304
  - cbe::delegate::object::MoveDelegate, 305
- onPublishError
  - cbe::delegate::PublishDelegate, 343
- onPublishSuccess
  - cbe::delegate::PublishDelegate, 343
- onQueryError
  - cbe::delegate::QueryDelegate, 355
- onQueryJoinError
  - cbe::delegate::QueryJoinDelegate, 357
- onQueryJoinSuccess
  - cbe::delegate::QueryJoinDelegate, 357
- onQuerySuccess
  - cbe::delegate::QueryDelegate, 355
  - cbe::delegate::QueryLoaded, 358
- onRemoteContainerAdded
  - cbe::delegate::CloudBackendListenerDelegate, 84
  - cbe::delegate::ICloudBackendListener, 281
- onRemoteContainerMoved
  - cbe::delegate::CloudBackendListenerDelegate, 85
  - cbe::delegate::ICloudBackendListener, 281
- onRemoteContainerRemoved
  - cbe::delegate::CloudBackendListenerDelegate, 85
  - cbe::delegate::ICloudBackendListener, 281
- onRemoteContainerRenamed
  - cbe::delegate::CloudBackendListenerDelegate, 85
  - cbe::delegate::ICloudBackendListener, 281
- onRemoteObjectAdded
  - cbe::delegate::CloudBackendListenerDelegate, 83
  - cbe::delegate::ICloudBackendListener, 280
- onRemoteObjectMoved
  - cbe::delegate::CloudBackendListenerDelegate, 84
  - cbe::delegate::ICloudBackendListener, 280
- onRemoteObjectRemoved
  - cbe::delegate::CloudBackendListenerDelegate, 84
  - cbe::delegate::ICloudBackendListener, 280
- onRemoteObjectRenamed
  - cbe::delegate::CloudBackendListenerDelegate, 84
  - cbe::delegate::ICloudBackendListener, 280
- onRemoveError
  - cbe::delegate::container::RemoveDelegate, 361
  - cbe::delegate::group::RemoveDelegate, 362
  - cbe::delegate::object::RemoveDelegate, 363
- onRemoveRoleError
  - cbe::delegate::RemoveRoleDelegate, 364
- onRemoveRoleMemberError
  - cbe::delegate::RemoveRoleMemberDelegate, 365
- onRemoveRoleMemberSuccess

- cbe::delegate::RemoveRoleMemberDelegate, 365
- onRemoveRoleSuccess
  - cbe::delegate::RemoveRoleDelegate, 364
- onRemoveSuccess
  - cbe::delegate::container::RemoveDelegate, 361
  - cbe::delegate::group::RemoveDelegate, 362
  - cbe::delegate::object::RemoveDelegate, 363
- onRenameError
  - cbe::delegate::container::RenameDelegate, 367
  - cbe::delegate::group::RenameDelegate, 368
  - cbe::delegate::object::RenameDelegate, 369
- onRenameSuccess
  - cbe::delegate::container::RenameDelegate, 367
  - cbe::delegate::group::RenameDelegate, 368
  - cbe::delegate::object::RenameDelegate, 369
- onSearchGroupsError
  - cbe::delegate::SearchGroupsDelegate, 377
- onSearchGroupsSuccess
  - cbe::delegate::SearchGroupsDelegate, 377
- onShareError
  - cbe::delegate::ShareDelegate, 378
- onShareSuccess
  - cbe::delegate::ShareDelegate, 378
- onSubscribeError
  - cbe::delegate::SubscribeDelegate, 386
- onSubscribeSuccess
  - cbe::delegate::SubscribeDelegate, 385
- onUnBanError
  - cbe::delegate::UnBanDelegate, 393
- onUnBanSuccess
  - cbe::delegate::UnBanDelegate, 392
- onUnPublishError
  - cbe::delegate::UnPublishDelegate, 394
- onUnPublishSuccess
  - cbe::delegate::UnPublishDelegate, 394
- onUnShareError
  - cbe::delegate::UnShareDelegate, 395
- onUnShareSuccess
  - cbe::delegate::UnShareDelegate, 395
- onUnSubscribeError
  - cbe::delegate::UnSubscribeDelegate, 397
- onUnSubscribeSuccess
  - cbe::delegate::UnSubscribeDelegate, 397
- onUpdateKeyValuesError
  - cbe::delegate::UpdateKeyValuesDelegate, 399
- onUpdateKeyValuesSuccess
  - cbe::delegate::UpdateKeyValuesDelegate, 398
- onUploadError
  - cbe::delegate::UploadDelegate, 400
- onUploadSuccess
  - cbe::delegate::UploadDelegate, 399
- operator bool
  - cbe::Account, 59
  - cbe::CloudBackend, 82
  - cbe::delegate::DownloadBinarySuccess, 129
  - cbe::delegate::DownloadSuccess, 131
  - cbe::delegate::Error, 132
  - cbe::delegate::group::RenameSuccess, 370
  - cbe::delegate::LeaveSuccess, 291
  - cbe::delegate::object::RemoveSuccess, 366
  - cbe::delegate::PublishSuccess, 344
  - cbe::delegate::ShareSuccess, 383
  - cbe::delegate::UnPublishSuccess, 395
  - cbe::delegate::UnShareSuccess, 396
  - cbe::delegate::UnSubscribeSuccess, 398
  - cbe::Group, 271
  - cbe::GroupManager, 277
  - cbe::Item, 285
  - cbe::Member, 303
  - cbe::Publish, 342
  - cbe::PublishManager, 344
  - cbe::QueryResult, 360
  - cbe::Role, 376
  - cbe::ShareManager, 382
  - cbe::util::Optional< T >, 339
- operator<<
  - cbe::delegate::Error, 133
  - cbe::GroupFilter, 273
- ownerId
  - cbe::Database, 126
  - cbe::Item, 284
- parentId
  - cbe::Group, 260
  - cbe::Item, 284
- password
  - cbe::Account, 58
- path
  - cbe::Item, 284
- Permissions
  - cbe, 44
- ProgressEventFn
  - cbe::delegate, 51
- Publish
  - cbe::Publish, 340
- publish
  - cbe::Container, 116, 117
  - cbe::Object, 334, 335
- PublishAccess
  - cbe, 45
- PublishDelegatePtr
  - cbe::Container, 95
  - cbe::delegate, 51
  - cbe::Object, 313
- PublishError
  - cbe::Container, 95
  - cbe::Object, 314
- PublishException
  - cbe::Container, 95
  - cbe::Object, 313
- PublishManager
  - cbe::PublishManager, 343
- publishManager
  - cbe::CloudBackend, 82
- PublishVisibility
  - cbe, 45

- query
  - cbe::CloudBackend, 74–77
  - cbe::Container, 107–109
- QueryDelegatePtr
  - cbe::CloudBackend, 68
  - cbe::Container, 92
  - cbe::delegate, 51
- QueryError
  - cbe::CloudBackend, 68
  - cbe::Container, 93
- QueryException
  - cbe::CloudBackend, 68
  - cbe::Container, 93
- QueryJoinDelegatePtr
  - cbe::CloudBackend, 68
  - cbe::Container, 93
  - cbe::delegate, 52
- QueryJoinError
  - cbe::CloudBackend, 68
  - cbe::Container, 93
- QueryResult
  - cbe::QueryResult, 359
- queryWithPath
  - cbe::CloudBackend, 78
  - cbe::Container, 109, 110
- queryWithPathError
  - cbe::Container, 93
- queryWithPathException
  - cbe::Container, 93
- readLocked
  - cbe::Database, 126
- reason
  - cbe::delegate::Error, 133
- remove
  - cbe::Container, 98, 99
  - cbe::Group, 264, 265
  - cbe::Object, 316, 317
- RemoveDelegatePtr
  - cbe::Container, 91
  - cbe::delegate::container, 54
  - cbe::delegate::group, 55
  - cbe::delegate::object, 56
  - cbe::Group, 258
  - cbe::Object, 310
- RemoveError
  - cbe::Container, 91
  - cbe::Group, 258
  - cbe::Object, 310
- RemoveException
  - cbe::Container, 91
  - cbe::Group, 258
  - cbe::Object, 310
- removeListener
  - cbe::CloudBackend, 74
- removeRole
  - cbe::Group, 270
- RemoveRoleDelegatePtr
  - cbe::delegate, 52
- cbe::Group, 260
- RemoveRoleError
  - cbe::Group, 260
- RemoveRoleException
  - cbe::Group, 260
- removeRoleMember
  - cbe::Role, 375
- RemoveRoleMemberDelegatePtr
  - cbe::delegate, 52
  - cbe::Role, 372
- RemoveRoleMemberError
  - cbe::Role, 372
- RemoveRoleMemberException
  - cbe::Role, 372
- rename
  - cbe::Container, 97, 98
  - cbe::Group, 265, 266
  - cbe::Object, 315, 316
- RenameDelegatePtr
  - cbe::Container, 91
  - cbe::delegate::container, 54
  - cbe::delegate::group, 55
  - cbe::delegate::object, 56
  - cbe::Group, 258
  - cbe::Object, 310
- RenameError
  - cbe::Container, 91
  - cbe::Group, 258
  - cbe::Object, 310
- RenameException
  - cbe::Container, 91
  - cbe::Group, 258
  - cbe::Object, 310
- requests
  - cbe::Group, 261
- Role
  - cbe::Role, 373
- rootContainer
  - cbe::Account, 58
  - cbe::Database, 127
- rootDatabase
  - cbe::Account, 59
- search
  - cbe::CloudBackend, 78–80
  - cbe::Container, 110–112
- SearchError
  - cbe::CloudBackend, 69
  - cbe::Container, 93
- SearchException
  - cbe::CloudBackend, 69
  - cbe::Container, 93
- searchGroups
  - cbe::GroupManager, 276, 277
- SearchGroupsDelegatePtr
  - cbe::delegate, 52
  - cbe::GroupManager, 275
- SearchGroupsError
  - cbe::GroupManager, 275

- SearchGroupsException
  - cbe::GroupManager, 275
- setAcl
  - cbe::Container, 112, 113
  - cbe::Object, 331, 332
- SetAclDelegatePtr
  - cbe::Container, 93
  - cbe::Object, 313
- SetAclError
  - cbe::Container, 94
  - cbe::Object, 313
- SetAclException
  - cbe::Container, 94
  - cbe::Object, 313
- setAscending
  - cbe::Filter, 250
  - cbe::GroupFilter, 273
- setByPassCache
  - cbe::Filter, 252
- setContainerFirst
  - cbe::Filter, 251
- setCount
  - cbe::Filter, 251
  - cbe::GroupFilter, 273
- setDataType
  - cbe::Filter, 250
- setDeleted
  - cbe::Filter, 251
  - cbe::GroupFilter, 273
- setDescription
  - cbe::Publish, 341
- setFilter
  - cbe::GroupFilter, 273
- setItemOrder
  - cbe::Filter, 252
- setOffset
  - cbe::Filter, 251
  - cbe::GroupFilter, 273
- setPassword
  - cbe::Publish, 341
- setPrivacy
  - cbe::Publish, 341
- setQuery
  - cbe::Filter, 250
  - cbe::GroupFilter, 273
- setSecurity
  - cbe::Publish, 340
- setTitle
  - cbe::Publish, 340
- share
  - cbe::Container, 114, 115
  - cbe::Object, 332, 333
- ShareDelegatePtr
  - cbe::Container, 94
  - cbe::delegate, 52
  - cbe::Object, 313
- ShareError
  - cbe::Container, 94
- cbe::Object, 313
- ShareException
  - cbe::Container, 94
  - cbe::Object, 313
- ShareManager
  - cbe::ShareManager, 380
- shareManager
  - cbe::CloudBackend, 82
- source
  - cbe::Account, 58
- streamId
  - cbe::Stream, 383
- Streams
  - cbe::Object, 312
- subscribe
  - cbe::SubscribeManager, 389, 390
- SubscribeDelegatePtr
  - cbe::delegate, 52
  - cbe::SubscribeManager, 387
- SubscribeError
  - cbe::SubscribeManager, 388
- SubscribeException
  - cbe::SubscribeManager, 387
- subscribeManager
  - cbe::CloudBackend, 82
- Success
  - cbe::delegate::LoginDelegate, 297
- tenantContainerId
  - cbe::Account, 58
- terminate
  - cbe::CloudBackend, 82
- totalCount
  - cbe::GroupQueryResult, 279
  - cbe::QueryResult, 360
- type
  - cbe::Item, 285
- unBan
  - cbe::Member, 302, 303
- UnBanDelegatePtr
  - cbe::delegate, 52
  - cbe::Member, 300
- UnBanError
  - cbe::Member, 300
- UnBanException
  - cbe::Member, 300
- unPublish
  - cbe::Container, 118
  - cbe::Object, 336
- UnPublishDelegatePtr
  - cbe::Container, 95
  - cbe::delegate, 53
  - cbe::Object, 314
- UnPublishError
  - cbe::Container, 95
  - cbe::Object, 314
- UnPublishException
  - cbe::Container, 95

- cbe::Object, 314
- unShare
  - cbe::Container, 115, 116
  - cbe::Object, 333, 334
- UnShareDelegatePtr
  - cbe::Container, 94
  - cbe::delegate, 53
  - cbe::Object, 313
- UnShareError
  - cbe::Container, 94
  - cbe::Object, 313
- UnShareException
  - cbe::Container, 94
  - cbe::Object, 313
- unSubscribe
  - cbe::Container, 119
  - cbe::Object, 337
  - cbe::Subscribe, 385
- UnSubscribeDelegatePtr
  - cbe::Container, 95
  - cbe::delegate, 53
  - cbe::Object, 314
- UnSubscribeError
  - cbe::Container, 95
  - cbe::Object, 314
- UnSubscribeException
  - cbe::Container, 95
  - cbe::Object, 314
- updated
  - cbe::Item, 284
- updateKeyValues
  - cbe::Object, 327, 328
- UpdateKeyValuesDelegatePtr
  - cbe::delegate, 53
  - cbe::Object, 312
- UpdateKeyValuesError
  - cbe::Object, 312
- UpdateKeyValuesException
  - cbe::Object, 312
- upload
  - cbe::Container, 101–107
- UploadDelegatePtr
  - cbe::Container, 92
  - cbe::delegate, 53
  - cbe::Object, 311
- UploadError
  - cbe::Container, 92
  - cbe::Object, 311
- UploadException
  - cbe::Container, 92
  - cbe::Object, 311
- uploadStream
  - cbe::Object, 323–326
- username
  - cbe::Account, 58
  - cbe::Item, 284
- version
  - cbe::CloudBackend, 81
- Visibility
  - cbe, 46
- visibility
  - cbe::Member, 303
- writeLocked
  - cbe::Database, 126