InfoReach

Engineered for what's next

InfoReach FIX Engine

FIX C++ API Reference Guide

Version 8.3

Contents

Core Classes

ELT

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELT.h**

Placeholder for FIX fields codes.

```
enum
{
    fieldInvalid            = 0  ,
    fieldAccount            = 1  ,
    fieldAdvID              = 2  ,
    fieldAdvRefID           = 3  ,
    fieldAdvSide            = 4  ,
    fieldAdvTransType       = 5  ,
    fieldAvgPx              = 6  ,
    fieldBeginString        = 8  ,
    fieldCheckSum           = 10 ,
    fieldClOrdID            = 11 ,
    fieldCommission         = 12 ,
    fieldCommType           = 13 ,
    fieldCumQty             = 14 ,
    fieldCurrency           = 15 ,
    fieldExecID             = 17 ,
    fieldExecInst           = 18 ,
    fieldExecRefID          = 19 ,
    fieldExecTransType      = 20 ,
    fieldHandlInst          = 21 ,
    fieldIDSource           = 22 ,
    fieldSecurityIDSource   = fieldIDSource , //was renamed in 4.3
    fieldIOIid              = 23 ,
    fieldIOIOthSvc          = 24 ,
    fieldIOIQltyInd         = 25 ,
    fieldIOIRefID           = 26 ,
    fieldIOIShares          = 27 ,
    fieldIOIQty             = fieldIOIShares , //was renamed in 4.3
    fieldIOITransType       = 28 ,
    fieldLastCapacity       = 29 ,
    fieldLastMkt            = 30 ,
    fieldLastPx             = 31 ,
    fieldLastShares         = 32 ,
    fieldLastQty            = fieldLastShares , //was renamed in 4.3
    fieldMsgSeqNum          = 34 ,
    fieldMsgType            = 35 ,
    fieldOrderID            = 37 ,
    fieldOrderQty           = 38 ,
    fieldOrdStatus          = 39 ,
    fieldOrdType            = 40 ,
```

```
fieldOrigClOrdID        = 41 ,
fieldOrigTime           = 42 ,
fieldPossDupFlag        = 43 ,
fieldPrice              = 44 ,
fieldRefSeqNum          = 45 ,
fieldRelatdSym          = 46 ,
fieldRule80A            = 47 ,
fieldSecurityID         = 48 ,
fieldSenderCompID       = 49 ,
fieldSenderSubID        = 50 ,
fieldSendingDate        = 51 ,
fieldSendingTime        = 52 ,
fieldShares             = 53 ,
fieldQuantity           = fieldShares , //was renamed in 4.3
fieldSide               = 54 ,
fieldSymbol             = 55 ,
fieldTargetCompID       = 56 ,
fieldTargetSubID        = 57 ,
fieldText               = 58 ,
fieldTimeInForce        = 59 ,
fieldTransactTime       = 60 ,
fieldUrgency            = 61 ,
fieldValidUntilTime     = 62 ,
fieldSettlmntTyp        = 63 ,
fieldSettlType          = fieldSettlmntTyp , //was renamed in 4.4
fieldFutSettDate        = 64 ,
fieldSettlDate          = fieldFutSettDate , //was renamed in 4.4
fieldSymbolSfx          = 65 ,
fieldListID             = 66 ,
fieldListSeqNo          = 67 ,
fieldListNoOrds         = 68 ,
fieldTotNoOrders        = fieldListNoOrds , //was renamed in 4.3
fieldListExecInst       = 69 ,
fieldAllocID            = 70 ,
fieldAllocTransType     = 71 ,
fieldRefAllocID         = 72 ,
fieldNoOrders           = 73 ,
fieldAvgPrxPrecision    = 74 ,
fieldAvgPxPrecision     = fieldAvgPrxPrecision , //renamed in 4.4
fieldTradeDate          = 75 ,
fieldExecBroker         = 76 ,
fieldOpenClose          = 77 ,
fieldPositionEffect     = fieldOpenClose , //was renamed in 4.3
fieldNoAllocs           = 78 ,
fieldAllocAccount       = 79 ,
fieldAllocShares        = 80 ,
fieldAllocQty           = fieldAllocShares , //was renamed in 4.3
fieldProcessCode        = 81 ,
fieldNoRpts             = 82 ,
fieldRptSeq             = 83 ,
fieldCxlQty             = 84 ,
fieldNoDvyInst          = 85 ,
```

```
fieldDlvyInst           = 86 ,
fieldAllocStatus        = 87 ,
fieldAllocRejCode       = 88 ,
fieldSignature          = 89 ,
fieldBrokerOfCredit     = 92 ,
fieldPossResend         = 97 ,
fieldStopPx             = 99 ,
fieldExDestination      = 100,
fieldCxlRejReason       = 102,
fieldOrdRejReason       = 103,
fieldIOIQualifier       = 104,
fieldWaveNo             = 105,
fieldIssuer             = 106,
fieldSecurityDesc       = 107,
fieldHeartBtInt         = 108,
fieldClientID           = 109,
fieldMinQty             = 110,
fieldMaxFloor           = 111,
fieldReportToExch       = 113,
fieldLocateReqd         = 114,
fieldOnBehalfOfCompID   = 115,
fieldOnBehalfOfSubID    = 116,
fieldQuoteID            = 117,
fieldNetMoney           = 118,
fieldSettlCurrAmt       = 119,
fieldSettlCurrency      = 120,
fieldForexReq           = 121,
fieldNoExecs            = 124,
fieldCxlType            = 125,
fieldExpireTime         = 126,
// new fields in FIX 4.0 below
fieldDKReason           = 127,
fieldDeliverToCompID    = 128,
fieldDeliverToSubID     = 129,
fieldIOINaturalFlag     = 130,
fieldQuoteReqID         = 131,
fieldBidPx              = 132,
fieldOfferPx            = 133,
fieldBidSize            = 134,
fieldOfferSize          = 135,
fieldNoMiscFees         = 136,
fieldMiscFeeAmt         = 137,
fieldMiscFeeCurr        = 138,
fieldMiscFeeType        = 139,
fieldPrevClosePx        = 140,
fieldResetSeqNumFlag                = 141,
fieldSenderLocationID               = 142,
fieldTargetLocationID               = 143,
fieldOnBehalfOfLocationID           = 144,
fieldDeliverToLocationID            = 145,
fieldNoRelatedSym                   = 146,
fieldSubject                        = 147,
```

```
fieldHeadline                       = 148,
fieldURLLink                        = 149,
fieldExecType                       = 150,
fieldLeavesQty                      = 151,
fieldCashOrderQty                   = 152,
fieldAllocAvgPx                     = 153,
fieldAllocNetMoney                  = 154,
fieldSettlCurrFxRate                = 155,
fieldSettlCurrFxRateCalc            = 156,
fieldNumDaysInterest                = 157,
fieldAccruedInterestRate            = 158,
fieldAccruedInterestAmt             = 159,
fieldSettlInstMode                  = 160,
fieldAllocText                      = 161,
fieldSettlInstID                    = 162,
fieldSettlInstTransType             = 163,
fieldEmailThreadID                  = 164,
fieldSettlInstSource                = 165,
fieldSettlLocation                  = 166,
fieldSecurityType                   = 167,
fieldEffectiveTime                  = 168,
fieldStandInstDbType                = 169,
fieldStandInstDbName                = 170,
fieldStandInstDbID                  = 171,
fieldSettlDeliveryType              = 172,
fieldSettlDepositoryCode            = 173,
fieldSettlBrkrCode                  = 174,
fieldSettlInstCode                  = 175,
fieldSecuritySettlAgentName         = 176,
fieldSecuritySettlAgentCode         = 177,
fieldSecuritySettlAgentAcctNum      = 178,
fieldSecuritySettlAgentAcctName     = 179,
fieldSecuritySettlAgentContactName  = 180,
fieldSecuritySettlAgentContactPhone = 181,
fieldCashSettlAgentName             = 182,
fieldCashSettlAgentCode             = 183,
fieldCashSettlAgentAcctNum          = 184,
fieldCashSettlAgentAcctName         = 185,
fieldCashSettlAgentContactName      = 186,
fieldCashSettlAgentContactPhone     = 187,
fieldBidSpotRate                    = 188,
fieldBidForwardPoints               = 189,
fieldOfferSpotRate                  = 190,
fieldOfferForwardPoints             = 191,
fieldOrderQty2                      = 192,
fieldFutSettDate2                   = 193,
fieldSettlDate2                     = fieldFutSettDate2
fieldLastSpotRate                   = 194,
fieldLastForwardPoints              = 195,
fieldAllocLinkID                    = 196,
fieldAllocLinkType                  = 197,
fieldSecondaryOrderID               = 198,
```

```
fieldNoIOIQualifiers                = 199,
fieldMaturityMonthYear              = 200,
fieldPutOrCall                      = 201,
fieldStrikePrice                    = 202,
fieldCoveredOrUncovered             = 203,
fieldCustomerOrFirm                 = 204,
fieldMaturityDay                    = 205,
fieldOptAttribute                   = 206,
fieldSecurityExchange               = 207,
fieldNotifyBrokerOfCredit           = 208,
fieldAllocHandlInst                 = 209,
fieldMaxShow                        = 210,
fieldPegDifference                  = 211,
fieldPegOffsetValue                 = fieldPegDifference,
fieldXMLDataLen                     = 212,
fieldXMLData                        = 213,

// FIX.4.2 fields
fieldSettlInstRefID                   = 214,
fieldNoRoutingIDs                     = 215,
fieldRoutingType                      = 216,
fieldRoutingID                        = 217,
fieldSpreadToBenchmark                = 218,
fieldSpread                           = fieldSpreadToBenchmark ,
fieldBenchmark                        = 219,
fieldCouponRate                       = 223,
fieldContractMultiplier               = 231,
fieldMDReqID                          = 262,
fieldSubscriptionRequestType          = 263,
fieldMarketDepth                      = 264,
fieldMDUpdateType                     = 265,
fieldAggregatedBook                   = 266,
fieldNoMDEntryTypes                   = 267,
fieldNoMDEntries                      = 268,
fieldMDEntryType                      = 269,
fieldMDEntryPx                        = 270,
fieldMDEntrySize                      = 271,
fieldMDEntryDate                      = 272,
fieldMDEntryTime                      = 273,
fieldTickDirection                    = 274,
fieldMDMkt                            = 275,
fieldQuoteCondition                   = 276,
fieldTradeCondition                   = 277,
fieldMDEntryID                        = 278,
fieldMDUpdateAction                   = 279,
fieldMDEntryRefID                     = 280,
fieldMDReqRejReason                   = 281,
fieldMDEntryOriginator                = 282,
fieldLocationID                       = 283,
fieldDeskID                           = 284,
fieldDeleteReason                     = 285,
fieldOpenCloseSettleFlag              = 286,
```

```
fieldOpenCloseSettlFlag              = fieldOpenCloseSettleFlag,
fieldSellerDays                      = 287,
fieldMDEntryBuyer                    = 288,
fieldMDEntrySeller                   = 289,
fieldMDEntryPositionNo               = 290,
fieldFinancialStatus                 = 291,
fieldCorporateAction                 = 292,
fieldDefBidSize                      = 293,
fieldDefOfferSize                    = 294,
fieldNoQuoteEntries                  = 295,
fieldNoQuoteSets                     = 296,
fieldQuoteAckStatus                  = 297,
fieldQuoteStatus                     = fieldQuoteAckStatus
fieldQuoteCancelType                 = 298,
fieldQuoteEntryID                    = 299,
fieldQuoteRejectReason               = 300,
fieldQuoteResponseLevel              = 301,
fieldQuoteSetID                      = 302,
fieldQuoteRequestType                = 303,
fieldTotQuoteEntries                 = 304,
fieldTotNoQuoteEntries               = fieldTotQuoteEntries ,
fieldUnderlyingIDSource              = 305,
fieldUnderlyingSecurityIDSource      = fieldUnderlyingIDSource ,
fieldUnderlyingIssuer                = 306,
fieldUnderlyingSecurityDesc          = 307,
fieldUnderlyingSecurityExchange      = 308,
fieldUnderlyingSecurityID            = 309,
fieldUnderlyingSecurityType          = 310,
fieldUnderlyingSymbol                = 311,
fieldUnderlyingSymbolSfx             = 312,
fieldUnderlyingMaturityMonthYear     = 313,
fieldUnderlyingMaturityDay           = 314,
fieldUnderlyingPutOrCall             = 315,
fieldUnderlyingStrikePrice           = 316,
fieldUnderlyingOptAttribute          = 317,
fieldUnderlyingCurrency              = 318,
fieldRatioQty                        = 319,
fieldSecurityReqID                   = 320,
fieldSecurityRequestType             = 321,
fieldSecurityResponseID              = 322,
fieldSecurityResponseType            = 323,
fieldSecurityStatusReqID             = 324,
fieldUnsolicitedIndicator            = 325,
fieldSecurityTradingStatus           = 326,
fieldHaltReason                      = 327,
fieldInViewOfCommon                  = 328,
fieldDueToRelated                    = 329,
fieldBuyVolume                       = 330,
fieldSellVolume                      = 331,
fieldHighPx                          = 332,
fieldLowPx                           = 333,
fieldAdjustment                      = 334,
```

```
fieldTradSesReqID                       = 335,
fieldTradingSessionID                   = 336,
fieldContraTrader                       = 337,
fieldTradSesMethod                      = 338,
fieldTradSesMode                        = 339,
fieldTradSesStatus                      = 340,
fieldTradSesStartTime                   = 341,
fieldTradSesOpenTime                    = 342,
fieldTradSesPreCloseTime                = 343,
fieldTradSesCloseTime                   = 344,
fieldTradSesEndTime                     = 345,
fieldNumberOfOrders                     = 346,
fieldMessageEncoding                    = 347,
fieldEncodedIssuerLen                   = 348,
fieldEncodedIssuer                      = 349,
fieldEncodedSecurityDescLen             = 350,
fieldEncodedSecurityDesc                = 351,
fieldEncodedListExecInstLen             = 352,
fieldEncodedListExecInst                = 353,
fieldEncodedTextLen                     = 354,
fieldEncodedText                        = 355,
fieldEncodedSubjectLen                  = 356,
fieldEncodedSubject                     = 357,
fieldEncodedHeadlineLen                 = 358,
fieldEncodedHeadline                    = 359,
fieldEncodedAllocTextLen                = 360,
fieldEncodedAllocText                   = 361,
fieldEncodedUnderlyingIssuerLen         = 362,
fieldEncodedUnderlyingIssuer            = 363,
fieldEncodedUnderlyingSecurityDescLen = 364,
fieldEncodedUnderlyingSecurityDesc    = 365,
fieldAllocPrice                         = 366,
fieldQuoteSetValidUntilTime             = 367,
fieldQuoteEntryRejectReason             = 368,
fieldLastMsgSeqNumProcessed             = 369,
fieldOnBehalfOfSendingTime              = 370,
fieldRefTagID                           = 371,
fieldRefMsgType                         = 372,
fieldSessionRejectReason                = 373,
fieldBidRequestTransType                = 374,
fieldContraBroker                       = 375,
fieldComplianceID                       = 376,
fieldSolicitedFlag                      = 377,
fieldExecRestatementReason              = 378,
fieldBusinessRejectRefID                = 379,
fieldBusinessRejectReason               = 380,
fieldNoContraBrokers                    = 382,
fieldMaxMessageSize                     = 383,
fieldNoMsgTypes                         = 384,
fieldMsgDirection                       = 385,
fieldNoTradingSessions                  = 386,
fieldTotalVolumeTraded                  = 387,
```

```
fieldDiscretionInst                 = 388,
fieldDiscretionOffset               = 389,
fieldDiscretionOffsetValue          = fieldDiscretionOffset ,
fieldBidID                          = 390,
fieldClientBidID                    = 391,
fieldListName                       = 392,
fieldTotalNumSecurities             = 393,
fieldTotNoRelatedSym                = fieldTotalNumSecurities ,
fieldBidType                        = 394,
fieldNumTickets                     = 395,
fieldSideValue1                     = 396,
fieldSideValue2                     = 397,
fieldNoBidDescriptors               = 398,
fieldBidDescriptorType              = 399,
fieldBidDescriptor                  = 400,
fieldSideValueInd                   = 401,
fieldLiquidityPctLow                = 402,
fieldLiquidityPctHigh               = 403,
fieldLiquidityValue                 = 404,
fieldEFPTrackingError               = 405,
fieldFairValue                      = 406,
fieldOutsideIndexPct                = 407,
fieldValueOfFutures                 = 408,
fieldLiquidityIndType               = 409,
fieldWtAverageLiquidity             = 410,
fieldExchangeForPhysical            = 411,
fieldOutSideMainCountryIndex        = 412,
fieldCrossPercent                   = 413,
fieldProgRptReqs                    = 414,
fieldProgPeriodInterval             = 415,
fieldIncTaxInd                      = 416,
fieldNumBidders                     = 417,
fieldTradeType                      = 418,
fieldBidTradeType                   = fieldTradeType ,
fieldBasisPxType                    = 419,
fieldNoBidComponents                = 420,
fieldCountry                        = 421,
fieldTotNoStrikes                   = 422,
fieldPriceType                      = 423,
fieldDayOrderQty                    = 424,
fieldDayCumQty                      = 425,
fieldDayAvgPx                       = 426,
fieldGTBookingInst                  = 427,
fieldNoStrikes                      = 428,
fieldListStatusType                 = 429,
fieldNetGrossInd                    = 430,
fieldListOrderStatus                = 431,
fieldExpireDate                     = 432,
fieldListExecInstType               = 433,
fieldCxlRejResponseTo               = 434,
fieldUnderlyingCouponRate           = 435,
fieldUnderlyingContractMultiplier   = 436,
```

```
fieldContraTradeQty                   = 437,
fieldContraTradeTime                  = 438,
fieldClearingFirm                     = 439,
fieldClearingAccount                  = 440,
fieldLiquidityNumSecurities           = 441,
fieldMultiLegReportingType            = 442,

//FIX 4.2 fields which were added in FIX 4.2 Errata
fieldStrikeTime                       = 443,
fieldListStatusText                   = 444,
fieldEncodedListStatusTextLen         = 445,
fieldEncodedListStatusText            = 446,
fieldListStatusEncodedText            = fieldEncodedListStatusText,
//FIX 4.3 fields
fieldPartyIDSource                    = 447,
fieldPartyID                          = 448,
fieldNetChgPrevDay                    = 451,
fieldPartyRole                        = 452,
fieldNoPartyIDs                       = 453,
fieldNoSecurityAltID                  = 454,
fieldSecurityAltID                    = 455,
fieldSecurityAltIDSource              = 456,
fieldNoUnderlyingSecurityAltID        = 457,
fieldUnderlyingSecurityAltID          = 458,
fieldUnderlyingSecurityAltIDSource    = 459,
fieldProduct                          = 460,
fieldCFICode                          = 461,
fieldUnderlyingProduct                = 462,
fieldUnderlyingCFICode                = 463,
fieldTestMessageIndicator             = 464,
fieldQuantityType                     = 465,
fieldBookingRefID                     = 466,
fieldIndividualAllocID                = 467,
fieldRoundingDirection                = 468,
fieldRoundingModulus                  = 469,
fieldCountryOfIssue                   = 470,
fieldStateOrProvinceOfIssue           = 471,
fieldLocaleOfIssue                    = 472,
fieldNoRegistDtls                     = 473,
fieldMailingDtls                      = 474,
fieldInvestorCountryOfResidence       = 475,
fieldPaymentRef                       = 476,
fieldDistribPaymentMethod             = 477,
fieldCashDistribCurr                  = 478,
fieldCommCurrency                     = 479,
fieldCancellationRights               = 480,
fieldMoneyLaunderingStatus            = 481,
fieldMailingInst                      = 482,
fieldTransBkdTime                     = 483,
fieldExecPriceType                    = 484,
fieldExecPriceAdjustment              = 485,
fieldDateOfBirth                      = 486,
```

```
fieldTradeReportTransType             = 487,
fieldCardHolderName                   = 488,
fieldCardNumber                       = 489,
fieldCardExpDate                      = 490,
fieldCardIssNum                       = 491,
fieldPaymentMethod                    = 492,
fieldRegistAcctType                   = 493,
fieldDesignation                      = 494,
fieldTaxAdvantageType                 = 495,
fieldRegistRejReasonText              = 496,
fieldFundRenewWaiv                    = 497,
fieldCashDistribAgentName             = 498,
fieldCashDistribAgentCode             = 499,
fieldCashDistribAgentAcctNumber       = 500,
fieldCashDistribPayRef                = 501,
fieldCashDistribAgentAcctName         = 502,
fieldCardStartDate                    = 503,
fieldPaymentDate                      = 504,
fieldPaymentRemitterID                = 505,
fieldRegistStatus                     = 506,
fieldRegistRejReasonCode              = 507,
fieldRegistRefID                      = 508,
fieldRegistDetls                      = 509,
fieldNoDistribInsts                   = 510,
fieldRegistEmail                      = 511,
fieldDistribPercentage                = 512,
fieldRegistID                         = 513,
fieldRegistTransType                  = 514,
fieldExecValuationPoint               = 515,
fieldOrderPercent                     = 516,
fieldOwnershipType                    = 517,
fieldNoContAmts                       = 518,
fieldContAmtType                      = 519,
fieldContAmtValue                     = 520,
fieldContAmtCurr                      = 521,
fieldOwnerType                        = 522,
fieldPartySubID                       = 523,
fieldNestedPartyID                    = 524,
fieldNestedPartyIDSource              = 525,
fieldSecondaryClOrdID                 = 526,
fieldSecondaryExecID                  = 527,
fieldOrderCapacity                    = 528,
fieldOrderRestrictions                = 529,
fieldMassCancelRequestType            = 530,
fieldMassCancelResponse               = 531,
fieldMassCancelRejectReason           = 532,
fieldTotalAffectedOrders              = 533,
fieldNoAffectedOrders                 = 534,
fieldAffectedOrderID                  = 535,
fieldAffectedSecondaryOrderID         = 536,
fieldQuoteType                        = 537,
fieldNestedPartyRole                  = 538,
```

```
fieldNoNestedPartyIDs                = 539,
fieldTotalAccruedInterestAmt         = 540,
fieldMaturityDate                    = 541,
fieldUnderlyingMaturityDate          = 542,
fieldInstrRegistry                   = 543,
fieldCashMargin                      = 544,
fieldNestedPartySubID                = 545,
fieldScope                           = 546,
fieldMDImplicitDelete                = 547,
fieldCrossID                         = 548,
fieldCrossType                       = 549,
fieldCrossPrioritization             = 550,
fieldOrigCrossID                     = 551,
fieldNoSides                         = 552,
fieldUsername                        = 553,
fieldPassword                        = 554,
fieldNoLegs                          = 555,
fieldLegCurrency                     = 556,
fieldTotalNumSecurityTypes           = 557,
fieldTotNoSecurityTypes              = fieldTotalNumSecurityTypes,
fieldNoSecurityTypes                 = 558,
fieldSecurityListRequestType         = 559,
fieldSecurityRequestResult           = 560,
fieldRoundLot                        = 561,
fieldMinTradeVol                     = 562,
fieldMultiLegRptTypeReq              = 563,
fieldLegPositionEffect               = 564,
fieldLegCoveredOrUncovered           = 565,
fieldLegPrice                        = 566,
fieldTradSesStatusRejReason          = 567,
fieldTradeRequestID                  = 568,
fieldTradeRequestType                = 569,
fieldPreviouslyReported              = 570,
fieldTradeReportID                   = 571,
fieldTradeReportRefID                = 572,
fieldMatchStatus                     = 573,
fieldMatchType                       = 574,
fieldOddLot                          = 575,
fieldNoClearingInstructions          = 576,
fieldClearingInstruction             = 577,
fieldTradeInputSource                = 578,
fieldTradeInputDevice                = 579,
fieldNoDates                         = 580,
fieldAccountType                     = 581,
fieldCustOrderCapacity               = 582,
fieldClOrdLinkID                     = 583,
fieldMassStatusReqID                 = 584,
fieldMassStatusReqType               = 585,
fieldOrigOrdModTime                  = 586,
fieldLegSettlmntTyp                  = 587,
fieldLegSettlType                    = fieldLegSettlmntTyp ,
fieldLegFutSettDate                  = 588,
```

```
fieldLegSettlDate                       = fieldLegFutSettDate ,
fieldDayBookingInst                     = 589,
fieldBookingUnit                        = 590,
fieldPreallocMethod                     = 591,
fieldUnderlyingCountryOfIssue           = 592,
fieldUnderlyingStateOrProvinceOfIssue = 593,
fieldUnderlyingLocaleOfIssue            = 594,
fieldUnderlyingInstrRegistry            = 595,
fieldLegCountryOfIssue                  = 596,
fieldLegStateOrProvinceOfIssue          = 597,
fieldLegLocaleOfIssue                   = 598,
fieldLegInstrRegistry                   = 599,
fieldLegSymbol                          = 600,
fieldLegSymbolSfx                       = 601,
fieldLegSecurityID                      = 602,
fieldLegSecurityIDSource                = 603,
fieldNoLegSecurityAltID                 = 604,
fieldLegSecurityAltID                   = 605,
fieldLegSecurityAltIDSource             = 606,
fieldLegProduct                         = 607,
fieldLegCFICode                         = 608,
fieldLegSecurityType                    = 609,
fieldLegMaturityMonthYear               = 610,
fieldLegMaturityDate                    = 611,
fieldLegStrikePrice                     = 612,
fieldLegOptAttribute                    = 613,
fieldLegContractMultiplier              = 614,
fieldLegCouponRate                      = 615,
fieldLegSecurityExchange                = 616,
fieldLegIssuer                          = 617,
fieldEncodedLegIssuerLen                = 618,
fieldEncodedLegIssuer                   = 619,
fieldLegSecurityDesc                    = 620,
fieldEncodedLegSecurityDescLen          = 621,
fieldEncodedLegSecurityDesc             = 622,
fieldLegRatioQty                        = 623,
fieldLegSide                            = 624,
fieldTradingSessionSubID                = 625,
fieldAllocType                          = 626,
fieldNoHops                             = 627,
fieldHopCompID                          = 628,
fieldHopSendingTime                     = 629,
fieldHopRefID                           = 630,
fieldMidPx                              = 631,
fieldBidYield                           = 632,
fieldMidYield                           = 633,
fieldOfferYield                         = 634,
fieldClearingFeeIndicator               = 635,
fieldWorkingIndicator                   = 636,
fieldLegLastPx                          = 637,
fieldPriorityIndicator                  = 638,
fieldPriceImprovement                   = 639,
```

```
fieldPrice2                         = 640,
fieldLastForwardPoints2             = 641,
fieldBidForwardPoints2              = 642,
fieldOfferForwardPoints2            = 643,
fieldRFQReqID                       = 644,
fieldMktBidPx                       = 645,
fieldMktOfferPx                     = 646,
fieldMinBidSize                     = 647,
fieldMinOfferSize                   = 648,
fieldQuoteStatusReqID               = 649,
fieldLegalConfirm                   = 650,
fieldUnderlyingLastPx               = 651,
fieldUnderlyingLastQty              = 652,
fieldLegRefID                       = 654,
fieldContraLegRefID                 = 655,
fieldSettlCurrBidFxRate             = 656,
fieldSettlCurrOfferFxRate           = 657,
fieldQuoteRequestRejectReason       = 658,
fieldSideComplianceID               = 659,

 //FIX 4.4 fields
fieldAcctIDSource                   = 660,
fieldAllocAcctIDSource              = 661,
fieldBenchmarkPrice                 = 662,
fieldBenchmarkPriceType             = 663,
fieldConfirmID                      = 664,
fieldConfirmStatus                  = 665,
fieldConfirmTransType               = 666,
fieldContractSettlMonth             = 667,
fieldDeliveryForm                   = 668,
fieldLastParPx                      = 669,
fieldNoLegAllocs                    = 670,
fieldLegAllocAccount                = 671,
fieldLegIndividualAllocID           = 672,
fieldLegAllocQty                    = 673,
fieldLegAllocAcctIDSource           = 674,
fieldLegSettlCurrency               = 675,
fieldLegBenchmarkCurveCurrency      = 676,
fieldLegBenchmarkCurveName          = 677,
fieldLegBenchmarkCurvePoint         = 678,
fieldLegBenchmarkPrice              = 679,
fieldLegBenchmarkPriceType          = 680,
fieldLegBidPx                       = 681,
fieldLegIOIQty                      = 682,
fieldNoLegStipulations              = 683,
fieldLegOfferPx                     = 684,
fieldLegOrderQty                    = 685,
fieldLegPriceType                   = 686,
fieldLegQty                         = 687,
fieldLegStipulationType             = 688,
fieldLegStipulationValue            = 689,
fieldLegSwapType                    = 690,
```

```
fieldPool                               = 691,
fieldQuotePriceType                     = 692,
fieldQuoteRespID                        = 693,
fieldQuoteRespType                      = 694,
fieldQuoteQualifier                     = 695,
fieldYieldRedemptionDate                = 696,
fieldYieldRedemptionPrice               = 697,
fieldYieldRedemptionPriceType           = 698,
fieldBenchmarkSecurityID                = 699,
fieldReversalIndicator                  = 700,
fieldYieldCalcDate                      = 701,
fieldNoPositions                        = 702,
fieldPosType                            = 703,
fieldLongQty                            = 704,
fieldShortQty                           = 705,
fieldPosQtyStatus                       = 706,
fieldPosAmtType                         = 707,
fieldPosAmt                             = 708,
fieldPosTransType                       = 709,
fieldPosReqID                           = 710,
fieldNoUnderlyings                      = 711,
fieldPosMaintAction                     = 712,
fieldOrigPosReqRefID                    = 713,
fieldPosMaintRptRefID                   = 714,
fieldClearingBusinessDate               = 715,
fieldSettlSessID                        = 716,
fieldSettlSessSubID                     = 717,
fieldAdjustmentType                     = 718,
fieldContraryInstructionIndicator       = 719,
fieldPriorSpreadIndicator               = 720,
fieldPosMaintRptID                      = 721,
fieldPosMaintStatus                     = 722,
fieldPosMaintResult                     = 723,
fieldPosReqType                         = 724,
fieldResponseTransportType              = 725,
fieldResponseDestination                = 726,
fieldTotalNumPosReports                 = 727,
fieldPosReqResult                       = 728,
fieldPosReqStatus                       = 729,
fieldSettlPrice                         = 730,
fieldSettlPriceType                     = 731,
fieldUnderlyingSettlPrice               = 732,
fieldUnderlyingSettlPriceType           = 733,
fieldPriorSettlPrice                    = 734,
fieldNoQuoteQualifiers                  = 735,
fieldAllocSettlCurrency                 = 736,
fieldAllocSettlCurrAmt                  = 737,
fieldInterestAtMaturity                 = 738,
fieldLegDatedDate                       = 739,
fieldLegPool                            = 740,
fieldAllocInterestAtMaturity            = 741,
fieldAllocAccruedInterestAmt            = 742,
```

```
fieldDeliveryDate                       = 743,
fieldAssignmentMethod                   = 744,
fieldAssignmentUnit                     = 745,
fieldOpenInterest                       = 746,
fieldExerciseMethod                     = 747,
fieldTotNumTradeReports                 = 748,
fieldTradeRequestResult                 = 749,
fieldTradeRequestStatus                 = 750,
fieldTradeReportRejectReason            = 751,
fieldSideMultiLegReportingType          = 752,
fieldNoPosAmt                           = 753,
fieldAutoAcceptIndicator                = 754,
fieldAllocReportID                      = 755,
fieldNoNested2PartyIDs                  = 756,
fieldNested2PartyID                     = 757,
fieldNested2PartyIDSource               = 758,
fieldNested2PartyRole                   = 759,
fieldNested2PartySubID                  = 760,
fieldBenchmarkSecurityIDSource          = 761,
fieldSecuritySubType                    = 762,
fieldUnderlyingSecuritySubType          = 763,
fieldLegSecuritySubType                 = 764,
fieldAllowableOneSidednessPct           = 765,
fieldAllowableOneSidednessValue         = 766,
fieldAllowableOneSidednessCurr          = 767,
fieldNoTrdRegTimestamps                 = 768,
fieldTrdRegTimestamp                    = 769,
fieldTrdRegTimestampType                = 770,
fieldTrdRegTimestampOrigin              = 771,
fieldConfirmRefID                       = 772,
fieldConfirmType                        = 773,
fieldConfirmRejReason                   = 774,
fieldBookingType                        = 775,
fieldIndividualAllocRejCode             = 776,
fieldSettlInstMsgID                     = 777,
fieldNoSettlInst                        = 778,
fieldLastUpdateTime                     = 779,
fieldAllocSettlInstType                 = 780,
fieldNoSettlPartyIDs                    = 781,
fieldSettlPartyID                       = 782,
fieldSettlPartyIDSource                 = 783,
fieldSettlPartyRole                     = 784,
fieldSettlPartySubID                    = 785,
fieldSettlPartySubIDType                = 786,
fieldDlvyInstType                       = 787,
fieldTerminationType                    = 788,
fieldNextExpectedMsgSeqNum              = 789,
fieldOrdStatusReqID                     = 790,
fieldSettlInstReqID                     = 791,
fieldSettlInstReqRejCode                = 792,
fieldSecondaryAllocID                   = 793,
fieldAllocReportType                    = 794,
```

```
fieldAllocReportRefID                = 795,
fieldAllocCancReplaceReason          = 796,
fieldCopyMsgIndicator                = 797,
fieldAllocAccountType                = 798,
fieldOrderAvgPx                      = 799,
fieldOrderBookingQty                 = 800,
fieldNoSettlPartySubIDs              = 801,
fieldNoPartySubIDs                   = 802,
fieldPartySubIDType                  = 803,
fieldNoNestedPartySubIDs             = 804,
fieldNestedPartySubIDType            = 805,
fieldNoNested2PartySubIDs            = 806,
fieldNested2PartySubIDType           = 807,
fieldAllocIntermedReqType            = 808,
fieldUnderlyingPx                    = 810,
fieldPriceDelta                      = 811,
fieldApplQueueMax                    = 812,
fieldApplQueueDepth                  = 813,
fieldApplQueueResolution             = 814,
fieldApplQueueAction                 = 815,
fieldNoAltMDSource                   = 816,
fieldAltMDSourceID                   = 817,
fieldSecondaryTradeReportID          = 818,
fieldAvgPxIndicator                  = 819,
fieldTradeLinkID                     = 820,
fieldOrderInputDevice                = 821,
fieldUnderlyingTradingSessionID      = 822,
fieldUnderlyingTradingSessionSubID   = 823,
fieldTradeLegRefID                   = 824,
fieldExchangeRule                    = 825,
fieldTradeAllocIndicator             = 826,
fieldExpirationCycle                 = 827,
fieldTrdType                         = 828,
fieldTrdSubType                      = 829,
fieldTransferReason                  = 830,
fieldAsgnReqID                       = 831,
fieldTotNumAssignmentReports         = 832,
fieldAsgnRptID                       = 833,
fieldThresholdAmount                 = 834,
fieldPegMoveType                     = 835,
fieldPegOffsetType                   = 836,
fieldPegLimitType                    = 837,
fieldPegRoundDirection               = 838,
fieldPeggedPrice                     = 839,
fieldPegScope                        = 840,
fieldDiscretionMoveType              = 841,
fieldDiscretionOffsetType            = 842,
fieldDiscretionLimitType             = 843,
fieldDiscretionRoundDirection        = 844,
fieldDiscretionPrice                 = 845,
fieldDiscretionScope                 = 846,
fieldTargetStrategy                  = 847,
```

```
fieldTargetStrategyParameters         = 848,
fieldParticipationRate                = 849,
fieldTargetStrategyPerformance        = 850,
fieldLastLiquidityInd                 = 851,
fieldPublishTrdIndicator              = 852,
fieldShortSaleReason                  = 853,
fieldQtyType                          = 854,
fieldSecondaryTrdType                 = 855,
fieldTradeReportType                  = 856,
fieldAllocNoOrdersType                = 857,
fieldSharedCommission                 = 858,
fieldConfirmReqID                     = 859,
fieldAvgParPx                         = 860,
fieldReportedPx                       = 861,
fieldNoCapacities                     = 862,
fieldOrderCapacityQty                 = 863,
fieldNoEvents                         = 864,
fieldEventType                        = 865,
fieldEventDate                        = 866,
fieldEventPx                          = 867,
fieldEventText                        = 868,
fieldPctAtRisk                        = 869,
fieldNoInstrAttrib                    = 870,
fieldInstrAttribType                  = 871,
fieldInstrAttribValue                 = 872,
fieldDatedDate                        = 873,
fieldInterestAccrualDate              = 874,
fieldCPProgram                        = 875,
fieldCPRegType                        = 876,
fieldUnderlyingCPProgram              = 877,
fieldUnderlyingCPRegType              = 878,
fieldUnderlyingQty                    = 879,
fieldTrdMatchID                       = 880,
fieldSecondaryTradeReportRefID        = 881,
fieldUnderlyingDirtyPrice             = 882,
fieldUnderlyingEndPrice               = 883,
fieldUnderlyingStartValue             = 884,
fieldUnderlyingCurrentValue           = 885,
fieldUnderlyingEndValue               = 886,
fieldNoUnderlyingStips                = 887,
fieldUnderlyingStipType               = 888,
fieldUnderlyingStipValue              = 889,
fieldMaturityNetMoney                 = 890,
fieldMiscFeeBasis                     = 891,
fieldTotNoAllocs                      = 892,
fieldLastFragment                     = 893,
fieldCollReqID                        = 894,
fieldCollAsgnReason                   = 895,
fieldCollInquiryQualifier             = 896,
fieldNoTrades                         = 897,
fieldMarginRatio                      = 898,
fieldMarginExcess                     = 899,
```

```
fieldTotalNetValue                  = 900,
fieldCashOutstanding                = 901,
fieldCollAsgnID                     = 902,
fieldCollAsgnTransType              = 903,
fieldCollRespID                     = 904,
fieldCollAsgnRespType               = 905,
fieldCollAsgnRejectReason           = 906,
fieldCollAsgnRefID                  = 907,
fieldCollRptID                      = 908,
fieldCollInquiryID                  = 909,
fieldCollStatus                     = 910,
fieldTotNumReports                  = 911,
fieldLastRptRequested               = 912,
fieldAgreementDesc                  = 913,
fieldAgreementID                    = 914,
fieldAgreementDate                  = 915,
fieldStartDate                      = 916,
fieldEndDate                        = 917,
fieldAgreementCurrency              = 918,
fieldDeliveryType                   = 919,
fieldEndAccruedInterestAmt          = 920,
fieldStartCash                      = 921,
fieldEndCash                        = 922,
fieldUserRequestID                  = 923,
fieldUserRequestType                = 924,
fieldNewPassword                    = 925,
fieldUserStatus                     = 926,
fieldUserStatusText                 = 927,
fieldStatusValue                    = 928,
fieldStatusText                     = 929,
fieldRefCompID                      = 930,
fieldRefSubID                       = 931,
fieldNetworkResponseID              = 932,
fieldNetworkRequestID               = 933,
fieldLastNetworkResponseID          = 934,
fieldNetworkRequestType             = 935,
fieldNoCompIDs                      = 936,
fieldNetworkStatusResponseType      = 937,
fieldNoCollInquiryQualifier         = 938,
fieldTrdRptStatus                   = 939,
fieldAffirmStatus                   = 940,
fieldUnderlyingStrikeCurrency       = 941,
fieldLegStrikeCurrency              = 942,
fieldTimeBracket                    = 943,
fieldCollAction                     = 944,
fieldCollInquiryStatus              = 945,
fieldCollInquiryResult              = 946,
fieldStrikeCurrency                 = 947,
fieldNoNested3PartyIDs              = 948,
fieldNested3PartyID                 = 949,
fieldNested3PartyIDSource           = 950,
fieldNested3PartyRole               = 951,
```

```
        fieldNoNested3PartySubIDs             = 952,
        fieldNested3PartySubID                = 953,
        fieldNested3PartySubIDType            = 954,
        fieldLegContractSettlMonth            = 955,
        fieldLegInterestAccrualDate           = 956,

        MAX_FIELD_TAG           = fieldLegInterestAccrualDate, //i.e. 956,
        NON_COMPOSING_FIELD     = -1,

        // Custom fields
           fieldMessageFormat                 = 16536,
           fieldIsFromCounterParty            = 16575,
           fieldConnectionName                = 16576,
           fieldTimestamp                     = 16577,
           fieldOriginatingConnectionName     =.16578,
           fieldOriginatingConnectionMsgId    = 16579,
           fieldConnectionMessageId           = 16580,
           fieldGlobalMessageId               = 16581,
           fieldMessageErrors                 = 16584,
           fieldIsOutOfSequence               = 16585,
           fieldWireProtocolVersion           = 16586,
           fieldDoNotPersistFlag              = 16588,
           fieldConnectionUniformMessageId    = 16589
    };
```

Tags `16502, 16503, 16504, 16535, 16579, 15682` and `15683` are used internally by InfoReach.

ELTAdminMessageEvent

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTAdminMessageEvent.h**

This class is derived from the **ELTEvent** class.

It separates events containing FIX administrative messages from the events that contain FIX application messages. **ELTAdminMessageEvent** uses the same methods as **ELTEvent** class.

ELTAppMessageEvent

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTAppMessageEvent.h**

This class is derived from the **ELTEvent** class.

It separates events containing FIX administrative messages from the events that contain FIX application messages. **ELTAppMessageEvent** class uses the same methods as **ELTEvent** class.

ELTConnectionState

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTConnectionState.h**

This class represents the engine connection state.

The connection state can be obtained by applications from **IELTEngine** by utilizing its **getConnectionState()** method.

Accessor methods:

```
bool isStateActive(long stateMask) const;
```

Parameters:
stateMask – one of CONNECTION_STATE values.

Returns: **True** if connection is in specified state.

```
long getHeartbeats() const;
```

Returns: Current heartbeat rate for this connection.

```
const std::string&  getLastError() const;
```

Returns: Description of the last error that was detected on this connection.

```
const std::string&  getLocalAddress() const;
```

Returns: String representation of local (client) address of the connection.

```
const std::string&  getRemoteAddress() const;
```

Returns: String representation of remote (server) address of the connection.

```
long getState() const;
```

Returns: Full state of the connection (bit set)

```
const ELTTime&  getTimeLastStateChange() const;
```

Returns: The last time the state has changed.

```
bool hasDisconnected() const;
```

Returns: **True** if socket connection was lost.

```
bool isLinkActive() const;
```

Returns: **True** if socket connection is alive.

```
bool isLoggedOut() const;
```

Returns: **True** if connection was disconnected on FIX level.

```
bool isMessagingIdle() const;
```

Returns: **True** if connection is idled (i.e. last received message was heartbeat message).

```
bool isSessionActive() const;
```

Returns: **True** if connection is active (i.e. last received message was application message).

```
bool isUnavailable() const;
```

Returns: **True** for client connections when server counterpart is unavailable.

ELTEngineProxy

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTEngineProxy.h**

Implement ***IELTEngine*** interface.

All engine functionality is made available to applications through methods of this class. Application components have access to the engine functionality only after the **proxy's** initialization is successfully completed.

Methods (besides those implementing IELTEngine **interface)**:

```
static void initialize(
      const IELTEngineConfigProviderPtr configProvider,
      const IELTEngineAdapterFactoryPtr adapterFactory)
      throw (
            ELTWrongConfigurationException,
            ELTAdapterCreationException,
            ELTCannotConnectException);
```

Parameters:
configProvider – configuration provider for engine adapter.
adapterFactory – specialized adapter factory for creating adapters.

Throws:
ELTWrongConfigurationException – if provided configuration is wrong or absent.
ELTAdapterCreationException – when adapter cannot be created using provided configuration and adapter factory.
ELTCannotConnectException – when adapter cannot be connected to the remote engine.

```
static bool isInitialized();
```

Returns: **True** if **Proxy** is initialized.

```
static void uninitialize();
```

Disconnects adapter and then destroys it.

```
static ELTEngineProxy& getInstance();
```

Returns: Initialized **Proxy** instance.

ELTEvent

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTEvent.h**

Applications are notified about messages sent or received on FIX connections through objects of this type. Each **ELTEvent** object contains single or multiple FIX messages that were sent or received on the connection, and the connection name.

Since listening components only receive these objects and never create them, this class only has accessor methods.

Accessor methods:

```
virtual const std::string&getConnectionName() const;
```

Returns: Reference to the source connection name.

```
virtual const ELTMessagePtr getMessage() const;
```

Returns: Smart pointer to the message.

ELTField

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTField.h**

Represents FIX field. A collection of **ELTField** objects can be combined into an **ELTFieldGroup** object from which the **ELTMessage** can be formed.

Constructors:
```
ELTField();
```
Creates empty field.

```
ELTField(int tag, int type, const ELTValue& value);
```
Creates a field, with specified tag, type and value.

Parameters:
tag – FIX tag. For convenience, most of FIX field's tags are provided as constants in ELT class.

Type – type of the field. All types defined as enumeration TYPES in ELTValue class.

value – the field value (see ELTValue).

Accessor methods:
```
virtual int getTag() const;
```

Returns: The field tag.

```
virtual int getType() const;
```

Returns: The field type (see TYPES).

```
virtual const ELTValue& getValue()    const;
```

Returns: The field's current value.

Mutator methods:

```
virtual void setTag  (int newTag);
```

Parameters:
newTag – new field tag.

```
virtual void setType (int newType);
```

Parameters:
newType – new field type, (see TYPES).

```
virtual void setValue(const ELTValue& newValue);

virtual ELTField& operator=(const ELTField& field);
```

Parameters:
field – source field for copying.

ELTFieldGroup

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTFieldGroup.h**

ELTFieldGroup is a collection of **ELTField** objects that provides convenient methods for iterating through fields.

Constructors:

```
ELTFieldGroup(int groupTag = GRPTAG_MSGFIELDS);
```
Creates an empty field group.

```
ELTFieldGroup(const ELTFieldGroup& origGroup);
```
Copy constructor. Performs deep copying. This gives the new group clones of source group fields.

Mutator methods:

```
ELTFieldGroup& operator=(const ELTFieldGroup& fieldGroup);
```
Performs deep copying. This gives the new group clones of source group fields..

```
virtual void setGroupTag(int newGroupTag);
```

Parameters:
newGroupTag – new tag value for the group.

```
virtual void addField(ELTFieldPtr field);
```

Parameters:
field – smart pointer to the added field.

```
virtual void setFieldValue(int fieldTag, ELTValue& fieldValue);
```

Sets specific field value in a group.

Parameters:
fieldTag – tag of changed field.
fieldValue – new field's value.

```
virtual void addGroup(ELTFieldGroupPtr subGroup);
```

Parameters:
subGroup – smart pointer to the added subgroup. The subgroup must have a tag of the field indicating the number of subgroups.

```
virtual void removeField(int fieldTag);
```

Parameters:
fieldTag – tag of the removed field.

```
virtual void removeGroup(int groupTag);
```

Parameters:
groupTag – tag of the removed subgroup. Removes all subgroups with specified tag.

```
virtual void removeGroup(int groupTag, int groupIdx);
```

Parameters:
groupTag – tag of the removed subgroup.
groupIdx – index of the subgroup to be removed from the list of subgroups.

Accessor methods:

```
virtual int getGroupTag() const;
```

Returns: The group's tag.

```
virtual ELTFieldPtr getField(int fieldTag);
```

Parameters:
fieldTag – tag of the requested field.

Returns: Smart pointer to field with specified fieldTag. This smart pointer will point to **NULL** if such field doesn't exist in the group.

```
virtual bool empty() const;
```

Returns: **True** if the group doesn't contain either field or subgroup.

```
virtual const ELTFieldPtr getField(int fieldTag) const;
```

Parameters:
fieldTag – tag of requested field.

Returns:
Smart pointer to field with specified fieldTag. This smart pointer will point to **NULL** if such field doesn't exist in the group.

```
virtual bool isFieldPresent(int fieldTag) const;
```

Returns: **True** if the field with specified tag is present in the group.

```
virtual FieldIteratorPtr fieldIterator();
```

Returns: Smart pointer to the fields iterator of declared as ELTIterator<ELTFieldPtr>. See ***ELTIterator***.

```
virtual const FieldIteratorPtr      fieldIterator() const;
```

Returns: Smart pointer to the fields iteratorof declared as ELTIterator<ELTFieldPtr>. See ***ELTIterator***.

```
virtual int getGroupCount(int groupTag) const;
```

Parameters:
groupTag – tag of the subgroup.

Returns: Number of subgroups with specified groupTag.

```
virtual ELTFieldGroupPtr  getGroup(int groupTag, int groupIdx);
```

Parameters:
groupTag – tag of the subgroup.
groupIdx – index of the subgroup to be returned from the list of subgroups.

Returns: Smart pointer to the subgroup. If group with such tag or at such index doesn't exist the smart pointer will point to the **NULL**.

```
virtual const ELTFieldGroupPtr getGroup(int groupTag, int groupIdx) const;
```

Parameters:
groupTag – tag of the subgroup.
groupIdx – index of the subgroup to be returned from the list of subgroups.

Returns: Smart pointer to the subgroup. If group with such tag or at such index doesn't exist the smart pointer will point to the **NULL**.

```
virtual bool isGroupPresent(int groupTag) const;
```

Returns: **True** if at least one subgroup with such tag presents.

```
virtual GroupIteratorPtr groupIterator(int groupTag);
```

Parameters:
groupTag – tag of requested subgroups.

Returns: Smart pointer to the iterator of all subgroups with specified tag. See ***ELTIterator***.

```
virtual const GroupIteratorPtr groupIterator(int groupTag) const;
```

Parameters:
groupTag – tag of requested subgroups.

Returns: Smart pointer to the iterator of all subgroups with specified tag. See ***ELTIterator***.

```
virtual void clear();
```

Removes all fields and subgroups.

ELTGuaranteedEngineMessageListener

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTGuaranteedEngineMessageListener.h**

ELTMessage

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTMessage.h**

Represents a FIX message.

For now, this is just a wrapper for the field group. Later it will contain a set of convenient methods.

Constructors:

```
ELTMessage();
```
Creates an empty message.

```
ELTMessage(ELTFieldGroupPtr fieldGroup);
```
Creates message from specified field group.

```
virtual void setFieldGroup(const ELTFieldGroupPtr newFieldGroup);
```
Set new field group.

```
virtual ELTFieldGroupPtr getFieldGroup();
```

Returns: Smart pointer to the field group. If it is an empty message it will point to **NULL**.

```
virtual const ELTFieldGroupPtr getFieldGroup() const;
```

Returns: Constant smart pointer to the field group. If it is an empty message it will point to **NULL**.

ELTTime

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTTime.h**

Represents time in milliseconds.

ELTTime is descendant from **tm** structure from standard <time.h>. It contains one additional field (**tm_millis)** to keep milliseconds.

ELTUserTicket

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTUserTicket.h**

User ticket, an inherent part of engine security, is passed as a parameter to every method of the ***IELTEngine*** interface. It is convenient for method-level authorization checks.

Constructors:
```
ELTUserTicket(const std::string& userId, const std::string sessionId);
```
Creates user's ticket with specified user id and session id.

```
virtual const std::string& getUserId() const;
```

Returns: User id.

```
virtual const std::string& getSessionId() const;
```

Returns: Session id.

```
static const ELTUserTicket& getSystemTicket();
```
Helper method, which allows acquisition of a single user ticket global for a given application.

```
static void setSystemTicket(const ELTUserTicket& newSystemTicket);
```
Helper method, which allows setting a single user ticket global for a given application.

ELTValue

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTValue.h**

Universal container for values of different types.

Constructors:
```
ELTValue();
```
Creates an empty value object.

```
ELTValue(const ELTValue& value);
```
Copy constructor.

Available data types:
```
enum TYPES
{
      typeNone,
      typeInteger,
      typeDouble,
      typeAlphaNumeric,
      typeRawDataLen,
      typeRawData,
      typeDate,
      typeTime,
      typeMonthYear,
      typeDayOfMonth,
      typeLong,
      typeBoolean,
      typeXMLData,
      typeQty,
      typePrice,
      typePriceOffset,
      typeAmt,
      typeCurrency,
      typeExchange,
      typeUTCTimestamp,
      typeUTCTimeOnly,
      typeLocalMktDate,
      typeUTCDate,
      typeMultipleValueString,
      typeChar,
      typeCharTimestamp,
      typeBigChar,
};
```

Mutator methods:

```
virtual ELTValue& operator=(const ELTValue& newValue);
```
Sets value from another value object.

```
virtual ELTValue& operator=(const std::string& newValue);
```
Sets value from string.

```
virtual ELTValue& operator=(const char* newValue);
```
Sets value from C string.

```
virtual ELTValue& operator=(const bool& newValue);
```
Sets value from bool value.

```
virtual ELTValue& operator=(const char& newValue);
```
Sets value from char value.

```
virtual ELTValue& operator=(const int& newValue);
```
Sets value from int value.

```
virtual ELTValue& operator=(const long& newValue);
```
Sets value from long value.

```
virtual ELTValue& operator=(const double& newValue);
```
Sets value from double value.

```
virtual void setType(int newType);
```
Sets data type of the value.

```
virtual void clear();
```
Clears the value.

Accessor methods:

```
virtual int getType() const;
```

Returns: Data type of the value.

```
virtual const std::string& getStringValue()     const;
```

Returns: String value of the object.

```
virtual bool getBoolValue() const;
```

Returns: bool value of the object.

```
virtual char getCharValue() const;
```

Returns: char value of the object.

```
virtual int getIntValue() const;
```

Returns: int value of the object.

```
virtual long getLongValue() const;
```

Returns: Long value of the object.

```
virtual double getDoubleValue() const;
```

Returns: Double value of the object.

```
virtual bool isNull() const;
```

Returns: **True** if this is an empty value.

IELTEngine

Library: **eltraderEngine.lib**
Header: **eltraderEngine/IELTEngine.h**

The main **El'Trader FIX Engine** interface. For descriptions of the all thrown exceptions, see

Exceptions below.

```
virtual void connect(
      const ELTUserTicket& userTicket,
      const std::string& connectionName)
      throw (
            ELTEngineUnknownRemoteException,
            ELTInvalidLogonException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Establishes a connection between two parties as represented by connectionName.

```
virtual void connect(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTFieldGroupPtr extraFields)
      throw (
            ELTEngineUnknownRemoteException,
            ELTInvalidLogonException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Establishes a connection between two parties as represented by connectionName with extra parameters.

```
virtual void disconnect(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const std::string& logoutText)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Closes the connection between two parties as represented by connectionName.

```
virtual void resetConnection(
      const ELTUserTicket& userTicket,
      const std::string& connectionName)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException,
            ELTNotConnectedException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Reset a connection to start message exchange with initial sequence numbers.

```
virtual bool pingConnection(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      long timeout)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Sends a test request message on given connection.

```
virtual bool isConnected(
      const ELTUserTicket& userTicket,
      const std::string& connectionName)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Check if session represented by connectionName is active.

```
virtual void subscribe(
      const ELTUserTicket& userTicket,
      IELTEventListenerPtr subscriber,
      const std::string& connectionName,
      const std::string& eventType)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException,
            ELTInvalidConfigPropertiesException) = 0;
```

WARNING: this method is deprecated. Use subscribeDirect() method instead.
Subscribes event listener for events of specified event type from given connection.

```
virtual void subscribeDirect(
      const ELTUserTicket& userTicket,
      const STDNAMESPACE string& connectionNameList,
      bool forAppMessage,
      bool forAdminMessage,
      bool forStateEvents,
      IELTEventListenerPtr subscriber)
            throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException) = 0;
```

Subscribes event listener for events of specified event type from given connection.

```
virtual void unsubscribe(
      const ELTUserTicket& userTicket,
      IELTEventListenerPtr subscriber,
      const std::string& connectionName,
      const std::string& eventType)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException,
            ELTInvalidConfigPropertiesException) = 0;
```

WARNING: this method is deprecated. Use unsubscribeDirect() method instead.
Un-subscribes event listener for events of specified event type from given connection.

```
virtual void unsubscribeDirect(
      const ELTUserTicket& userTicket,
      const STDNAMESPACE string& connectionNameList,
      bool fromAppMessage,
      bool fromAdminMessage,
      bool fromStateEvents,
      IELTEventListenerPtr subscriber)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException) = 0;
```

Un-subscribes event listener for events of specified event type from given connection.

```
virtual void destroyDirectSubscriber(
      IELTEventListenerPtr subscriber) = 0;
```

This method is useful if application needs to guarantee that engine stop receiving events for that particular subscriber. Even if subscriber was subscribed/unsubscribed the same number of times there are possibilities that engine core will still keep receiving events for it. To stop it this method should be invoked.
Typically is not used.

```
virtual void sendAdminMessage(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTMessagePtr message)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTInvalidMessageException,
            ELTNotConnectedException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Sends administrative message to the connection.

```
virtual void sendAdminMessage(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const std::string& message)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTInvalidMessageException,
            ELTNotConnectedException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Sends an administrative message represented by FIX string on given connection.

```
virtual long resendMessages(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTTime& startingTime = -1,
      const ELTTime& endingTime = -1,
      const ELTMessageBoolExpr& filter = "")
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTNotConnectedException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Resends messages that were sent on given connection between startingDate and endingDate.

```
virtual ELTMessageVector& retrieveMessagesByDateRange(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      ELTMessageVector& resultsHolder,
      const ELTTime& startingTime = -1,
      const ELTTime& endingTime = -1,
      const ELTMessageBoolExpr& filter = "")
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Retrieves messages, which were sent between specified time boundaries.

```
virtual ELTMessageVector& retrieveMessagesByDateRange(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      ELTMessageVector& resultsHolder,
      long sentRecievedFlag,
      const ELTTime& startingTime = -1,
      const ELTTime& endingTime = -1,
      const ELTMessageBoolExpr& filter = "")
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Retrieves messages, which were sent and/or received between specified time boundaries. See enumeration ***SendReceivedValues*** in ***ELTEngineDefs***.

```
virtual ELTMessageVector& retrieveMessagesByIdRange(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      ELTMessageVector& resultsHolder,
      long sentRecievedFlag,
      long lowerBoundMessageId,
      long upperBoundMessageId,
      const ELTMessageBoolExpr& filter = "")
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Retrieves sent and/or received messages, which have message id between specified boundaries. See enumeration ***SendReceivedValues*** in ELTEngineDefs.

```
virtual ELTMessageVector& retrieveMessagesByIdSet(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      ELTMessageVector& resultsHolder,
      long sentRecievedFlag,
      const ELTMessageIdVector& messageIds,
      const ELTMessageBoolExpr& filter = "")
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Retrieves sent and/or received messages with specified message ids. See enumeration ***SendReceivedValues*** in ***ELTEngineDefs***.

```
virtual ELTMessageVector& retrieveMessagesByQuery(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      ELTMessageVector& resultsHolder,
      long sentRecievedFlag,
      const ELTMessageBoolExpr& logQuery,
      const ELTMessageBoolExpr& filter = "")
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Retrieves sent and/or received messages, which satisfy the query. See enumeration ***SendReceivedValues*** in ***ELTEngineDefs***.

```
virtual ELTMessageVector& retrieveMessageChain(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      ELTMessageVector& resultsHolder,
      const ELTFieldTagVector& fieldTags,
      const std::vector<std::string>& initialValues)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

```
virtual ELTMessageVector& retrieveMessagesForResend(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      ELTMessageVector& resultsHolder,
      long beginSeqNum,
      long endSeqNum,
      const ELTMessageBoolExpr& filter = "")
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;


virtual const std::string retrieveMessageLogData(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      long sentReceivedFlag,
      const std::string& selectClause,
      const std::string& whereClause,
      const std::string& advancedClauses)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;


virtual long updateMessageLogData(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      long sentRecievedFlag,
      const std::string& setClause,
      const std::string& whereClause)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;


virtual long insertMessageLogData(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTMessageVector& messages)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
```

```
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;


virtual long deleteMessageLogData(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      long sentRecievedFlag,
      const std::string& whereClause)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;


virtual const ELTConnectionState getConnectionState(
      const ELTUserTicket& userTicket,
      const std::string& connectionName)
      throw (
            ELTEngineUnknownRemoteException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Returns: Current state of given connection.

```
virtual ELTMessageIdVector getMessageIds(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      long seqNum,
      bool excludePossDups)
      throw (
            ELTEngineUnknownRemoteException,
            ELTPersistentStorageException,
            ELTSecurityException,
            ELTConnectionUnavailableException,
            ELTConnectionUnknownException) = 0;
```

Returns: IDs of messages with specified sequence number excluding or including possible duplicates.

```
virtual ELTXmlString getConnectionProperties(
      const ELTUserTicket& userTicket,
      const std::string& connectionName) = 0;
```

Returns: XML formatted string with properties of given connection.

```
virtual void sendMessage(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTMessagePtr message)
      throw (
            ELTEngineUnknownRemoteException,
            ELTConnectionUnknownException,
            ELTConnectionUnavailableException,
            ELTSecurityException,ELTNotConnectedException,
            ELTInvalidMessageException,
            ELTPersistentStorageException) = 0;
```

Sends message on given connection.

```
virtual void sendMessage(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const std::string& message)
      throw (
            ELTEngineUnknownRemoteException,
            ELTConnectionUnknownException,
            ELTConnectionUnavailableException,
            ELTSecurityException,
            ELTNotConnectedException,
            ELTInvalidMessageException,
            ELTPersistentStorageException) = 0;
```

Sends message represented by FIX string on given connection.

```
virtual void sendMultipleMessages(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTMessageVector& messages)
      throw (
            ELTEngineUnknownRemoteException,
            ELTConnectionUnknownException,
            ELTConnectionUnavailableException,
            ELTSecurityException,
            ELTNotConnectedException,
            ELTInvalidMessageException,
            ELTPersistentStorageException) = 0;
```

Sends multiple messages on given connection.

```
virtual void postInternalMessage(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTMessagePtr message)
      throw(
            ELTEngineUnknownRemoteException,
            ELTConnectionUnknownException,
            ELTConnectionUnavailableException,
            ELTSecurityException,
            ELTInvalidMessageException,
            ELTPersistentStorageException) = 0;
```

Posts message on given connection.

```
virtual void postInternalMessage(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const std::string& message)
      throw (
            ELTEngineUnknownRemoteException,
            ELTConnectionUnknownException,
            ELTConnectionUnavailableException,
            ELTSecurityException,
            ELTInvalidMessageException,
            ELTPersistentStorageException) = 0;
```

Posts message represented by FIX string on given connection.

```
virtual void postMultipleInternalMessages(
      const ELTUserTicket& userTicket,
      const std::string& connectionName,
      const ELTMessageVector& messages)
      throw (
            ELTEngineUnknownRemoteException,
            ELTConnectionUnknownException,
            ELTConnectionUnavailableException,
            ELTSecurityException,
            ELTInvalidMessageException,
            ELTPersistentStorageException) = 0;
```

Posts multiple messages on given connection.

IELTEngineAdapter

Library: **eltraderEngine.lib**
Header: **eltraderEngine/IELTEngineAdapter.h**

El'Trader Engine's middleware adapters implement this interface. Provides basic functionality for all concrete adapters.

Constructor:
```
IELTEngineAdapter(const IELTEngineConfigProviderPtr configProvider)
      throw (ELTWrongConfigurationException);
```
Creates the adapter with provided configuration.

Methods:
```
virtual const IELTEngineConfigProvider& getConfigProvider() const;
```
Returns: Current adapter's configuration.

```
virtual void connectAdapter() throw (ELTCannotConnectException) = 0;
```
Connects adapter to designated middleware.

```
virtual void disconnectAdapter() throw () = 0;
```
Disconnects adapter from designated middleware.

IELTEngineAdapterFactory

Library: **eltraderEngine.lib**
Header: **eltraderEngine/IELTEngineAdapterFactory.h**

Implementers of this interface are responsible for designated adapter creation.
For example, **CORBA** knows how to create an adapter for connecting to a remote engine using either object names, or object **IORs**. An implementer might perform extra initializations required for the environment of the particular adapter.

Methods:

```
virtual IELTEngineAdapterPtr createAdapter(
      const IELTEngineConfigProviderPtr configProvider) const
      throw (
            ELTWrongConfigurationException,
            ELTAdapterCreationException) = 0;
```
Creates new adapter with provided configuration.

IELTEngineConfigProvider

Library: **eltraderEngine.lib**
Header: **eltraderEngine/IELTEngineConfigProvider.h**

The engine's adapter of a given kind can have its own configuration parameters. This interface generalizes an adapter configuration procedure during its creation. Each adapter has an attribute representing its kind, which is reflected by the single method in this interface.

As an implementer of this interface, a concrete configuration provider is responsible for providing all necessary configuration information for a particular adapter. For example, for a **CORBA** adapter, it could be a remote engine adapter's name (if a naming service is used), or a remote engine adapter's **IOR.** For a **SOCKET** adapter, it could be the remote engine adapter's host address and port.

Methods:

```
virtual const std::string& getAdapterKind() const = 0;
```

Returns: Type of given adapter.

IELTEventListener

Library: **eltraderEngine.lib**
Header: **eltraderEngine/IELTEventListener.h**

This interface's implementers can subscribe to the engine for the event flow on given connections.

```
virtual void processEvent(const ELTEventPtr event) = 0;
```
This method is invoked by the engine upon the occurrence of an event, which a given subscriber registered for (see ***IELTEngine.subscribe***.)

```
virtual void processStateEvent(const ELTConnectionStateEventPtr event) {};
```
This method is invoked by the engine upon the occurrence of an state event, which a given subscriber registered for (see ***IELTEngine.subscribe***.)

Utility Classes

ELTEngineDefs

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTEngineDefs.h**

Defines common constants and enumeration types.

```
enum SendReceivedValues
{
      SENT_ONLY         = 0,
      RECEIVED_ONLY     = 1,
      SENT_AND_RECEIVED = 2
};
enum MessagIdValues
{
      MAX_RESENT_MESSAGE_ID,
      INITIAL_RESENT_MESSAGE_ID,
      MAX_ADMIN_MESSAGE_FROM_COUNTER_PARTY_ID  ,
      INITIAL_ADMIN_MESSAGE_FROM_COUNTER_PARTY_ID,
      MAX_ADMIN_MESSAGE_TO_COUNTER_PARTY_ID,
      INITIAL_ADMIN_MESSAGE_TO_COUNTER_PARTY_ID,
      MAX_APP_MESSAGE_FROM_COUNTER_PARTY_ID,
      INITIAL_APP_MESSAGE_FROM_COUNTER_PARTY_ID,
      MAX_APP_MESSAGE_TO_COUNTER_PARTY_ID,
      INITIAL_APP_MESSAGE_TO_COUNTER_PARTY_ID
};
enum  CONNECTION_STATE
{
      CONNECTION_STATE_STARTING,
      CONNECTION_STATE_STOPPING,
      CONNECTION_STATE_LOGON_IN_PROGRESS,
      CONNECTION_STATE_LOGOUT_IN_PROGRESS,
      CONNECTION_STATE_EXPLICIT_LOGOUT,
      CONNECTION_STATE_LINK_ACTIVE,
      CONNECTION_STATE_SESSION_ACTIVE,
      CONNECTION_STATE_MESSAGING_IDLE,
      CONNECTION_STATE_HAS_DISCONNECTED,
      CONNECTION_STATE_TIMED_OUT,
      CONNECTION_STATE_BROKEN_PIPE,
      CONNECTION_STATE_DB_ERROR,
      CONNECTION_STATE_CRYPT_ERROR,
      CONNECTION_STATE_SERVER_UNREACHABLE,
      CONNECTION_STATE_SERVER_UNAVAILABLE,
};
static const std::string APP_EVENT_TYPE;
static const std::string ADMIN_EVENT_TYPE;
```

counted_ptr

Library: **eltraderEngine.lib**
Header: **eltraderEngine/counted_ptr.h**

This template class represents a "**smart**" **pointer**, which alleviates memory allocation and de-allocation issues. It automatically counts the number of object references, and destroys the object only when the count becomes equal to 0.

ELTIterator

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTIterator.h**

This class differs from STL iterators and allows iteration through sequences of big objects. A common way to determine the end of an STL sequence is by checking two objects for equality. For sequences of heavy objects (e.g. instances of **ELTFieldGroup** with many fields), **STL** iterators become inefficient.

```
template <class X> class ELTRADERENGINE_API ELTIterator
{
public:
    typedef X  element_type;
    typedef X& reference_type;
    typedef const X& const_reference_type;

    ELTIterator()
    {
    }

    virtual ~ELTIterator()
    {
    }

    virtual bool hasNext()       const = 0;
    virtual void toNext()        const = 0;
    virtual const_reference_type    getCurrent()    const = 0;
    virtual reference_type          getCurrent()          = 0;
};
```

Typical usage of it is:

```
    while (Iterator ->hasNext()) {
        DoSomething(Iterator->getCurrent());
        Iterator ->toNext();
    }
```

UtilityFunctions

Library: **infra.lib**
Header: **infra/UtilityFunctions.h**

```
static STDNAMESPACE string& longToString(
      UniLong i,
      STDNAMESPACE string& destination);
```

Converts long values to string and puts it into a provided destination buffer.

```
static UniLong stringToLong(
      const char* str);
```

Converts string to long value.
Returns: 0 in case if it's impossible to convert the provided string to the long.

```
static UniDouble timeToDouble(
      const INFRANAMESPACE Time& source);
```

Convenience method for converting time objects to doubles.

```
static INFRANAMESPACE Time& doubleToTime(
      UniDouble source,
      INFRANAMESPACE Time& destination);
```

Convenience method for converting doubles to time objects.

```
static STDNAMESPACE string& timeToString(
      const STDNAMESPACE string& format,
      const INFRANAMESPACE Time& eltTime,
      STDNAMESPACE string& destination);
```

Convenience method for converting time objects to strings.
Uses such format symbols:
%Y - year
%m - month
%d - day of month
%H - hour
%M - minute
%S - second
%s - milli second

```
static INFRANAMESPACE Time& stringToTime(
      const STDNAMESPACE string& format,
      const STDNAMESPACE string& stringValue,
      INFRANAMESPACE Time& destination,
      STDNAMESPACE string::size_type &endPos)
      throw (InfraException);
```

Converts string to time according to format.
Format should be expressed as described in the method timeToString().
After executing 'endPos' will contain index of the last processed character from the string.

```
static INFRANAMESPACE Time& timeLocalToGMT(
      const INFRANAMESPACE Time& local,
      INFRANAMESPACE Time& gmtDestination);
```

Converts local time into GMT.

```
static INFRANAMESPACE Time& timeGMTToLocal(
      const INFRANAMESPACE Time& gmt,
      INFRANAMESPACE Time& localDestination);
```

Converts GMT time to local.

Small example how it can be used:

```
ELTFieldGroupPtr fg = msg->getFieldGroup();
ELTFieldPtr field = fg->getField(ELT.fieldTimestamp);
UniDouble ms = field->getValue()->getDoulbeValue();
Time t;
UtilityFunctions.doubleToTime(ms, t);
STDNAMESPACE cout << "Year: " << t.tm_year;
STDNAMESPACE cout << " Month: " << t.tm_mon;
STDNAMESPACE cout << " Day: " << t.tm_mday << STDNAMESPACE endl;
```

Exceptions

ELTEngineException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTEngineException.h**

Base class for all exceptions that could be thrown by the engine.
Derived from **STL's std::exception**.

Constructors:

```
ELTEngineException();
```
Creates empty exception.

```
ELTEngineException(const std::string& message);
```
Creates exception with specified message.

Accessor methods:

```
virtual const std::string& getMessage() const;
```

Returns: Description of the exception.

```
virtual __exString what() const;
```
Derived method from std::exception. Returns description of the exception.

ELTEngineRemoteException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTEngineRemoteException.h**

Derived from ***ELTEngineException***.

Base class for all exceptions that could be received from a remote engine.
Provides methods, which allow bind C++ exception to the remote Java exception.

Accessor methods:

```
virtual const std::string& getType() const;
```

Returns: Type of the remote Java exception.

```
virtual void throwInstance (const std::string& message) = 0;
```

Throws instance of descendant exception.

ELTEngineUnknownRemoteException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTConnectionUnknownException.h**

Derived from ELTEngineRemoteException.

Thrown when the application receives a remote exception of an unknown type.

ELTAdapterCreationException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTAdapterCreationException.h**

Derived from ***ELTEngineException***.

Thrown when there is an engine adapter creation problem.

ELTCannotConnectException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTCannotConnectException.h**

Derived from ***ELTEngineException***.

Thrown when the adapter has a problem connecting to the middleware.

ELTConnectionUnavailableException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTConnectionUnavailableException.h**

Derived from ELTEngineRemoteException.

Thrown when an application tries to perform an operation on the remote FIX connection that is unreachable from the application's process space.

ELTConnectionUnknownException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTConnectionUnknownException.h**

Derived from ELTEngineRemoteException.

Thrown when an application tries to perform operation on a FIX connection with an unknown name.

ELTInvalidConfigPropertiesException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTInvalidConfigPropertiesException.h**

Derived from ***ELTEngineException***.

Thrown when there is a configuration problem.

ELTInvalidLogonException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTInvalidLogonException.h**

Derived from ELTEngineRemoteException.

Thrown when there is a FIX logon sequence problem.

ELTInvalidMessageException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTInvalidMessageException.h**

Derived from ELTEngineRemoteException.

Thrown when there is a message format problem.

ELTNotConnectedException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTNotConnectedException.h**

Derived from ELTEngineRemoteException.

Thrown when application tries to perform operation on a FIX connection that is not connected to the counter-party.

ELTPersistentStorageException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTPersistentStorageException.h**

Derived from ELTEngineRemoteException..

Thrown when there is a message persistence problem.

ELTSecurityException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTSecurityException.h**

Never thrown. Reserved for future use.

ELTWrongConfigurationException

Library: **eltraderEngine.lib**
Header: **eltraderEngine/ELTWrongConfigurationException.h**

Derived from ***ELTEngineException***.

Thrown when there is a configuration problem.