N_3053 : ['StockCode', 'Direction', 'OrderPrice', 'OrderCommand']:

Block:34756 line:821~822
```
int32_t CETFOrderInsertFlow::InsertRiskEntrust()
```

Block:34758 line:822~827
```
{
 // return 0;
 CSerialInfo* pSerial = lpContext->m_pDataImpl->GetSerialInfo();
 uint32_t nRiskEntrustNo = 0;
```

Block:34759 line:827~828
```
 if (!  false  )
```
T F

Block:34760 line:828~832
```
 {
  nRiskEntrus                ntext->GetRiskSerialNo();
  m_stPublicContext.nRiskEntrustNo = nRiskEntrustNo;
 }
```

Block:34761 line:832~833
```
 else
```

Block:34762 line:833~835
```
 {
  nRiskEntrustNo = m_stPublicContext.nRiskEntrustNo;
 }
```

Block:34763 line:835~860
```
 INFO_LOG(m_lpContext, "生成风控委托序号.", wrap("RiskEntrustNo", m_stPublicContext.nRiskEntrustNo));
 //lpContext->m_pRiskContext->m_pRiskData->m_iRiskEntrustNo = nRiskEntrustNo;
 CRiskEntrust* pRiskEntrust = lpContext->m_pDataImpl->NewRiskEntrust(pSerial, nRiskEntrustNo);
 CHECK_OBJ_ADD_FAIL(pRiskEntrust, m_nErrorNo, "新增委托记录失败",
 "EntrustNo:", wrap(nRiskEntrustNo),
 "EntrustReference:", wrap(m_stPublicContext.uiEntrustReference));
 pRiskEntrust->m_stEntField.RiskEntrustNo = nRiskEntrustNo;
 pRiskEntrust->m_stEntField.EntrustBs = m_stPublicContext.cEntrustDirection;
 pRiskEntrust->m_stEntField.EntrustPrice = 0; //m_stPublicContext.nOrderPrice;
 pRiskEntrust->m_stEntField.OrderCommand = 1; //m_stPublicContext.nOrderCommand;
 pRiskEntrust->m_stEntField.ContraOrderType = 0;
 //pRiskEntrust->m_stEntField.StockCode = m_stPublicContext.pStockCode->m_stCodeField.StockCode;
 pRiskEntrust->m_stEntField.OrderBalance = 0; //m_stPublicContext.nEntrustBalance;
 //pRiskEntrust->m_stEntField.OrderCommission = m_stPublicContext.nCommission;
 //pRiskEntrust->m_stEntField.OrderCommissionLeft = m_stPublicContext.nCommission;
 //只生成挂单委托,不插入挂单列表. 原因是目前风控更新F类流程与挂单对象强耦合,需要使用到挂单委托里的相关字段,待后续优化解耦
 m_stPublicContext.pRiskEntrust = pRiskEntrust;
 return ERR_OK;
```

Block:34764 line:860~860
```
}
```

N_2775 : ['StockCode', 'SeatNo', 'Direction']:

Block:29273 line:2030~2031
```
int32_t CNewStockOrderInsertFlow::UpdateHoldReal()
```

Block:29275 line:2031~2036
```
{
 //批量报单存在一种情况：第一笔无证券代码持仓，创建该证券代码持仓；由于批量数据存在缓存数组中，无法矫正，所以不能用数组中缓存的持仓指针。
 CStockHoldReal* pHoldReal = m_stStockPublicContext.pCombi->GetHoldRealByCode(m_stStockPublicContext.nStockCode, m_stStockPublicContext.nExchIndex,
 m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stStockPublicContext.SeatIndex);
```

Block:29276 line:2036~2037
```
if (  unlikely(pHoldReal == nullptr)  )
```
T F

Block:29277 line:2037~2064
```
 {
  pHoldReal = m_stStockPublicContext.pCombi->NewStockHoldReal(m_stStockPublicContext.pStockHolder,m_stStockPublicContext.nStockCode,m_stStockPublicContext.nExchIndex,
  m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stStockPublicContext.SeatIndex);
  CHECK_BIZ_OBJ_ADD_FAIL(pHoldReal, m_nErrorNo, "新增持仓记录失败",
  "CombiID:", wrap(m_stStockPublicContext.pCombi->m_stCombiField.CombiID),
  "StockCode:", wrap(m_stStockPublicContext.nStockCode));
  pHoldReal->m_stHoldReal.AssetID = m_stStockPublicContext.pCombi->m_stCombiField.AssetID;
  pHoldReal->m_stHoldReal.CombiID = m_stStockPublicContext.pCombi->m_stCombiField.CombiID;
  memcpy(pHoldReal->m_stHoldReal.ExchangeType, m_stStockPublicContext.pStockCode->m_stCodeField.ExchangeType, sizeof(pHoldReal->m_stHoldReal.ExchangeType));
  pHoldReal->m_stHoldReal.StockCode = m_stStockPublicContext.pStockCode->m_stCodeField.StockCode;
  memcpy(pHoldReal->m_stHoldReal.StockAccount, m_stStockPublicContext.pStockHolder->m_stStockHolder.StockAccount, sizeof(pHoldReal->m_stHoldReal.StockAccount));
  pHoldReal->m_stHoldReal.StockHoldIndex = m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex;
  pHoldReal->m_stHoldReal.BeginAmount = 0;
  pHoldReal->m_stHoldReal.CurrentAmount = 0;
  pHoldReal->m_stHoldReal.TemporaryVolume = 0;
  pHoldReal->m_stHoldReal.BeginCarryoverCost = 0;
  strncpy(pHoldReal->m_stHoldReal.BindSeat, m_stStockPublicContext.pPositionSeat->m_stSeatField.SeatNo, sizeof(pHoldReal->m_stHoldReal.BindSeat) - 1);
  pHoldReal->m_stHoldReal.BindSeat[sizeof(pHoldReal->m_stHoldReal.BindSeat) - 1] = '\0';
  pHoldReal->m_stHoldReal.SeatIndex = m_stStockPublicContext.SeatIndex;
  pHoldReal->m_pExcharg = m_stStockPublicContext.pExcharg;
  pHoldReal->m_pStockCode = m_stStockPublicContext.pStockCode;
  m_stStockPublicContext.pStockHold = pHoldReal;
  m_stStockPublicContext.cHoldRealDealType = CNST_DEALTYPE_ADD;
 }
```

Block:29278 line:2064~2065
```
 else
```

Block:29279 line:2065~2068
```
 {
  m_stStockPublicContext.pStockHold = pHoldReal;
  m_stStockPublicContext.cHoldRealDealType = CNST_DEALTYPE_MOD;
 }
```

Block:29280 line:2068~2086
```
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(this->m_lpContext->GetThreadNo(), 115);
 //更新持仓
 UpdateHoldEnableValue<CStockHoldReal>(lpContext, pHoldReal, pHoldReal->m_stHoldReal.ExchangeType,
 pHoldReal->m_pStockCode->m_stCodeField.StockType,
 pHoldReal->m_pStockCode->m_stCodeField.HzjyFlag,
 m_stStockPublicContext.cEntrustDirection,
 CNST_BUSINPROTYPE_ENTRUST,
 m_stStockPublicContext.nFrozenVolume);
 INFO_LOG(lpContext,"持仓更新UpdateHoldReal",wrap("AssetID",pHoldReal->m_stHoldReal.AssetID),
 wrap("CombiID", pHoldReal->m_stHoldReal.CombiID),
 wrap("ExchangeType", pHoldReal->m_stHoldReal.ExchangeType),
 wrap("StockCode", pHoldReal->m_stHoldReal.StockCode),
 wrap("BeginAmount", pHoldReal->m_stHoldReal.BeginAmount),
 wrap("CurrentAmount", pHoldReal->m_stHoldReal.CurrentAmount),
 wrap("StockAccount", pHoldReal->m_stHoldReal.StockAccount));
 return m_nErrorNo;
```

Block:29281 line:2086~2086
```
}
```

N_2774 : ['Direction']:

Block:29269 line:2014~2015
```
int32_t CNewStockOrderInsertFlow::UpdateAssetday()
```

Block:29271 line:2015~2023
```
{
 //pstOrderField->pAssetday->EnableBalance -= pstOrderField->nFrozenBalance;
 m_nErrorNo = UpdateCashEnableValue(lpContext, m_stStockPublicContext.pAsset, m_stStockPublicContext.nExchIndex, CNST_STOCKKIND_STOCK_INDEX, m_stStockPublicContext.cEntrustDirection,
 CNST_BUSINPROTYPE_ENTRUST, CNST_CASH_TYPE_NO_DISTINCTION, m_stStockPublicContext.nFrozenBalance);
 m_stStockPublicContext.cAssetdayDealType = CNST_DEALTYPE_MOD;
 return m_nErrorNo;
```

Block:29272 line:2023~2023
```
}
```

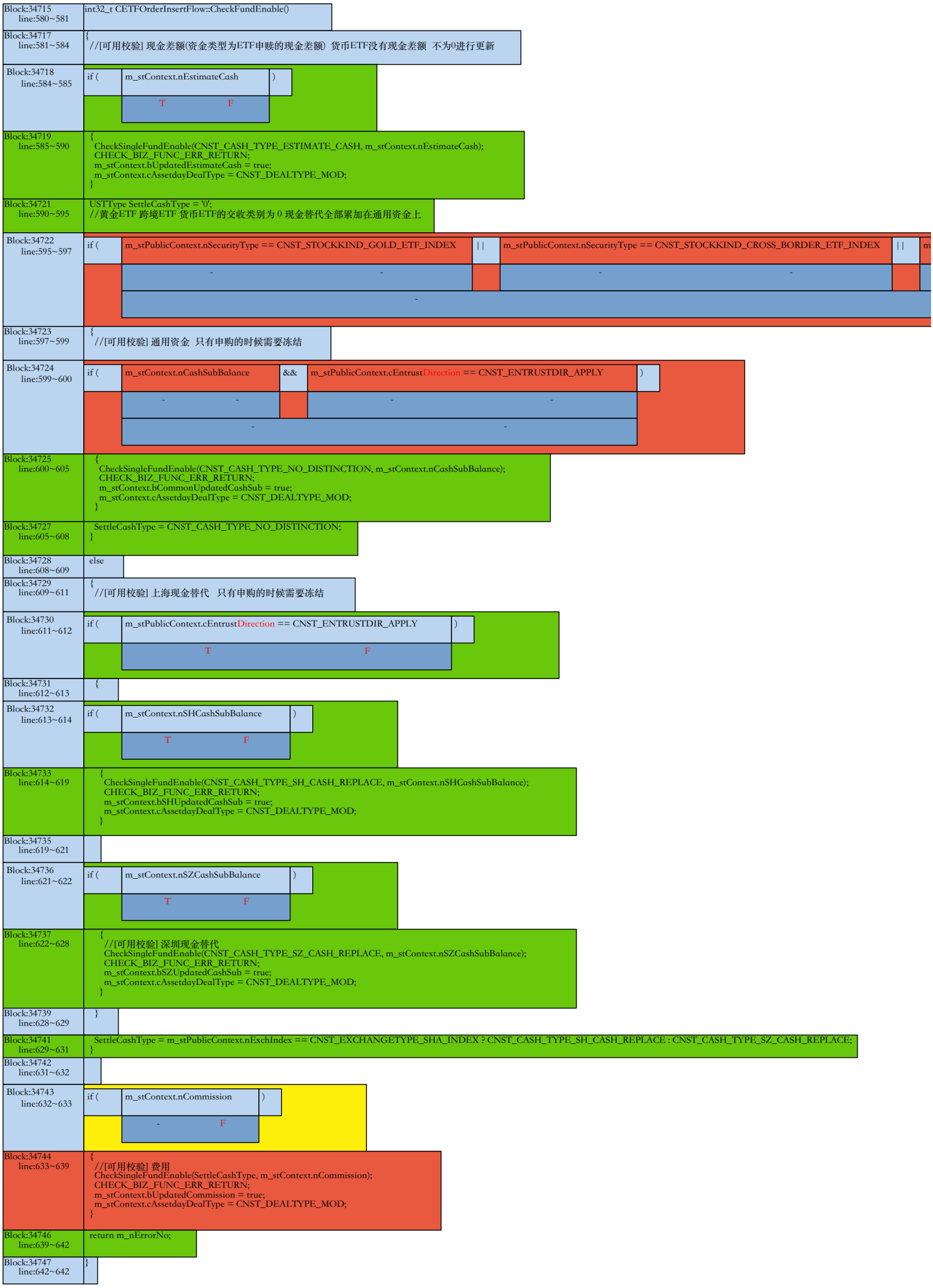
N_39 : ['OrderCommand']:

```
Block:697 line:1321~1322   int32_t CHeckPriceType(CStockBizContext* lpContext, USTExchangeIndex nExchIndex, HSOrderCommand OrderCommand, USTType StockProperty,USTType cFlagRegistra)
Block:699 line:1322~1323   {
Block:700 line:1323~1324   if ( nExchIndex == CNST_EXCHANGETYPE_SHA_INDEX )
                             T                F
Block:701 line:1324~1326     {
                               //上海科创板：1-限价 13-五档市价即时成交剩余撤销 14-五档市价即时成交剩余转限价 18-本方最优价转限价 19-对手方最优价申报
Block:702 line:1326~1327     if( CNST_STOCK_SCIENTIFIC_BOARD == StockProperty )
                               T                F
Block:703 line:1327~1328     {
Block:704 line:1328~1333     if ( unlikely( (OrderCommand != CNST_UST_ORDERCOMMAND_LIMIT) &&  (OrderCommand != CNST_UST_ORDERCOMMAND_MARKET5FAK) &&  (OrderCommand != CNS
                                                                                            T
Block:705 line:1333~1337       {
                                 AMUST_LOGBIZERR_RETURN(lpContext, ERR_NOT_SUPPORT_CURRENT_ORDER_COMMAND_TYPE,  wrap("ExchangeType", GetExchangeType(nExchIndex)),
                                 wrap("StockProperty", StockProperty), wrap("OrderCommand", OrderCommand));
                                 return lpContext->m_nErrorNo;
                               }
Block:707 line:1337~1340     }
                               //上海主板：1-限价 13-五档市价即时成交剩余撤销 14-五档市价即时成交剩余转限价 18-本方最优价转限价 19-对手方最优价申报
Block:708 line:1340~1341     else
Block:709 line:1341~1342     {
Block:710 line:1342~1346     if ( unlikely((OrderCommand != CNST_UST_ORDERCOMMAND_LIMIT) &&  (OrderCommand != CNST_UST_ORDERCOMMAND_MARKET5FAK) &&  (OrderCommand != CNS
                                                                                            -
Block:711 line:1346~1350       {
                                 AMUST_LOGBIZERR_RETURN(lpContext, ERR_NOT_SUPPORT_CURRENT_ORDER_COMMAND_TYPE,  wrap("ExchangeType", GetExchangeType(nExchIndex)),
                                 wrap("StockProperty", StockProperty), wrap("OrderCommand", OrderCommand));
                                 return lpContext->m_nErrorNo;
                               }
Block:713 line:1350~1351     }
Block:714 line:1351~1354     }
                               //深圳主板、创业板：1-限价 7-市价即时全部成交否则撤销 8-市价任意数量即时成交剩余撤销 13-五档市价即时成交剩余撤销 18-本方最优价转限价 19-对手方最优价申报
Block:717 line:1354~1355     else if ( nExchIndex == CNST_EXCHANGETYPE_SZA_INDEX )
716                                  T                -
715
Block:718 line:1355~1356     {
Block:719 line:1356~1362     if ( unlikely((OrderCommand != CNST_UST_ORDERCOMMAND_LIMIT) &&  (OrderCommand != CNST_UST_ORDERCOMMAND_MARKETFOK) &&  (OrderCommand != CNST_UST
                                                                                            T
Block:720 line:1362~1367       {
                                 AMUST_LOGBIZERR_RETURN(lpContext, ERR_NOT_SUPPORT_CURRENT_ORDER_COMMAND_TYPE,  wrap("ExchangeType", GetExchangeType(nExchIndex)),
                                 wrap("StockProperty", StockProperty), wrap("OrderCommand", OrderCommand));
                                 return lpContext->m_nErrorNo;
                               }
Block:722 line:1367~1369     }
Block:723 line:1369~1370     else
Block:724 line:1370~1374     {
                               AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_IN_PARAM_FAIL, wrap("ExchangeType", GetExchangeType(nExchIndex)),
                               wrap("OrderCommand", OrderCommand));
                               return lpContext->m_nErrorNo;
                             }
Block:726 line:1374~1377     return ERR_OK;
Block:727 line:1377~1377   }
```

```
Block:10735 line:1010~1011   CSeat* GetSeat(const string& seatNo, const string& exchangeType) const
Block:10737 line:1011~1014   {
                               string id(seatNo + COLON_SEPERATOR + exchangeType );
                               return GetObj(m_mapSeat, id);
Block:10738 line:1014~1014   }
```

N_1445 : ['SeatNo']:

N_3050 : ['Direction']:


Block:34715 line:580~581
int32_t CETFOrderInsertFlow::CheckFundEnable()
Block:34717 line:581~584
{
//[可用校验] 现金差额(资金类型为ETF申赎的现金差额) 货币ETF没有现金差额 不为0进行更新
Block:34718 line:584~585
if (m_stContext.nEstimateCash)
T F
Block:34719 line:585~590
{
CheckSingleFundEnable(CNST_CASH_TYPE_ESTIMATE_CASH, m_stContext.nEstimateCash);
CHECK_BIZ_FUNC_ERR_RETURN;
m_stContext.bUpdatedEstimateCash = true;
m_stContext.cAssetdayDealType = CNST_DEALTYPE_MOD;
}
Block:34721 line:590~595
USTType SettleCashType = '0';
//黄金ETF 跨境ETF 货币ETF的交收类别为 0 现金替代全部累加在通用资金上
Block:34722 line:595~597
if (m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX ||
Block:34723 line:597~599
{
//[可用校验] 通用资金 只有申购的时候需要冻结
Block:34724 line:599~600
if (m_stContext.nCashSubBalance && m_stPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_APPLY)
Block:34725 line:600~605
{
CheckSingleFundEnable(CNST_CASH_TYPE_NO_DISTINCTION, m_stContext.nCashSubBalance);
CHECK_BIZ_FUNC_ERR_RETURN;
m_stContext.bCommonUpdatedCashSub = true;
m_stContext.cAssetdayDealType = CNST_DEALTYPE_MOD;
}
Block:34727 line:605~608
SettleCashType = CNST_CASH_TYPE_NO_DISTINCTION;
}
Block:34728 line:608~609
else
Block:34729 line:609~611
{
//[可用校验] 上海现金替代 只有申购的时候需要冻结
Block:34730 line:611~612
if (m_stPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_APPLY)
T F
Block:34731 line:612~613
{
Block:34732 line:613~614
if (m_stContext.nSHCashSubBalance)
T F
Block:34733 line:614~619
{
CheckSingleFundEnable(CNST_CASH_TYPE_SH_CASH_REPLACE, m_stContext.nSHCashSubBalance);
CHECK_BIZ_FUNC_ERR_RETURN;
m_stContext.bSHUpdatedCashSub = true;
m_stContext.cAssetdayDealType = CNST_DEALTYPE_MOD;
}
Block:34735 line:619~621
Block:34736 line:621~622
if (m_stContext.nSZCashSubBalance)
T F
Block:34737 line:622~628
{
//[可用校验] 深圳现金替代
CheckSingleFundEnable(CNST_CASH_TYPE_SZ_CASH_REPLACE, m_stContext.nSZCashSubBalance);
CHECK_BIZ_FUNC_ERR_RETURN;
m_stContext.bSZUpdatedCashSub = true;
m_stContext.cAssetdayDealType = CNST_DEALTYPE_MOD;
}
Block:34739 line:628~629
}
Block:34741 line:629~631
SettleCashType = m_stPublicContext.nExchIndex == CNST_EXCHANGETYPE_SHA_INDEX ? CNST_CASH_TYPE_SH_CASH_REPLACE : CNST_CASH_TYPE_SZ_CASH_REPLACE;
}
Block:34742 line:631~632
Block:34743 line:632~633
if (m_stContext.nCommission)
F
Block:34744 line:633~639
{
//[可用校验] 费用
CheckSingleFundEnable(SettleCashType, m_stContext.nCommission);
CHECK_BIZ_FUNC_ERR_RETURN;
m_stContext.bUpdatedCommission = true;
m_stContext.cAssetdayDealType = CNST_DEALTYPE_MOD;
}
Block:34746 line:639~642
return m_nErrorNo;
Block:34747 line:642~642
}

```
Block:8176 line:1221~1222   CRuleStockHold* CLayer::GetRuleStockHold(USTNumID iGlobalIndex, char* pStockCode)
Block:8178 line:1222~1223   {
Block:8179 line:1223~1224   if ( m_arrStockHold[iGlobalIndex] != NULL )
                              T  F
Block:8180 line:1224~1226   {
                              return m_arrStockHold[iGlobalIndex];
                            }
Block:8182 line:1226~1232   uint32_t nRecordCount = 0;
                            int32_t  iErrorNo = 0;
                            CRuleStockHold* pRuleStockHold = m_arrStockHoldTable.CreateRecord(&nRecordCount, &iErrorNo);
Block:8183 line:1232~1233   if ( unlikely(pRuleStockHold == NULL) )
                              -  F
Block:8184 line:1233~1235   {
                              return pRuleStockHold;
                            }
Block:8186 line:1235~1238   m_arrStockHold[iGlobalIndex] = pRuleStockHold;
Block:8187 line:1238~1239   if ( likely(pStockCode != NULL) )
                              T  F
Block:8188 line:1239~1241   {
                              snprintf(pRuleStockHold->m_stStockHold.SecurityCode, sizeof(USTSecurityCode), "%s", pStockCode);
                            }
Block:8190 line:1241~1244   return m_arrStockHold[iGlobalIndex];
Block:8191 line:1244~1244   }
```

N_1109 : ['StockCode']:

N_2776 : ['StockCode', 'Direction']:

```
Block:29282 line:2094~2095  int32_t CNewStockOrderInsertFlow::UpdateInstanceHold()
Block:29284 line:2095~2098  {
                              CInstanceHold* pInstanceHold = m_stStockPublicContext.pInstance->GetInstanceHoldByCode(m_stStockPublicContext.nStockCode, m_stStockPublicContext.nExchIndex,
                              m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stStockPublicContext.SeatIndex);
Block:29285 line:2098~2099  if ( unlikely(pInstanceHold == nullptr) )
                              T  F
Block:29286 line:2099~2116  {
                              pInstanceHold = m_stStockPublicContext.pInstance->NewInstanceHold(m_stStockPublicContext.nStockCode, m_stStockPublicContext.nExchIndex,
                              m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stStockPublicContext.SeatIndex);
                              CHECK_BIZ_OBJ_ADD_FAIL(pInstanceHold, m_nErrorNo, "新增实例持仓记录失败",
                              "InstanceID:", wrap(m_stStockPublicContext.pInstance->m_stInstanceField.InstanceID),
                              "StockCode:", wrap(m_stStockPublicContext.nStockCode));
                              pInstanceHold->m_stInsHoldField.InstanceID = m_stStockPublicContext.pInstance->m_stInstanceField.InstanceID;
                              pInstanceHold->m_stInsHoldField.StockCode = m_stStockPublicContext.nStockCode;
                              memcpy(pInstanceHold->m_stInsHoldField.ExchangeType, m_stStockPublicContext.pStockCode->m_stCodeField.ExchangeType, sizeof(pInstanceHold->m_stInsHoldField.ExchangeType));
                              pInstanceHold->m_stInsHoldField.CurrentAmount = 0;
                              pInstanceHold->m_stInsHoldField.TargetAmount = 0;
                              pInstanceHold->m_stInsHoldField.StockHoldIndex = m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex;
                              m_stStockPublicContext.pInstanceHold = pInstanceHold;
                              m_stStockPublicContext.cInstanceDealType = CNST_DEALTYPE_ADD;
                            }
Block:29287 line:2116~2117  else
Block:29288 line:2117~2120  {
                              m_stStockPublicContext.pInstanceHold = pInstanceHold;
                              m_stStockPublicContext.cInstanceDealType = CNST_DEALTYPE_MOD;
                            }
Block:29289 line:2120~2136  lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(this->m_lpContext->GetThreadNo(), 116);
                            //更新实例持仓
                            CStockCode* pStockCode = lpContext->m_pDataImpl->GetStockCode(pInstanceHold->m_stInsHoldField.ExchangeType,
                            pInstanceHold->m_stInsHoldField.StockCode);
                            AMUST_LOGBIZERR_CHECK_OBJ_NOT_FOUND_INNER_ASSIGN(pStockCode, "找不到该证券代码信息", "StockCode:", wrap(pInstanceHold->m_stInsHoldField.StockCode));
                            UpdateHoldEnableValue<CInstanceHold>(lpContext, pInstanceHold, pInstanceHold->m_stInsHoldField.ExchangeType,
                            pStockCode->m_stCodeField.StockType,
                            pStockCode->m_stCodeField.HzjyFlag,
                            m_stStockPublicContext.cEntrustDirection,
                            CNST_BUSINPROTYPE_ENTRUST,
                            m_stStockPublicContext.nFrozenVolume);
                            return m_nErrorNo;
Block:29290 line:2136~2136  }
```

N_2777 : ['StockCode', 'SeatNo', 'Direction', 'OrderPrice', 'OrderVolume', 'OrderCommand']:

```
Block:29291 line:2146~2147  int32_t CNewStockOrderInsertFlow::InsertEntrust(uint32_t nBathcNo, uint32_t& nEntrustIndex)
Block:29293 line:2147~2150  {
                              lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(this->m_lpContext->GetThreadNo(), 117);
                              CSerialInfo* pSerial = lpContext->m_pDataImpl->GetSerialInfo();
Block:29294 line:2150~2151  if ( strcmp(lpContext->m_pDataImpl->GetSysParamVal(10000), "0") == 0 )
                              T  -
Block:29295 line:2151~2153  {
                              m_nEntrustNo = pSerial->GetEntrustSerialNo();
Block:29296 line:2153~2154  if ( unlikely(m_nEntrustNo <= 0) )
                              -  F
Block:29297 line:2154~2162  {
                              m_nErrorNo = ERR_USER_ADD_TABLERECORD_FAIL;
                              AMUST_LOGBIZERR_RETURN(lpContext, m_nErrorNo, "新增委托序号失败",
                              wrap("CombiId", m_stStockPublicContext.pCombi->m_stCombiField.CombiID),
                              wrap("StockCode", m_stStockPublicContext.pStockCode->m_stCodeField.StockCode),
                              wrap("OrderVolume", m_stStockPublicContext.nOrderVolume),
                              wrap("OrderPrice", ConvertDouble(m_stStockPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT)
                              );
                            }
```

Block:29299 line:2162~2163
```
}
```

Block:29301 line:2163~2178
```
int iError = 0;
CStockEntrust* pEntrust = lpContext->m_pDataImpl->NewStockEntrust(pSerial, m_nEntrustNo, nEntrustIndex);
CHECK_BIZ_OBJ_ADD_FAIL(pEntrust, m_nErrorNo, "新增委托记录失败".
wrap("CombiId", m_stStockPublicContext.pCombi->m_stCombiField.CombiID),
wrap("StockCode", m_stStockPublicContext.pStockCode->m_stCodeField.StockCode),
wrap("OrderVolume", m_stStockPublicContext.nOrderVolume),
wrap("OrderPrice", ConvertDouble(m_stStockPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT),
wrap("EntrustNo", m_nEntrustNo),
wrap("EntrustReference", m_stStockPublicContext.EntrustReference));
lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(this->m_lpContext->GetThreadNo(), 118);
//将本笔委托插入到批号内
iError = pSerial->BatchBindEntrust(nBathcNo, pEntrust, nEntrustIndex, CNST_MACHINE_MAIN);
lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 119);
```

Block:29302 line:2178~2179
```
if ( iError != ERR_OK )
```
- F

Block:29303 line:2179~2182
```
{
  m_nErrorNo = ERR_USER_TABLERECORD_NOTEXISTS;
  AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_TABLERECORD_NOTEXISTS,"委托绑定批号失败",wrap("nBathcNo", nBathcNo),wrap("nEntrustNo", m_nEntrustNo));
}
```

Block:29305 line:2182~2183
```

```

Block:29306 line:2183~2184
```
if ( likely(nullptr != m_stStockPublicContext.pSession) && m_stStockPublicContext.EntrustReference > 0 )
```
T - T F

T F

Block:29307 line:2184~2186
```
{
  m_stStockPublicContext.pSession->AddEntrust(m_stStockPublicContext.EntrustReference, pEntrust);
}
```

Block:29309 line:2186~2243
```
lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(this->m_lpContext->GetThreadNo(), 120);
//插入挂单列表
lpContext->m_pDataImpl->m_mapStockPendingOrder.insert(std::pair<uint32_t, CStockEntrust*>(m_nEntrustNo, pEntrust));
//m_pSerial->SetEntBatchNo(nBathcNo, pEntrust);
pEntrust->m_stEntField.EntrustNo = m_nEntrustNo;
pEntrust->m_stEntField.BatchNo = nBathcNo;
pEntrust->m_stEntField.FundIndex = m_stStockPublicContext.pFund->m_stFundField.GlobalIndex;
pEntrust->m_stEntField.AssetIndex = m_stStockPublicContext.pAsset->m_stAssetField.GlobalIndex;
pEntrust->m_stEntField.CombiIndex = m_stStockPublicContext.pCombi->m_stCombiField.GlobalIndex;
pEntrust->m_stEntField.InstanceIndex = m_stStockPublicContext.StrategyIndex;
memcpy(pEntrust->m_stEntField.ExchangeType, m_stStockPublicContext.pStockCode->m_stCodeField.ExchangeType, sizeof(pEntrust->m_stEntField.ExchangeType));
pEntrust->m_stEntField.EntrustAmount = m_stStockPublicContext.nOrderVolume;
pEntrust->m_stEntField.OrderCommand = m_stStockPublicContext.nOrderCommand;
pEntrust->m_stEntField.EntrustBs = m_stStockPublicContext.cEntrustBs;
pEntrust->m_stEntField.EntrustDirection = m_stStockPublicContext.cEntrustBs - '0';
pEntrust->m_stEntField.CurrDate = m_stStockPublicContext.mCurrDate;
pEntrust->m_stEntField.CurrTime = m_stStockPublicContext.mCurrTime;
pEntrust->m_stEntField.ConfirmTime = 0;
pEntrust->m_stEntField.EntrustPrice = m_stStockPublicContext.nOrderPrice;
pEntrust->m_stEntField.EntrustBalance = m_stStockPublicContext.nOrderVolume * m_stStockPublicContext.nOrderPrice;
pEntrust->m_stEntField.ExchIndex = m_stStockPublicContext.nExchIndex;
pEntrust->m_stEntField.FrozenBalance = m_stStockPublicContext.nFrozenBalance;
pEntrust->m_stEntField.FrozenCommission = m_stStockPublicContext.nFrozenCommission;
pEntrust->m_stEntField.SessionNo = m_stStockPublicContext.nSessionIndex;
pEntrust->m_stEntField.TraderID = m_stStockPublicContext.pTrader->m_sTraderField.TraderID;
pEntrust->m_stEntField.SubscribeID = lpContext->m_pDataImpl->GetSubIdByConnectId(m_stStockPublicContext.nConnectID, m_stStockPublicContext.nApiConnectionID);
pEntrust->m_stEntField.ExternReportNo = m_stStockPublicContext.nExternReportNo;
pEntrust->m_stEntField.EntrustOrigin = m_stStockPublicContext.cEntrustOrigin;
memcpy(pEntrust->m_stEntField.StockAccount, m_stStockPublicContext.pStockHolder->m_stStockHolder.StockAccount, sizeof(pEntrust->m_stEntField.StockAccount));
pEntrust->m_stEntField.StockCode = m_stStockPublicContext.nStockCode;
pEntrust->m_stEntField.EntrustReference = m_stStockPublicContext.EntrustReference;
pEntrust->m_stEntField.WithdrawAmount = 0;
pEntrust->m_stEntField.BusinessAmount = 0;
pEntrust->m_stEntField.BusinessBalance = 0;
pEntrust->m_stEntField.EntrustType = CNST_ENTRUSTTYPE_NORMAL;
pEntrust->m_stEntField.ErrorNo = 0;
pEntrust->m_stEntField.EntrustStatus = CNST_ENTRUSTSTATUS_NOREPORT;
pEntrust->m_stEntField.OrigEntrustNo = 0;
pEntrust->m_stEntField.RecordNo = 0;
pEntrust->m_stEntField.Busiflag = CNST_BUSINFLAG_BUYSALE;
memcpy(pEntrust->m_stEntField.SeatNo, m_stStockPublicContext.pTradeSeat->m_stSeatField.SeatNo, sizeof(pEntrust->m_stEntField.SeatNo));
pEntrust->m_stEntField.UnionRiskResult = CNST_UNION_RISK_NOT_REPLY;
pEntrust->m_pAsset = m_stStockPublicContext.pAsset;
pEntrust->m_pCombi = m_stStockPublicContext.pCombi;
pEntrust->m_pHoldReal = m_stStockPublicContext.pStockHold;
pEntrust->m_pStockCode = m_stStockPublicContext.pStockCode;
pEntrust->pSecurityInfo = m_stStockPublicContext.m_pSecurityInfo;
pEntrust->m_pInstance = m_stStockPublicContext.pInstance;
pEntrust->m_pStockHolder = m_stStockPublicContext.pStockHolder;
pEntrust->m_pSeat = m_stStockPublicContext.pTradeSeat;
pEntrust->m_pEntrustExpandInfo = nullptr;
m_stStockPublicContext.pEntrust = pEntrust;
```

Block:29310 line:2243~2244
```
if ( lpContext->m_pDataImpl->m_pRiskData->m_iRiskUpdate != 0 )
```
T -

Block:29311 line:2244~2248
```
{
  pEntrust->pRiskEntrust = m_stStockPublicContext.pRiskEntrust;
  pEntrust->m_stEntField.RiskEntrustNo = m_stStockPublicContext.pRiskEntrust->m_stEntField.RiskEntrustNo;
  m_stStockPublicContext.pRiskEntrust->m_pStockEntrust = pEntrust;
}
```

Block:29313 line:2248~2269
```
INFO_LOG(lpContext, "插入委托表", wrap("AssetIndex", pEntrust->m_stEntField.AssetIndex),
wrap("EntrustNo", pEntrust->m_stEntField.EntrustNo),
wrap("BatchNo", pEntrust->m_stEntField.BatchNo),
wrap("BusinessAmount", pEntrust->m_stEntField.BusinessAmount),
wrap("BusinessBalance", ConvertDouble(pEntrust->m_stEntField.BusinessBalance, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
wrap("CombiIndex", pEntrust->m_stEntField.CombiIndex),
wrap("ConfirmTime", pEntrust->m_stEntField.ConfirmTime),
wrap("EntrustAmount", pEntrust->m_stEntField.EntrustAmount),
wrap("EntrustBs", pEntrust->m_stEntField.EntrustBs),
wrap("ExchIndex", pEntrust->m_stEntField.ExchIndex),
wrap("EntrustPrice", ConvertDouble(pEntrust->m_stEntField.EntrustPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT),
//wrap("EntrustReference", m_stContext.pEntrust->m_stEntField.EntrustReference),
wrap("EntrustStatus", pEntrust->m_stEntField.EntrustStatus),
wrap("StockCode", pEntrust->m_stEntField.StockCode),
wrap("FrozenBalance", ConvertDouble(pEntrust->m_stEntField.FrozenBalance, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
wrap("FrozenCommission", ConvertDouble(pEntrust->m_stEntField.FrozenCommission, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
wrap("StockAccount", pEntrust->m_stEntField.StockAccount));
lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 121);
```

```
return ERR_OK;
```

Block:29314 line:2269~2269
```
}
```

N_2788 : ['StockCode', 'SeatNo', 'Direction', 'OrderPrice', 'OrderVolume', 'OrderCommand']:

Block:29409 line:2499~2500
```
int32_t CNewStockOrderInsertFlow::SendToUnionRisk()
```

Block:29411 line:2500~2504
```
{
 //计算委托编号
 CSerialInfo* pSerial = lpContext->m_pDataImpl->GetSerialInfo();
 m_nEntrustNo = pSerial->GetEntrustSerialNo(m_stStockPublicContext.nExchIndex);
```

Block:29412 line:2504~2505
```
if( unlikely(m_nEntrustNo <= 0) )
    -                    -
```

Block:29413 line:2505~2513
```
 {
  m_nErrorNo = ERR_USER_ADD_TABLERECORD_FAIL;
  AMUST_LOGBIZERR_RETURN(lpContext, m_nErrorNo, "新增委托序号失败",
  wrap("CombiId", m_stStockPublicContext.pCombi->m_stCombiField.CombiID),
  wrap("StockCode", m_stStockPublicContext.pStockCode->m_stCodeField.StockCode),
  wrap("OrderVolume", m_stStockPublicContext.nOrderVolume),
  wrap("OrderPrice", ConvertDouble(m_stStockPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT)
  );
 }
```

Block:29415 line:2513~2531
```
 REQ_UNION_RISK_ORDER_MSG sReportMsg;
 sReportMsg.LdpHead.FunctionID = CNST_TO_UNIRISK_ORDER_INSERT;
 auto& toUniRisk = sReportMsg.ReqUnionRiskOrderField;
 toUniRisk.ExternSysNo = lpContext->m_iExternSysNo;
 toUniRisk.SubSysNo = (EXTERNAL_SYT_TYPE_O45 == lpContext->m_pDataImpl->GetExternSysType()) ? 3335 : 0;
 toUniRisk.ExternReportNo = m_nEntrustNo;
 toUniRisk.EntrustDate = m_stStockPublicContext.mCurrDate;
 memcpy(toUniRisk.FundCode, m_stStockPublicContext.pFund->m_stFundField.FundCode, sizeof(toUniRisk.FundCode));
 memcpy(toUniRisk.AssetCode, m_stStockPublicContext.pAsset->m_stAssetField.AssetCode, sizeof(toUniRisk.AssetCode));
 memcpy(toUniRisk.CombiCode, m_stStockPublicContext.pCombi->m_stCombiField.CombiCode, sizeof(toUniRisk.CombiCode));
 memcpy(toUniRisk.ExchangeType, m_stStockPublicContext.pStockCode->m_stCodeField.ExchangeType, sizeof(toUniRisk.ExchangeType));
 memcpy(toUniRisk.SecurityCode, m_stStockPublicContext.pStockCode->m_stCodeField.StockCode, sizeof(USTStockCodeStr));
 toUniRisk.ExchangePlatform = (m_stStockPublicContext.pExcharg->m_ExchangeIndex == 0) ? CNST_TRADE_PLATFORM_SH_BIDDING : CNST_TRADE_PLATFORM_SZ_BIDDING;
 memcpy(toUniRisk.StockAccount, m_stStockPublicContext.pStockHolder->m_stStockHolder.StockAccount, sizeof(toUniRisk.StockAccount));
 CSeat* pBindSeat = lpContext->m_pDataImpl->GetSeatByIndex(m_stStockPublicContext.SeatIndex);
```

Block:29416 line:2531~2532
```
if ( pBindSeat )
   -      -
```

Block:29417 line:2532~2534
```
 {
  memcpy(toUniRisk.BindSeat, pBindSeat->m_stSeatField.SeatNo, sizeof(toUniRisk.BindSeat));
 }
```

Block:29419 line:2534~2538
```
 memcpy(toUniRisk.SeatNo, m_stStockPublicContext.pTradeSeat->m_stSeatField.SeatNo, sizeof(toUniRisk.SeatNo));
 toUniRisk.OrderPrice = ConvertDouble(m_stStockPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);
```

Block:29420 line:2538~2539
```
if( unlikely(m_stStockPublicContext.pInstance != nullptr) )
                -                    -
```

Block:29421 line:2539~2542
```
 {
  memcpy(toUniRisk.InstanceCode, m_stStockPublicContext.pInstance->m_stInstanceField.InstanceCode, sizeof(USTCodeStr));
 }
```

Block:29422 line:2542~2543
```
else
```

Block:29423 line:2543~2545
```
 {
  toUniRisk.InstanceCode[0] = '\0';
 }
```

Block:29424 line:2545~2553
```
 toUniRisk.EntrustDirection = m_stStockPublicContext.cEntrustDirection;
 toUniRisk.OrderVolume = m_stStockPublicContext.nOrderVolume;
 toUniRisk.OrderCommand = m_stStockPublicContext.nOrderCommand;
 toUniRisk.OrderBalance = m_stStockPublicContext.nOrderVolume * toUniRisk.OrderPrice;
 int32_t nRet = lpContext->PostMsg2Risk((void*)&sReportMsg, sizeof(REQ_UNION_RISK_ORDER_MSG));
```

Block:29425 line:2553~2554
```
if ( unlikely(nRet != ERR_OK) )
   -                    -
```

Block:29426 line:2554~2559
```
 {
  m_nErrorNo = ERR_PUSH_MESSAGE_FAIL_TO_UNION_RISK;
  SET_CONTEXT_BIZ_ERRMSG(lpContext, m_nErrorNo);
  ERROR_LOG(lpContext, ERR_PUSH_MESSAGE_FAIL_TO_UNION_RISK, wrap("CurrentIndex", lpContext->m_nCurrentIndex),
  wrap("EntrustNo", m_nEntrustNo), wrap("返回错误码", nRet));
 }
```

Block:29428 line:2559~2576
```
 INFO_LOG(lpContext, "推送给联合风控",
 wrap("FunctionID", sReportMsg.LdpHead.FunctionID),
 wrap("ExternSysNo", toUniRisk.ExternSysNo),
 wrap("SubSysNo", toUniRisk.SubSysNo),
 wrap("ExternReportNo", toUniRisk.ExternReportNo),
 wrap("ExchangeType", toUniRisk.ExchangeType),
 wrap("CombiCode", toUniRisk.CombiCode),
 wrap("SecurityCode", toUniRisk.SecurityCode),
 wrap("EntrustDirection", toUniRisk.EntrustDirection),
 wrap("OrderPrice", toUniRisk.OrderPrice, PRICE_DECIMAL_DIGIT),
 wrap("OrderVolume", toUniRisk.OrderVolume),
 wrap("OrderCommand", toUniRisk.OrderCommand),
 wrap("StockAccount", toUniRisk.StockAccount));
 return m_nErrorNo;
```

Block:29429 line:2576~2576
```
}
```

N_1083 : ['StockCode']:

Block:7973 line:802~803
```
CStockHoldReal* CCombi::NewStockHoldReal(CStockHolder* pStockHolder, uint32_t nStockCode,uint32_t exchangeindex,uint32_t StockHoldIndex, uint32_t SeatIndex /*=0*/)
```

Block:7975 line:803~807
```
{
 uint32_t nRecordCount = 0;
 int32_t nErrorNo = 0;
 CStockHoldReal* pStockHoldReal = m_tStockHoldReal.CreateRecord(&nRecordCount, &nErrorNo);
```

Block:7976 line:807~808
```
if (  nErrorNo != 0  )
```
- F

Block:7977 line:808~812
```
 {
  //string sErrorInfo = "CCombi::NewStockHoldReal,分配内存失败:" + to_string(nErrorNo);
  //LOGERROR(sErrorInfo); //TODO 暂不支持
 }
```

Block:7978 line:812~813
```
else
```

Block:7979 line:813~816
```
 {
  uint64_t nKey = ((1LL * nStockCode << 32) | (1LL * exchangeindex << 28) | (StockHoldIndex << 12) | SeatIndex );
  m_mStockHoldRealMap.insert({nKey, pStockHoldReal});
```

Block:7980 line:816~817
```
if (  likely(pStockHoldReal)  )
```
T -

Block:7981 line:817~819
```
   {
    pStockHolder->InsertStockHoldReal(pStockHoldReal, nStockCode, exchangeindex, SeatIndex);
   }
```

Block:7983 line:819~820
```
 }
```

Block:7984 line:820~823
```
 return pStockHoldReal;
```

Block:7985 line:823~823
```
}
```

N_3051 : ['StockCode', 'Direction', 'OrderCommand']:

Block:34748 line:644~645
```
int32_t CETFOrderInsertFlow::CheckRisk()
```

Block:34750 line:645~796
```
{
 int32_t iRet = ERR_OK;
 //hh 暂时注释
 #if 0
 //获取交易平台类型
 if (m_stPublicContext.pExcharg->m_ExchangeIndex == 0)
{
 m_stPublicContext.TradePlatForm = CNST_TRADE_PLATFORM_SH_BIDDING;
}
 else
{
 m_stPublicContext.TradePlatForm = CNST_TRADE_PLATFORM_SZ_BIDDING;
}
 auto& orderInfo = lpContext->m_pRiskContext->m_szBatchOrderInfo[0];
 //申购
 if (m_stPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_APPLY)
{
 //等效于买入ETF合约,判风控
 orderInfo.clear();
 orderInfo.m_iEntrustDirection = CNST_ENTRUSTDIR_BUY;
 orderInfo.m_iOriginEntrustDirection = m_stPublicContext.cEntrustDirection;
 orderInfo.m_iPlatForm = m_stPublicContext.TradePlatForm;
 orderInfo.m_iRiskEntrustNo = m_stPublicContext.nRiskEntrustNo;
 orderInfo.m_nErrorNo = 0;
 orderInfo.m_nOrderCommand = m_stPublicContext.nOrderCommand;
 orderInfo.m_pCombi = m_stPublicContext.pCombi;
 orderInfo.m_pExcharg = m_stPublicContext.pExcharg;
 orderInfo.m_pSeat = m_stPublicContext.pPositionSeat;
 orderInfo.m_pSecurityInfo = m_stContext.pEtfBasicInfo->m_pSecurityInfo;
 orderInfo.m_pStockHolder = m_stPublicContext.pStockHolder;
 orderInfo.m_pTrader = m_stPublicContext.pTrader;
 orderInfo.m_pStockCode = m_stContext.pEtfBasicInfo->m_pStockCode;
 orderInfo.m_iContraReportNo = 0;
 orderInfo.m_iContraSysNo = 0;
 orderInfo.m_iReportNo = m_stPublicContext.nExternReportNo;
 orderInfo.m_iSysNo = m_stPublicContext.nExternSysNo;
 orderInfo.mCurrDate = m_stPublicContext.CurrDate;
 orderInfo.mCurrTime = m_stPublicContext.CurrTime;
 // lpContext->GetDateTime(&orderInfo.mCurrDate, &orderInfo.mCurrTime);
 m_nErrorNo = RiskCheck(lpContext, 1);
 if (m_nErrorNo != ERR_OK)
{
 return m_nErrorNo;
}
 if (m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX
 || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX
 || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CURRENCY_ETF_INDEX
 )
{
 //黄金 货币  跨境ETF没有成分股,无需做成分股风控校验
}
 else
{
 //等效于买入ETF成分股,判风控
 orderInfo.m_iEntrustDirection = CNST_ENTRUSTDIR_SALE;
 int32_t i = 0;
 for (auto& etfStockDetail : m_stContext.pEtfBasicInfo->m_vecEtfStocklist)
{
 if (etfStockDetail->m_pSecurityInfo == NULL)
{
 continue;
}
 orderInfo.m_pSecurityInfo = etfStockDetail->m_pSecurityInfo;
 CExcharg* pExcharg = lpContext->GetExcharg(GetExchangeIndex(etfStockDetail->mstEtfStocklist.ExchangeTypeElement));
 CHECK_BIZ_OBJ_NOT_FOUND(pExcharg, "找不到成分股交易市场对象 ", "ExchangeType:", wrap(etfStockDetail->mstEtfStocklist.ExchangeTypeElement));
 orderInfo.m_pStockCode = pExcharg->GetStockCode(atoi(etfStockDetail->mstEtfStocklist.SecurityCodeElement));
 orderInfo.m_RuleCaleResult.clear();
 m_nErrorNo = RiskCheck(lpContext, 1);
 if (m_nErrorNo != ERR_OK)
{
 return m_nErrorNo;
}
 ++i;
}
}
 return iRet;
}
 //赎回
 else
{
 //等效卖出入ETF合约,判风控
 orderInfo.clear();
 orderInfo.m_iEntrustDirection = CNST_ENTRUSTDIR_SALE;
 orderInfo.m_iOriginEntrustDirection = m_stPublicContext.cEntrustDirection;
 orderInfo.m_iPlatForm = m_stPublicContext.TradePlatForm;
 orderInfo.m_iRiskEntrustNo = m_stPublicContext.nRiskEntrustNo;
 orderInfo.m_nErrorNo = 0;
 orderInfo.m_nOrderCommand = m_stPublicContext.nOrderCommand;
 orderInfo.m_pCombi = m_stPublicContext.pCombi;
 orderInfo.m_pExcharg = m_stPublicContext.pExcharg;
 orderInfo.m_pSeat = m_stPublicContext.pPositionSeat;
```

```
orderInfo.m_pSecurityInfo = m_stContext.pEtfBasicInfo->m_pSecurityInfo;
orderInfo.m_pStockHolder = m_stPublicContext.pStockHolder;
orderInfo.m_pTrader = m_stPublicContext.pTrader;
orderInfo.m_pStockCode = m_stContext.pEtfBasicInfo->m_pStockCode;
orderInfo.m_iContraReportNo = 0;
orderInfo.m_iContraSysNo = 0;
orderInfo.m_iReportNo = m_stPublicContext.nExternReportNo;
orderInfo.m_iSysNo = m_stPublicContext.nExternSysNo;
orderInfo.mCurrDate = m_stPublicContext.CurrDate;
orderInfo.mCurrTime = m_stPublicContext.CurrTime;
// lpContext->GetDateTime(&orderInfo.mCurrDate, &orderInfo.mCurrTime);
m_nErrorNo = RiskCheck(lpContext, 1);
if (m_nErrorNo != ERR_OK)
{
return m_nErrorNo;
}
if (m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX
|| m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX
|| m_stPublicContext.nSecurityType == CNST_STOCKKIND_CURRENCY_ETF_INDEX
)
{
//黄金 货币  跨境ETF没有成分股,无需做成分股风控校验
}
else
{
//等效于买入ETF成分股,判风控
orderInfo.m_iEntrustDirection = CNST_ENTRUSTDIR_BUY;
for (auto& etfStockDetail : m_stContext.pEtfBasicInfo->m_vecEtfStocklist)
{
orderInfo.m_pSecurityInfo = etfStockDetail->m_pSecurityInfo;
CExcharg* pExcharg = lpContext->GetExcharg(GetExchangeIndex(etfStockDetail->mstEtfStocklist.ExchangeTypeElement));
CHECK_BIZ_OBJ_NOT_FOUND(pExcharg, "找不到成分股交易市场对象 ", "ExchangeType:", wrap(etfStockDetail->mstEtfStocklist.ExchangeTypeElement));
orderInfo.m_pStockCode = pExcharg->GetStockCode(atoi(etfStockDetail->mstEtfStocklist.SecurityCodeElement));
orderInfo.m_RuleCalcResult.clear();
m_nErrorNo = RiskCheck(lpContext, 1);
if (m_nErrorNo != ERR_OK)
{
return m_nErrorNo;
}
}
}
return iRet;
}
#endif
return iRet;
```

Block:34751
line:796~796

```
}
```

N_3723 : ['StockCode']:

```
Block:42871 line:1146~1147   int32_t CRuleStockHold::RiskUpdateOrderVirtualGroup(SRiskUpdateField& tRiskUpdateField, USTDouble& dCostBalance, bool& bIsLock)
Block:42873 line:1147~1148   {
Block:42874 line:1148~1149   if ( bIsLock )
                                 - -
Block:42875 line:1149~1151   {
                               Lock();
                             }
Block:42877 line:1151~1156   //初始化的时候赋值证券代码
                             //m_stStockHold.StockCode = tStockOrderContextField.nStockCode;
                             /*买方向*/
Block:42878 line:1156~1157   if ( tRiskUpdateField.m_cEntrustBs == '1' )
                                 -                    -
Block:42879 line:1157~1165   {
                               //m_stStockHold.PreBuyFee += tRiskUpdateField.m_nCommission;
                               //m_stStockHold.PreBuyBalance += tRiskUpdateField.m_nEntrustBalance;
                               m_stStockHold.PreBuyAmount += tRiskUpdateField.m_iEntrustAmount;
                               //m_stStockHold.LastBuyCarryoverCost += dCostBalance;
                             }
                              /*卖方向*/
Block:42882 line:1165~1166   else if ( tRiskUpdateField.m_cEntrustBs == '2' )
            42881                -                    -
            42880
Block:42883 line:1166~1172   {
                               //m_stStockHold.PreSellFee += tRiskUpdateField.m_nCommission;
                               //m_stStockHold.PreSellBalance += tRiskUpdateField.m_nEntrustBalance;
                               m_stStockHold.PreSellAmount += tRiskUpdateField.m_iEntrustAmount;
                               //m_stStockHold.LastSellCarryoverCost += dCostBalance;
                             }
Block:42886 line:1172~1174
Block:42887 line:1174~1175   if ( bIsLock )
                                 - -
Block:42888 line:1175~1177   {
                               Unlock();
                             }
Block:42890 line:1177~1180   return 0;
Block:42891 line:1180~1180   }
```

N_3535 : ['OrderVolume']:

```
Block:39820 line:55~56       int32_t CStockRiskUpdate::RiskOrderUpdate(SRiskUpdateField& tRiskUpdateField)
Block:39822 line:56~60       {
                             INFO_LOG(tRiskUpdateField.m_lpContext, tRiskUpdateField.GetDetail().c_str())
                             auto& riskField = tRiskUpdateField.m_pRiskEntrust->m_stEntField;
                             const auto& secuinfoField = tRiskUpdateField.m_pSecurityInfo->m_stSecurityInfo;
Block:39823 line:60~61       if( tRiskUpdateField.m_pRiskEntrust )
                                 T                -
Block:39824 line:61~64       {
                               memcpy(riskField.SecurityCode, secuinfoField.SecurityCode, sizeof(riskField.SecurityCode));
                               riskField.OrderVolumeLeft = tRiskUpdateField.m_iEntrustAmount;
                             }
Block:39826 line:64~89       /*更新组合*/
                             CCombi* pCombi = tRiskUpdateField.m_pCombi;
                             tRiskUpdateField.m_iUpdateContra = 1;
                             CStockRiskUpdate::RiskOrderUpdateByLayer(pCombi, tRiskUpdateField, false);
                             CHECK_RISK_UPATE_FUNCTION_RESULT
                             /*更新单元*/
                             tRiskUpdateField.m_iUpdateContra = 0;
                             CStockRiskUpdate::RiskOrderUpdateByLayer(pCombi->m_pAsset, tRiskUpdateField, false);
                             CHECK_RISK_UPATE_FUNCTION_RESULT
                             /*更新产品*/
                             CFund* pFund = pCombi->m_pFund;
                             tRiskUpdateField.m_iUpdateContra = 0;
                             CStockRiskUpdate::RiskOrderUpdateByLayer(pFund, tRiskUpdateField, false);
                             CHECK_RISK_UPATE_FUNCTION_RESULT
                             /*更新公司*/
                             /*CCompany* pCompany = pCombi->m_pCompany;
                             CStockRiskUpdate::RiskOrderUpdateByLayer(pCompany, tRiskUpdateField, false);
                             CHECK_RISK_UPATE_FUNCTION_RESULT
                             tRiskUpdateField.m_lpContext->getLdpContext()->lpTimeStamp->Add("RiskOrderUpdate_6");  //458ns*/
                             return 0;
Block:39827 line:89~89       }
```

N_16 : ['StockCode', 'Direction']:

```
Block:294 line:552~553
int32_t CheckStockEnable(CStockBizContext* lpContext, CStockHoldReal* lpHoldReal, USTStockCode nStockCode, HSNum RealAmount, USTEntrustDirection EntrustDirection)
Block:296 line:553~554
{
Block:297 line:554~555
if ( unlikely(lpHoldReal == nullptr || lpHoldReal->m_pStockCode == nullptr) )
  -  F
Block:298 line:555~564
{
 ///报错返回，可用证券不足
 USTStockCodeStr StockCodeStr;
 itoa(nStockCode, StockCodeStr, 10, 6);
 StockCodeStr[6] = '\0';
 AMUST_LOGBIZERR_RETURN(lpContext, ERR_SECU_STOCK_CANUSEAMT_NOTENOUGH, wrap("StockCode", StockCodeStr),
 wrap("FrozenVolume", RealAmount),
 "找不到该证券的持仓，EnableAmount:0");
 return lpContext->m_nErrorNo;
}
Block:300 line:564~569
USTVolume EnableAmount = CalcHoldEnableValue<CStockHoldReal>(lpContext, lpHoldReal, lpHoldReal->m_stHoldReal.ExchangeType,
lpHoldReal->m_pStockCode->m_stCodeField.StockType, lpHoldReal->m_pStockCode->m_stCodeField.HzjyFlag, EntrustDirection);
Block:301 line:569~570
if ( unlikely(RealAmount > EnableAmount) )
  T  F
Block:302 line:570~583
{
 ///报错返回，可用证券不足
 USTStockCodeStr StockCodeStr;
 itoa(nStockCode, StockCodeStr, 10, 6);
 StockCodeStr[6] = '\0';
 AMUST_LOGBIZERR_RETURN(lpContext, ERR_SECU_STOCK_CANUSEAMT_NOTENOUGH, wrap("StockCode", StockCodeStr),
 wrap("FrozenVolume", RealAmount),
 wrap("EnableAmount", EnableAmount),
 wrap("CombiID", lpHoldReal->m_stHoldReal.CombiID),
 wrap("BindSeat", lpHoldReal->m_stHoldReal.BindSeat),
 wrap("StockAccount", lpHoldReal->m_stHoldReal.StockAccount)
 );
 return lpContext->m_nErrorNo;
}
Block:304 line:583~586
return 0;
Block:305 line:586~586
}
```

N_3069 : ['ExchangeID', 'StockCode', 'Direction', 'OrderPrice', 'OrderVolume']:

```
Block:35016 line:1859~1860
int32_t CETFOrderInsertFlow::InsertEntrust(uint32_t nBathcNo, uint32_t& nEntrustIndex)
Block:35018 line:1860~1873
{
 CSerialInfo* pSerial = lpContext->m_pDataImpl->GetSerialInfo();
 uint32_t nEntrustNo = pSerial->GetEntrustSerialNo();
 //uint32_t nEntrustIndex = 0;
 m_stContext.pEntrust = lpContext->m_pDataImpl->NewEtfStockEntrust(pSerial, nEntrustNo, nEntrustIndex);
 CHECK_BIZ_OBJ_ADD_FAIL(m_stContext.pEntrust, m_nErrorNo, "新增ETF委托记录失败",
 wrap("AccountIndex", m_stPublicContext.nAssetIndex),
 wrap("EntrustNo", nEntrustNo),
 wrap("EntrustReference", m_stPublicContext.uiEntrustReference));
 //获取委托引用
Block:35019 line:1873~1874
if ( likely(nullptr != m_stPublicContext.pSession) && m_stPublicContext.uiEntrustReference > 0 )
  T  -  T  F
  T  F
Block:35020 line:1874~1876
 {
  m_stPublicContext.pSession->AddEntrust(m_stPublicContext.uiEntrustReference, m_stContext.pEntrust);
 }
Block:35022 line:1876~1878
Block:35023 line:1878~1879
if ( m_Mode == ORDER_SVERICE_TYPE::TS || m_Mode == ORDER_SVERICE_TYPE::CMC_BEFOR| | m_Mode == ORDER_SVERICE_TYPE::CMC_AFTER| | m_Mode == ORDE
  -  F  -  F  -  F  -
  -  F
Block:35024 line:1879~1881
 {
  lpContext->m_pDataImpl->SetStockEntrustSysNo(m_stPublicContext.nExternSysNo, m_stPublicContext.nExternReportNo, nEntrustNo);
 }
Block:35026 line:1881~1887
 //委托批号（与买卖保持一致）
 //uint32_t nBathcNo;
 //SBatchEntrustInfo *pBatchInfo = lpContext->m_pDataImpl->GetSerialInfo()->NewBatchEntrustInfo(nBathcNo);
 int32_t iError = lpContext->m_pDataImpl->GetSerialInfo()->BatchBindEntrust(nBathcNo, m_stContext.pEntrust, nEntrustIndex, CNST_MACHINE_MAIN);
Block:35027 line:1887~1888
if ( iError != ERR_OK )
  -  F
Block:35028 line:1888~1891
 {
  m_nErrorNo = ERR_USER_TABLERECORD_NOTEXISTS;
  AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_TABLERECORD_NOTEXISTS, "委托绑定批号失败", wrap("nBathcNo", nBathcNo), wrap("nEntrustNo", nEntrustNo));
 }
Block:35030 line:1891~1913
 //委托表赋值
 m_stContext.pEntrust->m_stEntField.CurrDate = lpContext->m_pDataImpl->GetSysarg()->m_nInitDate;
 lpContext->GetDateTime(NULL, &m_stContext.pEntrust->m_stEntField.CurrTime);
 m_stContext.pEntrust->m_stEntField.EntrustNo = nEntrustNo;
 m_stContext.pEntrust->m_stEntField.BatchNo = nBathcNo;
 m_stContext.pEntrust->m_stEntField.AssetIndex = m_stPublicContext.pAsset->m_stAssetField.GlobalIndex;
 m_stContext.pEntrust->m_stEntField.CombiIndex = m_stPublicContext.pCombi->m_stCombiField.GlobalIndex;
 m_stContext.pEntrust->m_stEntField.FundIndex = m_stPublicContext.pFund->m_stFundField.GlobalIndex;
 m_stContext.pEntrust->m_stEntField.CombiID = m_stPublicContext.pCombi->m_stCombiField.CombiID;
 m_stContext.pEntrust->m_stEntField.InstanceIndex = m_stPublicContext.nInstanceIndex;
 memcpy(m_stContext.pEntrust->m_stEntField.ExchangeType, m_stPublicContext.ExchangeID, sizeof(m_stContext.pEntrust->m_stEntField.ExchangeType));
 m_stContext.pEntrust->m_stEntField.ExchIndex = m_stPublicContext.nExchIndex;
 itoa(m_stPublicContext.nSecurityCode, m_stContext.pEntrust->m_stEntField.SecurityCode, 10, 6); //申赎代码
 m_stContext.pEntrust->m_stEntField.EntrustAmount = m_stPublicContext.nOrderVolume;
 m_stContext.pEntrust->m_stEntField.EntrustPrice = m_stPublicContext.nOrderPrice;
 m_stContext.pEntrust->m_stEntField.EntrustStatus = CNST_ENTRUSTSTATUS_NOREPORT;
 m_stContext.pEntrust->m_stEntField.ConfirmTime = 0;
 m_stContext.pEntrust->m_stEntField.EntrustDirection = m_stPublicContext.cEntrustDirection;
 m_stContext.pEntrust->m_stEntField.BusinessAmount = 0;
 m_stContext.pEntrust->m_stEntField.EntrustType = CNST_ENTRUSTTYPE_NORMAL;
```

Block:35031 line:1913~1914	if (m_stPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_REDEEM) T F

Block:35032
line:1914~1918
	{
	m_stContext.pEntrust->m_stEntField.FrozenBalance = m_stContext.nEstimateCash + m_stContext.nCommission; //冻结总金额
	m_stContext.pEntrust->m_stEntField.FrozenSubstituteBalance = 0; //冻结现金替代
	}

Block:35033 line:1918~1919	else

Block:35034
line:1919~1922
	{
	m_stContext.pEntrust->m_stEntField.FrozenBalance = m_stContext.nEstimateCash + m_stContext.nCashSubBalance + m_stContext.nCommission; //冻结总金额
	m_stContext.pEntrust->m_stEntField.FrozenSubstituteBalance = m_stContext.nCashSubBalance; //冻结现金替代
	}

Block:35035
line:1922~1928
	m_stContext.pEntrust->m_stEntField.FrozenEstimateCash = m_stContext.nEstimateCash; //冻结预估现金差额
	m_stContext.pEntrust->m_stEntField.FrozenCommission = m_stContext.nCommission; //冻结手续费
	m_stContext.pEntrust->m_stEntField.ErrorNo = 0;
	m_stContext.pEntrust->m_stEntField.EntrustReference = m_stPublicContext.uiEntrustReference;

Block:35036 line:1928~1929	if(m_stPublicContext.pSession != nullptr) T -

Block:35037
line:1929~1931
	{
	m_stContext.pEntrust->m_stEntField.SessionIndex = m_stPublicContext.pSession->SessionIndex;
	}

Block:35039
line:1931~1982
	m_stContext.pEntrust->m_stEntField.TraderID = m_stPublicContext.pTrader->m_sTraderField.TraderID;
	m_stContext.pEntrust->m_stEntField.SubscribeID = lpContext->m_pDataImpl->GetSubIdByConnectId(lpContext->GetConnectID());
	m_stContext.pEntrust->m_stEntField.ExecType = m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.ExecType;
	m_stContext.pEntrust->m_stEntField.ExternReportNo = m_stPublicContext.nExternReportNo;
	m_stContext.pEntrust->m_stEntField.EntrustOrigin = m_stPublicContext.cEntrustOrigin;
	m_stContext.pEntrust->m_stEntField.StockHoldIndex = m_stPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex;
	m_stContext.pEntrust->m_stEntField.SeatIndex = m_stPublicContext.pTraderSeat->m_stSeatField.SeatIndex;
	INFO_LOG(lpContext, "插入委托表", wrap("AssetIndex", m_stContext.pEntrust->m_stEntField.AssetIndex),
	wrap("EntrustNo", m_stContext.pEntrust->m_stEntField.EntrustNo),
	wrap("BatchNo", m_stContext.pEntrust->m_stEntField.BatchNo),
	wrap("BusinessAmount", m_stContext.pEntrust->m_stEntField.BusinessAmount),
	wrap("CombiID", m_stContext.pEntrust->m_stEntField.CombiID),
	wrap("ConfirmTime", m_stContext.pEntrust->m_stEntField.ConfirmTime),
	wrap("EntrustAmount", m_stContext.pEntrust->m_stEntField.EntrustAmount),
	wrap("EntrustDirection", m_stContext.pEntrust->m_stEntField.EntrustDirection),
	wrap("EntrustPrice", ConvertDouble(m_stContext.pEntrust->m_stEntField.EntrustPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT),
	//wrap("EntrustReference", m_stContext.pEntrust->m_stEntField.EntrustReference),
	wrap("EntrustStatus", m_stContext.pEntrust->m_stEntField.EntrustStatus),
	wrap("ExchIndex", m_stContext.pEntrust->m_stEntField.ExchIndex),
	wrap("ExecType", m_stContext.pEntrust->m_stEntField.ExecType),
	wrap("FrozenBalance", ConvertDouble(m_stContext.pEntrust->m_stEntField.FrozenBalance, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
	wrap("FrozenCommission", ConvertDouble(m_stContext.pEntrust->m_stEntField.FrozenCommission, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
	//wrap("FrozenEstimateCash", ConvertDouble(m_stContext.pEntrust->m_stEntField.FrozenEstimateCash, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
	//wrap("nCashSubBalance", ConvertDouble(m_stContext.pEntrust->m_stEntField.nCashSubBalance, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
	wrap("SecurityCode", m_stContext.pEntrust->m_stEntField.SecurityCode));
	//wrap("SubscribeID", m_stContext.pEntrust->m_stEntField.SubscribeID),
	//wrap("TraderID", m_stContext.pEntrust->m_stEntField.TraderID));
	m_stContext.pEntrust->m_pAsset = m_stPublicContext.pCombi->m_pAsset;
	m_stContext.pEntrust->m_pCombi = m_stPublicContext.pCombi;
	m_stContext.pEntrust->m_pFund = m_stPublicContext.pFund;
	m_stContext.pEntrust->m_pHoldReal = m_stContext.pETFHold;
	m_stContext.pEntrust->m_pStockCode = m_stContext.pETFCode;
	m_stContext.pEntrust->pEtfBasicInfo = m_stContext.pEtfBasicInfo;
	m_stContext.pEntrust->m_lstEtfStockDetail = m_lstEtfStockDetail;
	m_stContext.pEntrust->pSecurityInfo = m_stPublicContext.pSecurityInfo;
	m_stContext.pEntrust->iMode = m_Mode;
	m_stContext.pEntrust->m_pStockHolder = m_stPublicContext.pStockHolder;
	m_stContext.pEntrust->m_pSeat = m_stPublicContext.pTraderSeat;
	m_stContext.pEntrust->pRiskEntrust = m_stPublicContext.pRiskEntrust;
	m_stContext.pEntrust->m_pEntrustExpandInfo = nullptr;
	//插入委托明细
	InsertEntrustDeatil();
	CHECK_BIZ_FUNC_ERR_RETURN;
	return m_nErrorNo;

Block:35040 line:1982~1982	}

N_3041 : ['OrderVolume']:

Block:34518 line:116~117	int32_t CETFOrderInsertFlow::CheckCreationredemNum()
Block:34520 line:117~119	{ //申购
Block:34521 line:119~120	if(unlikely(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfReportUnit <= 0)) - F

Block:34522
line:120~124
	{
	///报错返回
	AMUST_LOGBIZERR_ASSIGN(lpContext, m_nErrorNo, ERR_ETF_UNIT_CHECK_FAIL, wrap("EtfReportUnit", m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfReportUnit));
	return m_nErrorNo;
	}

Block:34524 line:124~126	
Block:34525 line:126~127	if(unlikely(m_stPublicContext.nOrderVolume <= 0 || m_stPublicContext.nOrderVolume % m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfReportUnit != 0)) - F

Block:34526
line:127~132
	{
	///报错返回
	AMUST_LOGBIZERR_ASSIGN(lpContext, m_nErrorNo,ERR_ETF_REPORT_NUM_CHECK_FAIL, wrap("nOrderVolume", m_stPublicContext.nOrderVolume),
	wrap("EtfReportUnit", m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfReportUnit));
	return m_nErrorNo;
	}

Block:34528 line:132~135	return m_nErrorNo;
Block:34529 line:135~135	}

Block:8368 line:1581~1582
```
CStockCode* CExcharg::GetStockCode(int32_t nStockCode)
```

Block:8370 line:1582~1584
```
{
 CStockCode* pStockcode = NULL;
```

Block:8371 line:1584~1585
```
if (  m_mapStockCode.find(nStockCode) != m_mapStockCode.end()  )
```
T F

Block:8372 line:1585~1587
```
 {
  pStockcode = m_mapStockCode.at(nStockCode);
 }
```

Block:8374 line:1587~1590
```
return pStockcode;
```

Block:8375 line:1590~1590
```
}
```

N_1130 : ['StockCode']:

N_3055 : ['StockCode', 'Direction', 'OrderVolume']:

Block:34769 line:867~868
```
int32_t CETFOrderInsertFlow::UpdateRiskData()
```

Block:34771 line:868~1009
```
{
 //hh 暂时注释
 #if 0
 //更新风控标志位
 m_stPublicContext.bIsRiskUpdate = true;
 //获取ETF交易代码对象
 CSecurityInfo* pSecurityInfo = m_stPublicContext.pExcharg->GetSecurityInfo(const_cast<char*>(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode2));
 if (nullptr == pSecurityInfo)
 {
 char szErrorInfo[200];
 snprintf(szErrorInfo, sizeof(szErrorInfo),
 "ETF申赎代码[%s]对应的二级市场交易代码[%s]在系统内不存在对应的证券对象!",
 m_stContext.pEtfBasicInfo->m_pSecurityInfo->m_stSecurityInfo.SecurityCode,
 m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode2
 );
 LOGERROR(lpContext, ERR_USER_TABLERECORD_NOTEXISTS, szErrorInfo);
 return ERR_USER_TABLERECORD_NOTEXISTS;
 }
 //填写风控更新入参
 tRiskUpdateField.Clear();
 tRiskUpdateField.m_lpContext = lpContext;
 tRiskUpdateField.m_pCombi = m_stPublicContext.pCombi;
 tRiskUpdateField.m_pTrader = m_stPublicContext.pTrader;
 //申购
 if (m_stPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_APPLY)
 {
 tRiskUpdateField.m_pSecurityInfo = pSecurityInfo;
 tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_BUY + '0';
 tRiskUpdateField.m_iEntrustAmount = m_stPublicContext.nOrderVolume;
 USTBalance nNoneRepBalance(0);
 for (size_t i = 0; i < m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size(); ++i)
 {
 auto* pStockCode = m_stPublicContext.pExcharg->GetStockCode(m_stContext.vStockCode[i]);
 if (pStockCode == nullptr)
 continue;
 nNoneRepBalance += pStockCode->m_lpStockQuote->YesterdayClosePrice * (m_stContext.vStockAmount[i] - m_stContext.vStockRepAmount[i]);
 }
 INFO_LOG(lpContext, wrap("现金替代金额", m_stContext.nCashSubBalance), wrap("预估现金差额", m_stContext.nEstimateCash),
 wrap("成分股未替代数量*昨日结算价", nNoneRepBalance))
 tRiskUpdateField.m_nEntrustBalance = m_stContext.nCashSubBalance + std::max(m_stContext.nEstimateCash, 0L) + nNoneRepBalance;
 tRiskUpdateField.m_nCommission = m_stContext.nCommission;
 tRiskUpdateField.m_pRiskEntrust = m_stPublicContext.pRiskEntrust;
 tRiskUpdateField.m_pRiskEntrust->m_stEntField.OrderBalance = tRiskUpdateField.m_nEntrustBalance;
 tRiskUpdateField.m_pRiskEntrust->m_stEntField.OrderCommission = m_stContext.nCommission;
 //ETF合约风控数据更新
 //        CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
 //AddEtfOrderCashInfo(m_stPublicContext.pCombi->m_pAsset, m_stContext.nCashSubBalance, m_stContext.nCommission, m_stContext.nEstimateCash);
 if (m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX
 || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX
 || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CURRENCY_ETF_INDEX
 )
 {
 //黄金 货币 跨境ETF不更新成份股持仓
 }
 else
 {
 //成分股风控数据更新
 int32_t i = 0;
 for (auto& pEtfStocklistField : m_stContext.pEtfBasicInfo->m_vecEtfStocklist)
 {
 if (pEtfStocklistField->m_pSecurityInfo == NULL)
 {
 continue;
 }
 tRiskUpdateField.m_pSecurityInfo = pEtfStocklistField->m_pSecurityInfo;
 tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_SALE + '0';
 //tRiskUpdateField.m_iEntrustAmount = m_stPublicContext.nOrderVolume * pEtfStocklistField->mstEtfStocklist.StockAmount / ReportUnitAmount;
 tRiskUpdateField.m_iEntrustAmount = m_stContext.vStockAmount[i] - m_stContext.vStockRepAmount[i];
 tRiskUpdateField.m_nEntrustBalance = 0;
 tRiskUpdateField.m_nCommission = 0;
 //成分股风控代码更新
 //            CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
 ++i;
 }
 }
 }
 //赎回
 else
 {
 tRiskUpdateField.m_pSecurityInfo = pSecurityInfo;
 tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_SALE + '0';
 tRiskUpdateField.m_iEntrustAmount = m_stPublicContext.nOrderVolume;
 tRiskUpdateField.m_nEntrustBalance = 0;
 tRiskUpdateField.m_nCommission = m_stContext.nCommission;
 tRiskUpdateField.m_pRiskEntrust = m_stPublicContext.pRiskEntrust;
 tRiskUpdateField.m_pRiskEntrust->m_stEntField.OrderCommission = m_stContext.nCommission;
 //ETF风控数据更新
 //        CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
 if (m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX
 || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX
 || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CURRENCY_ETF_INDEX
 )
 {
 //黄金 货币 跨境ETF不更新成份股持仓
 }
 else
 {
 int32_t i = 0;
 //成分股风控数据更新
 for (auto& pEtfStocklistField : m_stContext.pEtfBasicInfo->m_vecEtfStocklist)
 {
 if (pEtfStocklistField->m_pSecurityInfo == NULL)
 {
 continue;
 }
 CStockCode* pStockCode = m_stPublicContext.pExcharg->GetStockCode(m_stContext.vStockCode[i]);
 if(nullptr == pStockCode && CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType)
 {
 pStockCode = m_stPublicContext.pOtherExcharg->GetStockCode(m_stContext.vStockCode[i]);
 }
 tRiskUpdateField.m_pSecurityInfo = pEtfStocklistField->m_pSecurityInfo;
 tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_BUY + '0';
 //tRiskUpdateField.m_iEntrustAmount = m_stPublicContext.nOrderVolume * pEtfStocklistField->mstEtfStocklist.StockAmount / ReportUnitAmount;
 tRiskUpdateField.m_iEntrustAmount = m_stContext.vStockAmount[i] - m_stContext.vStockRepAmount[i];
 //ETF赎回 成分股委托金额 = 成分股未替代数量*昨日结算价
 tRiskUpdateField.m_nEntrustBalance = tRiskUpdateField.m_iEntrustAmount * pStockCode->m_lpStockQuote->YesterdayClosePrice;
 tRiskUpdateField.m_nCommission = 0;
```

```
//成分股风控代码更新
//                CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
++i;
}
}
}
#endif
return ERR_OK;
```

Block:34772 line:1009~1009 `}`

N_2767 :

['StockCode', 'OrderPrice', 'OrderCommand']:

Block:29089 line:1591~1592
```
int32_t CNewStockOrderInsertFlow::InsertRiskEntrust()
```

Block:29091 line:1592~1597
```
{
// return 0;
CSerialInfo* pSerial = lpContext->m_pDataImpl->GetSerialInfo();
uint32_t nRiskEntrustNo = 0;
```

Block:29092 line:1597~1598
```
if (! false )
```
- F

Block:29093 line:1598~1602
```
{
  nRiskEntrus                ntext->GetRiskSerialNo();
  m_stStockPublicContext.nRiskEntrustNo = nRiskEntrustNo;
}
```

Block:29094 line:1602~1603
```
else
```

Block:29095 line:1603~1605
```
{
  nRiskEntrustNo = m_stStockPublicContext.nRiskEntrustNo;
}
```

Block:29096 line:1605~1631
```
INFO_LOG(m_lpContext, "生成风控委托序号,", wrap("RiskEntrustNo", m_stStockPublicContext.nRiskEntrustNo));
//lpContext->m_pRiskContext->m_pRiskData->m_iRiskEntrustNo = nRiskEntrustNo;
CRiskEntrust* pRiskEntrust = lpContext->m_pDataImpl->NewRiskEntrust(pSerial, nRiskEntrustNo);
CHECK_BIZ_OBJ_ADD_FAIL(pRiskEntrust, lpContext->m_nErrorNo, "新增委托记录失败",
"EntrustNo:", wrap(nRiskEntrustNo),
"EntrustReference:", wrap(m_stStockPublicContext.EntrustReference));
pRiskEntrust->m_stEntField.RiskEntrustNo = nRiskEntrustNo;
pRiskEntrust->m_stEntField.EntrustBs = m_stStockPublicContext.cEntrustBs;
pRiskEntrust->m_stEntField.EntrustPrice = m_stStockPublicContext.nOrderPrice;
pRiskEntrust->m_stEntField.OrderCommand = m_stStockPublicContext.nOrderCommand;
pRiskEntrust->m_stEntField.ContraOrderType = 0;
pRiskEntrust->m_stEntField.StockCode = m_stStockPublicContext.pStockCode->m_stCodeField.StockCode;
pRiskEntrust->m_stEntField.OrderBalance = m_stStockPublicContext.nEntrustBalance;
pRiskEntrust->m_stEntField.OrderCommission = m_stStockPublicContext.nCommission;
pRiskEntrust->m_stEntField.OrderCommissionLeft = m_stStockPublicContext.nCommission;
m_stStockPublicContext.pRiskEntrust = pRiskEntrust;
int exchargeIndex = GetExchangeIndex(m_stStockPublicContext.pStockCode->m_stCodeField.ExchangeType);
CSecurityInfo* pSecurityInfo = lpContext->m_pDataImpl->GetExcharg()[exchargeIndex].GetSecurityInfo(m_stStockPublicContext.pStockCode->m_stCodeField.StockCodeStr);
pSecurityInfo->m_pSelfDealCheckMgr->InsertEntrust(pRiskEntrust);
const auto &nOrderCommand = m_stStockPublicContext.nOrderCommand;
```

Block:29097 line:1631~1632
```
if ( nOrderCommand != CNST_UST_ORDERCOMMAND_LIMIT )
```
T F

Block:29098 line:1632~1633
```
{
```

Block:29099 line:1633~1634
```
if ( exchargeIndex == CNST_EXCHANGETYPE_SHA_INDEX )
```
T F

Block:29100 line:1634~1642
```
  {
  //上交所是否全面注册制 '0'否, '1'是, 默认为'0'
  char cSwitch = '0';
  //市价委托计算委托金额的价格类型 '1'最新价 '2'保护限价或涨跌停价, 默认为'1'
  char cPriceType = '1';
  lpContext->m_pDataImpl->GetSysParamVal(10014, cSwitch);
  lpContext->m_pDataImpl->GetSysParamVal(10031, cPriceType, '1');
```

Block:29101 line:1642~1643
```
if ( cSwitch == '1' )
```
- F

Block:29102 line:1643~1644
```
  {
```

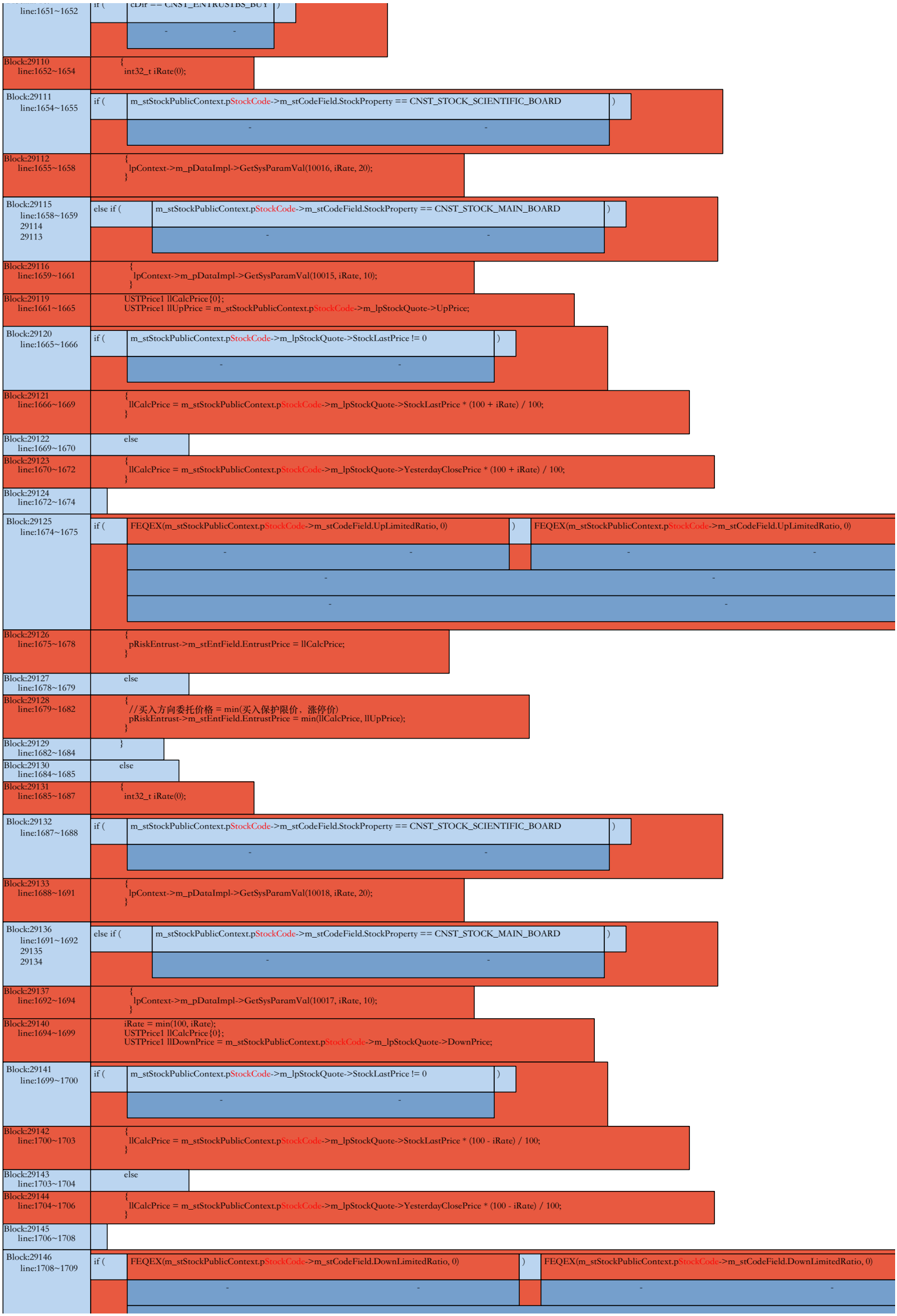
Block:29103 line:1644~1645
```
if ( cPriceType == '1' )
```
- -

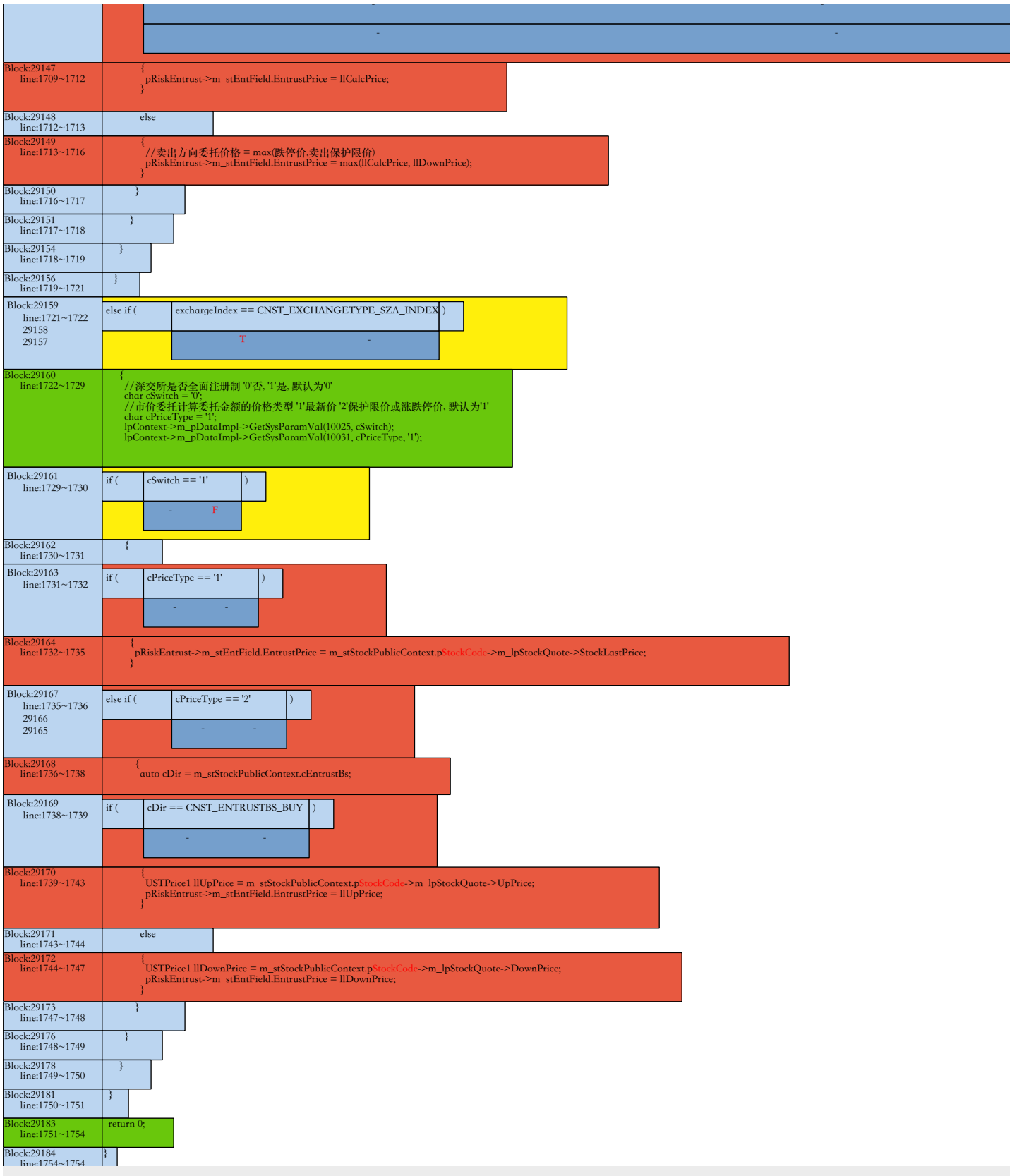
Block:29104 line:1645~1648
```
    {
     pRiskEntrust->m_stEntField.EntrustPrice = m_stStockPublicContext.pStockCode->m_lpStockQuote->StockLastPrice;
    }
```

Block:29107 line:1648~1649 29106 29105
```
else if ( cPriceType == '2' )
```
- -

Block:29108 line:1649~1651
```
    {
     auto cDir = m_stStockPublicContext.cEntrustBs;
```

Block:29109
```
if ( cDir == CNST_ENTRUSTBS_BUY )
```


line:1651~1652
Block:29110
line:1652~1654
int32_t iRate(0);
Block:29111
line:1654~1655
if (m_stStockPublicContext.pStockCode->m_stCodeField.StockProperty == CNST_STOCK_SCIENTIFIC_BOARD)
Block:29112
line:1655~1658
lpContext->m_pDataImpl->GetSysParamVal(10016, iRate, 20);
Block:29115
line:1658~1659
29114
29113
else if (m_stStockPublicContext.pStockCode->m_stCodeField.StockProperty == CNST_STOCK_MAIN_BOARD)
Block:29116
line:1659~1661
lpContext->m_pDataImpl->GetSysParamVal(10015, iRate, 10);
Block:29119
line:1661~1665
USTPrice1 llCalcPrice{0};
USTPrice1 llUpPrice = m_stStockPublicContext.pStockCode->m_lpStockQuote->UpPrice;
Block:29120
line:1665~1666
if (m_stStockPublicContext.pStockCode->m_lpStockQuote->StockLastPrice != 0)
Block:29121
line:1666~1669
llCalcPrice = m_stStockPublicContext.pStockCode->m_lpStockQuote->StockLastPrice * (100 + iRate) / 100;
Block:29122
line:1669~1670
else
Block:29123
line:1670~1672
llCalcPrice = m_stStockPublicContext.pStockCode->m_lpStockQuote->YesterdayClosePrice * (100 + iRate) / 100;
Block:29124
line:1672~1674
Block:29125
line:1674~1675
if (FEQEX(m_stStockPublicContext.pStockCode->m_stCodeField.UpLimitedRatio, 0)) FEQEX(m_stStockPublicContext.pStockCode->m_stCodeField.UpLimitedRatio, 0)
Block:29126
line:1675~1678
pRiskEntrust->m_stEntField.EntrustPrice = llCalcPrice;
Block:29127
line:1678~1679
else
Block:29128
line:1679~1682
//买入方向委托价格 = min(买入保护限价，涨停价)
pRiskEntrust->m_stEntField.EntrustPrice = min(llCalcPrice, llUpPrice);
Block:29129
line:1682~1684
Block:29130
line:1684~1685
else
Block:29131
line:1685~1687
int32_t iRate(0);
Block:29132
line:1687~1688
if (m_stStockPublicContext.pStockCode->m_stCodeField.StockProperty == CNST_STOCK_SCIENTIFIC_BOARD)
Block:29133
line:1688~1691
lpContext->m_pDataImpl->GetSysParamVal(10018, iRate, 20);
Block:29136
line:1691~1692
29135
29134
else if (m_stStockPublicContext.pStockCode->m_stCodeField.StockProperty == CNST_STOCK_MAIN_BOARD)
Block:29137
line:1692~1694
lpContext->m_pDataImpl->GetSysParamVal(10017, iRate, 10);
Block:29140
line:1694~1699
iRate = min(100, iRate);
USTPrice1 llCalcPrice{0};
USTPrice1 llDownPrice = m_stStockPublicContext.pStockCode->m_lpStockQuote->DownPrice;
Block:29141
line:1699~1700
if (m_stStockPublicContext.pStockCode->m_lpStockQuote->StockLastPrice != 0)
Block:29142
line:1700~1703
llCalcPrice = m_stStockPublicContext.pStockCode->m_lpStockQuote->StockLastPrice * (100 - iRate) / 100;
Block:29143
line:1703~1704
else
Block:29144
line:1704~1706
llCalcPrice = m_stStockPublicContext.pStockCode->m_lpStockQuote->YesterdayClosePrice * (100 - iRate) / 100;
Block:29145
line:1706~1708
Block:29146
line:1708~1709
if (FEQEX(m_stStockPublicContext.pStockCode->m_stCodeField.DownLimitedRatio, 0)) FEQEX(m_stStockPublicContext.pStockCode->m_stCodeField.DownLimitedRatio, 0)



N_2772 : ['StockCode', 'OrderVolume']:

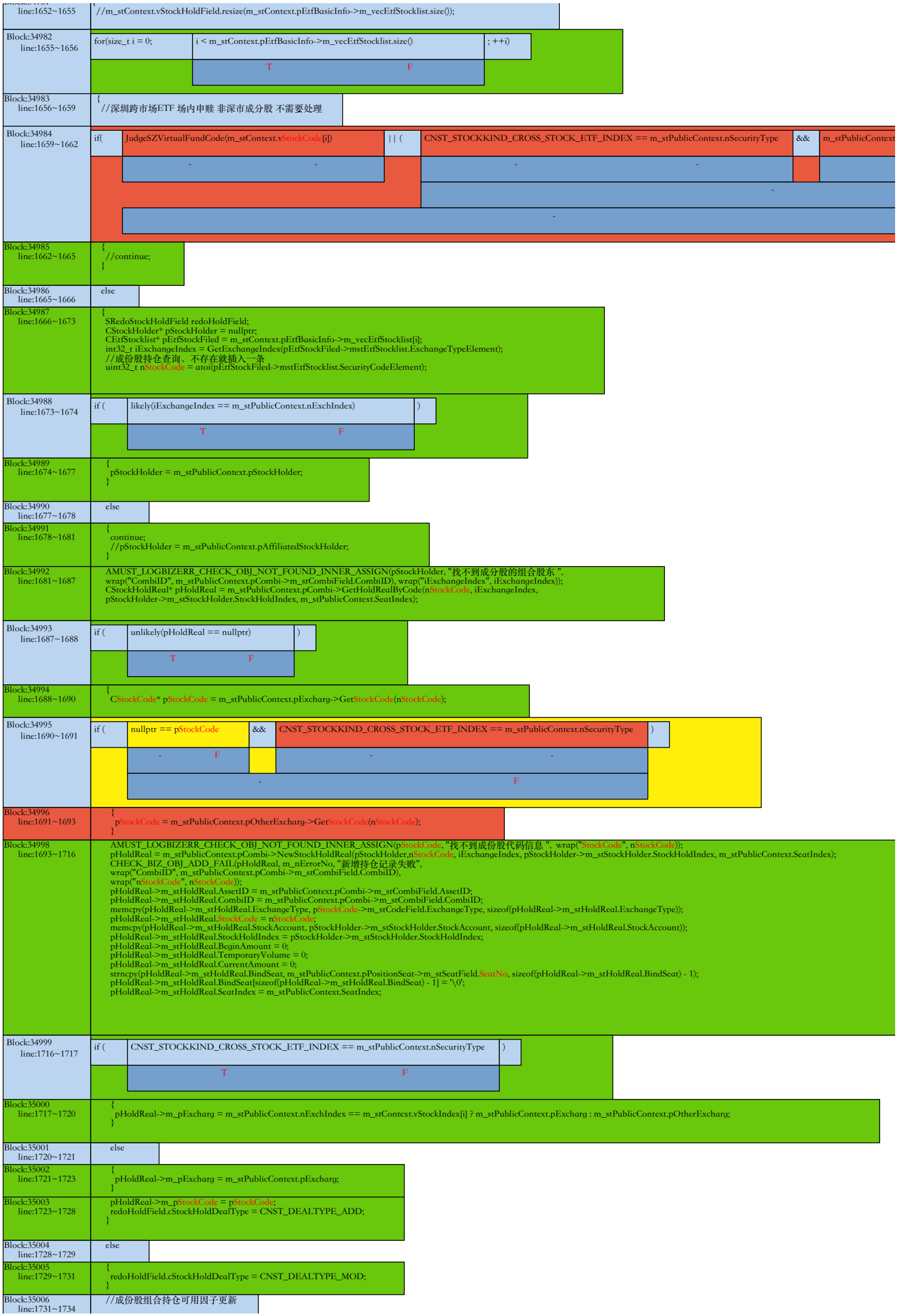
```
Block:29249 line:1953~1954   int32_t CNewStockOrderInsertFlow::RollBackRiskData()
Block:29251 line:1954~1955   {
Block:29252 line:1955~1956   if ( m_nErrorNo != ERR_OK )
                               T -
Block:29253 line:1956~1958   {
                               //风控数据回滚
Block:29254 line:1958~1959   if ( lpContext->m_pDataImpl->m_pRiskData->m_iRiskUpdate != 0 && m_stStockPublicContext.m_bIsRiskUpdated )
                               T -
                               T F
                               T F
Block:29255 line:1959~1961   {
                               //挂单列表回滚
Block:29256 line:1961~1962   if ( likely(m_stStockPublicContext.pRiskEntrust != NULL) )
                               T -
Block:29257 line:1962~1964   {
                               m_stStockPublicContext.pExcharg->GetSecurityInfo(m_stStockPublicContext.pStockCode->m_stCodeField.StockCodeStr)->m_pSelfDealCheckMgr->Remove(m_stStockPublicContext.pRiskEntrust);
                             }
Block:29259 line:1964~1989   /*填写风控更新需要的入参*/
                             SRiskUpdateField tRiskUpdateField;
                             tRiskUpdateField.m_lpContext = lpContext;
                             //账户对象
                             tRiskUpdateField.m_pCombi = m_stStockPublicContext.pCombi;
                             tRiskUpdateField.m_pTrader = m_stStockPublicContext.pTrader;
                             tRiskUpdateField.m_pInstance = m_stStockPublicContext.pInstance;
                             //证券对象
                             tRiskUpdateField.m_pStockCode = m_stStockPublicContext.pStockCode;
                             tRiskUpdateField.m_pSecurityInfo = m_stStockPublicContext.pExcharg->GetSecurityInfo(m_stStockPublicContext.pStockCode->m_stCodeField.StockCodeStr);
                             //AB类需要
                             tRiskUpdateField.m_iEntrustAmount = -m_stStockPublicContext.nOrderVolume;
                             tRiskUpdateField.m_nCommission = -m_stStockPublicContext.nCommission;
                             //F类需要
                             tRiskUpdateField.m_iPreSum = lpContext->m_pDataImpl->m_pRiskData->GetPreSumFlag();
                             tRiskUpdateField.m_cEntrustBs = m_stStockPublicContext.cEntrustBs;
                             tRiskUpdateField.m_nEntrustBalance = -m_stStockPublicContext.nEntrustBalance;
                             tRiskUpdateField.m_pRiskEntrust = m_stStockPublicContext.pRiskEntrust;
                             tRiskUpdateField.m_iRollback = 1;
                             //其他风控数据回滚
                             CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
                             CHECK_BIZ_FUNC_ERR_RETURN;
                           }
Block:29261 line:1989~1990   }
Block:29263 line:1990~1992   return lpContext->m_nErrorNo;
Block:29264 line:1992~1992   }
```

N_3040 : ['Direction']:

```
Block:34501 line:80~81   int32_t CETFOrderInsertFlow::CheckCreationredemType()
Block:34503 line:81~82   {
Block:34504 line:82~83   if ( m_stPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_APPLY )
                           T F
Block:34505 line:83~85   {
                           //申购
Block:34506 line:85~87   if( unlikely(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.CreationredemType != CNST_ENABLE_PURCHASE_AND_REDEMPTION && m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.Creationred
                           -
Block:34507 line:87~91   {
                           ///报错返回
                           AMUST_LOGBIZERR_ASSIGN(lpContext, m_nErrorNo, ERR_FORBID_PURCHASE, wrap("CreationredemType", m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.CreationredemType));
                           return m_nErrorNo;
                         }
Block:34509 line:91~95   m_stContext.nStockEntrustDirection = CNST_ENTRUSTDIR_SALE;
                         m_stContext.nPreDealDataFuncIndex = 0;
                       }
Block:34510 line:95~96   else
Block:34511 line:96~98   {
                           //赎回
Block:34512 line:98~100  if( unlikely(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.CreationredemType != CNST_ENABLE_PURCHASE_AND_REDEMPTION && m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.Creationre
                           -
Block:34513 line:100~104 {
                           ///报错返回
                           AMUST_LOGBIZERR_ASSIGN(lpContext, m_nErrorNo, ERR_RORBID_REDEMPTION, wrap("CreationredemType", m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.CreationredemType));
                           return m_nErrorNo;
                         }
Block:34515 line:104~107   m_stContext.nStockEntrustDirection = CNST_ENTRUSTDIR_BUY;
                           m_stContext.nPreDealDataFuncIndex = 1;
                         }
Block:34516 line:107~110 return m_nErrorNo;
Block:34517 line:110~110 }
```

N_3068 : ['StockCode', 'SeatNo', 'Direction']:

```
Block:34979 line:1651~1652   int32_t CETFOrderInsertFlow::UpdateStockHoldReal()
Block:34981                  {
```


line:1652~1655
//m_stContext.vStockHoldField.resize(m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size());
Block:34982
line:1655~1656
for(size_t i = 0;
i < m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size()
; ++i)
T
F
Block:34983
line:1656~1659
{
//深圳跨市场ETF 场内申赎 非深市成分股 不需要处理
Block:34984
line:1659~1662
if(
JudgeSZVirtualFundCode(m_stContext.vStockCode[i])
||(
CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType
&&
m_stPublicContext
Block:34985
line:1662~1665
{
//continue;
}
Block:34986
line:1665~1666
else
Block:34987
line:1666~1673
{
SRedoStockHoldField redoHoldField;
CStockHolder* pStockHolder = nullptr;
CEtfStocklist* pEtfStockFiled = m_stContext.pEtfBasicInfo->m_vecEtfStocklist[i];
int32_t iExchangeIndex = GetExchangeIndex(pEtfStockFiled->mstEtfStocklist.ExchangeTypeElement);
//成份股持仓查询，不存在就插入一条
uint32_t nStockCode = atoi(pEtfStockFiled->mstEtfStocklist.SecurityCodeElement);
Block:34988
line:1673~1674
if (
likely(iExchangeIndex == m_stPublicContext.nExchIndex)
)
T
F
Block:34989
line:1674~1677
{
pStockHolder = m_stPublicContext.pStockHolder;
}
Block:34990
line:1677~1678
else
Block:34991
line:1678~1681
{
continue;
//pStockHolder = m_stPublicContext.pAffiliatedStockHolder;
}
Block:34992
line:1681~1687
AMUST_LOGBIZERR_CHECK_OBJ_NOT_FOUND_INNER_ASSIGN(pStockHolder, "找不到成分股的组合股东 ",
wrap("CombiID", m_stPublicContext.pCombi->m_stCombiField.CombiID), wrap("iExchangeIndex", iExchangeIndex));
CStockHoldReal* pHoldReal = m_stPublicContext.pCombi->GetHoldRealByCode(nStockCode, iExchangeIndex,
pStockHolder->m_stStockHolder.StockHoldIndex, m_stPublicContext.SeatIndex);
Block:34993
line:1687~1688
if (
unlikely(pHoldReal == nullptr)
)
T
F
Block:34994
line:1688~1690
{
CStockCode* pStockCode = m_stPublicContext.pExcharg->GetStockCode(nStockCode);
Block:34995
line:1690~1691
if (
nullptr == pStockCode
&&
CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType
)
F
Block:34996
line:1691~1693
{
pStockCode = m_stPublicContext.pOtherExcharg->GetStockCode(nStockCode);
}
Block:34998
line:1693~1716
AMUST_LOGBIZERR_CHECK_OBJ_NOT_FOUND_INNER_ASSIGN(pStockCode, "找不到成份股代码信息 ", wrap("StockCode", nStockCode));
pHoldReal = m_stPublicContext.pCombi->NewStockHoldReal(pStockHolder,nStockCode, iExchangeIndex, pStockHolder->m_stStockHolder.StockHoldIndex, m_stPublicContext.SeatIndex);
CHECK_BIZ_OBJ_ADD_FAIL(pHoldReal, m_nErrorNo, "新增持仓记录失败",
wrap("CombiID", m_stPublicContext.pCombi->m_stCombiField.CombiID),
wrap("nStockCode", nStockCode));
pHoldReal->m_stHoldReal.AssetID = m_stPublicContext.pCombi->m_stCombiField.AssetID;
pHoldReal->m_stHoldReal.CombiID = m_stPublicContext.pCombi->m_stCombiField.CombiID;
memcpy(pHoldReal->m_stHoldReal.ExchangeType, pStockCode->m_stCodeField.ExchangeType, sizeof(pHoldReal->m_stHoldReal.ExchangeType));
pHoldReal->m_stHoldReal.StockCode = nStockCode;
memcpy(pHoldReal->m_stHoldReal.StockAccount, pStockHolder->m_stStockHolder.StockAccount, sizeof(pHoldReal->m_stHoldReal.StockAccount));
pHoldReal->m_stHoldReal.StockHoldIndex = pStockHolder->m_stStockHolder.StockHoldIndex;
pHoldReal->m_stHoldReal.BeginAmount = 0;
pHoldReal->m_stHoldReal.TemporaryVolume = 0;
pHoldReal->m_stHoldReal.CurrentAmount = 0;
strncpy(pHoldReal->m_stHoldReal.BindSeat, m_stPublicContext.pPositionSeat->m_stSeatField.SeatNo, sizeof(pHoldReal->m_stHoldReal.BindSeat) - 1);
pHoldReal->m_stHoldReal.BindSeat[sizeof(pHoldReal->m_stHoldReal.BindSeat) - 1] = '\0';
pHoldReal->m_stHoldReal.SeatIndex = m_stPublicContext.SeatIndex;
Block:34999
line:1716~1717
if (
CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType
)
T
F
Block:35000
line:1717~1720
{
pHoldReal->m_pExcharg = m_stPublicContext.nExchIndex == m_stContext.vStockIndex[i] ? m_stPublicContext.pExcharg : m_stPublicContext.pOtherExcharg;
}
Block:35001
line:1720~1721
else
Block:35002
line:1721~1723
{
pHoldReal->m_pExcharg = m_stPublicContext.pExcharg;
}
Block:35003
line:1723~1728
pHoldReal->m_pStockCode = pStockCode;
redoHoldField.cStockHoldDealType = CNST_DEALTYPE_ADD;
}
Block:35004
line:1728~1729
else
Block:35005
line:1729~1731
{
redoHoldField.cStockHoldDealType = CNST_DEALTYPE_MOD;
}
Block:35006
line:1731~1734
//成份股组合持仓可用因子更新

```
Block:35007 line:1734~1735   if ( m_stPublicContext.cEntrustDirection != CNST_ENTRUSTDIR_REDEEM )
                                 T                                F
Block:35008 line:1735~1746   {
                              UpdateHoldEnableValueAndHoldFactor<CStockHoldReal>(lpContext,
                              pHoldReal,
                              pHoldReal->m_stHoldReal.ExchangeType,
                              pHoldReal->m_pStockCode->m_stCodeField.StockType,
                              pHoldReal->m_pStockCode->m_stCodeField.HzjyFlag,
                              m_stPublicContext.cEntrustDirection,
                              CNST_BUSINPROTYPE_ENTRUST,
                              m_stContext.vStockAmount[i] - m_stContext.vStockRepAmount[i],
                              m_stContext.vHoldEnableFactorField,
                              pHoldReal->m_stHoldReal.StockCode);
                             }
Block:35010 line:1746~1764   memcpy(&redoHoldField.stETFHoldRealField, &pHoldReal->m_stHoldReal,sizeof(redoHoldField.stETFHoldRealField));
                             m_stContext.vStockHoldField.push_back(redoHoldField);
                             INFO_LOG(lpContext, "持仓redo", wrap("cStockHoldDealType", redoHoldField.cStockHoldDealType),
                             wrap("AssetID", redoHoldField.stETFHoldRealField.AssetID),
                             wrap("BeginAmount", redoHoldField.stETFHoldRealField.BeginAmount),
                             wrap("CombiID", redoHoldField.stETFHoldRealField.CombiID),
                             wrap("BeginCarryoverCost", redoHoldField.stETFHoldRealField.BeginCarryoverCost, PRICE_DECIMAL_DIGIT),
                             wrap("CurrentAmount", redoHoldField.stETFHoldRealField.CurrentAmount),
                             wrap("ExchangeType", redoHoldField.stETFHoldRealField.ExchangeType),
                             wrap("StockAccount", redoHoldField.stETFHoldRealField.StockAccount),
                             wrap("StockCode", redoHoldField.stETFHoldRealField.StockCode),
                             wrap("StockHoldIndex", redoHoldField.stETFHoldRealField.StockHoldIndex),
                             wrap("TemporaryVolume", redoHoldField.stETFHoldRealField.TemporaryVolume),
                             wrap("WaitMarketVolume", redoHoldField.stETFHoldRealField.WaitMarketVolume));
                            }
Block:35011 line:1764~1767   }
Block:35014 line:1767~1770   return m_nErrorNo;
Block:35015 line:1770~1770   }
```

N_17 : ['StockCode', 'Direction']:

```
Block:306 line:589~590   int32_t CheckInsHoldEnable(CStockBizContext *lpContext, CInstanceHold *pInstanceHold, USTVolume &nOccurAmount, USTStockCode &nStockCode, USTNumID &InstanceID, USTEntrustDirection EntrustD
Block:308 line:590~591   {
Block:309 line:591~592   if ( unlikely(pInstanceHold == nullptr) )
                             -                F
Block:310 line:592~598   {
                          ///报错返回，可用持仓不足
                          AMUST_LOGBIZERR_RETURN(lpContext, ERR_INSTANCE_AMOUNT_NOTENOUGH, "EnableAmount:0", wrap("StockCode",nStockCode),
                          wrap("InstanceID", InstanceID),
                          wrap("nOccurAmount", nOccurAmount));
                          return lpContext->m_nErrorNo;
                         }
Block:312 line:598~602   CStockCode* pStockCode = lpContext->m_pDataImpl->GetStockCode(pInstanceHold->m_stInsHoldField.ExchangeType,
                         pInstanceHold->m_stInsHoldField.StockCode);
Block:313 line:602~603   if( unlikely(nullptr == pStockCode) )
                             -                F
Block:314 line:603~606   {
                          AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_TABLERECORD_NOTEXISTS,"证券代码信息不存在", wrap("StockCode", nStockCode));
                          return lpContext->m_nErrorNo;
                         }
Block:316 line:606~614   USTVolume EnableAmount = CalcHoldEnableValue<CInstanceHold>(lpContext, pInstanceHold,
                         pInstanceHold->m_stInsHoldField.ExchangeType,
                         pStockCode->m_stCodeField.StockType,
                         pStockCode->m_stCodeField.HzjyFlag, EntrustDirection);
                         DEBUG_LOG_TRACE("INSTANCE EnableAmount=%ld\n", EnableAmount);
Block:317 line:614~615   if ( nOccurAmount > EnableAmount )
                             T                F
Block:318 line:615~622   {
                          ///报错返回，可用持仓不足
                          AMUST_LOGBIZERR_RETURN(lpContext, ERR_INSTANCE_AMOUNT_NOTENOUGH, wrap("StockCode", nStockCode),
                          wrap("InstanceID", InstanceID),
                          wrap("OccurAmount", nOccurAmount),
                          wrap("EnableAmount", EnableAmount));
                          return lpContext->m_nErrorNo;
                         }
Block:320 line:622~625   return 0;
Block:321 line:625~625   }
```

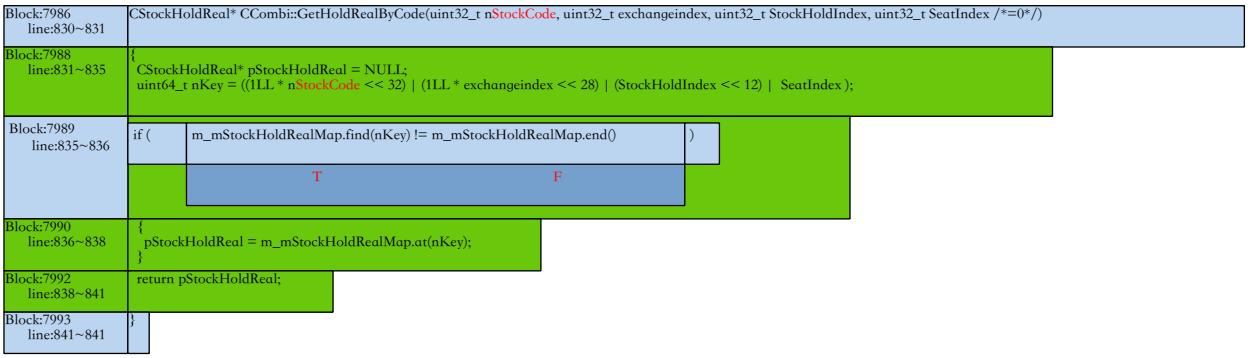
```
Block:10727 line:998~999   CStockCode *GetStockCode(const string &exchangeType, int32_t id) const
Block:10729 line:999~1002  {
                           string key(exchangeType + COLON_SEPERATOR + to_string(id));
                           return GetObj(m_mapStockCode, key);
Block:10730 line:1002~1002 }
```

N_1443 : ['StockCode']:

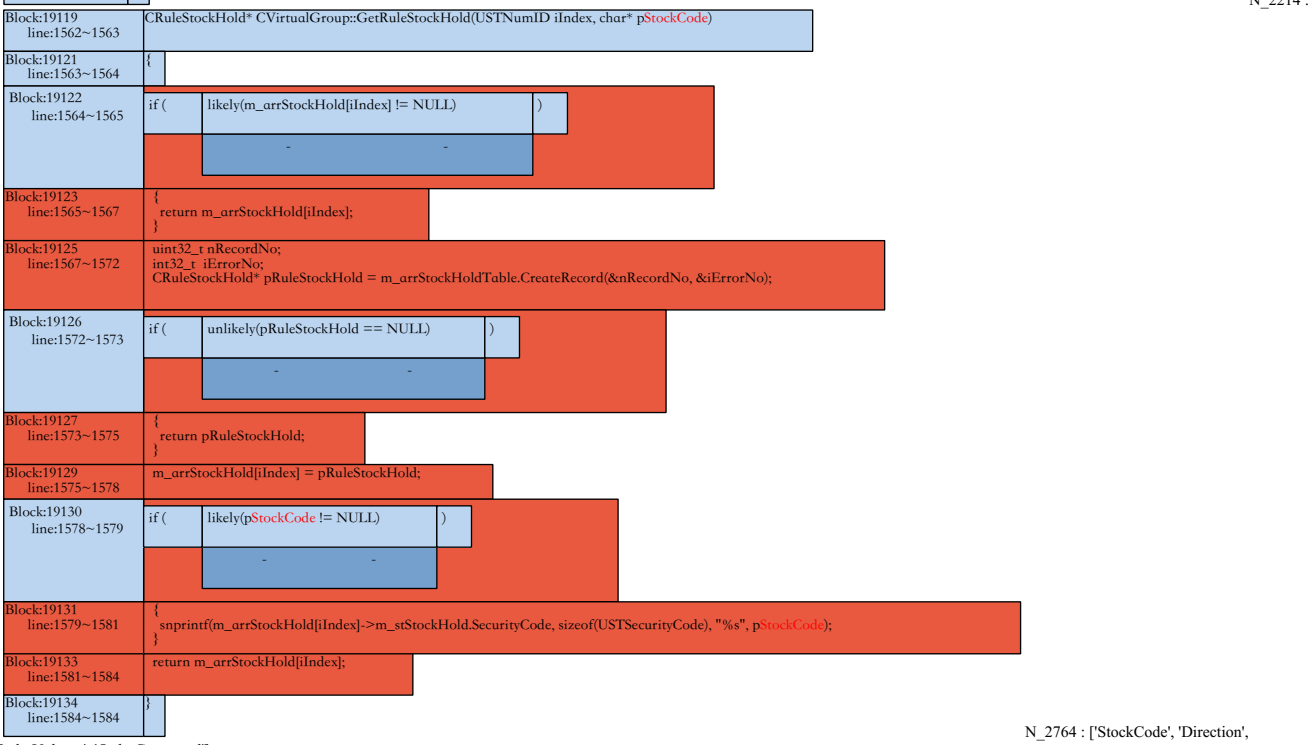
N_262 : ['StockCode', 'Direction']:

```
Block:2596 line:2240~2241   inline uint64_t GetSecondsHashKey(int32_t AssetID,int32_t StockType, int32_t EntrustDirectionIndex, int32_t StockCode, int32_t nStockFeeType)
Block:2598 line:2241~2247   {
                            return  ( (1L*(-1 == AssetID ? 0xffffff : AssetID) << 40)
                            | (1L * (-1 == StockType ? 0xff : StockType) << 32)
                            | (1L*(-1 == EntrustDirectionIndex ? 0xff : EntrustDirectionIndex) << 24)
                            | (1L*(-1 == StockCode ? 0xffffff : StockCode) << 4)
                            | (-1 == nStockFeeType ? 0xf : nStockFeeType) );
Block:2599 line:2247~2247   }
```

N_1084 :



['StockCode']: N_2214 :



['StockCode']: N_2764 : ['StockCode', 'Direction', 'OrderPrice', 'OrderVolume', 'OrderCommand']:

Block:29063 line:1471~1472
```
int32_t CNewStockOrderInsertFlow::CheckBusiness()
```

Block:29065 line:1472~1473
```
{
```

Block:29066 line:1473~1474
```
if (  lpContext->m_nNeedBusiCheck == CNST_TRUE  )
```
T -

Block:29067 line:1474~1506
```
 {
  //会话信息校验
  INFO_LOG(lpContext, "会话信息校验");
  m_nErrorNo = CheckSessionRef(lpContext, m_stStockPublicContext.pSession, m_stStockPublicContext.nSessionIndex, m_stStockPublicContext.EntrustReference);
  CHECK_BIZ_FUNC_ERR_RETURN;
  //价格类型校验
  INFO_LOG(lpContext, "价格类型校验");
  m_nErrorNo = CHeckPriceType(lpContext, m_stStockPublicContext.nExchIndex, m_stStockPublicContext.nOrderCommand, m_stStockPublicContext.pStockCode->m_stCodeField.StockProperty,m_stStockPu
  CHECK_BIZ_FUNC_ERR_RETURN;
  //停牌校验
  INFO_LOG(lpContext, "停牌校验");
  m_nErrorNo = CheckStockStop(lpContext, m_stStockPublicContext.pStockCode);
  CHECK_BIZ_FUNC_ERR_RETURN;
  //市价申报校验
  INFO_LOG(lpContext, "市价申报校验");
  m_nErrorNo = CHeckMarketOrderCommand(lpContext, m_stStockPublicContext.nExchIndex, m_stStockPublicContext.nOrderCommand, m_stStockPublicContext.pStockCode);
  CHECK_BIZ_FUNC_ERR_RETURN;
  //涨跌停校验
  INFO_LOG(lpContext, "涨跌停校验");
  m_nErrorNo = CheckOrderPriceLimit(lpContext, m_stStockPublicContext.nOrderPrice, m_stStockPublicContext.pStockCode->m_lpStockQuote->UpPrice, m_stStockPublicContext.pStockCode->m_lpStockQ
  CHECK_BIZ_FUNC_ERR_RETURN;
  //最小价差校验
  INFO_LOG(lpContext, "最小价差校验");
  m_nErrorNo = CheckPriceInterval(lpContext, ConvertDouble(m_stStockPublicContext.nOrderPrice, PRICE_MULTIPLIER,PRICE_DECIMAL_DIGIT), m_stStockPublicContext.pStockCode->m_stCodeField
  CHECK_BIZ_FUNC_ERR_RETURN;
  //科创板
```

Block:29068 line:1506~1507
```
if(  CNST_STOCK_SCIENTIFIC_BOARD == m_stStockPublicContext.pStockCode->m_stCodeField.StockProperty  )
```
T F

Block:29069 line:1507~1514
```
  {
   INFO_LOG(lpContext, "科创板零股卖出校验、上下限校验");
   m_nErrorNo = CheckSTBoardOrderAmountAndUnit(lpContext, m_stStockPublicContext.nOrderVolume, m_stStockPublicContext.nExchIndex,
   m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stStockPublicContext.nOrderCommand,
   m_stStockPublicContext.cEntrustDirection, m_stStockPublicContext.pStockCode, m_stStockPublicContext.SeatIndex, m_stStockPublicContext.pStockHold);
   CHECK_BIZ_FUNC_ERR_RETURN;
  }
```

Block:29070 line:1514~1515
```
  else
```

Block:29071 line:1515~1527
```
  {
   //数量上下限校验
   INFO_LOG(lpContext, "数量上下限校验");
   m_nErrorNo = CheckOrderAmountLimit(lpContext, m_stStockPublicContext.nOrderVolume, m_stStockPublicContext.nOrderCommand, m_stStockPublicContext.pStockCode);
   CHECK_BIZ_FUNC_ERR_RETURN;
   //买卖最小单位校验
   INFO_LOG(lpContext, "买卖最小单位校验");
   m_nErrorNo = CheckOrderUnit(lpContext, m_stStockPublicContext.cEntrustDirection, m_stStockPublicContext.nOrderVolume, m_stStockPublicContext.nExchIndex,
   m_stStockPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stStockPublicContext.pStockCode,
   m_stStockPublicContext.cEntrustDirection, m_stStockPublicContext.SeatIndex, m_stStockPublicContext.pStockHold);
   CHECK_BIZ_FUNC_ERR_RETURN;
  }
```

Block:29072 line:1527~1528
```
 }
```

Block:29074 line:1528~1530
```
 return m_nErrorNo;
```

Block:29075 line:1530~1530
```
}
```

N_2770 : ['StockCode', 'Direction', 'OrderPrice', 'OrderVolume', 'OrderCommand']:

Block:29220 line:1865~1866
```
int32_t CNewStockOrderInsertFlow::CheckRisk()
```
Block:29222 line:1866~1867
```
{
```
Block:29223 line:1867~1868
```
if (  m_nErrorNo == ERR_OK  )
```
T -

Block:29224 line:1868~1887
```
{
 CRiskOrderInfo *pOrderInfo = &(lpContext->m_pRiskContext->m_szBatchOrderInfo[0]);
 pOrderInfo->m_nErrorNo                = 0;
 pOrderInfo->m_pCombi                  = m_stStockPublicContext.pCombi;
 pOrderInfo->m_pSeat                   = m_stStockPublicContext.pPositionSeat;
 pOrderInfo->m_pStockHolder            = m_stStockPublicContext.pStockHolder;
 //pOrderInfo->m_pStockHolder          = lpContext->m_pDataImpl->GetStkHolder(m_stStockPublicContext.pStockHolder->StockAccount, m_stStockPublicContext.pStockHolder->ExchangeType);
 pOrderInfo->m_pTrader                 = m_stStockPublicContext.pTrader;
 pOrderInfo->m_pSecurityInfo           = m_stStockPublicContext.pExcharg->GetSecurityInfo(m_stStockPublicContext.pStockCode->m_stCodeField.StockCodeStr);
 pOrderInfo->m_pStockCode              = m_stStockPublicContext.pStockCode;
 pOrderInfo->mCurrDate                 = m_stStockPublicContext.mCurrDate;
 pOrderInfo->mCurrTime                 = m_stStockPublicContext.mCurrTime;
 pOrderInfo->m_iSysNo                  = 0;
 pOrderInfo->m_iReportNo               = 0/*pstOrderField->nExternReportNo*/;
 pOrderInfo->m_iRiskEntrustNo          = m_stStockPublicContext.nRiskEntrustNo;
 pOrderInfo->m_iOriginEntrustDirection = m_stStockPublicContext.cEntrustDirection;
 pOrderInfo->m_iEntrustDirection       = m_stStockPublicContext.cEntrustDirection;
```
Block:29225 line:1887~1888
```
if (  m_stStockPublicContext.pStockCode->m_stCodeField.ExchangeType[0] == '1'  )
```
T F

Block:29226 line:1888~1891
```
 {
  pOrderInfo->m_iPlatForm = 1;
 }
```
Block:29229 line:1891~1892 29228 29227
```
else if (  m_stStockPublicContext.pStockCode->m_stCodeField.ExchangeType[0] == '2'  )
```
T -

Block:29230 line:1892~1894
```
 {
  pOrderInfo->m_iPlatForm = 4;
 }
```
Block:29233 line:1894~1909
```
 pOrderInfo->m_nOrderCommand           = m_stStockPublicContext.nOrderCommand;
 pOrderInfo->m_nOrderPrice             = m_stStockPublicContext.pRiskEntrust->m_stEntField.EntrustPrice;
 pOrderInfo->m_iEntrustAmount          = m_stStockPublicContext.nOrderVolume;
 pOrderInfo->m_iContraReportNo         = 0;
 pOrderInfo->m_iContraSysNo            = 0;
 pOrderInfo->m_pExcharg                = m_stStockPublicContext.pExcharg;
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 108);
 m_nErrorNo= RiskCheck((CStockBizContext *)lpContext, 1);
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 109);
 INFO_LOG(lpContext, "CNewStockOrderInsertFlow::CheckRisk", wrap("m_nErrorNo", m_nErrorNo));
 CHECK_FUNC_ERR_RETURN;
}
```
Block:29235 line:1909~1911
```
return m_nErrorNo;
```
Block:29236 line:1911~1911
```
}
```

Block:29237 line:1919~1920
```
int32_t CNewStockOrderInsertFlow::UpdateBusinData( int32_t nBatchNo)
```
Block:29239 line:1920~1922
```
{
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 110);
```
Block:29240 line:1922~1923
```
if (  m_stStockPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_BUY  )
```
T F

Block:29241 line:1923~1926
```
 {
  m_nErrorNo = UpdateAssetday();
  CHECK_BIZ_FUNC_ERR_RETURN;
 }
```
Block:29243 line:1926~1931
```
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 111);
 m_nErrorNo = UpdateHoldReal();
 CHECK_BIZ_FUNC_ERR_RETURN;
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 112);
```
Block:29244 line:1931~1932
```
if (  m_stStockPublicContext.StrategyIndex > 0  )
```
T F

Block:29245 line:1932~1935
```
 {
  m_nErrorNo = UpdateInstanceHold();
  CHECK_BIZ_FUNC_ERR_RETURN;
 }
```
Block:29247 line:1935~1943
```
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 113);
 uint32_t nEntrustIndex;
 m_nErrorNo = InsertEntrust(nBatchNo, nEntrustIndex);
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 114);
 CHECK_BIZ_FUNC_ERR_RETURN;
 return m_nErrorNo;
```
Block:29248 line:1943~1943
```
}
```

N_2771 : ['Direction']: 'OrderVolume']:

N_2765 : ['StockCode', 'Direction', 'OrderPrice',

Block:29076 line:1538~1539
```
int32_t CNewStockOrderInsertFlow::CalcBusinData()
```

Block:29078 line:1539~1541
```
{
```

Block:29079 line:1541~1542
```
if ( m_stStockPublicContext.cEntrustBs == CNST_ENTRUSTBS_BUY )
```
T F

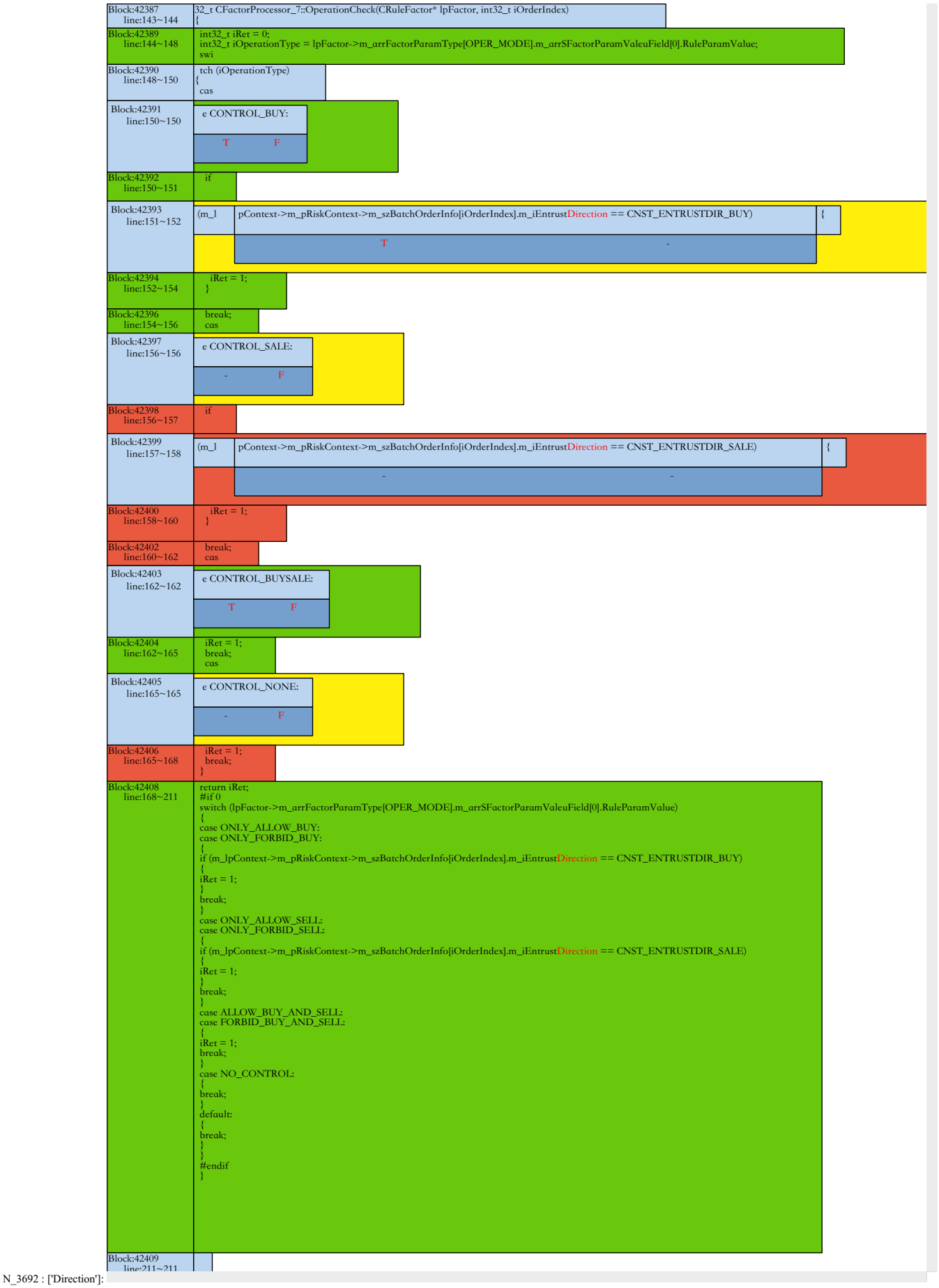
Block:29080 line:1542~1555
```
 {
  //计算费用
  m_stStockPublicContext.nCommission = CalcFee(lpContext, m_stStockPublicContext.pAsset, m_stStockPublicContext.nExchIndex,
  GetEntrustDirectionIndex(m_stStockPublicContext.cEntrustDirection), m_stStockPublicContext.pStockCode, m_stStockPublicContext.nOrderVolume, m_stStockPublicContext.nOrderPrice);
  //计算委托金额
  m_stStockPublicContext.nEntrustBalance = m_stStockPublicContext.nOrderPrice * m_stStockPublicContext.nOrderVolume;
  //冻结金额=委托金额+费用
  m_stStockPublicContext.nFrozenBalance = m_stStockPublicContext.nEntrustBalance + m_stStockPublicContext.nCommission;
  m_stStockPublicContext.nFrozenCommission = m_stStockPublicContext.nCommission;
  m_stStockPublicContext.nFrozenVolume = 0;
 }
```

Block:29081 line:1555~1556
```
else
```

Block:29082 line:1556~1567
```
 {
  //计算费用
  m_stStockPublicContext.nCommission = CalcFee(lpContext, m_stStockPublicContext.pAsset, m_stStockPublicContext.nExchIndex,
  GetEntrustDirectionIndex(m_stStockPublicContext.cEntrustDirection), m_stStockPublicContext.pStockCode, m_stStockPublicContext.nOrderVolume, m_stStockPublicContext.nOrderPrice);
  //计算委托金额
  m_stStockPublicContext.nEntrustBalance = m_stStockPublicContext.nOrderPrice * m_stStockPublicContext.nOrderVolume;
  m_stStockPublicContext.nFrozenCommission = 0;
  m_stStockPublicContext.nFrozenBalance = 0;
  m_stStockPublicContext.nFrozenVolume = m_stStockPublicContext.nOrderVolume;
 }
```

Block:29083 line:1567~1575
```
 INFO_LOG(lpContext,"业务数据计算",
 wrap("nEntrustBalance", ConvertDouble(m_stStockPublicContext.nEntrustBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
 wrap("nFrozenCommission", ConvertDouble(m_stStockPublicContext.nFrozenCommission, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
 wrap("nFrozenBalance", ConvertDouble(m_stStockPublicContext.nFrozenBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
 wrap("nFrozenVolume", m_stStockPublicContext.nFrozenVolume));
 return lpContext->m_nErrorNo;
```

Block:29084 line:1575~1575
```
}
```



N_3692 : ['Direction']:

N_3725 :

Block:42913 line:1218~1219	int32_t CRuleStockHold::RiskUpdateOrder(SRiskUpdateField& tRiskUpdateField, USTDouble& dCostBalance, bool& bIsLock)
Block:42915 line:1219~1220	{
Block:42916 line:1220~1221	if (bIsLock)
	- F

Block:42917
line:1221~1223
	{
	Lock();
	}

Block:42919
line:1223~1228
	//初始化的时候赋值证券代码
	//m_stStockHold.StockCode = tStockOrderContextField.nStockCode;
	/*买方向*/

Block:42920 line:1228~1229	if (tRiskUpdateField.m_cEntrustBs == '1')
	T F

Block:42921
line:1229~1241
	{
	m_stStockHold.PreBuyFee += tRiskUpdateField.m_nCommission;
	m_stStockHold.PreBuyBalance += tRiskUpdateField.m_nEntrustBalance;
	m_stStockHold.PreBuyAmount += tRiskUpdateField.m_iEntrustAmount;
	/*持仓总成本*/
	USTDouble dHoldCostTotal = BuyHoldCostCal(0, 0);
	/*成本金额*/
	dCostBalance = dHoldCostTotal - m_stStockHold.LastBuyCarryoverCost;
	m_stStockHold.LastBuyCarryoverCost = dHoldCostTotal;
	}
	/*卖方向*/

Block:42924 line:1241~1242 42923 42922	else if (tRiskUpdateField.m_cEntrustBs == '2')
	T -

Block:42925
line:1242~1252
	{
	m_stStockHold.PreSellFee += tRiskUpdateField.m_nCommission;
	m_stStockHold.PreSellBalance += tRiskUpdateField.m_nEntrustBalance;
	m_stStockHold.PreSellAmount += tRiskUpdateField.m_iEntrustAmount;
	///*持仓总成本*/
	USTDouble dHoldCostTotal = SellHoldCostCal(0);
	///*成本金额*/
	dCostBalance = dHoldCostTotal - m_stStockHold.LastSellCarryoverCost;
	m_stStockHold.LastSellCarryoverCost = dHoldCostTotal;
	}

Block:42928 line:1252~1254	
Block:42929 line:1254~1255	if (bIsLock)
	- F

Block:42930
line:1255~1257
	{
	Unlock();
	}

Block:42932 line:1257~1260	return 0;
Block:42933 line:1260~1260	}

['StockCode']: N_3043 : ['StockCode', 'OrderPrice', 'OrderVolume']:

Block:34547 line:167~168	int32_t CETFOrderInsertFlow::CalcBusinData()

Block:34549
line:168~189
	{
	m_stContext.nOrderHands = (m_stPublicContext.nOrderVolume / m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfReportUnit);
	//委托价格 = (t-1日基准单位净值)/ETF申报单位（放大10000倍）
	bool bFlag = false;
	m_stPublicContext.nOrderPrice = ConvertInt(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfLastNavPercu / m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfReportUnit, &bFlag);
	//提前分配记录大小(避免自动扩容，影响性能)
	m_stContext.vStockRepAmount.resize(m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size());
	m_stContext.vStockRepBalance.resize(m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size());
	m_stContext.vCommission.resize(m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size());
	m_stContext.vStockCode.resize(m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size());
	m_stContext.vStockIndex.resize(m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size());
	m_stContext.vStockAmount.resize(m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size());
	//预处理申购成份股数据
	//PreDealBusinData();
	(this->*m_arrPreDealBusinDataFunc[m_stContext.nPreDealDataFuncIndex])();
	return m_nErrorNo;

Block:34550 line:189~189	}

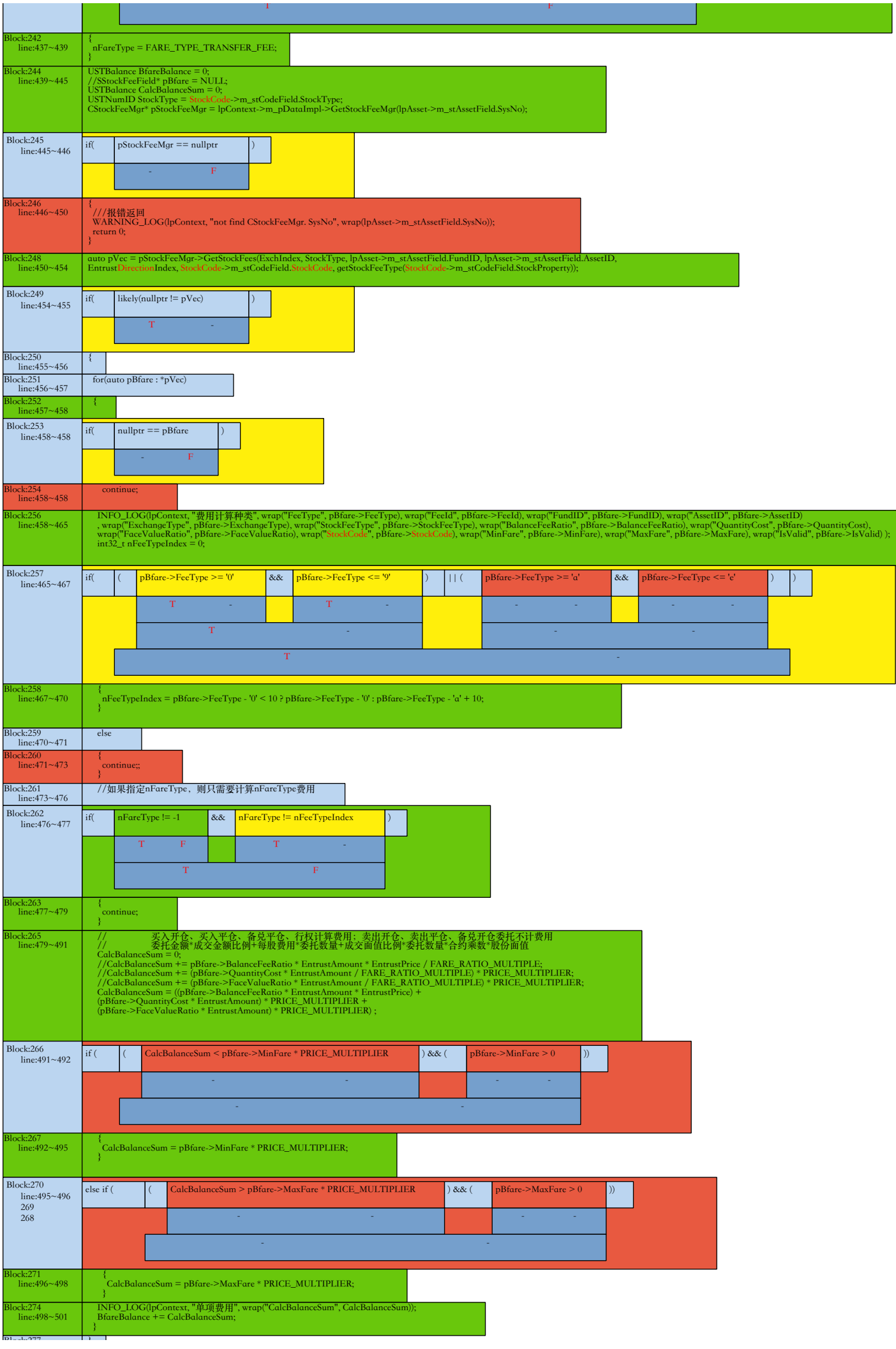
N_3057 : ['ExchangeID', 'StockCode', 'SeatNo', 'Direction', 'OrderPrice', 'OrderVolume', 'OrderCommand']:

Block:34786 line:1030~1031	int32_t CETFOrderInsertFlow::InnerMarketOfferOrder()
Block:34788 line:1031~1033	{ USTNumID BusinFlag = CNST_BUSINFLAG_ETF_APPLYREDEEM;
Block:34789 line:1033~1034	if(CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType)
	T F

Block:34790
line:1034~1036
	{
	BusinFlag = CNST_BUSINFLAG_ETF_INCROSSMARKET;
	}

Block:34792
line:1036~1056
	//报盘结构体
	STOCK_OFFER_ORDER_WTHDRW_MSG sReportMsg;
	sReportMsg.MsgHead.FunctionID = FUNC_OFFER_REPORT;
	sReportMsg.MsgHead.UserDefined = 0; //普通交易写死0，报盘根据这个识别什么业务
	sReportMsg.MsgHead.Token = m_stPublicContext.pAsset->m_stAssetField.GlobalIndex;
	sReportMsg.OrderWthdrw.ReportPrice = ConvertDouble(m_stPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);
	sReportMsg.OrderWthdrw.ReportAmount = m_stPublicContext.nOrderVolume;
	sReportMsg.OrderWthdrw.OrderCommand = 0;
	sReportMsg.OrderWthdrw.EntrustNo = m_stContext.pEntrust->m_stEntField.EntrustNo;
	sReportMsg.OrderWthdrw.OrigEntrustNo = 0;
	sReportMsg.OrderWthdrw.RecordNo = 0;
	memcpy(sReportMsg.OrderWthdrw.ExchangeType, m_stPublicContext.ExchangeID, sizeof (sReportMsg.OrderWthdrw.ExchangeType));
	sReportMsg.OrderWthdrw.EntrustType = CNST_ENTRUSTTYPE_NORMAL;
	sReportMsg.OrderWthdrw.BusinFlag = BusinFlag;
	sReportMsg.OrderWthdrw.EntrustBs = m_stPublicContext.cEntrustDirection + '0' - 2; //申购传'1' 赎回传'2'
	//上海 实物申赎：单市场、跨市场 现金申赎：跨境、黄金、货币
	sReportMsg.OrderWthdrw.ExecType = CNST_EXEC_TYPE_PHYSICAL;

```
Block:34793 line:1056~1058
if ( ( m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX
Block:34794 line:1058~1062
{
 sReportMsg.OrderWthdrw.ExecType = CNST_EXEC_TYPE_CASH;
 memcpy(sReportMsg.OrderWthdrw.StockCode, m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode2, sizeof(sReportMsg.OrderWthdrw.StockCode));
}
Block:34795 line:1062~1063
else
Block:34796 line:1063~1065
{
 memcpy(sReportMsg.OrderWthdrw.StockCode, m_stPublicContext.pSecurityInfo->m_stSecurityInfo.SecurityCode, sizeof(sReportMsg.OrderWthdrw.StockCode));
}
Block:34797 line:1065~1066
Block:34798 line:1066~1067
if ( CNST_STOCKKIND_GOLD_ETF_INDEX == m_stPublicContext.nSecurityType )
T F
Block:34799 line:1067~1070
{
 sReportMsg.OrderWthdrw.StockCategory = '4';
}
Block:34802 line:1070~1071 34801 34800
else if ( CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX == m_stPublicContext.nSecurityType )
T F
Block:34803 line:1071~1074
{
 sReportMsg.OrderWthdrw.StockCategory = '6';
}
Block:34806 line:1074~1075 34805 34804
else if ( CNST_STOCKKIND_CURRENCY_ETF_INDEX == m_stPublicContext.nSecurityType )
T F
Block:34807 line:1075~1077
{
 sReportMsg.OrderWthdrw.StockCategory = '5';
}
Block:34811 line:1077~1087
memcpy(sReportMsg.OrderWthdrw.SeatNo, m_stPublicContext.pTraderSeat->m_stSeatField.SeatNo, sizeof (sReportMsg.OrderWthdrw.SeatNo));
memcpy(sReportMsg.OrderWthdrw.BranchFlag, m_stPublicContext.pTraderSeat->m_stSeatField.BranchFlag, sizeof (sReportMsg.OrderWthdrw.BranchFlag));
memcpy(sReportMsg.OrderWthdrw.StockAccount, m_stPublicContext.pStockHolder->m_stStockHolder.StockAccount, sizeof (sReportMsg.OrderWthdrw.StockAccount));
//多码合一之后都是使用综合业务平台申报
int OfferType = 1;
INFO_LOG(lpContext, "报盘编号", wrap("OfferType", OfferType));
Block:34812 line:1087~1089
if ( unlikely((m_stPublicContext.pTraderSeat->GetTransParam(OfferType) == nullptr) || (m_stPublicContext.pTraderSeat->GetTransParam(OfferType)->ReportStatus != CNST_TRANSPARAM_R
T
Block:34813 line:1089~1092
{
 INFO_LOG(lpContext,"未开报盘", wrap("EntrustNo", sReportMsg.OrderWthdrw.EntrustNo));
 return ERR_OK;
}
Block:34815 line:1092~1096
//int32_t nRet = lpContext->PostMsg2Offer(m_stPublicContext.pSeat->GetTransParam(1)->TransIndex, (void *) &sReportMsg, sizeof(STOCK_OFFER_ORDER_WTHDRW_MSG));
int32_t nRet = lpContext->PostMsg2Offer(m_stPublicContext.pTraderSeat->GetTransParam(OfferType)->TransIndex, (void *) &sReportMsg, sizeof(STOCK_OFFER_ORDER_WTHDRW_MSG));
Block:34816 line:1096~1097
if ( nRet == ERR_OK )//组播新方案，不论成功还是失败都置成待报
Block:34817 line:1097~1100
{
 m_stContext.pEntrust->m_stEntField.EntrustStatus = CNST_ENTRUSTSTATUS_WAITREPORT;
}
Block:34818 line:1100~1101
else
Block:34819 line:1101~1103
{
 ERROR_LOG(lpContext, ERR_OFFER_INTERATION, "报单失败", wrap("EntrustNo", sReportMsg.OrderWthdrw.EntrustNo));
}
Block:34820 line:1103~1105
return ERR_OK;
Block:34821 line:1105~1105
}
N_14 : ['StockCode', 'Direction']:
Block:234 line:425~426
int32_t CalcFee(CStockBizContext* lpContext, CAsset* lpAsset, int32_t ExchIndex,int32_t EntrustDirectionIndex, CStockCode* StockCode,HSNum EntrustAmount, USTPrice1 EntrustPrice, int32_t nFareType /*=
Block:236 line:426~432
{
 INFO_LOG(lpContext,"费用计算入参",wrap("ExchIndex", ExchIndex),
 wrap("EntrustDirectionIndex", EntrustDirectionIndex),
 wrap("EntrustAmount", EntrustAmount),
 wrap("EntrustPrice", ConvertDouble(EntrustPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT),
 wrap("nFareType", nFareType));
Block:237 line:432~433
if( EntrustAmount == 0 )
T F
Block:238 line:433~435
{
 return 0;
}
Block:240 line:435~436
Block:241 line:436~437
if ( EntrustDirectionIndex == (CNST_ENTRUSTDIR_APPLY-1) || EntrustDirectionIndex == (CNST_ENTRUSTDIR_REDEEM-1) ) //申赎只计算过户费
T F T F
```


Block:242 line:437~439
{ nFareType = FARE_TYPE_TRANSFER_FEE; }
Block:244 line:439~445
USTBalance BfareBalance = 0;
//SStockFeeField* pBfare = NULL;
USTBalance CalcBalanceSum = 0;
USTNumID StockType = StockCode->m_stCodeField.StockType;
CStockFeeMgr* pStockFeeMgr = lpContext->m_pDataImpl->GetStockFeeMgr(lpAsset->m_stAssetField.SysNo);
Block:245 line:445~446
if(pStockFeeMgr == nullptr)
Block:246 line:446~450
{ ///报错返回 WARNING_LOG(lpContext, "not find CStockFeeMgr. SysNo", wrap(lpAsset->m_stAssetField.SysNo)); return 0; }
Block:248 line:450~454
auto pVec = pStockFeeMgr->GetStockFees(ExchIndex, StockType, lpAsset->m_stAssetField.FundID, lpAsset->m_stAssetField.AssetID, EntrustDirectionIndex, StockCode->m_stCodeField.StockCode, getStockFeeType(StockCode->m_stCodeField.StockProperty));
Block:249 line:454~455
if(likely(nullptr != pVec))
Block:250 line:455~456
{
Block:251 line:456~457
for(auto pBfare : *pVec)
Block:252 line:457~458
{
Block:253 line:458~458
if(nullptr == pBfare)
Block:254 line:458~458
continue;
Block:256 line:458~465
INFO_LOG(lpContext, "费用计算种类", wrap("FeeType", pBfare->FeeType), wrap("FeeId", pBfare->FeeId), wrap("FundID", pBfare->FundID), wrap("AssetID", pBfare->AssetID) , wrap("ExchangeType", pBfare->ExchangeType), wrap("StockFeeType", pBfare->StockFeeType), wrap("BalanceFeeRatio", pBfare->BalanceFeeRatio), wrap("QuantityCost", pBfare->QuantityCost), wrap("FaceValueRatio", pBfare->FaceValueRatio), wrap("StockCode", pBfare->StockCode), wrap("MinFare", pBfare->MinFare), wrap("MaxFare", pBfare->MaxFare), wrap("IsValid", pBfare->IsValid));
int32_t nFeeTypeIndex = 0;
Block:257 line:465~467
if((pBfare->FeeType >= '0' && pBfare->FeeType <= '9') || (pBfare->FeeType >= 'a' && pBfare->FeeType <= 'e'))
Block:258 line:467~470
{ nFeeTypeIndex = pBfare->FeeType - '0' < 10 ? pBfare->FeeType - '0' : pBfare->FeeType - 'a' + 10; }
Block:259 line:470~471
else
Block:260 line:471~473
{ continue;; }
Block:261 line:473~476
//如果指定nFareType，则只需要计算nFareType费用
Block:262 line:476~477
if(nFareType != -1 && nFareType != nFeeTypeIndex)
Block:263 line:477~479
{ continue; }
Block:265 line:479~491
// 买入开仓、买入平仓、备兑平仓、行权计算费用：卖出开仓、卖出平仓、备兑开仓委托不计费用
// 委托金额*成交金额比例+每股费用*委托数量+成交面值比例*委托数量*合约乘数*股份面值
CalcBalanceSum = 0;
//CalcBalanceSum += pBfare->BalanceFeeRatio * EntrustAmount * EntrustPrice / FARE_RATIO_MULTIPLE;
//CalcBalanceSum += (pBfare->QuantityCost * EntrustAmount / FARE_RATIO_MULTIPLE) * PRICE_MULTIPLIER;
//CalcBalanceSum += (pBfare->FaceValueRatio * EntrustAmount / FARE_RATIO_MULTIPLE) * PRICE_MULTIPLIER;
CalcBalanceSum = ((pBfare->BalanceFeeRatio * EntrustAmount * EntrustPrice) + (pBfare->QuantityCost * EntrustAmount) * PRICE_MULTIPLIER + (pBfare->FaceValueRatio * EntrustAmount) * PRICE_MULTIPLIER) ;
Block:266 line:491~492
if ((CalcBalanceSum < pBfare->MinFare * PRICE_MULTIPLIER) && (pBfare->MinFare > 0))
Block:267 line:492~495
{ CalcBalanceSum = pBfare->MinFare * PRICE_MULTIPLIER; }
Block:270 line:495~496 269 268
else if ((CalcBalanceSum > pBfare->MaxFare * PRICE_MULTIPLIER) && (pBfare->MaxFare > 0))
Block:271 line:496~498
{ CalcBalanceSum = pBfare->MaxFare * PRICE_MULTIPLIER; }
Block:274 line:498~501
INFO_LOG(lpContext, "单项费用", wrap("CalcBalanceSum", CalcBalanceSum));
BfareBalance += CalcBalanceSum;
}

```
Block:277
    line:501~502    }

Block:279
    line:502~507    /// 四舍五入第五位小数
                    USTDoubleBalance dfare = ConvertDouble(BfareBalance , FARE_RATIO_MULTIPLE, BALANCE_DECIMAL_DIGIT);
                    bool flag = false;
                    USTBalance ifare = ConvertInt(dfare, &flag, 1.0);

Block:280
    line:507~508    if (    flag    )
                        T   -

Block:281
    line:508~511    {
                     BfareBalance = ifare;
                    }

Block:282
    line:511~512    else

Block:283
    line:512~514    {
                     BfareBalance = BfareBalance / FARE_RATIO_MULTIPLE;
                    }

Block:284
    line:514~518    INFO_LOG(lpContext,"费用计算出参", wrap("BfareBalance", ConvertDouble(BfareBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT));
                    return BfareBalance;

Block:285
    line:518~518    }
```

N_3232 : ['Direction', 'OrderPrice', 'OrderCommand']:

```
Block:36648
    line:1113~1113  void CRuleProcessor::SetEntrustInfo(CRiskOrderInfo *pOrderInfo, SRuleWarningInfoField *pRecord)

Block:36650
    line:1113~1133  {
                     pRecord->ExternSysNo      = pOrderInfo->m_iSysNo;
                     pRecord->ExternReportNo   = pOrderInfo->m_iReportNo;
                     // EntrustNo将会在触警落库时再赋值，这里先赋值为-1表示无效
                     pRecord->EntrustNo            = -1;
                     pRecord->RiskEntrustNo        = pOrderInfo->m_iRiskEntrustNo;
                     // SerialNo在pRecord的GetNewRecord中已赋值
                     strncpy(pRecord->SecurityCode, pOrderInfo->m_pSecurityInfo->m_stSecurityInfo.SecurityCode, sizeof(USTSecurityCode));
                     strncpy(pRecord->SecurityName, pOrderInfo->m_pSecurityInfo->m_stSecurityInfo.SecurityName, sizeof(USTSecurityCode));
                     pRecord->BusinessDate         = pOrderInfo->mCurrDate;
                     pRecord->CurrDate             = pOrderInfo->mCurrDate;
                     pRecord->EntrustDate          = pOrderInfo->mCurrDate;
                     pRecord->EntrustTime          = pOrderInfo->mCurrTime;
                     pRecord->EntrustDirection     = pOrderInfo->m_iOriginEntrustDirection;
                     strncpy(pRecord->ExchangeType, pOrderInfo->m_pSecurityInfo->m_stSecurityInfo.ExchangeType, sizeof(pRecord->ExchangeType));
                     pRecord->Sharding = m_lpContext->m_pDataImpl->m_pRiskData->m_iShardingNo;

Block:36651
    line:1133~1135  if (    pRecord->RuleInternalId < pOrderInfo->m_RuleCalcResult.size()    &&    pOrderInfo->m_RuleCalcResult[pRecord->RuleInternalId].size()    )
                        -   F   -   -
                        -   F

Block:36652
    line:1135~1144  {
                     auto& vec = pOrderInfo->m_RuleCalcResult[pRecord->RuleInternalId];
                     pRecord->RiskContraEntrustSysNo    = vec[0];
                     pRecord->RiskContraEntrustReportNo = vec[1];
                     pRecord->LayerId                   = vec[2];
                     pRecord->ContraCombiId             = vec[2];
                     pRecord->ContraEntrustPrice            = ConvertDouble(vec[3], PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);
                     pRecord->ContraEntrustAmount           = vec[4];
                    }

Block:36653
    line:1144~1145  else

Block:36654
    line:1145~1152  {
                     pRecord->RiskContraEntrustSysNo    = 0;
                     pRecord->RiskContraEntrustReportNo = 0;
                     pRecord->LayerId                   = 0;
                     pRecord->ContraCombiId             = 0;
                     pRecord->ContraEntrustPrice            = 0;
                     pRecord->ContraEntrustAmount           = 0;
                    }

Block:36655
    line:1152~1159  pRecord->EntrustAmount = pOrderInfo->m_iEntrustAmount;
                    pRecord->OffsetFlag = '-';
                    pRecord->HedgeType = HS_HT_Speculation;
                    pRecord->PriceType = pOrderInfo->m_nOrderCommand;
                    pRecord->EntrustPrice = ConvertDouble(pOrderInfo->m_nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);

Block:36656
    line:1159~1159  }
```

```
Block:36405
    line:696~697    int32_t CRuleProcessor::RiskWarnJudge(int32_t iOrderIndex, SRuleProcessorMember* lpMember)

Block:36407
    line:697~702    {
                     int iRet = 0;
                     USTType Operation = '0'; //触警操作
                     USTType WarnLevel = '0'; //触警阈值挡位

Block:36408
    line:702~703    if (    lpMember->m_lpRuleSet->m_stRuleSetField.RuleType == 'I'    )
                        T   F

Block:36409
    line:703~704    {

Block:36410
    line:704~705    if (    lpMember->m_dFormulaValue == 1    )
                        T   -

Block:36411
    line:705~708        {
                         Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1;
                         WarnLevel = '1';
                        }

Block:36413
    line:708~710        goto svr_end;
                    }

Block:36415
    line:710~712

Block:36416
    line:712~713    if (    lpMember->m_lpRuleSet->m_stRuleSetField.RuleType == 'J'    )
                        -   F

Block:36417
    line:713~714    {
```


Block:36418 line:714~715
if (lpMember->m_dFormulaValue == ERR_RISK_WARN)
Block:36419 line:715~718
{ Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1; WarnLevel = '1'; }
Block:36421 line:718~720
goto svr_end; }
Block:36423 line:720~722
Block:36424 line:722~723
if (lpMember->m_lpRuleSet->m_pRuleSetBak->m_bIsFormula) // 一般是Z类风控
F
Block:36425 line:723~724
{
Block:36426 line:724~725
if (lpMember->m_bLogicalResult) // 逻辑表达式结果，没有那么多挡位，结果为真即触警
Block:36427 line:725~728
{ Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1; WarnLevel = '1'; }
Block:36429 line:728~730
goto svr_end; }
Block:36431 line:730~733
Block:36432 line:733~735
switch (lpMember->m_lpRuleSet->m_stRuleSetField.CompareDirection)
{
Block:36433 line:735~735
case LARGER_THAN:
T F
Block:36434 line:735~737
//大于
{
Block:36435 line:737~738
if (FBGEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue1)) // >阈值1
F
Block:36436 line:738~742
{ Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1; WarnLevel = '1';
Block:36437 line:742~743
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1 == '2')
Block:36438 line:743~745
{ goto svr_end; }
Block:36440 line:745~747
Block:36441 line:747~748
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 != RISK_OPERATION_DEFAUL_VALUE) //有设阈值2
Block:36442 line:748~749
{
Block:36443 line:749~750
if (FBGEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue2)) // >阈值2
Block:36444 line:750~754
{ Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2; WarnLevel = '2';
Block:36445 line:754~755
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 == '2')
Block:36446 line:755~757
{ goto svr_end; }
Block:36448 line:757~759
Block:36449 line:759~760
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 != RISK_OPERATION_DEFAUL_VALUE) //有设阈值3
Block:36450 line:760~761
{
Block:36451 line:761~762
if (FBGEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue3)) // >阈值3
Block:36452 line:762~766
{ Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3; WarnLevel = '3';
Block:36453 line:766~767
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 == '2')


Block:36454 line:767~769
{ goto svr_end; }
Block:36456 line:769~770
}
Block:36458 line:770~771
}
Block:36460 line:771~772
}
Block:36462 line:772~773
}
Block:36464 line:773~774
}
Block:36466 line:774~777
break; }
Block:36467 line:777~777
case LOWER_THAN:
Block:36468 line:777~779
//小于 {
Block:36469 line:779~780
if (FLSEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue1)) // <阈值1
Block:36470 line:780~784
{ Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1; WarnLevel = '1';
Block:36471 line:784~785
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1 == '2')
Block:36472 line:785~787
{ goto svr_end; }
Block:36474 line:787~789
Block:36475 line:789~790
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 != RISK_OPERATION_DEFAUL_VALUE) //有设置阈值2
Block:36476 line:790~791
{
Block:36477 line:791~792
if (FLSEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue2)) // <阈值2
Block:36478 line:792~796
{ Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2; WarnLevel = '2';
Block:36479 line:796~797
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 == '2')
Block:36480 line:797~799
{ goto svr_end; }
Block:36482 line:799~801
Block:36483 line:801~802
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 != RISK_OPERATION_DEFAUL_VALUE) //有设置阈值3
Block:36484 line:802~803
{
Block:36485 line:803~804
if (FLSEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue3)) //<阈值3
Block:36486 line:804~809
{ //阈值3触警结果落地 Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3; WarnLevel = '3';
Block:36487 line:809~810
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 == '2')
Block:36488 line:810~812
{ goto svr_end; }
Block:36490 line:812~813
}
Block:36492 line:813~814
}
Block:36494 line:814~815
}
Block:36496 line:815~816
}
Block:36498 line:816~817
}
Block:36500 line:817~820
break; }


Block:36501 line:820~820
case LAGER_EQUAL_THAN:
T F
Block:36502 line:820~822
//大于等于
{
Block:36503 line:822~823
if (! FLSEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue1)) // >=阈值1
T -
- F
Block:36504 line:823~827
{
Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1;
WarnLevel = '1';
Block:36505 line:827~828
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1 == '2')
- -
Block:36506 line:828~830
{
goto svr_end;
}
Block:36508 line:830~832
Block:36509 line:832~833
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 != RISK_OPERATION_DEFAUL_VALUE) //有设置阈值2
- -
Block:36510 line:833~834
{
Block:36511 line:834~835
if (! FLSEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue2)) // >=阈值2
- -
- -
Block:36512 line:835~839
{
Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2;
WarnLevel = '2';
Block:36513 line:839~840
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 == '2')
- -
Block:36514 line:840~842
{
goto svr_end;
}
Block:36516 line:842~844
Block:36517 line:844~845
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 != RISK_OPERATION_DEFAUL_VALUE) //有设置阈值3
- -
Block:36518 line:845~846
{
Block:36519 line:846~847
if (! FLSEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue3)) // >=阈值3
- -
- -
Block:36520 line:847~851
{
Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3;
WarnLevel = '3';
Block:36521 line:851~852
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 == '2')
- -
Block:36522 line:852~854
{
goto svr_end;
}
Block:36524 line:854~855
}
Block:36526 line:855~856
}
Block:36528 line:856~857
}
Block:36530 line:857~858
}
Block:36532 line:858~859
}
Block:36534 line:859~862
break;
}
Block:36535 line:862~862
case LOWER_EQUAL_THAN:
- F
Block:36536 line:862~864
//小于等于
{
Block:36537 line:864~865
if (! FBGEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue1)) // <=阈值1


Block:36538 line:865~869
{
Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1;
WarnLevel = '1';
Block:36539 line:869~870
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation1 == '2')
Block:36540 line:870~872
{
goto svr_end;
}
Block:36542 line:872~874
Block:36543 line:874~875
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 != RISK_OPERATION_DEFAUL_VALUE) //有设置阈值2
Block:36544 line:875~876
{
Block:36545 line:876~877
if (! FBGEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue2)) // <=阈值2
Block:36546 line:877~881
{
Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2;
WarnLevel = '2';
Block:36547 line:881~882
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation2 == '2')
Block:36548 line:882~884
{
goto svr_end;
}
Block:36550 line:884~886
Block:36551 line:886~887
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 != RISK_OPERATION_DEFAUL_VALUE) //有设置阈值3
Block:36552 line:887~888
{
Block:36553 line:888~889
if (! FBGEX(lpMember->m_dFormulaValue, lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue3)) // <=阈值3
Block:36554 line:889~893
{
Operation = lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3;
WarnLevel = '3';
Block:36555 line:893~894
if (lpMember->m_lpRuleSet->m_stRuleSetField.WarnOperation3 == '2')
Block:36556 line:894~896
{
goto svr_end;
}
Block:36558 line:896~897
}
Block:36560 line:897~898
}
Block:36562 line:898~899
}
Block:36564 line:899~900
}
Block:36566 line:900~901
}
Block:36568 line:901~904
break;
}
Block:36569 line:904~904
default:
F
Block:36570 line:904~909
{
//规则比较类型设置错误,默认不控
return 0;
}
}
Block:36571 line:909~911
Block:36572 line:911~911
svr_end:
Block:36573 line:911~912
Block:36574 line:912~913
if (Operation != '0')
T
F
Block:36575
{

line:913~914	
Block:36576 line:914~915	if (lpMember->m_bStockCollect)
	- F

Block:36577
line:915~920
	{
	m_lpContext->m_pRiskContext->m_sBufAllStock.m_szbRuleWarning[lpMember->m_iRuleId] = true;
	m_lpContext->m_pRiskContext->m_sBufAllStock.m_szWarnRecordBuff[lpMember->m_iRuleId].Operation = Operation;
	m_lpContext->m_pRiskContext->m_sBufAllStock.m_szWarnRecordBuff[lpMember->m_iRuleId].WarnLevel = WarnLevel;
	}

Block:36578 line:920~921	else

Block:36579
line:921~925
	{
	m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szbRuleWarning[lpMember->m_iRuleId][lpMember->m_iStockIndex] = true;
	m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szWarnRecordBuff[lpMember->m_iRuleId][lpMember->m_iStockIndex].Operation = Operation;
	m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szWarnRecordBuff[lpMember->m_iRuleId][lpMember->m_iStockIndex].WarnLevel = WarnLevel;
	}

Block:36580 line:925~931	m_lpContext->m_pRiskContext->m_bIsHasWanningInfo = true; WarnInfoLanding(Operation, WarnLevel, iOrderIndex, lpMember);
Block:36581 line:931~932	if (Operation == '2')
	- F

Block:36582
line:932~940
	{
	iRet = ERR_RISK_WARN;
	//设置委托与篮子的禁止标志位为true
	m_lpContext->m_pRiskContext->m_szbOrderForbid[lpMember->m_iOrderIndex] = true;
	m_lpContext->m_pRiskContext->m_bBucketForbid = true;
	//设置发生阻断时的委托下标与规则id
	m_lpContext->m_pRiskContext->m_iForbidOrderIndex = iOrderIndex;
	m_lpContext->m_pRiskContext->m_iForbidRuleId = lpMember->m_iRuleId;
	}

Block:36584 line:940~942	}
Block:36585 line:942~943	else //没触警
Block:36586 line:943~944	{
Block:36587 line:944~945	if (lpMember->m_bStockCollect)
	- F

Block:36588
line:945~948
	{
	m_lpContext->m_pRiskContext->m_sBufAllStock.m_szbRuleWarning[lpMember->m_iRuleId] = false;
	}

Block:36589 line:948~949	else

Block:36590
line:949~951
	{
	m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szbRuleWarning[lpMember->m_iRuleId][lpMember->m_iStockIndex] = false;
	}

Block:36591 line:951~954	iRet = 0; }
Block:36592 line:954~957	return iRet;
Block:36593 line:957~957	}

N_3227 : ['Direction']:
N_3233 : ['Direction']:

Block:36657 line:1161~1162	void CRuleProcessor::SetRuleInfo(const USTType &Operation, const USTType &WarnLevel, const SRuleProcessorMember *lpMember, SRuleWarningInfoField *pRecord)

Block:36659
line:1162~1182
	{
	const SRuleSetField &field = lpMember->m_lpRuleSet->m_stRuleSetField;
	pRecord->RuleType = field.RuleType; //规则类型
	pRecord->RuleSubType = field.RuleSubType; //规则控制子类型
	pRecord->RuleId = field.RuleId; //规则id
	pRecord->RuleInternalId = field.RuleInternalId;
	pRecord->DisplayUnit = field.DisplayUnit;
	snprintf(pRecord->RiskSummary, sizeof(pRecord->RiskSummary),
	"%s",
	lpMember->m_lpRuleSet->m_pRuleSetBak->m_stRuleSetBakField.RiskSummary); //规则概要
	snprintf(pRecord->RuleSummary, sizeof(pRecord->RuleSummary),
	"%s",
	lpMember->m_lpRuleSet->m_pRuleSetBak->m_stRuleSetBakField.RiskSummary); //规则概要
	pRecord->Operation = Operation; //触警操作
	pRecord->WarnLevel = WarnLevel; //触警挡位
	pRecord->RiskThreshold = (WarnLevel == '1') ? field.CompareValue1
	: (WarnLevel == '2') ? field.CompareValue2 : field.CompareValue3;
	//从上下文缓存获取的数据

Block:36660 line:1182~1183	if (lpMember->m_bStockCollect)
	- F

Block:36661
line:1183~1189
	{
	pRecord->NumeratorCalValue = m_lpContext->m_pRiskContext->m_sBufAllStock.m_szdNumeratorCalValue[lpMember->m_iRuleId]; //分子
	pRecord->DenominatorCalValue = m_lpContext->m_pRiskContext->m_sBufAllStock.m_szdDenominatorCalValue[lpMember->m_iRuleId]; //分母
	pRecord->RuleCalcResult = m_lpContext->m_pRiskContext->m_sBufAllStock.m_szdFormulaValue[lpMember->m_iRuleId]; //公式值
	pRecord->CalcValue = m_lpContext->m_pRiskContext->m_sBufAllStock.m_szdFormulaValue[lpMember->m_iRuleId]; //公式值
	}

Block:36662 line:1189~1190	else

Block:36663
line:1190~1195
	{
	pRecord->NumeratorCalValue = m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szdNumeratorCalValue[lpMember->m_iRuleId][lpMember->m_iStockIndex]; //分子
	pRecord->DenominatorCalValue = m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szdDenominatorCalValue[lpMember->m_iRuleId][lpMember->m_iStockIndex]; //分母
	pRecord->RuleCalcResult = m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szdFormulaValue[lpMember->m_iRuleId][lpMember->m_iStockIndex]; //公式值
	pRecord->CalcValue = m_lpContext->m_pRiskContext->m_sBufAllStock.m_szdFormulaValue[lpMember->m_iRuleId]; //公式值
	}

Block:36664
line:1195~1206
	pRecord->CompareDirection = field.CompareDirection; //比较方向
	pRecord->CompareValue1 = field.CompareValue1; //比较阀值1
	pRecord->CompareValue2 = field.CompareValue2; //比较阀值2
	pRecord->CompareValue3 = field.CompareValue3; //比较阀值3
	pRecord->WarnOperation1 = field.WarnOperation1; //阀值1触警操作
	pRecord->WarnOperation2 = field.WarnOperation2; //阀值2触警操作
	pRecord->WarnOperation3 = field.WarnOperation3; //阀值3触警操作
	pRecord->ControlLayer = field.ControlLayer; //控制层次

Block:36665 line:1206~1208	switch(pRecord->ControlLayer) {

```
Block:36666 line:1208~1208
    case '1':
        -        F
Block:36667 line:1208~1211
    strncpy(pRecord->ControlLayerName, u8"公司", sizeof(pRecord->ControlLayerName));
    break;
Block:36668 line:1211~1211
    case '2':
        -        F
Block:36669 line:1211~1214
    strncpy(pRecord->ControlLayerName, u8"部门", sizeof(pRecord->ControlLayerName));
    break;
Block:36670 line:1214~1214
    case '3':
        T        F
Block:36671 line:1214~1217
    strncpy(pRecord->ControlLayerName, u8"产品", sizeof(pRecord->ControlLayerName));
    break;
Block:36672 line:1217~1217
    case '4':
        -        F
Block:36673 line:1217~1220
    strncpy(pRecord->ControlLayerName, u8"组合", sizeof(pRecord->ControlLayerName));
    break;
Block:36674 line:1220~1220
    case '5':
        T        F
Block:36675 line:1220~1223
    strncpy(pRecord->ControlLayerName, u8"单元", sizeof(pRecord->ControlLayerName));
    break;
Block:36676 line:1223~1223
    case '9':
        -        F
Block:36677 line:1223~1226
    strncpy(pRecord->ControlLayerName, u8"股东", sizeof(pRecord->ControlLayerName));
    break;
Block:36678 line:1226~1226
    case 'a':
        -        F
Block:36679 line:1226~1229
    strncpy(pRecord->ControlLayerName, u8"对接系统", sizeof(pRecord->ControlLayerName));
    break;
  }
Block:36681 line:1229~1234
  pRecord->LayerUnion   = field.LayerUnion;              //层次联合
  pRecord->StockCollect = field.StockCollect;            //证券汇总
  pRecord->ControlType  = field.ControlType;             //控制方式
Block:36682 line:1234~1234
}
```

```
Block:36367 line:588~589
int32_t CRuleProcessor::FactorsRelativeCheck(int32_t OrderIndex, SRuleProcessorMember* lpMember)
Block:36369 line:589~595
{
  // 由于本类被各种规则继承，特殊规则的相关性特化可以考虑放到子类 override 本方法改为 virtual
  // 本基类实现 只考虑最基本的单因子处理 供参考 hrh30668
  int32_t iRet = 0;
  CRuleFactor* lpRuleFactor;
Block:36370 line:595~596
  if (    lpMember->m_lpRuleSet->m_stRuleSetField.RuleType == 'A'    )
              -                                                F
Block:36371 line:596~598
  {
    //逐个因子处理
Block:36372 line:598~599
  for (int32_t i = 0;    i < lpMember->m_lpRuleSet->m_iCount    ; ++i)
                              -                          -
Block:36373 line:599~617
    {
      lpRuleFactor = &lpMember->m_lpRuleSet->m_arrRiskFactor[i];
      USTNumID iFactorId = lpRuleFactor->m_stSRiskFactorField.FactorInternalId;
      DEBUG_LOG(m_lpContext,
      "\n==================FactorsRelativeCheck!==========",
      "\n\t 当前因子：  ", wrap(i),
      "\n\t FactorDirection: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorDirection),
      "\n\t FactorId: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorId),
      "\n\t FactorInternalId: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorInternalId),
      "\n\t FactorMode: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorMode),
      "\n\t FactorType: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorType),
      "\n\t RuleId: ", wrap(lpRuleFactor->m_stSRiskFactorField.RuleId),
      "\n**********************************************\n\n"
      );
      //因子相关性判断
      iRet = m_pFactorsProcMgr->FactorParamRelativeCheck(lpRuleFactor, lpMember->m_lpRuleSet, OrderIndex);
Block:36374 line:617~618
    if (    1 == iRet    )//因子相关性判断校验通过再进行预汇总计算
                -      -
Block:36375 line:618~624
    {
      //PreCalc(OrderIndex, lpMember->m_lpRuleSet);
      //根据相关性校验结果置规则相关性标志位
      m_lpContext->m_pRiskContext->m_bSzFactorRelevance[OrderIndex][iFactorId] = true;
      //DEBUG_LOG_TRACE("FactorsRelativeCheck orderIndex:%d, factorInternalId:%d, relvance:true\n", OrderIndex, iFactorId );
    }
```

Block:36376 line:624~625	else

Block:36377
line:625~629
	{
	//根据相关性校验结果置规则相关性标志位
	m_lpContext->m_pRiskContext->m_bSzFactorRelevance[OrderIndex][iFactorId] = false;
	//DEBUG_LOG_TRACE("FactorsRelativeCheck orderIndex:%d, factorInternalId:%d, relvance:false\n", OrderIndex, iFactorId);

Block:36378 line:629~630	if (lpMember->m_lpRuleSet->m_pRuleSetBak->m_bIsFormula)

Block:36379
line:630~634
	{ // 对于规则结果靠因子逻辑运算给出的，因子不相关给相应标志置为false，然后继续检查，并不退出
	iRet = 1;
	continue;
	}

Block:36380 line:634~635	else

Block:36381
line:635~637
	{ // 一般的因子相关性检查在这里，不相关就直接退出了
	break;
	}

Block:36382 line:637~639	}
Block:36383 line:639~640	}
Block:36386 line:640~642	}
Block:36387 line:642~643	else

Block:36388
line:643~661
	{//目前只考虑单个因子处理
	lpRuleFactor = &lpMember->m_lpRuleSet->m_arrRiskFactor[0];
	USTNumID iFactorId = lpRuleFactor->m_stSRiskFactorField.FactorInternalId;
	DEBUG_LOG(m_lpContext,
	"\n==========FactorsRelativeCheck!==========",
	"\n\t 当前因子： ", wrap(0),
	"\n\t FactorDirection: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorDirection),
	"\n\t FactorId: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorId),
	"\n\t FactorInternalId: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorInternalId),
	"\n\t FactorMode: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorMode),
	"\n\t FactorType: ", wrap(lpRuleFactor->m_stSRiskFactorField.FactorType),
	"\n\t RuleId: ", wrap(lpRuleFactor->m_stSRiskFactorField.RuleId),
	"\n**********\n\n"
	);
	//因子相关性判断
	iRet = m_pFactorsProcMgr->FactorParamRelativeCheck(lpRuleFactor, lpMember->m_lpRuleSet, OrderIndex);

Block:36389 line:661~662	if (1 == iRet)//因子相关性判断校验通过再进行预汇总计算 T F

Block:36390
line:662~666
	{
	//PreCalc(OrderIndex, lpMember->m_lpRuleSet);
	//根据相关性校验结果置规则相关性标志位
	m_lpContext->m_pRiskContext->m_bSzFactorRelevance[OrderIndex][iFactorId] = true;
	}

Block:36391 line:666~667	else

Block:36392
line:667~670
	{
	//根据相关性校验结果置规则相关性标志位
	m_lpContext->m_pRiskContext->m_bSzFactorRelevance[OrderIndex][iFactorId] = false;
	}

Block:36393 line:670~671	}
Block:36394 line:671~674	return iRet;
Block:36395 line:674~674	}

N_3225 : ['Direction']:

N_3231 : ['StockCode']:

Block:36636 line:1069~1070	int32_t CRuleProcessor::WarnInfoLanding(USTType& Operation, USTType& WarnLevel, int32_t iOrderIndex, SRuleProcessorMember* lpMember)

Block:36638
line:1070~1073
	{
	int32_t iRet = 0;
	CRiskOrderInfo* pOrderInfo = &(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex]);

Block:36639 line:1073~1074	if (m_lpContext->m_pDataImpl->m_pRiskData->m_iWriteRiskWarn == 1) T -

Block:36640
line:1074~1077
	{
	//获取一条触警记录指针
	SRuleWarningInfoField* pRecord = m_lpContext->m_pDataImpl->GetNewRecord();

Block:36641 line:1077~1078	if (pRecord == NULL) - F

Block:36642
line:1078~1082
	{
	m_lpContext->m_nErrorNo = ERR_USER_ADD_TABLERECORD_FAIL;
	ERROR_LOG(m_lpContext, ERR_USER_ADD_TABLERECORD_FAIL, "触警信息内存分配失败");
	return m_lpContext->m_nErrorNo;
	}

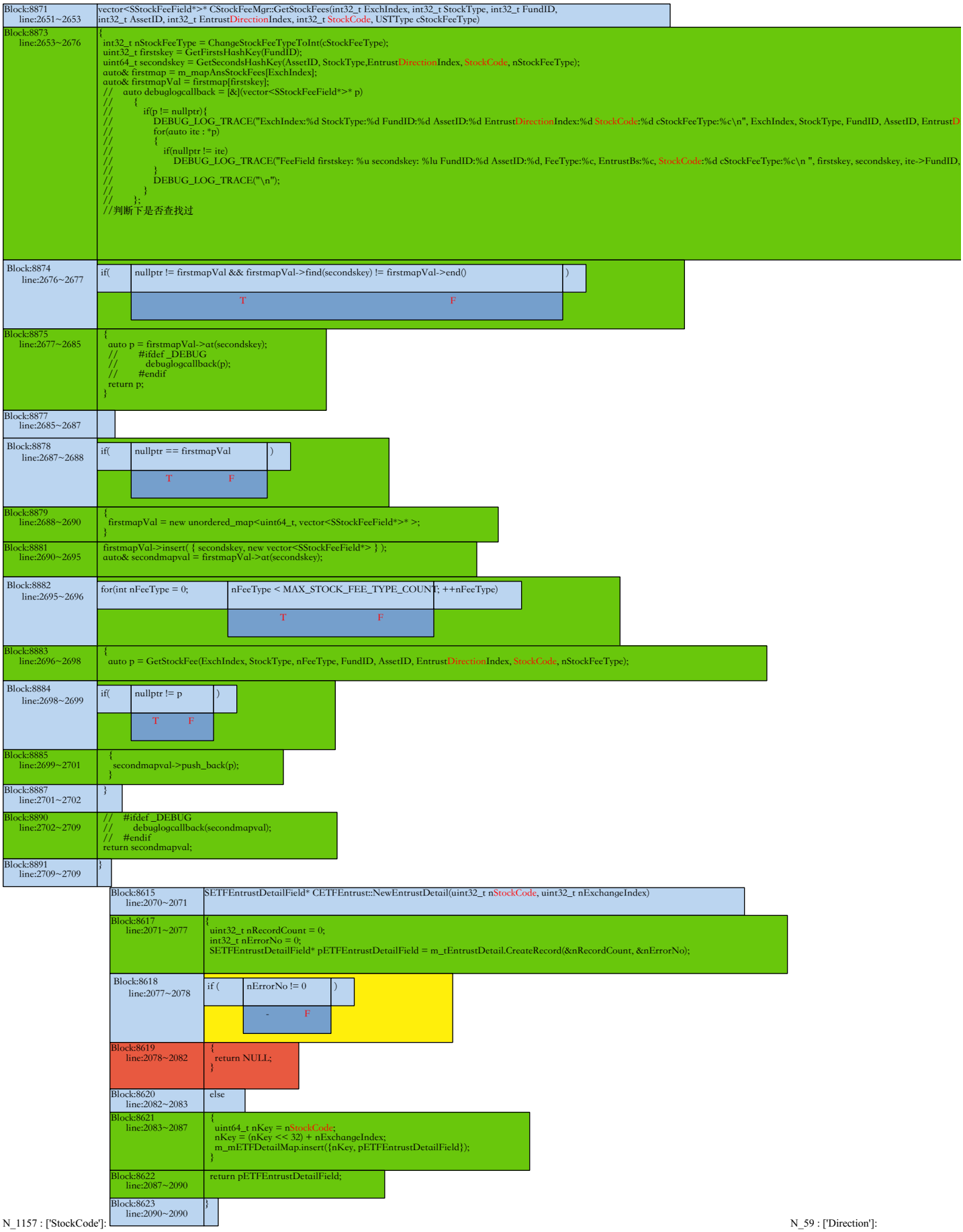
Block:36644
line:1082~1108
	SetLayerInfo(pOrderInfo, pRecord);
	SetRuleInfo(Operation, WarnLevel, lpMember, pRecord);
	SetEntrustInfo(pOrderInfo, pRecord);
	SetContraOrderInfo(pOrderInfo, lpMember->m_lpRuleSet, pRecord);
	// auto iExternSysType = m_lpContext->m_pDataImpl->m_pRiskData->m_iRiskWarnExternSysType;
	// if (iExternSysType == EXTERNAL_SYS_O32)
	// {
	// WarnInfoToDB(m_lpContext, iOrderIndex, pRecord);
	// }
	//风控触警禁止条数统计
	StatisticForbidWarningInfo(Operation, pRecord, m_lpContext->m_pDataImpl->m_pRiskData);
	WarnInfoLanding2Rsp(Operation, WarnLevel, iOrderIndex, lpMember, pRecord);
	INFO_LOG(m_lpContext,
	"\n\n\n**********",GetWarnOperation(Operation),"**********\n"
	"A Risk Warning Record is Created! The Detail is below:"
	"\n\t StockCode: ", wrap(pRecord->SecurityCode),
	"\n\t Rule Id: ", wrap(lpMember->m_lpRuleSet->m_stRuleSetField.RuleId),
	"\n\t Operation: ", wrap(Operation),
	"\n\t Warn Level: ", wrap(WarnLevel),
	"\n\t Set Value: ", wrap(lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue1), wrap(lpMember->m_lpRuleSet->m_stRuleSetField.CompareValue2), wrap(lpMember->m_lpRuleSet->m_stRuleSetFiel
	"\n\t Calc Value: ", wrap(pRecord->RuleCalcResult),
	"\n**********\n\n\n"
	);
	}

Block:36646 line:1108~1111	return iRet;
Block:36647 line:1111~1111	}

N_64 : ['StockCode', 'SeatNo', 'OrderCommand']:

```
Block:1024 line:2360~2361
int32_t OfferStockOrder(CStockBizContext* lpContext, CStockEntrust& pEntrust)
Block:1026 line:2361~2385
{
 STOCK_OFFER_ORDER_WTHDRW_MSG sReportMsg;
 sReportMsg.MsgHead.FunctionID = FUNC_OFFER_REPORT;
 sReportMsg.MsgHead.UserDefined = 0; //普通交易写死0，报盘根据这个识别什么业务
 auto& orderWthdrw = sReportMsg.OrderWthdrw;
 auto& stEntField = pEntrust.m_stEntField;
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 94);
 orderWthdrw.ReportPrice = ConvertDouble(stEntField.EntrustPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);
 orderWthdrw.ReportAmount = stEntField.EntrustAmount;
 orderWthdrw.OrderCommand = stEntField.OrderCommand;
 orderWthdrw.EntrustNo = stEntField.EntrustNo;
 memcpy(orderWthdrw.ExchangeType, stEntField.ExchangeType, sizeof (orderWthdrw.ExchangeType));
 orderWthdrw.EntrustBs = stEntField.EntrustBs;
 memcpy(orderWthdrw.StockAccount, stEntField.StockAccount, sizeof (orderWthdrw.StockAccount));
 itoa(stEntField.StockCode, orderWthdrw.StockCode, 10, 6);
 memcpy(orderWthdrw.SeatNo, stEntField.SeatNo, sizeof (orderWthdrw.SeatNo));
 orderWthdrw.OrigEntrustNo = 0;
 orderWthdrw.RecordNo = 0;
 orderWthdrw.BusinFlag = CNST_BUSINFLAG_BUYSALE;
 orderWthdrw.EntrustType = CNST_ENTRUSTTYPE_NORMAL;
 sReportMsg.OrderWthdrw.ExecType = '\0';  //买卖业务 不需要
 int OfferType = 0;
Block:1027 line:2385~2386
if (  pEntrust.pSecurityInfo->m_stSecurityInfo.SecurityType == CNST_STOCKKIND_CONVERTIBLE_BOND_INDEX  )
                                                -                                                F
Block:1028 line:2386~2388
 {
  orderWthdrw.BusinFlag = CNST_BSUINFLAG_CONVERTIBLE_BOND_INDEX;
Block:1029 line:2388~2389
if (  stEntField.ExchIndex == CNST_EXCHANGETYPE_SHA_INDEX  )
                                -                                -
Block:1030 line:2389~2391
  {
   OfferType = 2;
  }
Block:1032 line:2391~2392
 }
Block:1034 line:2392~2396
CSeat *pSeat = lpContext->m_pDataImpl->GetSeat(orderWthdrw.SeatNo, orderWthdrw.ExchangeType);
Block:1035 line:2396~2398
if (  unlikely((pSeat == nullptr) || (pSeat->GetTransParam(OfferType) == nullptr) ||       (pSeat->GetTransParam(OfferType)->ReportStatus != CNST_TRANSPARAM_REPORTSTATUS))
                                                T                                                                                                                -
Block:1036 line:2398~2401
 {
  INFO_LOG(lpContext, "未开报盘", wrap("EntrustNo", stEntField.EntrustNo));
  return -1;
 }
Block:1038 line:2401~2406
 memcpy(orderWthdrw.BranchFlag, pSeat->m_stSeatField.BranchFlag, sizeof(orderWthdrw.BranchFlag));
 lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(lpContext->GetThreadNo(), 95);
 int32_t nRet = lpContext->PostMsg2Offer(pSeat->GetTransParam(OfferType)->TransIndex, (void *) &sReportMsg, sizeof(STOCK_OFFER_ORDER_WTHDRW_MSG));
Block:1039 line:2406~2407
if (  unlikely(ERR_OK != nRet)  )
                -                -
Block:1040 line:2407~2410
 {
  ERROR_LOG(lpContext, ERR_OFFER_INTERATION, "报单失败", wrap("EntrustNo", stEntField.EntrustNo));
  return ERR_OFFER_INTERATION;
 }
Block:1042 line:2410~2413
 stEntField.EntrustStatus = CNST_ENTRUSTSTATUS_WAITREPORT;
 return ERR_OK;
Block:1043 line:2413~2413
}
```

N_1181 : ['StockCode', 'Direction']:


Block:8871 line:2651~2653
vector<SStockFeeField*>* CStockFeeMgr::GetStockFees(int32_t ExchIndex, int32_t StockType, int32_t FundID, int32_t AssetID, int32_t EntrustDirectionIndex, int32_t StockCode, USTType cStockFeeType)
Block:8873 line:2653~2676
{
int32_t nStockFeeType = ChangeStockFeeTypeToInt(cStockFeeType);
uint32_t firstskey = GetFirstsHashKey(FundID);
uint64_t secondskey = GetSecondsHashKey(AssetID, StockType,EntrustDirectionIndex, StockCode, nStockFeeType);
auto& firstmap = m_mapAnsStockFees[ExchIndex];
auto& firstmapVal = firstmap[firstskey];
// auto debuglogcallback = [&](vector<SStockFeeField*>* p)
// {
// if(p != nullptr){
// DEBUG_LOG_TRACE("ExchIndex:%d StockType:%d FundID:%d AssetID:%d EntrustDirectionIndex:%d StockCode:%d cStockFeeType:%c\n", ExchIndex, StockType, FundID, AssetID, EntrustD
// for(auto ite : *p)
// {
// if(nullptr != ite)
// DEBUG_LOG_TRACE("FeeField firstskey: %u secondskey: %lu FundID:%d AssetID:%d, FeeType:%c, EntrustBs:%c, StockCode:%d cStockFeeType:%c\n ", firstskey, secondskey, ite->FundID,
// }
// DEBUG_LOG_TRACE("\n");
// }
// };
//判断下是否查找过
Block:8874 line:2676~2677
if(
nullptr != firstmapVal && firstmapVal->find(secondskey) != firstmapVal->end()
)
T
F
Block:8875 line:2677~2685
{
auto p = firstmapVal->at(secondskey);
// #ifdef _DEBUG
// debuglogcallback(p);
// #endif
return p;
}
Block:8877 line:2685~2687
Block:8878 line:2687~2688
if(
nullptr == firstmapVal
)
T
F
Block:8879 line:2688~2690
{
firstmapVal = new unordered_map<uint64_t, vector<SStockFeeField*>* >;
}
Block:8881 line:2690~2695
firstmapVal->insert({ secondskey, new vector<SStockFeeField*> });
auto& secondmapval = firstmapVal->at(secondskey);
Block:8882 line:2695~2696
for(int nFeeType = 0;
nFeeType < MAX_STOCK_FEE_TYPE_COUNT; ++nFeeType)
T
F
Block:8883 line:2696~2698
{
auto p = GetStockFee(ExchIndex, StockType, nFeeType, FundID, AssetID, EntrustDirectionIndex, StockCode, nStockFeeType);
Block:8884 line:2698~2699
if(
nullptr != p
)
T
F
Block:8885 line:2699~2701
{
secondmapval->push_back(p);
}
Block:8887 line:2701~2702
}
Block:8890 line:2702~2709
// #ifdef _DEBUG
// debuglogcallback(secondmapval);
// #endif
return secondmapval;
Block:8891 line:2709~2709
}
Block:8615 line:2070~2071
SETFEntrustDetailField* CETFEntrust::NewEntrustDetail(uint32_t nStockCode, uint32_t nExchangeIndex)
Block:8617 line:2071~2077
{
uint32_t nRecordCount = 0;
int32_t nErrorNo = 0;
SETFEntrustDetailField* pETFEntrustDetailField = m_tEntrustDetail.CreateRecord(&nRecordCount, &nErrorNo);
Block:8618 line:2077~2078
if (
nErrorNo != 0
)
-
F
Block:8619 line:2078~2082
{
return NULL;
}
Block:8620 line:2082~2083
else
Block:8621 line:2083~2087
{
uint64_t nKey = nStockCode;
nKey = (nKey << 32) + nExchangeIndex;
m_mETFDetailMap.insert({nKey, pETFEntrustDetailField});
}
Block:8622 line:2087~2090
return pETFEntrustDetailField;
Block:8623 line:2090~2090
}
N_1157 : ['StockCode']:
N_59 : ['Direction']:

Block:971 line:2173~2175	SCashEnableFactorField* UpdateCashEnableValueAndCashFactor(CStockBizContext* lpContext, CAsset* pAsset, USTExchangeIndex ExchangeIndex, USTNumID SecurityType, USTEntrustDirection EntrustD USTType BusinProType, USTType SettleCashType, USTBalance UpdateBalance)
Block:973 line:2175~2176	{
Block:974 line:2176~2176	if (0 == UpdateBalance)
	- F
Block:975 line:2176~2176	return ERR_OK;
Block:977 line:2176~2179	int32_t nHashKey = GetCashEnableFormulaHashKey(ExchangeIndex, SecurityType, EntrustDirection, BusinProType, SettleCashType); SCashEnableCfgField* pCashEnableCfgField = lpContext->m_pDataImpl->GetCashEnableCfgFieldByKey(nHashKey);
Block:978 line:2179~2180	if (pCashEnableCfgField == nullptr)
	- F

Block:979
line:2180~2185
	{
	///报错返回
	AMUST_LOGBIZERR(lpContext, ERR_ETF_NOT_FIND_CASH_ENABLE_CFG, wrap("ExchangeIndex:", ExchangeIndex), wrap("SecurityType", SecurityType),
	wrap("EntrustDirection", EntrustDirection), wrap("BusinProType", BusinProType), wrap("SettleCashType", SettleCashType));
	return nullptr;
	}

Block:981
line:2185~2191
	//找到对应更新的资金因子
	uint32_t nUpdateTradeDay = lpContext->m_pDataImpl->GetNextNTradeDay(lpContext->m_pDataImpl->GetSysarg()->m_nInitDate, pCashEnableCfgField->UpdateTnDays); //计算更新到T+N 日期
	SCashEnableFactorField* pCashEnableFactorField = pAsset->GetCashEnableFactorField(nUpdateTradeDay, pCashEnableCfgField->UpdatePoint);

Block:982 line:2191~2192	if (pCashEnableCfgField->CashChangeType == '+')
	- F

Block:983
line:2192~2195
	{
	pCashEnableFactorField->CashFactorValue += UpdateBalance;
	}

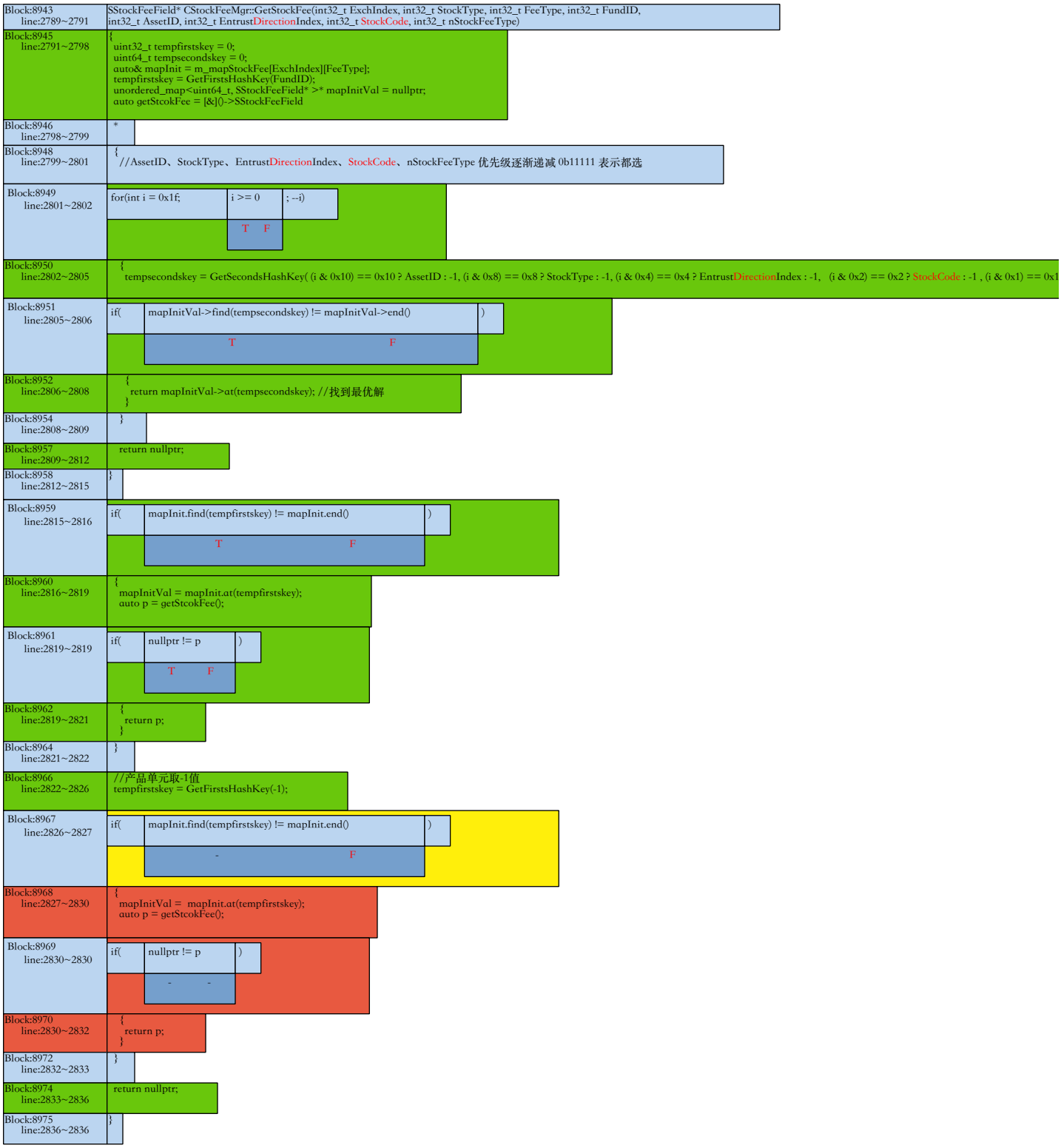
Block:986 line:2195~2196 985 984	else if (pCashEnableCfgField->CashChangeType == '-')
	T -

Block:987
line:2196~2198
	{
	pCashEnableFactorField->CashFactorValue -= UpdateBalance;
	}

Block:990 line:2198~2204	INFO_LOG(lpContext, "资金可用更新" , wrap("SettlePoint", pCashEnableFactorField->SettlePoint) , wrap("SettleTnDays", pCashEnableFactorField->SettleTnDays) , wrap("CashFactorValue", ConvertDouble(pCashEnableFactorField->CashFactorValue, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT),BALANCE_DECIMAL_DIGIT)); return pCashEnableFactorField;
Block:991 line:2204~2204	}

N_3234 : ['SeatNo']:

N_1184 : ['StockCode', 'Direction']:


Block:8943 line:2789~2791
SStockFeeField* CStockFeeMgr::GetStockFee(int32_t ExchIndex, int32_t StockType, int32_t FeeType, int32_t FundID, int32_t AssetID, int32_t EntrustDirectionIndex, int32_t StockCode, int32_t nStockFeeType)
Block:8945 line:2791~2798
{
uint32_t tempfirstskey = 0;
uint64_t tempsecondskey = 0;
auto& mapInit = m_mapStockFee[ExchIndex][FeeType];
tempfirstskey = GetFirstsHashKey(FundID);
unordered_map<uint64_t, SStockFeeField* >* mapInitVal = nullptr;
auto getStcokFee = [&]()->SStockFeeField
Block:8946 line:2798~2799
*
Block:8948 line:2799~2801
{
//AssetID、StockType、EntrustDirectionIndex、StockCode、nStockFeeType 优先级逐渐递减 0b11111 表示都选
Block:8949 line:2801~2802
for(int i = 0x1f;
i >= 0
; --i)
T
F
Block:8950 line:2802~2805
{
tempsecondskey = GetSecondsHashKey((i & 0x10) == 0x10 ? AssetID : -1, (i & 0x8) == 0x8 ? StockType : -1, (i & 0x4) == 0x4 ? EntrustDirectionIndex : -1, (i & 0x2) == 0x2 ? StockCode : -1 , (i & 0x1) == 0x1
Block:8951 line:2805~2806
if(
mapInitVal->find(tempsecondskey) != mapInitVal->end()
)
T
F
Block:8952 line:2806~2808
{
return mapInitVal->at(tempsecondskey); //找到最优解
}
Block:8954 line:2808~2809
}
Block:8957 line:2809~2812
return nullptr;
Block:8958 line:2812~2815
}
Block:8959 line:2815~2816
if(
mapInit.find(tempfirstskey) != mapInit.end()
)
T
F
Block:8960 line:2816~2819
{
mapInitVal = mapInit.at(tempfirstskey);
auto p = getStcokFee();
Block:8961 line:2819~2819
if(
nullptr != p
)
T
F
Block:8962 line:2819~2821
{
return p;
}
Block:8964 line:2821~2822
}
Block:8966 line:2822~2826
//产品单元取-1值
tempfirstskey = GetFirstsHashKey(-1);
Block:8967 line:2826~2827
if(
mapInit.find(tempfirstskey) != mapInit.end()
)
-
F
Block:8968 line:2827~2830
{
mapInitVal = mapInit.at(tempfirstskey);
auto p = getStcokFee();
Block:8969 line:2830~2830
if(
nullptr != p
)
-
-
Block:8970 line:2830~2832
{
return p;
}
Block:8972 line:2832~2833
}
Block:8974 line:2833~2836
return nullptr;
Block:8975 line:2836~2836
}

N_3355 : ['Direction']:
N_1185 : ['StockCode', 'Direction']:

```
Block:8946 line:2798~2799   *
Block:8948 line:2799~2801   {
                              //AssetID、StockType、EntrustDirectionIndex、StockCode、nStockFeeType 优先级逐渐递减 0b11111 表示都选
Block:8949 line:2801~2802   for(int i = 0x1f;  i >= 0  ; --i)
                                                  T   F
Block:8950 line:2802~2805     {
                                tempsecondskey = GetSecondsHashKey( (i & 0x10) == 0x10 ? AssetID : -1, (i & 0x8) == 0x8 ? StockType : -1, (i & 0x4) == 0x4 ? EntrustDirectionIndex : -1,  (i & 0x2) == 0x2 ? StockCode : -1 , (i & 0x1) == 0x1
Block:8951 line:2805~2806     if(  mapInitVal->find(tempsecondskey) != mapInitVal->end()  )
                                                  T                                F
Block:8952 line:2806~2808       {
                                  return mapInitVal->at(tempsecondskey); //找到最优解
                                }
Block:8954 line:2808~2809     }
Block:8957 line:2809~2812     return nullptr;
Block:8958 line:2812~2812   }
```

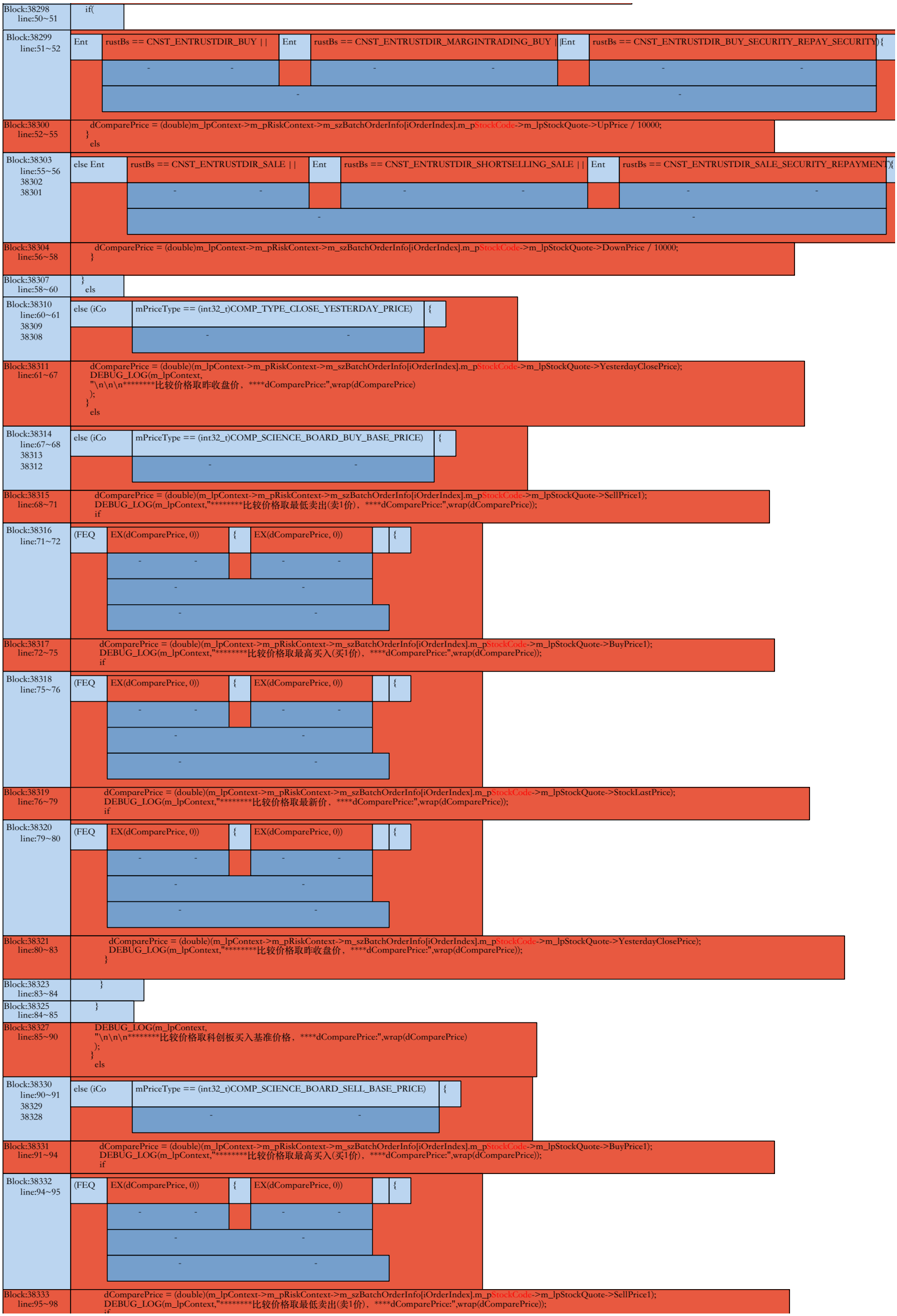
```
Block:35994 line:313~316    32_t CFactorProcessor_4::CalcFactorValue(int32_t iOrderIndex, CRuleSet* lpRuleSet,
                            CRuleFactor* lpFactor, double* lpCalcResult,
                            SFactorProcessorMember* lpMember)
                            {
Block:35996 line:316~329      // 计算方式 - 按数量 按金额
                              USTNumID iCalType = lpMember->m_lpFactor->m_arrFactorParamType[CALC_MODE]
                              .m_arrSFactorParamValeuField[0]
                              .RuleParamValue;
                              FORMULA_CALC_MODE_TYPE calMode = static_cast<FORMULA_CALC_MODE_TYPE>(iCalType);
                              CRiskOrderInfo& order = m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex];
                              *lpCalcResult = (calMode == FORMULA_CALC_MODE_TYPE::CALC_BY_BALANCE)
                              ? (order.m_iEntrustAmount * order.m_nOrderPrice / PRICE_MULTIPLIER)
                              : order.m_iEntrustAmount;
                              return 0;
                              }
Block:35997 line:329~329
```

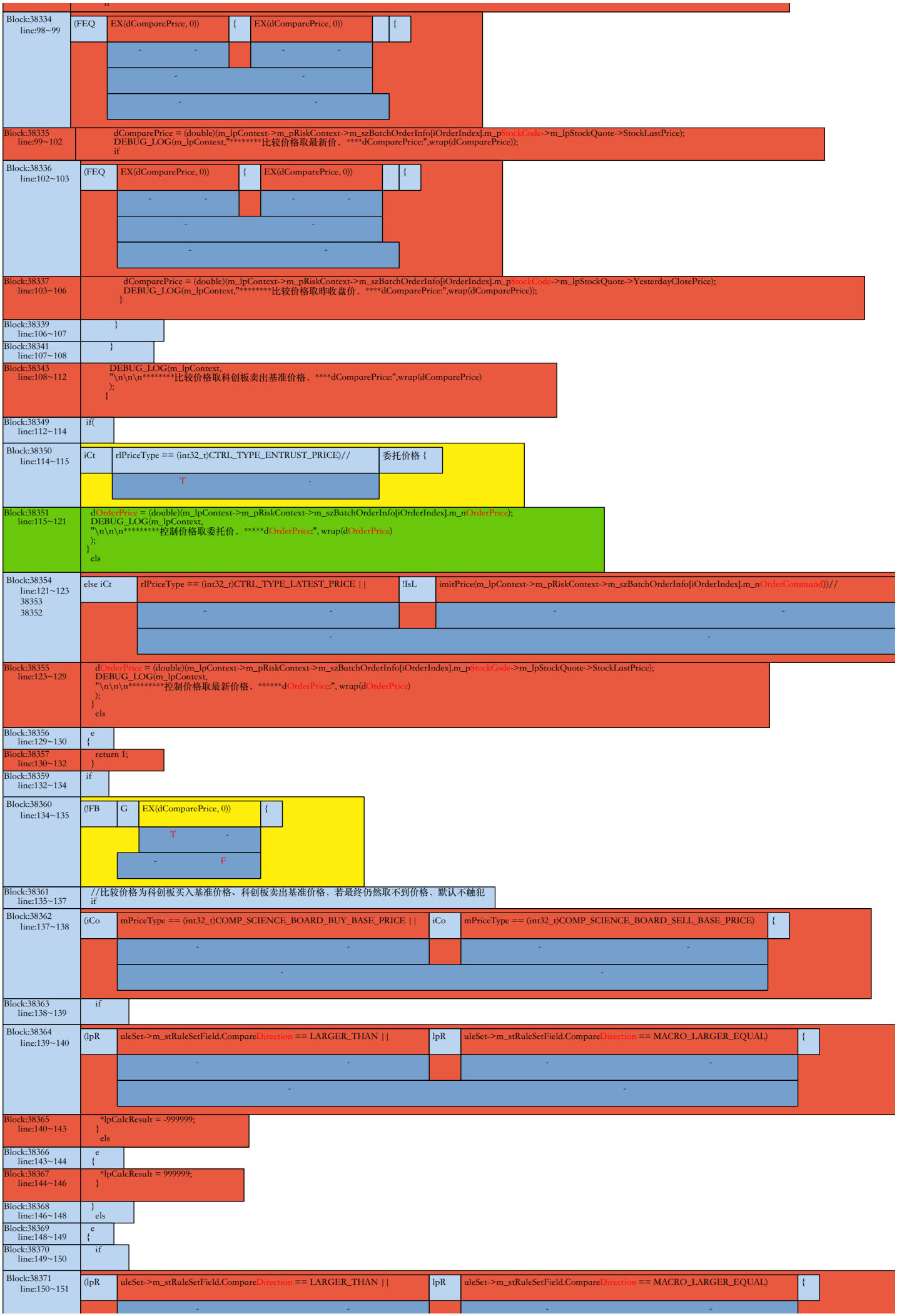
N_3207 : ['OrderPrice']: N_1148 : ['StockCode']:

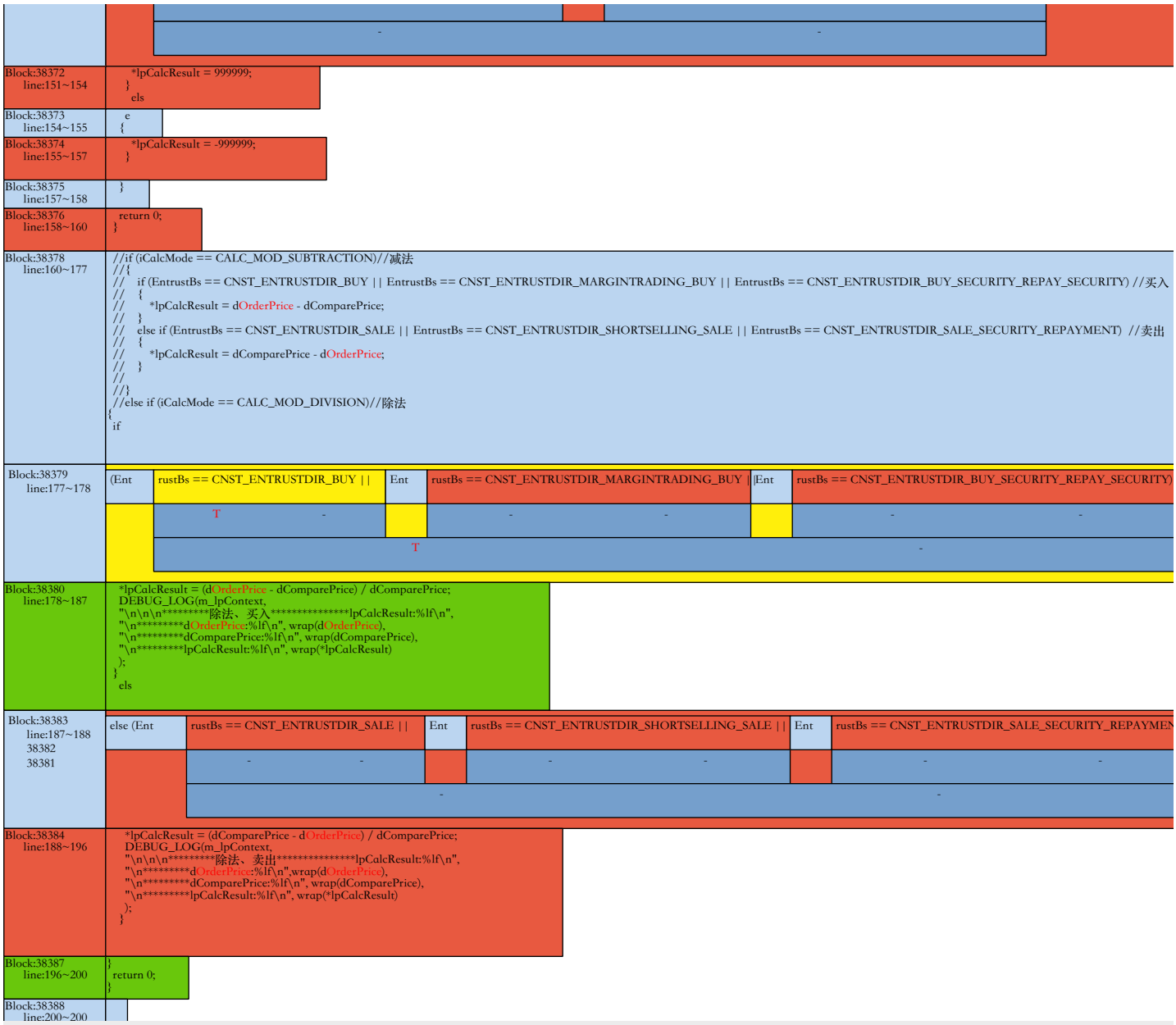
```
Block:8548 line:1935~1936   int32_t CStockHolder::InsertStockHoldReal(CStockHoldReal* pStockHoldReal,uint32_t nStockCode, uint32_t exchangeindex, uint32_t SeatIndex)
Block:8550 line:1936~1938   {
                              uint64_t nKey = ((1LL * nStockCode << 32) | (1LL * exchangeindex << 28) | SeatIndex);
Block:8551 line:1938~1939   if (  m_mStockHoldReal.find(nKey) != m_mStockHoldReal.end()  )
                                                  T                                F
Block:8552 line:1939~1943     {
                                auto tHoldReal = &m_mStockHoldReal.at(nKey);
                                tHoldReal->push_back(pStockHoldReal);
                              }
Block:8553 line:1943~1944   else
Block:8554 line:1944~1949     {
                                vector<CStockHoldReal*> vetHoldReal;
                                vetHoldReal.reserve(128);
                                vetHoldReal.push_back(pStockHoldReal);
                                m_mStockHoldReal.insert(std::pair<uint64_t, vector<CStockHoldReal*>>(nKey, vetHoldReal));
                              }
Block:8555 line:1949~1951   return 0;
Block:8556 line:1951~1951   }
```

N_3367 : ['StockCode', 'Direction', 'OrderPrice', 'OrderCommand']:

```
Block:38286 line:24~25      32_t CFactorProcessor_6::CalcFactorValue(int32_t iOrderIndex, CRuleSet* lpRuleSet, CRuleFactor *lpFactor, double *lpCalcResult, SFactorProcessorMember *lpMember)
                            {
Block:38288 line:25~34        double dOrderPrice = 0;
                              double dComparePrice = 0;
                              int32_t EntrustBs =  m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_iEntrustDirection;
                              int32_t iCtrlPriceType = lpMember->m_lpFactor->m_arrFactorParamType[CONTROL_PRICE].m_arrSFactorParamValeuField[0].RuleParamValue;
                              int32_t iComPriceType = lpMember->m_lpFactor->m_arrFactorParamType[COMPARE_PRICE].m_arrSFactorParamValeuField[0].RuleParamValue;
                              //    int32_t iCalcMode = lpMember->m_lpFactor->m_arrFactorParamType[CALC_MODE].m_arrSFactorParamValeuField[0].RuleParamValue;
                              if(
Block:38289 line:34~35        iCo  mPriceType == (int32_t)COMP_TYPE_LATEST_PRICE)  //最新价格 {
                                                  T                                -
Block:38290 line:35~41        dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->StockLastPrice);
                              DEBUG_LOG(m_lpContext,
                              "\n\n\n********比较价格取最新价，****dComparePrice:",wrap(dComparePrice)
                              );
                              //追加兜底处理：当分母为最新价，且最新价为0时，取昨收盘价参与计算
                              if
Block:38291 line:41~42        (!FB  G  EX(dComparePrice, 0))  {
                                                  T                -
                                                  -                F
Block:38292 line:42~47        dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->YesterdayClosePrice);
                              DEBUG_LOG(m_lpContext,
                              "\n\n\n********比较价格取昨收盘价，****dComparePrice:",wrap(dComparePrice)
                              );
                              }
Block:38294 line:47~49      }  els
Block:38297 line:49~50      else iCo  mPriceType == (int32_t)COMP_TYPE_LIMIT_PRICE)  //涨跌停价格 {
38296                                                 -                -
38295
```


Block:38298 line:50~51
if(
Block:38299 line:51~52
Ent
rustBs == CNST_ENTRUSTDIR_BUY | |
Ent
rustBs == CNST_ENTRUSTDIR_MARGINTRADING_BUY | |Ent
rustBs == CNST_ENTRUSTDIR_BUY_SECURITY_REPAY_SECURITY){
Block:38300 line:52~55
dComparePrice = (double)m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->UpPrice / 10000;
}
els
Block:38303 line:55~56 38302 38301
else Ent
rustBs == CNST_ENTRUSTDIR_SALE | |
Ent
rustBs == CNST_ENTRUSTDIR_SHORTSELLING_SALE | |
Ent
rustBs == CNST_ENTRUSTDIR_SALE_SECURITY_REPAYMENT){
Block:38304 line:56~58
dComparePrice = (double)m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->DownPrice / 10000;
}
Block:38307 line:58~60
}
els
Block:38310 line:60~61 38309 38308
else (iCo
mPriceType == (int32_t)COMP_TYPE_CLOSE_YESTERDAY_PRICE)
{
Block:38311 line:61~67
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->YesterdayClosePrice);
DEBUG_LOG(m_lpContext,
"\n\n\n********比较价格取昨收盘价，****dComparePrice:",wrap(dComparePrice)
);
}
els
Block:38314 line:67~68 38313 38312
else (iCo
mPriceType == (int32_t)COMP_SCIENCE_BOARD_BUY_BASE_PRICE)
{
Block:38315 line:68~71
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->SellPrice1);
DEBUG_LOG(m_lpContext,"********比较价格取最低卖出(卖1价)，****dComparePrice:",wrap(dComparePrice));
if
Block:38316 line:71~72
(FEQ
EX(dComparePrice, 0))
{
EX(dComparePrice, 0))
{
Block:38317 line:72~75
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->BuyPrice1);
DEBUG_LOG(m_lpContext,"********比较价格取最高买入(买1价)，****dComparePrice:",wrap(dComparePrice));
if
Block:38318 line:75~76
(FEQ
EX(dComparePrice, 0))
{
EX(dComparePrice, 0))
{
Block:38319 line:76~79
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->StockLastPrice);
DEBUG_LOG(m_lpContext,"********比较价格取最新价，****dComparePrice:",wrap(dComparePrice));
if
Block:38320 line:79~80
(FEQ
EX(dComparePrice, 0))
{
EX(dComparePrice, 0))
{
Block:38321 line:80~83
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->YesterdayClosePrice);
DEBUG_LOG(m_lpContext,"********比较价格取昨收盘价，****dComparePrice:",wrap(dComparePrice));
}
Block:38323 line:83~84
}
Block:38325 line:84~85
}
Block:38327 line:85~90
DEBUG_LOG(m_lpContext,
"\n\n\n********比较价格取科创板买入基准价格，****dComparePrice:",wrap(dComparePrice)
);
}
els
Block:38330 line:90~91 38329 38328
else (iCo
mPriceType == (int32_t)COMP_SCIENCE_BOARD_SELL_BASE_PRICE)
{
Block:38331 line:91~94
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->BuyPrice1);
DEBUG_LOG(m_lpContext,"********比较价格取最高买入(买1价)，****dComparePrice:",wrap(dComparePrice));
if
Block:38332 line:94~95
(FEQ
EX(dComparePrice, 0))
{
EX(dComparePrice, 0))
{
Block:38333 line:95~98
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->SellPrice1);
DEBUG_LOG(m_lpContext,"********比较价格取最低卖出(卖1价)，****dComparePrice:",wrap(dComparePrice));
if


Block:38334 line:98~99
(FEQ EX(dComparePrice, 0)) { EX(dComparePrice, 0)) {
Block:38335 line:99~102
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->StockLastPrice);
DEBUG_LOG(m_lpContext,"********比较价格取最新价，****dComparePrice:",wrap(dComparePrice));
if
Block:38336 line:102~103
(FEQ EX(dComparePrice, 0)) { EX(dComparePrice, 0)) {
Block:38337 line:103~106
dComparePrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->YesterdayClosePrice);
DEBUG_LOG(m_lpContext,"********比较价格取昨收盘价，****dComparePrice:",wrap(dComparePrice));
}
Block:38339 line:106~107
}
Block:38341 line:107~108
}
Block:38343 line:108~112
DEBUG_LOG(m_lpContext,
"\n\n\n********比较价格取科创板卖出基准价格，****dComparePrice:",wrap(dComparePrice)
);
}
Block:38349 line:112~114
if(
Block:38350 line:114~115
iCt rlPriceType == (int32_t)CTRL_TYPE_ENTRUST_PRICE)// 委托价格 {
T
Block:38351 line:115~121
dOrderPrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_nOrderPrice);
DEBUG_LOG(m_lpContext,
"\n\n\n*********控制价格取委托价，*****dOrderPrice:", wrap(dOrderPrice)
);
}
els
Block:38354 line:121~123 38353 38352
else iCt rlPriceType == (int32_t)CTRL_TYPE_LATEST_PRICE || !IsL imitPrice(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_nOrderCommand))//
Block:38355 line:123~129
dOrderPrice = (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pStockCode->m_lpStockQuote->StockLastPrice);
DEBUG_LOG(m_lpContext,
"\n\n\n*********控制价格取最新价格，******dOrderPrice:", wrap(dOrderPrice)
);
}
els
Block:38356 line:129~130
e
{
Block:38357 line:130~132
return 1;
}
Block:38359 line:132~134
if
Block:38360 line:134~135
(!FB G EX(dComparePrice, 0)) {
T
F
Block:38361 line:135~137
//比较价格为科创板买入基准价格、科创板卖出基准价格，若最终仍然取不到价格，默认不触犯
if
Block:38362 line:137~138
(iCo mPriceType == (int32_t)COMP_SCIENCE_BOARD_BUY_BASE_PRICE || iCo mPriceType == (int32_t)COMP_SCIENCE_BOARD_SELL_BASE_PRICE) {
Block:38363 line:138~139
if
Block:38364 line:139~140
(lpR uleSet->m_stRuleSetField.CompareDirection == LARGER_THAN || lpR uleSet->m_stRuleSetField.CompareDirection == MACRO_LARGER_EQUAL) {
Block:38365 line:140~143
*lpCalcResult = -999999;
}
els
Block:38366 line:143~144
e
{
Block:38367 line:144~146
*lpCalcResult = 999999;
}
Block:38368 line:146~148
}
els
Block:38369 line:148~149
e
{
Block:38370 line:149~150
if
Block:38371 line:150~151
(lpR uleSet->m_stRuleSetField.CompareDirection == LARGER_THAN || lpR uleSet->m_stRuleSetField.CompareDirection == MACRO_LARGER_EQUAL) {



N_3372 : ['OrderCommand']:

```
Block:38446 line:309~310
32_t CFactorProcessor_6::PriceTypeCheck(int32_t iOrderIndex, CRuleFactor* lpFactor)
{
Block:38448 line:310~313
 bool bIsLimitPrice = IsLimitPrice(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_nOrderCommand);
 auto& arrPriceType = lpFactor->m_arrFactorParamType[PRICE_TYPE];
 if
Block:38449 line:313~314
(arr | PriceType.m_iCount > 0) | {
T -
Block:38450 line:314~315
 for
Block:38451 line:315~316
(int i = 0; i < | arrPriceType.m_iCount; + | +i) {
T -
Block:38452 line:316~318
  auto iPriceType = arrPriceType.m_arrSFactorParamValeuField[i].RuleParamValue;
  if
Block:38453 line:318~319
(PRI | CE_TYPE_ALL == iPriceType) | {
T F
Block:38454 line:319~322
   return 1;
  }
  els
Block:38457 line:322~323 38456 38455
else (PRI | CE_TYPE_LIMITED == iPriceType && | bIs | LimitPrice) | {
T - | T -
T -
Block:38458 line:323~326
   return 1;
  }
  els
Block:38461 line:326~327 38460 38459
else (PRI | CE_TYPE_MARKET == iPriceType && | !bIs | LimitPrice) | {
- - | - -
- -
Block:38462 line:327~329
   return 1;
  }
Block:38466 line:329~330
 }
Block:38469 line:330~336
  DEBUG_LOG(m_lpContext,"\n\n\n价格类型判断不相关", wrap("OrderCommand", m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_nOrderCommand),
  wrap(bIsLimitPrice ? "限价" : "非限价"), "\n");
  return 0;
 }
  els
Block:38470 line:336~337
 e
{
Block:38471 line:337~340
  DEBUG_LOG(m_lpContext, "\n\n\n无价格类型，价格类型判断不相关\n");
  return 0;
 }
}
Block:38472 line:340~341
Block:38473 line:341~341
```

N_3038 :

['ExchangeID', 'StockCode']:

```
Block:34489 line:24~25
int32_t CETFOrderInsertFlow::GetBasicData()
Block:34491 line:25~38
{
 //获取ETF基础信息
 CEtfBasicInfo* pEtfBasicInfo = m_stPublicContext.pExcharg->GetEtfBasicInfoByEtfCode(m_stPublicContext.pSecurityInfo->m_stSecurityInfo.SecurityCode, m_stPublicContext.nExecType);
 AMUST_LOGBIZERR_CHECK_OBJ_NOT_FOUND_INNER_ASSIGN(pEtfBasicInfo, "找不到ETF基础信息", wrap("StockCodeStr", m_stPublicContext.pSecurityInfo->m_stSecurityInfo.SecurityCode),
 wrap("nExecType", m_stPublicContext.nExecType));
 m_stContext.pEtfBasicInfo = pEtfBasicInfo;
 //查找ETF持仓
 m_stContext.pETFHold = m_stPublicContext.pCombi->GetHoldRealByCode(atoi(pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode),m_stStockPublicContext.nExchIndex,
 m_stPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex,m_stPublicContext.SeatIndex);
 //查找实例持仓（找不到插入一条）
Block:34492 line:38~39
if( | m_stPublicContext.nInstanceIndex > 0 | )
- F
Block:34493 line:39~45
 {
  USTType cInstanceHoldDealType; // 实例持仓处理类别
  m_stPublicContext.pInstanceHold = GetInstanceHold(lpContext, m_stPublicContext.pInstance, m_stPublicContext.ExchangeID,
  atoi(pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode), cInstanceHoldDealType,m_stPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex,m_stPublicContext.SeatIndex);
  AMUST_LOGBIZERR_CHECK_OBJ_NOT_FOUND_INNER_ASSIGN(m_stPublicContext.pInstanceHold, "找不到实例持仓",  wrap("ExchangeID", m_stPublicContext.ExchangeID),
  wrap("nSecurityCode", pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode),  wrap("InstanceHoldDealType", cInstanceHoldDealType));
 }
Block:34495 line:45~54
 //查找代码
 m_stContext.pETFCode = m_stPublicContext.pExcharg->GetStockCode(atoi(pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode));
 AMUST_LOGBIZERR_CHECK_OBJ_NOT_FOUND_INNER_ASSIGN(m_stContext.pETFCode, "找不到代码信息",  wrap("ExchangeIndex", m_stPublicContext.nExchIndex),
 wrap("SecurityCode", pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode));
 return m_nErrorNo;
Block:34496 line:54~54
}
```

N_53 : ['Direction']:

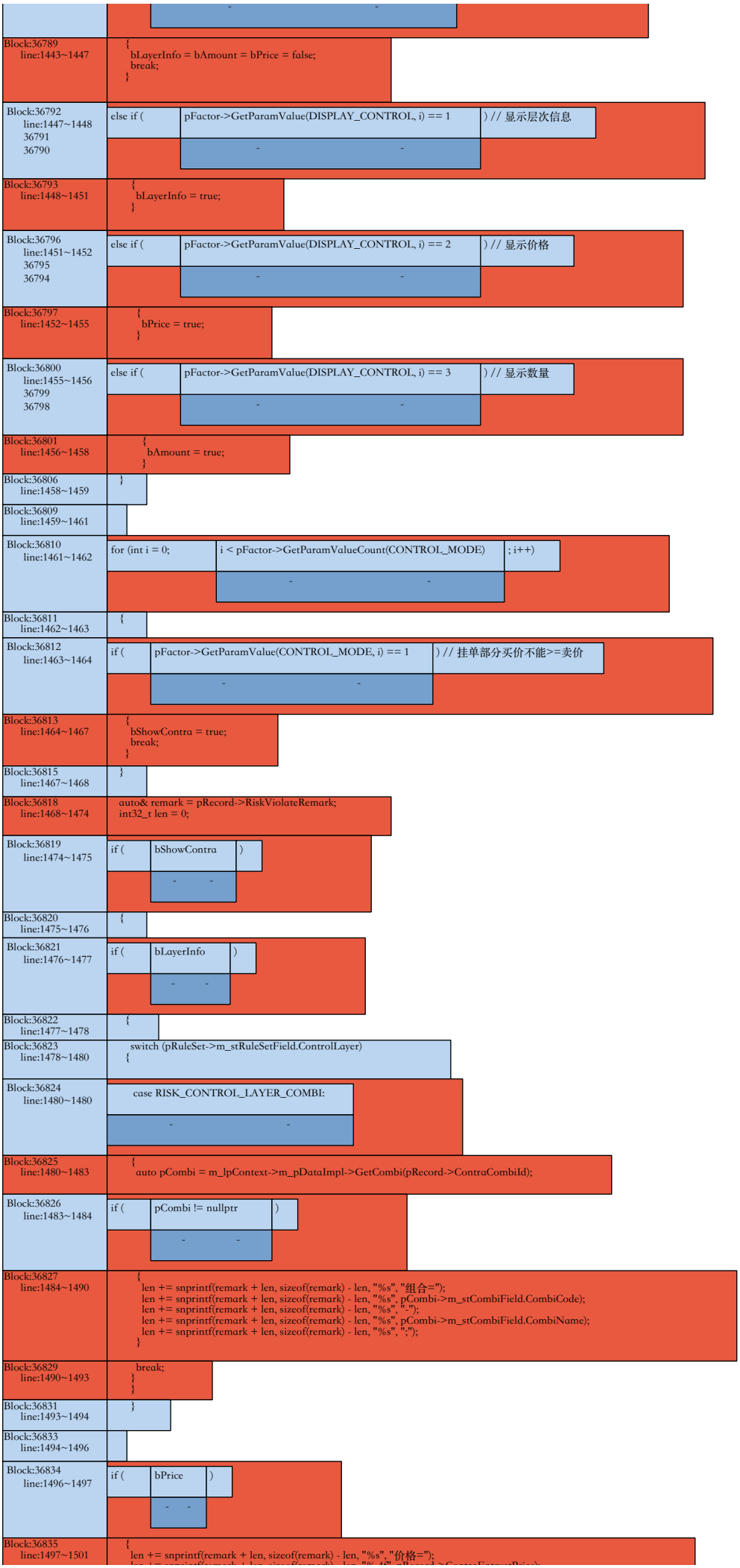
```
Block:884 line:1984~1985
int32_t GetCashEnableFormulaHashKey(USTExchangeIndex nExchangeIndex, USTNumID nSecurityType, USTEntrustDirection nEntrustDirection, USTType BusinProType, USTType SettleCashType)
Block:886 line:1985~1990
{
 int32_t nBusinProType = BusinProType - CNST_BUSINPROTYPE_ENTRUST;
 int32_t nSettleCashType = SettleCashType - CNST_CASH_TYPE_NO_DISTINCTION;
 return nExchangeIndex*100000000 + nSecurityType*1000000 + nEntrustDirection*10000 + nBusinProType*100 + nSettleCashType;
Block:887 line:1990~1990
}
```

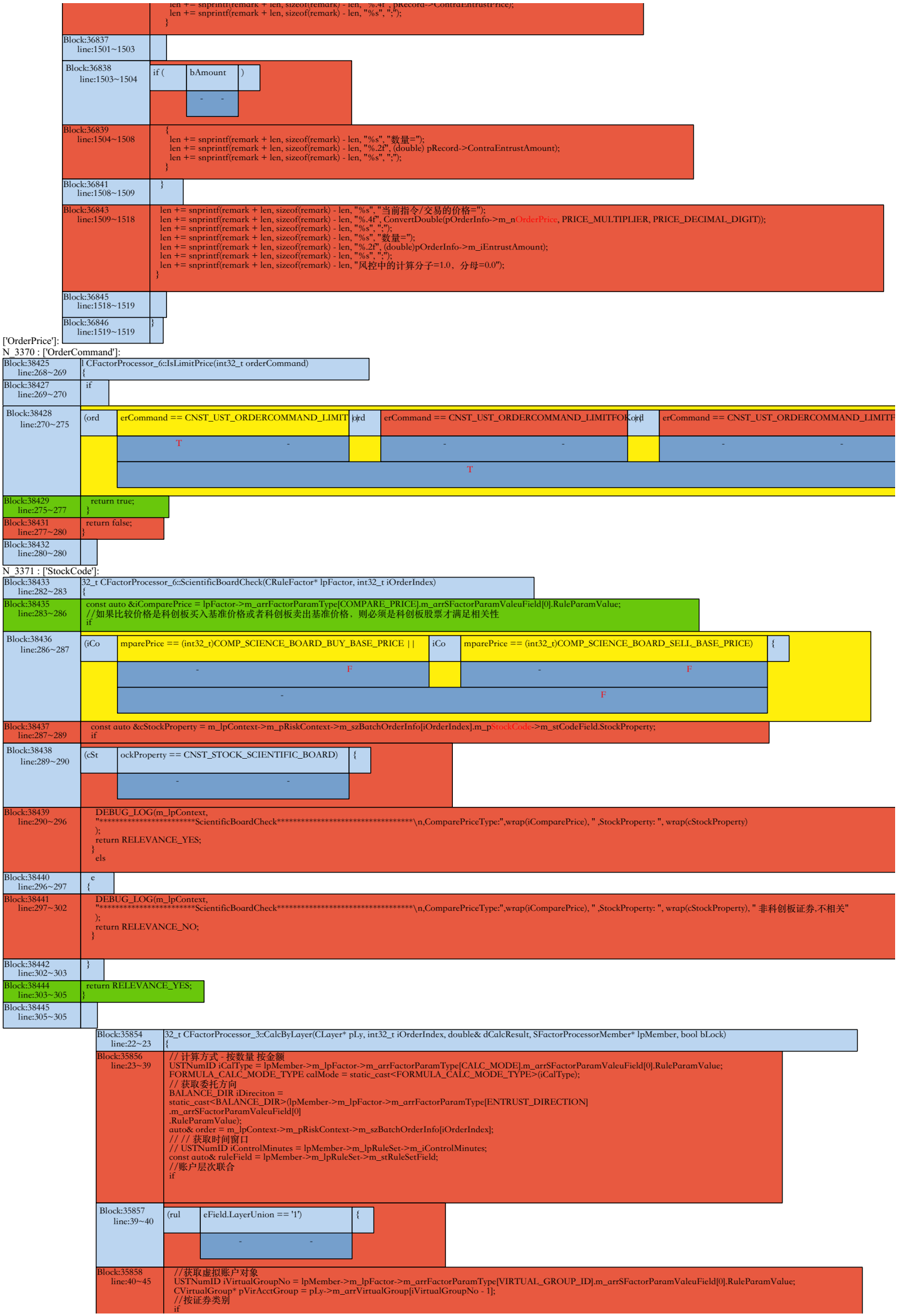
```
Block:35968 line:248~249   l CFactorProcessor_3::DirectionRelativeCheck(CRuleFactor* lpFactor, int32_t iOrderIndex, SFactorProcessorMember* lpMember)
                           {
Block:35970 line:249~257   int32_t iRet = 0;
                           // 委托方向
                           auto enDir = m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_iEntrustDirection;
                           // 规则比较方向
                           auto comDir = lpMember->m_lpRuleSet->m_stRuleSetField.CompareDirection;
                           // 控制方向
                           swi
Block:35971 line:257~259   tch (lpFactor->m_arrFactorParamType[ENTRUST_DIRECTION].m_arrSFactorParamValeuField[0].RuleParamValue)
                           {
                            cas
Block:35972 line:259~259   e BALANCE_DIR::BUY: //
                                  -                -
Block:35973 line:259~264    只控制买
                           {
                            iRet = (enDir == CNST_ENTRUSTDIR_BUY);
                            break;
                           }
                            cas
Block:35974 line:264~264   e BALANCE_DIR::SELL: //
                                  -                -
Block:35975 line:264~269    只控制卖
                           {
                            iRet = (enDir == CNST_ENTRUSTDIR_SALE);
                            break;
                           }
                            cas
Block:35976 line:269~269   e BALANCE_DIR::BUY_PLUS_SELL: //
                                  -                -
Block:35977 line:269~274    控制“买+卖”
                           {
                            iRet = 1;
                            break;
                           }
                            cas
Block:35978 line:274~274   e BALANCE_DIR::BUY_MINUS_SELL: //
                                  -                -
Block:35979 line:274~282    控制“买-卖”
                           {
                            iRet = (((MACRO_LARGER_THAN == comDir || MACRO_LARGER_EQUAL == comDir) &&
                            enDir == CNST_ENTRUSTDIR_BUY) ||
                            ((MACRO_LOWER_THAN == comDir || MACRO_LOWER_EQUAL == comDir) &&
                            enDir == CNST_ENTRUSTDIR_SALE));
                            break;
                           }
                            cas
Block:35980 line:282~282   e BALANCE_DIR::SELL_MINUS_BUY:
                                  -                -
Block:35981 line:282~290   {
                            iRet = (((MACRO_LARGER_THAN == comDir || MACRO_LARGER_EQUAL == comDir) &&
                            enDir == CNST_ENTRUSTDIR_SALE) ||
                            ((MACRO_LOWER_THAN == comDir || MACRO_LOWER_EQUAL == comDir) &&
                            enDir == CNST_ENTRUSTDIR_BUY));
                            break;
                           }
                            def
Block:35982 line:290~290   ault:
                              -      -
Block:35983 line:290~293   {
                           }
                           }
Block:35984 line:293~296   return iRet;
                           }
Block:35985 line:296~296
```

N_3204 : ['Direction']:

N_3238 :

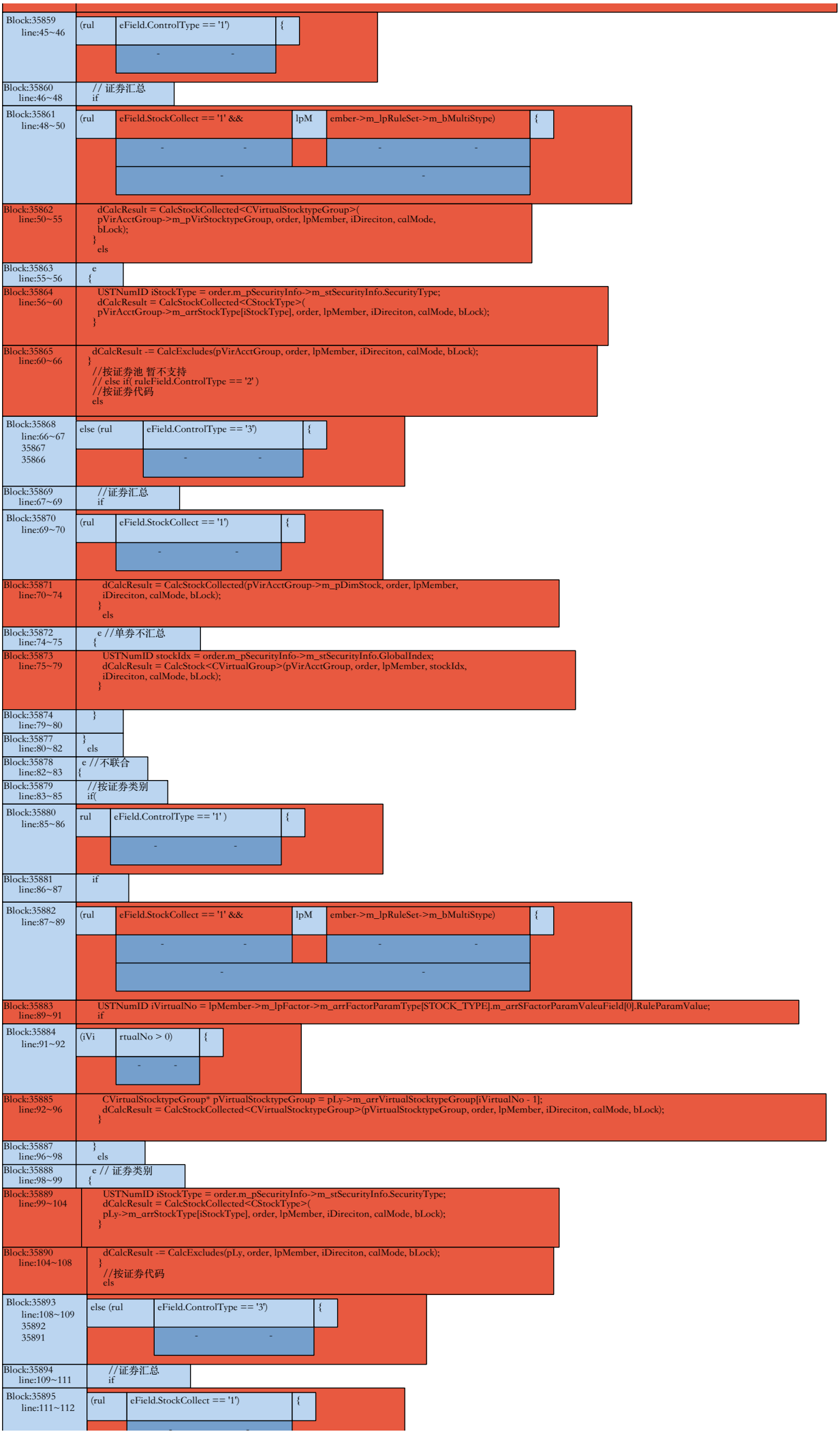
```
Block:36781 line:1429~1430   void CRuleProcessor::SetContraOrderInfo(CRiskOrderInfo *pOrderInfo, CRuleSet *pRuleSet, SRuleWarningInfoField *pRecord)
Block:36783 line:1430~1431   {
Block:36784 line:1431~1432   if (   pRecord->RuleType == 'J'   )
                                        -                F
Block:36785 line:1432~1440   {
                              pRecord->CalcValue = 1;
                              pRecord->RuleCalcResult = 1;
                              pRecord->NumeratorCalValue = 1;
                              bool bLayerInfo, bAmount, bPrice, bShowContra/*反向信息只支持挂单部分买价不能大于等于卖价*/;
                              bLayerInfo = bAmount = bPrice = bShowContra = false;
                              // 如果设置了反向信息，则不显示
                              CRuleFactor* pFactor = &pRuleSet->m_arrRiskFactor[0];
Block:36786 line:1440~1441   for (int i = 0;   i < pFactor->GetParamValueCount(DISPLAY_CONTROL)   ; i++)
                                                      -                -
Block:36787 line:1441~1442    {
Block:36788 line:1442~1443   if (   pFactor->GetParamValue(DISPLAY_CONTROL, i) == -1   ) // 不显示
```

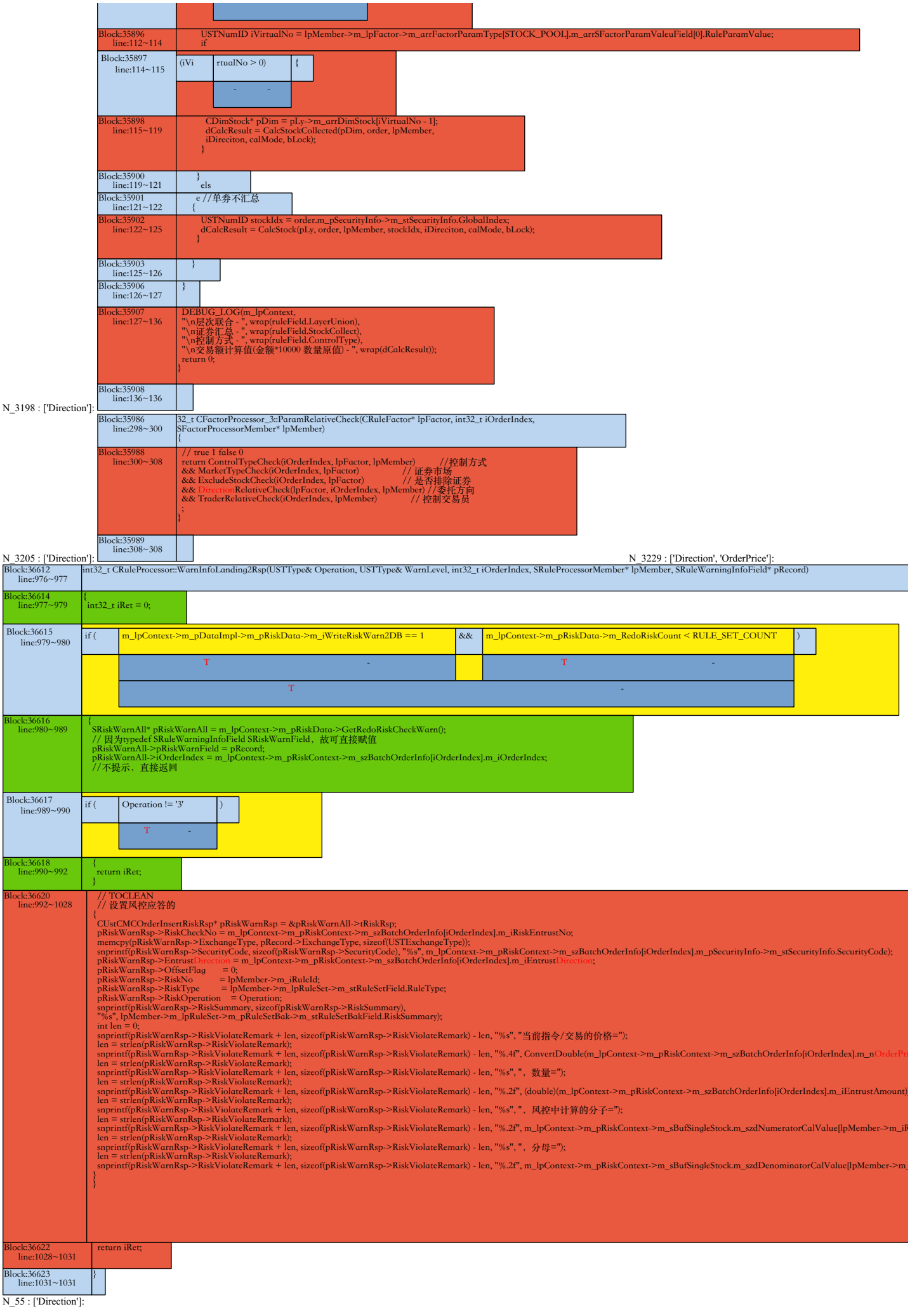

Block:36789
line:1443~1447
{
bLayerInfo = bAmount = bPrice = false;
break;
}
Block:36792
line:1447~1448
36791
36790
else if (
pFactor->GetParamValue(DISPLAY_CONTROL, i) == 1
) // 显示层次信息
Block:36793
line:1448~1451
{
bLayerInfo = true;
}
Block:36796
line:1451~1452
36795
36794
else if (
pFactor->GetParamValue(DISPLAY_CONTROL, i) == 2
) // 显示价格
Block:36797
line:1452~1455
{
bPrice = true;
}
Block:36800
line:1455~1456
36799
36798
else if (
pFactor->GetParamValue(DISPLAY_CONTROL, i) == 3
) // 显示数量
Block:36801
line:1456~1458
{
bAmount = true;
}
Block:36806
line:1458~1459
}
Block:36809
line:1459~1461
Block:36810
line:1461~1462
for (int i = 0;
i < pFactor->GetParamValueCount(CONTROL_MODE)
; i++)
Block:36811
line:1462~1463
{
Block:36812
line:1463~1464
if (
pFactor->GetParamValue(CONTROL_MODE, i) == 1
) // 挂单部分买价不能>=卖价
Block:36813
line:1464~1467
{
bShowContra = true;
break;
}
Block:36815
line:1467~1468
}
Block:36818
line:1468~1474
auto& remark = pRecord->RiskViolateRemark;
int32_t len = 0;
Block:36819
line:1474~1475
if (
bShowContra
)
Block:36820
line:1475~1476
{
Block:36821
line:1476~1477
if (
bLayerInfo
)
Block:36822
line:1477~1478
{
Block:36823
line:1478~1480
switch (pRuleSet->m_stRuleSetField.ControlLayer)
{
Block:36824
line:1480~1480
case RISK_CONTROL_LAYER_COMBI:
Block:36825
line:1480~1483
{
auto pCombi = m_lpContext->m_pDataImpl->GetCombi(pRecord->ContraCombiId);
Block:36826
line:1483~1484
if (
pCombi != nullptr
)
Block:36827
line:1484~1490
{
len += snprintf(remark + len, sizeof(remark) - len, "%s", "组合=");
len += snprintf(remark + len, sizeof(remark) - len, "%s", pCombi->m_stCombiField.CombiCode);
len += snprintf(remark + len, sizeof(remark) - len, "%s", "-");
len += snprintf(remark + len, sizeof(remark) - len, "%s", pCombi->m_stCombiField.CombiName);
len += snprintf(remark + len, sizeof(remark) - len, "%s", ";");
}
Block:36829
line:1490~1493
break;
}
}
Block:36831
line:1493~1494
}
Block:36833
line:1494~1496
Block:36834
line:1496~1497
if (
bPrice
)
Block:36835
line:1497~1501
{
len += snprintf(remark + len, sizeof(remark) - len, "%s", "价格=");

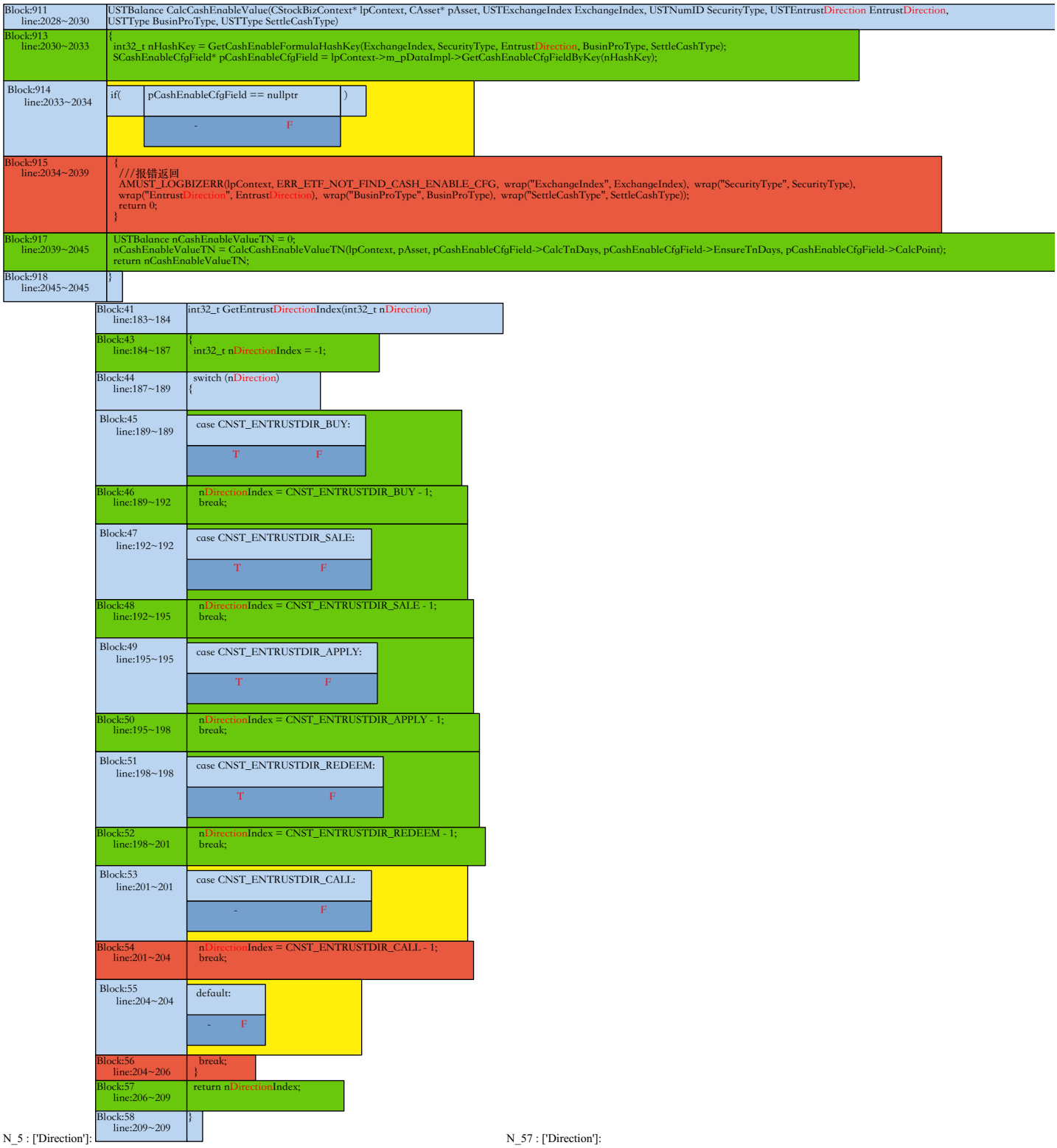


['OrderPrice']:
N_3370 : ['OrderCommand']:

N_3371 : ['StockCode']:


Block:35859 line:45~46
(rul eField.ControlType == '1') {
Block:35860 line:46~48
// 证券汇总
if
Block:35861 line:48~50
(rul eField.StockCollect == '1' && lpM ember->m_lpRuleSet->m_bMultiStype) {
Block:35862 line:50~55
dCalcResult = CalcStockCollected<CVirtualStocktypeGroup>(
pVirAcctGroup->m_pVirStocktypeGroup, order, lpMember, iDireciton, calMode,
bLock);
}
els
Block:35863 line:55~56
e
{
Block:35864 line:56~60
USTNumID iStockType = order.m_pSecurityInfo->m_stSecurityInfo.SecurityType;
dCalcResult = CalcStockCollected<CStockType>(
pVirAcctGroup->m_arrStockType[iStockType], order, lpMember, iDireciton, calMode, bLock);
}
Block:35865 line:60~66
dCalcResult -= CalcExcludes(pVirAcctGroup, order, lpMember, iDireciton, calMode, bLock);
}
//按证券池 暂不支持
// else if(ruleField.ControlType == '2')
//按证券代码
els
Block:35868 line:66~67 35867 35866
else (rul eField.ControlType == '3') {
Block:35869 line:67~69
//证券汇总
if
Block:35870 line:69~70
(rul eField.StockCollect == '1') {
Block:35871 line:70~74
dCalcResult = CalcStockCollected(pVirAcctGroup->m_pDimStock, order, lpMember,
iDireciton, calMode, bLock);
}
els
Block:35872 line:74~75
e //单券不汇总
{
Block:35873 line:75~79
USTNumID stockIdx = order.m_pSecurityInfo->m_stSecurityInfo.GlobalIndex;
dCalcResult = CalcStock<CVirtualGroup>(pVirAcctGroup, order, lpMember, stockIdx,
iDireciton, calMode, bLock);
}
Block:35874 line:79~80
}
Block:35877 line:80~82
}
els
Block:35878 line:82~83
e //不联合
{
Block:35879 line:83~85
//按证券类别
if(
Block:35880 line:85~86
rul eField.ControlType == '1') {
Block:35881 line:86~87
if
Block:35882 line:87~89
(rul eField.StockCollect == '1' && lpM ember->m_lpRuleSet->m_bMultiStype) {
Block:35883 line:89~91
USTNumID iVirtualNo = lpMember->m_lpFactor->m_arrFactorParamType[STOCK_TYPE].m_arrSFactorParamValeuField[0].RuleParamValue;
if
Block:35884 line:91~92
(iVi rtualNo > 0) {
Block:35885 line:92~96
CVirtualStocktypeGroup* pVirtualStocktypeGroup = pLy->m_arrVirtualStocktypeGroup[iVirtualNo - 1];
dCalcResult = CalcStockCollected<CVirtualStocktypeGroup>(pVirtualStocktypeGroup, order, lpMember, iDireciton, calMode, bLock);
}
Block:35887 line:96~98
}
els
Block:35888 line:98~99
e // 证券类别
{
Block:35889 line:99~104
USTNumID iStockType = order.m_pSecurityInfo->m_stSecurityInfo.SecurityType;
dCalcResult = CalcStockCollected<CStockType>(
pLy->m_arrStockType[iStockType], order, lpMember, iDireciton, calMode, bLock);
}
Block:35890 line:104~108
dCalcResult -= CalcExcludes(pLy, order, lpMember, iDireciton, calMode, bLock);
}
//按证券代码
els
Block:35893 line:108~109 35892 35891
else (rul eField.ControlType == '3') {
Block:35894 line:109~111
//证券汇总
if
Block:35895 line:111~112
(rul eField.StockCollect == '1') {


Block:35896 line:112~114
USTNumID iVirtualNo = lpMember->m_lpFactor->m_arrFactorParamType[STOCK_POOL].m_arrSFactorParamValeuField[0].RuleParamValue;
if
Block:35897 line:114~115
(iVi
rtualNo > 0)
{
-
-
Block:35898 line:115~119
CDimStock* pDim = pLy->m_arrDimStock[iVirtualNo - 1];
dCalcResult = CalcStockCollected(pDim, order, lpMember,
iDireciton, calMode, bLock);
}
Block:35900 line:119~121
}
els
Block:35901 line:121~122
e //单券不汇总
{
Block:35902 line:122~125
USTNumID stockIdx = order.m_pSecurityInfo->m_stSecurityInfo.GlobalIndex;
dCalcResult = CalcStock(pLy, order, lpMember, stockIdx, iDireciton, calMode, bLock);
}
Block:35903 line:125~126
}
Block:35906 line:126~127
}
Block:35907 line:127~136
DEBUG_LOG(m_lpContext,
"\n层次联合 - ", wrap(ruleField.LayerUnion),
"\n证券汇总 - ", wrap(ruleField.StockCollect),
"\n控制方式 - ", wrap(ruleField.ControlType),
"\n交易额计算值(金额*10000 数量原值) - ", wrap(dCalcResult));
return 0;
}
Block:35908 line:136~136
N_3198 : ['Direction']:
Block:35986 line:298~300
32_t CFactorProcessor_3::ParamRelativeCheck(CRuleFactor* lpFactor, int32_t iOrderIndex,
SFactorProcessorMember* lpMember)
{
Block:35988 line:300~308
// true 1 false 0
return ControlTypeCheck(iOrderIndex, lpFactor, lpMember) //控制方式
&& MarketTypeCheck(iOrderIndex, lpFactor) // 证券市场
&& ExcludeStockCheck(iOrderIndex, lpFactor) // 是否排除证券
&& DirectionRelativeCheck(lpFactor, iOrderIndex, lpMember) //委托方向
&& TraderRelativeCheck(iOrderIndex, lpMember) // 控制交易员
;
}
Block:35989 line:308~308
N_3205 : ['Direction']:
N_3229 : ['Direction', 'OrderPrice']:
Block:36612 line:976~977
int32_t CRuleProcessor::WarnInfoLanding2Rsp(USTType& Operation, USTType& WarnLevel, int32_t iOrderIndex, SRuleProcessorMember* lpMember, SRuleWarningInfoField* pRecord)
Block:36614 line:977~979
{
int32_t iRet = 0;
Block:36615 line:979~980
if (
m_lpContext->m_pDataImpl->m_pRiskData->m_iWriteRiskWarn2DB == 1
&&
m_lpContext->m_pRiskData->m_RedoRiskCount < RULE_SET_COUNT
)
T
-
T
-
T
-
Block:36616 line:980~989
{
SRiskWarnAll* pRiskWarnAll = m_lpContext->m_pRiskData->GetRedoRiskCheckWarn();
// 因为typedef SRuleWarningInfoField SRiskWarnField，故可直接赋值
pRiskWarnAll->pRiskWarnField = pRecord;
pRiskWarnAll->iOrderIndex = m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_iOrderIndex;
//不提示，直接返回
Block:36617 line:989~990
if (
Operation != '3'
)
T
-
Block:36618 line:990~992
{
return iRet;
}
Block:36620 line:992~1028
// TOCLEAN
// 设置风控应答的
{
CUstCMCOrderInsertRiskRsp* pRiskWarnRsp = &pRiskWarnAll->tRiskRsp;
pRiskWarnRsp->RiskCheckNo = m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_iRiskEntrustNo;
memcpy(pRiskWarnRsp->ExchangeType, pRecord->ExchangeType, sizeof(USTExchangeType));
snprintf(pRiskWarnRsp->SecurityCode, sizeof(pRiskWarnRsp->SecurityCode), "%s", m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_pSecurityInfo->m_stSecurityInfo.SecurityCode);
pRiskWarnRsp->EntrustDirection = m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_iEntrustDirection;
pRiskWarnRsp->OffsetFlag = 0;
pRiskWarnRsp->RiskNo = lpMember->m_iRuleId;
pRiskWarnRsp->RiskType = lpMember->m_lpRuleSet->m_stRuleSetField.RuleType;
pRiskWarnRsp->RiskOperation = Operation;
snprintf(pRiskWarnRsp->RiskSummary, sizeof(pRiskWarnRsp->RiskSummary),
"%s", lpMember->m_lpRuleSet->m_pRuleSetBak->m_stRuleSetBakField.RiskSummary);
int len = 0;
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%s", "当前指令/交易的价格=");
len = strlen(pRiskWarnRsp->RiskViolateRemark);
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%.4f", ConvertDouble(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_nOrderPr
len = strlen(pRiskWarnRsp->RiskViolateRemark);
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%s", "， 数量=");
len = strlen(pRiskWarnRsp->RiskViolateRemark);
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%.2f", (double)(m_lpContext->m_pRiskContext->m_szBatchOrderInfo[iOrderIndex].m_iEntrustAmount)
len = strlen(pRiskWarnRsp->RiskViolateRemark);
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%s", "， 风控中计算的分子=");
len = strlen(pRiskWarnRsp->RiskViolateRemark);
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%.2f", m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szdNumeratorCalValue[lpMember->m_iR
len = strlen(pRiskWarnRsp->RiskViolateRemark);
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%s", "， 分母=");
len = strlen(pRiskWarnRsp->RiskViolateRemark);
snprintf(pRiskWarnRsp->RiskViolateRemark + len, sizeof(pRiskWarnRsp->RiskViolateRemark) - len, "%.2f", m_lpContext->m_pRiskContext->m_sBufSingleStock.m_szdDenominatorCalValue[lpMember->m_
}
}
Block:36622 line:1028~1031
return iRet;
Block:36623 line:1031~1031
}
N_55 : ['Direction']:



N_5 : ['Direction']:

N_57 : ['Direction']:

Block:938 line:2098~2100	int32_t UpdateCashEnableValue(CStockBizContext* lpContext, CAsset* pAsset, USTExchangeIndex ExchangeIndex, USTNumID SecurityType, USTEntrustDirection EntrustDirection, USTType BusinProType, USTType SettleCashType, USTBalance UpdateBalance)
Block:940 line:2100~2101	{
Block:941 line:2101~2101	if(0 == UpdateBalance) - F
Block:942 line:2101~2101	return ERR_OK;
Block:944 line:2101~2105	int32_t nHashKey = GetCashEnableFormulaHashKey(ExchangeIndex, SecurityType, EntrustDirection, BusinProType, SettleCashType); SCashEnableCfgField* pCashEnableCfgField = lpContext->m_pDataImpl->GetCashEnableCfgFieldByKey(nHashKey);
Block:945 line:2105~2106	if(pCashEnableCfgField == nullptr) - F

Block:946
line:2106~2111
	{
	///报错返回
	AMUST_LOGBIZERR(lpContext, ERR_ETF_NOT_FIND_CASH_ENABLE_CFG, wrap("ExchangeIndex", ExchangeIndex), wrap("SecurityType", SecurityType),
	wrap("EntrustDirection", EntrustDirection), wrap("BusinProType", BusinProType), wrap("SettleCashType", SettleCashType));
	return ERR_ETF_NOT_FIND_CASH_ENABLE_CFG;
	}

Block:948
line:2111~2117
	//找到对应更新的资金因子
	uint32_t nUpdateTradeDay = lpContext->m_pDataImpl->GetNextNTradeDay(lpContext->m_pDataImpl->GetSysarg()->m_nInitDate, pCashEnableCfgField->UpdateTnDays); //计算更新到T+N 日期
	SCashEnableFactorField* pCashEnableFactorField = pAsset->GetCashEnableFactorField(nUpdateTradeDay, pCashEnableCfgField->UpdatePoint);

Block:949 line:2117~2118	if(pCashEnableCfgField->CashChangeType == '+') T F

Block:950
line:2118~2121
	{
	pCashEnableFactorField->CashFactorValue += UpdateBalance;
	}

Block:953 line:2121~2122 952 951	else if(pCashEnableCfgField->CashChangeType == '-') T -

Block:954
line:2122~2124
	{
	pCashEnableFactorField->CashFactorValue -= UpdateBalance;
	}

Block:957 line:2124~2130	INFO_LOG(lpContext,"资金可用更新" ,wrap("SettlePoint", pCashEnableFactorField->SettlePoint) ,wrap("SettleTnDays", pCashEnableFactorField->SettleTnDays) ,wrap("CashFactorValue", ConvertDouble(pCashEnableFactorField->CashFactorValue, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT),BALANCE_DECIMAL_DIGIT)); return ERR_OK;
Block:958 line:2130~2130	}

N_42 : ['StockCode']:

Block:754 line:1439~1440	CInstanceHold* GetInstanceHold(CStockBizContext* lpContext, CInstance* pInstance, USTExchangeType sExchangeType, USTStockCode nStockCode, USTType& cInstanceHoldDealType, USTNumID StockH

Block:756
line:1440~1443
	{
	int32_t iExchangeIndex = GetExchangeIndex(sExchangeType);
	CInstanceHold* pInstanceHold = pInstance->GetInstanceHoldByCode(nStockCode, iExchangeIndex,StockHoldIndex, SeatIndex);

Block:757 line:1443~1444	if (unlikely(pInstanceHold == nullptr)) - -
Block:758 line:1444~1446	{ pInstanceHold = pInstance->NewInstanceHold(nStockCode, iExchangeIndex, StockHoldIndex, SeatIndex);
Block:759 line:1446~1447	if (unlikely((pInstanceHold) == nullptr)) - -

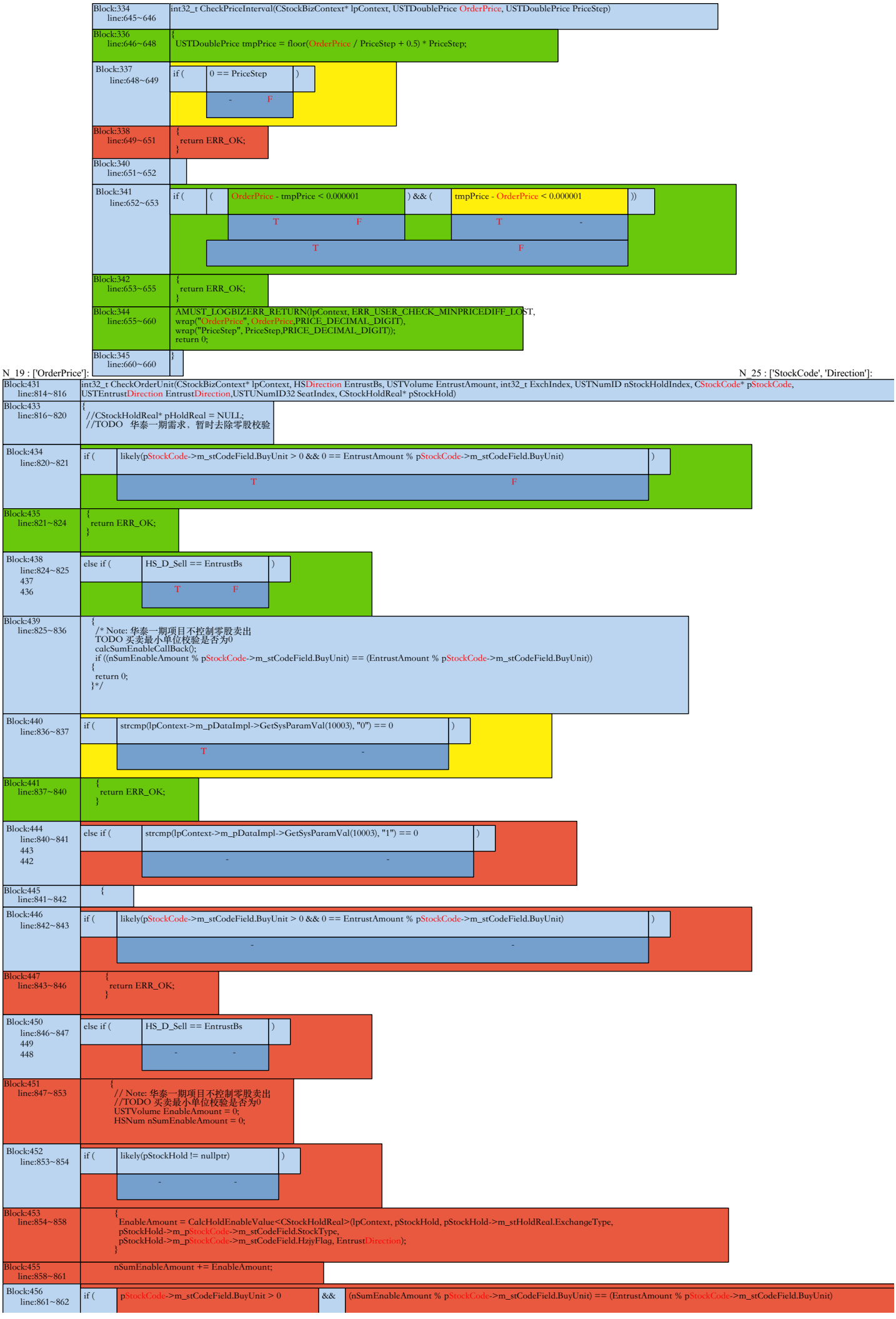
Block:760
line:1447~1452
	{
	/*LOGERROR(lpContext, ERR_USER_ADD_TABLERECORD_FAIL, "新增实例持仓记录失败",
	"StrategyID:", wrap(pInstance->m_stInstanceField.StrategyID),
	"OptionCode:", wrap(nOptionCode));*/
	return pInstanceHold;
	}

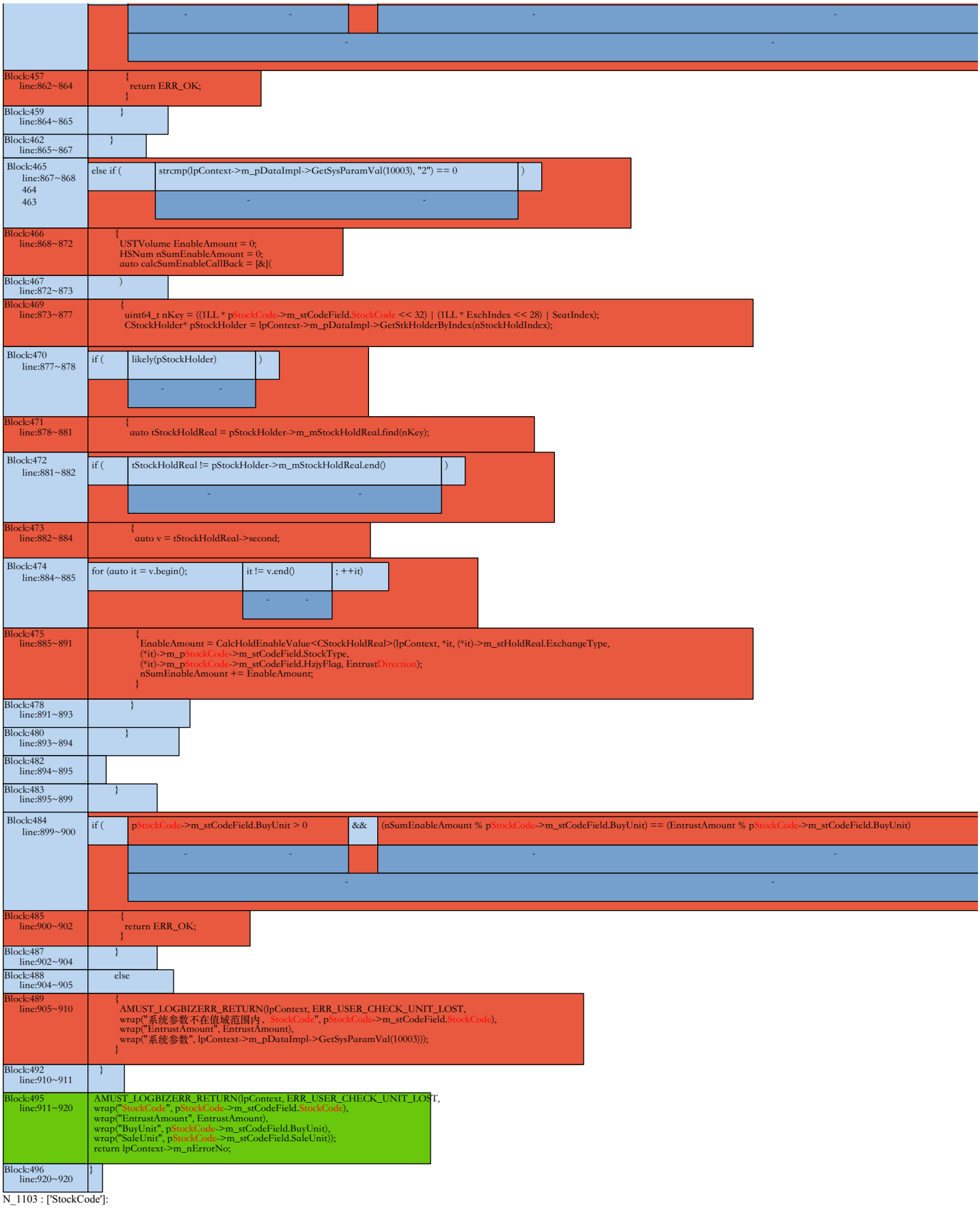
Block:762
line:1452~1464
	memset(&pInstanceHold->m_stInsHoldField, 0, sizeof(pInstanceHold->m_stInsHoldField));
	pInstanceHold->m_stInsHoldField.InstanceID = pInstance->m_stInstanceField.InstanceID;
	memcpy(pInstanceHold->m_stInsHoldField.ExchangeType, sExchangeType, sizeof(pInstanceHold->m_stInsHoldField.ExchangeType));
	pInstanceHold->m_stInsHoldField.StockCode = nStockCode;
	pInstanceHold->m_stInsHoldField.CurrentAmount = 0;
	pInstanceHold->m_stInsHoldField.TargetAmount = 0;
	pInstanceHold->m_stInsHoldField.TemporaryVolume = 0;
	pInstanceHold->m_stInsHoldField.StockHoldIndex = StockHoldIndex;
	cInstanceHoldDealType = CNST_DEALTYPE_ADD;
	}

Block:763 line:1464~1465	else

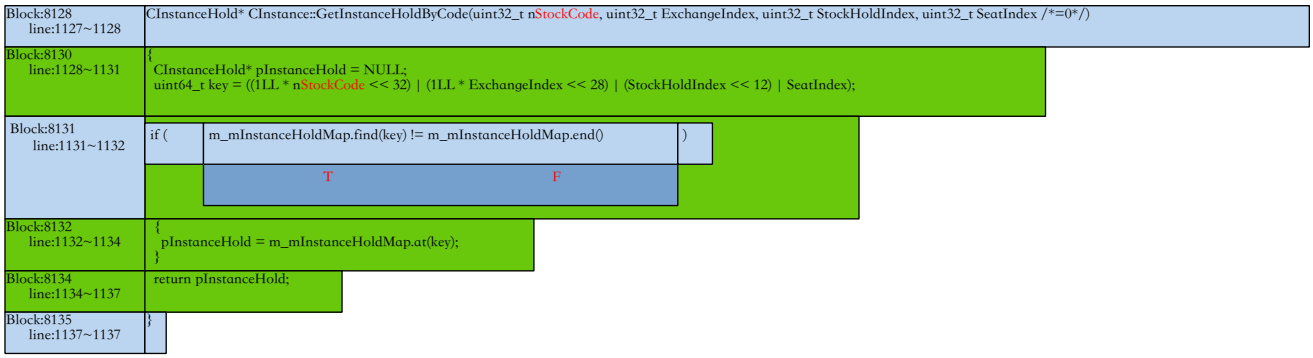
Block:764
line:1465~1467
	{
	cInstanceHoldDealType = CNST_DEALTYPE_MOD;
	}

Block:765 line:1467~1470	return pInstanceHold;
Block:766 line:1470~1470	}

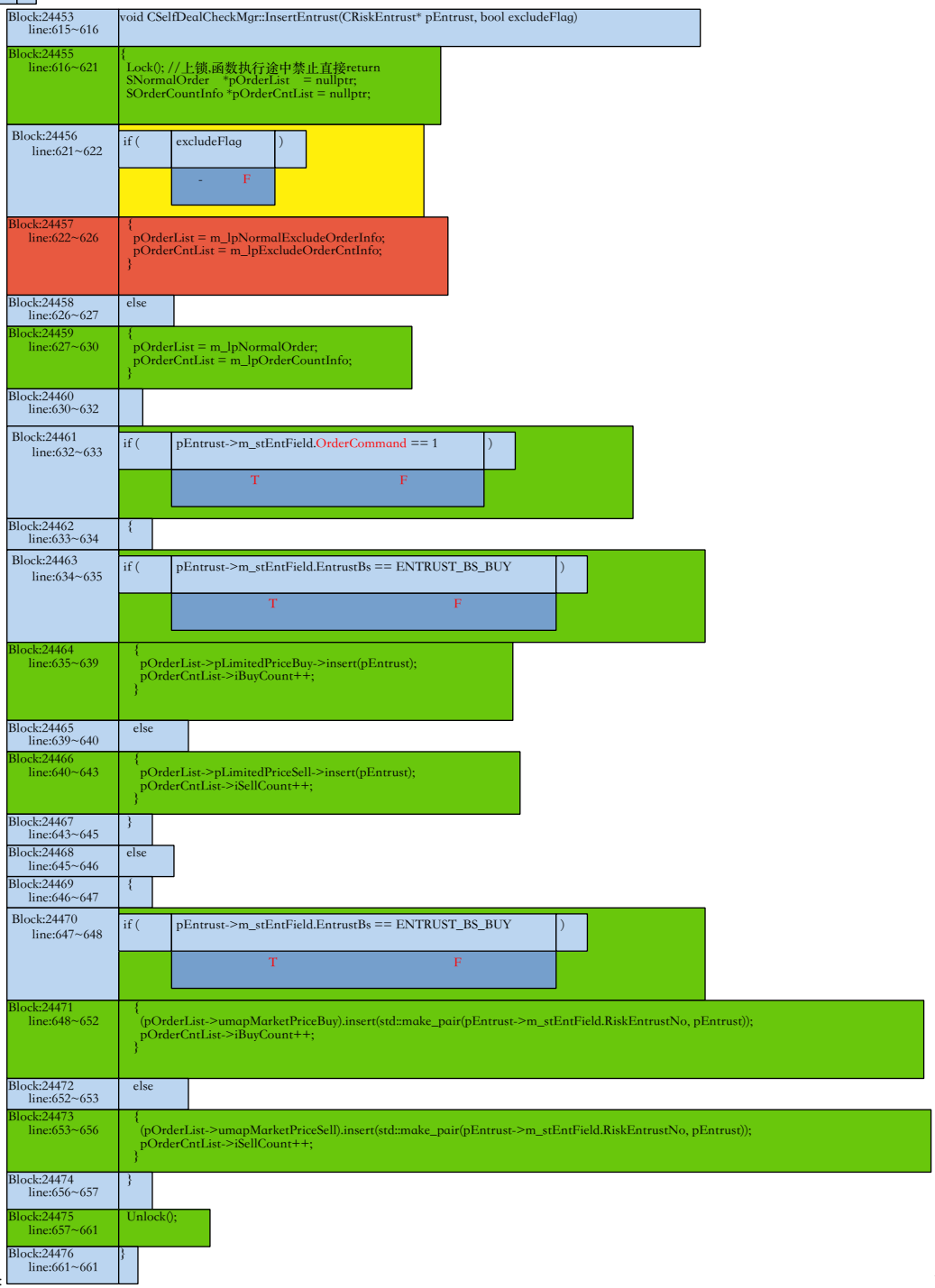

Block:334 line:645~646
int32_t CheckPriceInterval(CStockBizContext* lpContext, USTDoublePrice OrderPrice, USTDoublePrice PriceStep)
Block:336 line:646~648
{
USTDoublePrice tmpPrice = floor(OrderPrice / PriceStep + 0.5) * PriceStep;
Block:337 line:648~649
if (0 == PriceStep)
- F
Block:338 line:649~651
{
return ERR_OK;
}
Block:340 line:651~652
Block:341 line:652~653
if ((OrderPrice - tmpPrice < 0.000001) && (tmpPrice - OrderPrice < 0.000001))
T F T -
T F
Block:342 line:653~655
{
return ERR_OK;
}
Block:344 line:655~660
AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_CHECK_MINPRICEDIFF_LOST,
wrap("OrderPrice", OrderPrice,PRICE_DECIMAL_DIGIT),
wrap("PriceStep", PriceStep,PRICE_DECIMAL_DIGIT));
return 0;
Block:345 line:660~660
}
N_19 : ['OrderPrice']:
N_25 : ['StockCode', 'Direction']:
Block:431 line:814~816
int32_t CheckOrderUnit(CStockBizContext* lpContext, HSDirection EntrustBs, USTVolume EntrustAmount, int32_t ExchIndex, USTNumID nStockHoldIndex, CStockCode* pStockCode,
USTEntrustDirection EntrustDirection,USTUNumID32 SeatIndex, CStockHoldReal* pStockHold)
Block:433 line:816~820
{
//CStockHoldReal* pHoldReal = NULL;
//TODO 华泰一期需求，暂时去除零股校验
Block:434 line:820~821
if (likely(pStockCode->m_stCodeField.BuyUnit > 0 && 0 == EntrustAmount % pStockCode->m_stCodeField.BuyUnit))
T F
Block:435 line:821~824
{
return ERR_OK;
}
Block:438 line:824~825 437 436
else if (HS_D_Sell == EntrustBs)
T F
Block:439 line:825~836
{
/* Note: 华泰一期项目不控制零股卖出
TODO 买卖最小单位校验是否为0
calcSumEnableCallBack();
if ((nSumEnableAmount % pStockCode->m_stCodeField.BuyUnit) == (EntrustAmount % pStockCode->m_stCodeField.BuyUnit))
{
return 0;
}*/
Block:440 line:836~837
if (strcmp(lpContext->m_pDataImpl->GetSysParamVal(10003), "0") == 0)
T -
Block:441 line:837~840
{
return ERR_OK;
}
Block:444 line:840~841 443 442
else if (strcmp(lpContext->m_pDataImpl->GetSysParamVal(10003), "1") == 0)
- -
Block:445 line:841~842
{
Block:446 line:842~843
if (likely(pStockCode->m_stCodeField.BuyUnit > 0 && 0 == EntrustAmount % pStockCode->m_stCodeField.BuyUnit))
- -
Block:447 line:843~846
{
return ERR_OK;
}
Block:450 line:846~847 449 448
else if (HS_D_Sell == EntrustBs)
- -
Block:451 line:847~853
{
// Note: 华泰一期项目不控制零股卖出
//TODO 买卖最小单位校验是否为0
USTVolume EnableAmount = 0;
HSNum nSumEnableAmount = 0;
Block:452 line:853~854
if (likely(pStockHold != nullptr))
- -
Block:453 line:854~858
{
EnableAmount = CalcHoldEnableValue<CStockHoldReal>(lpContext, pStockHold, pStockHold->m_stHoldReal.ExchangeType,
pStockHold->m_pStockCode->m_stCodeField.StockType,
pStockHold->m_pStockCode->m_stCodeField.HzjyFlag, EntrustDirection);
}
Block:455 line:858~861
nSumEnableAmount += EnableAmount;
Block:456 line:861~862
if (pStockCode->m_stCodeField.BuyUnit > 0 && (nSumEnableAmount % pStockCode->m_stCodeField.BuyUnit) == (EntrustAmount % pStockCode->m_stCodeField.BuyUnit)



N_1103 : ['StockCode']:



N_2542 :



['OrderCommand']:

N_2768 : ['StockCode', 'OrderVolume']:

Block:29185 line:1762~1763	int32_t CNewStockOrderInsertFlow::UpdateRiskData()

Block:29187
line:1763~1788
	{
	/*填写风控更新需要的入参*/
	SRiskUpdateField tRiskUpdateField;
	tRiskUpdateField.m_lpContext = lpContext;
	//账户对象
	tRiskUpdateField.m_pCombi = m_stStockPublicContext.pCombi;
	tRiskUpdateField.m_pTrader = m_stStockPublicContext.pTrader;
	tRiskUpdateField.m_pInstance = m_stStockPublicContext.pInstance;
	//证券对象
	tRiskUpdateField.m_pStockCode = m_stStockPublicContext.pStockCode;
	tRiskUpdateField.m_pSecurityInfo = m_stStockPublicContext.pExcharg->GetSecurityInfo(m_stStockPublicContext.pStockCode->m_stCodeField.StockCodeStr);
	//AB类
	tRiskUpdateField.m_iEntrustAmount = m_stStockPublicContext.nOrderVolume;
	tRiskUpdateField.m_nCommission = m_stStockPublicContext.nCommission;
	//F类
	tRiskUpdateField.m_cEntrustBs = m_stStockPublicContext.cEntrustBs;
	tRiskUpdateField.m_iPreSum = lpContext->m_pDataImpl->m_pRiskData->GetPreSumFlag();
	tRiskUpdateField.m_nEntrustBalance = m_stStockPublicContext.nEntrustBalance;
	//J类
	tRiskUpdateField.m_pRiskEntrust = m_stStockPublicContext.pRiskEntrust;
	CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
	m_stStockPublicContext.m_bIsRiskUpdated = true;
	return lpContext->m_nErrorNo;

Block:29188 line:1788~1788	}

N_2769 :

['StockCode', 'Direction']:

```
Block:29189 line:1796~1797   int32_t CNewStockOrderInsertFlow::CheckEnable()
Block:29191 line:1797~1799   {
                               INFO_LOG(lpContext,"可用相关检查",wrap("cEntrustBs", m_stStockPublicContext.cEntrustBs));
Block:29192 line:1799~1800   if ( m_stStockPublicContext.cEntrustBs == CNST_ENTRUSTBS_BUY )
                                 T                                F
Block:29193 line:1800~1808     {
                                 //现金差额可用校验(资金类型为ETF申赎的现金差额)
                                 USTBalance nEnableValue = CalcCashEnableValue(lpContext, m_stStockPublicContext.pAsset, m_stStockPublicContext.nExchIndex, CNST_STOCKKIND_STOCK_INDEX,
                                 m_stStockPublicContext.cEntrustDirection, CNST_BUSINPROTYPE_ENTRUST, CNST_CASH_TYPE_NO_DISTINCTION);
                                 CHECK_FUNC_ERR_RETURN;
                                 INFO_LOG(lpContext, "委托冻结金额", wrap("nFrozenBalance", m_stStockPublicContext.nFrozenBalance), wrap("nTotalFrozenBalance", m_stStockPublicContext.pAssetday->TempTotalFrozenBalance),w
Block:29194 line:1808~1809     if ( unlikely((m_stStockPublicContext.pAssetday->TempTotalFrozenBalance + m_stStockPublicContext.nFrozenBalance) > nEnableValue) )
                                   -                                                                                                                                F
Block:29195 line:1809~1817     {
                                 m_nErrorNo = ERR_FUND_CANUSEFUND_NOTENOUGH;
                                 ///报错返回，可用资金不足
                                 AMUST_LOGBIZERR_RETURN(lpContext, ERR_FUND_CANUSEFUND_NOTENOUGH,  wrap("AssetID", m_stStockPublicContext.pAsset->m_stAssetField.AssetID),
                                 wrap("nFrozenBalance", ConvertDouble(m_stStockPublicContext.nFrozenBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
                                 wrap("nTotalFrozenBalance", ConvertDouble(m_stStockPublicContext.pAssetday->TempTotalFrozenBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
                                 wrap("EnableBalance", ConvertDouble(nEnableValue, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT));
                               }
Block:29197 line:1817~1820   }
Block:29200 line:1820~1821   else if ( m_stStockPublicContext.cEntrustBs == CNST_ENTRUSTBS_SALE )
29199                              T                                    -
29198
Block:29201 line:1821~1823     {
                                 //持仓组合可用校验
Block:29202 line:1823~1824     if ( m_stStockPublicContext.pStockHold )
                                   T                          F
Block:29203 line:1824~1827       {
                                   m_nErrorNo = CheckStockEnable(lpContext, m_stStockPublicContext.pStockHold, m_stStockPublicContext.nStockCode, m_stStockPublicContext.nFrozenVolume + m_stStockPublicContext.pStockHold->r
                                 }
Block:29204 line:1827~1828       else
Block:29205 line:1828~1833       {
                                   m_nErrorNo = ERR_SECU_STOCK_CANUSEAMT_NOTENOUGH;
                                   AMUST_LOGBIZERR_RETURN(lpContext, ERR_SECU_STOCK_CANUSEAMT_NOTENOUGH, wrap("StockCode", m_stStockPublicContext.pStockCode->m_stCodeField.StockCodeStr),
                                   wrap("FrozenVolume", m_stStockPublicContext.nFrozenVolume),
                                   "找不到该证券的持仓，EnableAmount:0");
                                 }
Block:29206 line:1833~1837       CHECK_BIZ_FUNC_ERR_RETURN;
Block:29207 line:1837~1838     if ( m_stStockPublicContext.StrategyIndex > 0 )
                                   T                                F
Block:29208 line:1838~1840       {
                                   //实例持仓可用校验
Block:29209 line:1840~1841     if ( m_stStockPublicContext.pInstanceHold )
                                   T                            F
Block:29210 line:1841~1845       {
                                   USTVolume FrozenVolume = m_stStockPublicContext.nFrozenVolume + m_stStockPublicContext.pInstanceHold->m_stInsHoldField.TemporaryVolume;
                                   m_nErrorNo = CheckInsHoldEnable(lpContext, m_stStockPublicContext.pInstanceHold, FrozenVolume, m_stStockPublicContext.nStockCode, m_stStockPublicContext.pInstance->m_stInstanceField.Inst
                                 }
Block:29211 line:1845~1846       else
Block:29212 line:1846~1851       {
                                   m_nErrorNo = ERR_SECU_STOCK_CANUSEAMT_NOTENOUGH;
                                   AMUST_LOGBIZERR_RETURN(lpContext, ERR_INSTANCE_AMOUNT_NOTENOUGH, "EnableAmount:0", wrap("StockCode", m_stStockPublicContext.pStockCode->m_stCodeField.StockCodeStr),
                                   wrap("InstanceID", m_stStockPublicContext.pInstance->m_stInstanceField.InstanceID),
                                   wrap("nOccurAmount", m_stStockPublicContext.nFrozenVolume));
                                 }
Block:29213 line:1851~1854       CHECK_BIZ_FUNC_ERR_RETURN;
                               }
Block:29215 line:1854~1855   }
Block:29218 line:1855~1858   return m_nErrorNo;
Block:29219 line:1858~1858   }
```

N_3067 : ['ExchangeID', 'StockCode', 'SeatNo', 'Direction', 'OrderVolume']:

Block:34966 line:1566~1567
```
int32_t CETFOrderInsertFlow::UpdateETFHoldReal()
```

Block:34968 line:1567~1571
```
{
 //持仓查询，不存在就插入一条
 CStockHoldReal* pHoldReal = m_stPublicContext.pCombi->GetHoldRealByCode(atoi(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode),
 m_stPublicContext.nExchIndex, m_stPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stPublicContext.SeatIndex);
```

Block:34969 line:1571~1572
```
if ( unlikely(pHoldReal == nullptr) )
```
T F

Block:34970 line:1572~1603
```
{
 pHoldReal = m_stPublicContext.pCombi->NewStockHoldReal(m_stPublicContext.pStockHolder,atoi(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode),
 m_stPublicContext.nExchIndex, m_stPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex, m_stPublicContext.SeatIndex);
 CHECK_BIZ_OBJ_ADD_FAIL(pHoldReal, m_nErrorNo, "新增持仓记录失败"
 , wrap("CombiID", m_stPublicContext.pCombi->m_stCombiField.CombiID)
 , wrap("nExchIndex", m_stPublicContext.nExchIndex)
 , wrap("StockHoldIndex", m_stPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex)
 , wrap("SecurityCode", m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode));
 pHoldReal->m_stHoldReal.AssetID = m_stPublicContext.pCombi->m_stCombiField.AssetID;
 pHoldReal->m_stHoldReal.CombiID = m_stPublicContext.pCombi->m_stCombiField.CombiID;
 memcpy(pHoldReal->m_stHoldReal.ExchangeType, m_stPublicContext.ExchangeID, sizeof(pHoldReal->m_stHoldReal.ExchangeType));
 pHoldReal->m_stHoldReal.StockCode = atoi(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode);
 memcpy(pHoldReal->m_stHoldReal.StockAccount, m_stPublicContext.pStockHolder->m_stStockHolder.StockAccount, sizeof(pHoldReal->m_stHoldReal.StockAccount));
 pHoldReal->m_stHoldReal.StockHoldIndex = m_stPublicContext.pStockHolder->m_stStockHolder.StockHoldIndex;
 pHoldReal->m_stHoldReal.BeginAmount = 0;
 pHoldReal->m_stHoldReal.TemporaryVolume = 0;
 pHoldReal->m_stHoldReal.CurrentAmount = 0;
 pHoldReal->m_pExcharg = m_stPublicContext.pExcharg;
 pHoldReal->m_pStockCode = m_stContext.pETFCode;
 strncpy(pHoldReal->m_stHoldReal.BindSeat, m_stPublicContext.pPositionSeat->m_stSeatField.SeatNo, sizeof(pHoldReal->m_stHoldReal.BindSeat) - 1);
 pHoldReal->m_stHoldReal.BindSeat[sizeof(pHoldReal->m_stHoldReal.BindSeat) - 1] = '\0';
 pHoldReal->m_stHoldReal.SeatIndex = m_stPublicContext.SeatIndex;
 m_stContext.pETFHold = pHoldReal;
 m_stContext.cHoldRealDealType = CNST_DEALTYPE_ADD;
}
```

Block:34971 line:1603~1604
```
else
```

Block:34972 line:1604~1607
```
{
 m_stContext.pETFHold = pHoldReal;
 m_stContext.cHoldRealDealType = CNST_DEALTYPE_UNDO;
}
```

Block:34973 line:1607~1628
```
/*持仓redo*/
SRedoStockHoldField redoHoldField;
redoHoldField.cStockHoldDealType = m_stContext.cHoldRealDealType;
memcpy(&redoHoldField.stETFHoldRealField, &pHoldReal->m_stHoldReal, sizeof(redoHoldField.stETFHoldRealField));
m_stContext.vStockHoldField.push_back(redoHoldField);
INFO_LOG(lpContext, "持仓redo", wrap("cStockHoldDealType", m_stContext.cHoldRealDealType),
wrap("AssetID", redoHoldField.stETFHoldRealField.AssetID),
wrap("BeginAmount", redoHoldField.stETFHoldRealField.BeginAmount),
wrap("CombiID", redoHoldField.stETFHoldRealField.CombiID),
wrap("BeginCarryoverCost", redoHoldField.stETFHoldRealField.BeginCarryoverCost, PRICE_DECIMAL_DIGIT),
wrap("CurrentAmount", redoHoldField.stETFHoldRealField.CurrentAmount),
wrap("ExchangeType", redoHoldField.stETFHoldRealField.ExchangeType),
wrap("StockAccount", redoHoldField.stETFHoldRealField.StockAccount),
wrap("StockCode", redoHoldField.stETFHoldRealField.StockCode),
wrap("StockHoldIndex", redoHoldField.stETFHoldRealField.StockHoldIndex),
wrap("TemporaryVolume", redoHoldField.stETFHoldRealField.TemporaryVolume),
wrap("WaitMarketVolume", redoHoldField.stETFHoldRealField.WaitMarketVolume));
//ETF组合持仓可用因子更新
```

Block:34974 line:1628~1629
```
if ( m_stPublicContext.cEntrustDirection != CNST_ENTRUSTDIR_APPLY )
```
T F

Block:34975 line:1629~1640
```
{
 UpdateHoldEnableValueAndHoldFactor<CStockHoldReal>(lpContext,
 m_stContext.pETFHold,
 m_stContext.pETFHold->m_stHoldReal.ExchangeType,
 m_stContext.pETFHold->m_pStockCode->m_stCodeField.StockType,
 m_stContext.pETFHold->m_pStockCode->m_stCodeField.HzjyFlag,
 m_stPublicContext.cEntrustDirection,
 CNST_BUSINPROTYPE_ENTRUST,
 m_stPublicContext.nOrderVolume,
 m_stContext.vHoldEnableFactorField,
 m_stContext.pETFHold->m_stHoldReal.StockCode);
}
```

Block:34977 line:1640~1645
```
//ETF持仓redo数据拷贝
return m_nErrorNo;
```

Block:34978 line:1645~1645
```
}
```

Block:8119 line:1108~1109
```
CInstanceHold* CInstance::NewInstanceHold(uint32_t nStockCode, uint32_t ExchangeIndex, uint32_t StockHoldIndex, uint32_t SeatIndex /*=0*/)
```

Block:8121 line:1109~1113
```
{
 uint32_t nRecordCount = 0;
 int32_t nErrorNo = 0;
 CInstanceHold* pInstanceHold = m_tInstanceHold.CreateRecord(&nRecordCount, &nErrorNo);
```

Block:8122 line:1113~1114
```
if ( nErrorNo != 0 )
```
- F

Block:8123 line:1114~1118
```
{
 string sErrorInfo = "CInstanceHold::NewInstanceHold,分配内存失败:" + to_string(nErrorNo);
 //LOGERROR(sErrorInfo); //TODO 暂不支持
}
```

Block:8124 line:1118~1119
```
else
```

Block:8125 line:1119~1122
```
{
 uint64_t key = ((1LL * nStockCode << 32) | (1LL * ExchangeIndex << 28) | (StockHoldIndex << 12) | SeatIndex);
 m_mInstanceHoldMap.insert({ key, pInstanceHold });
}
```

Block:8126 line:1122~1125
```
return pInstanceHold;
```

Block:8127 line:1125~1125
```
}
```

N_1102 : ['StockCode']:
N_24 : ['StockCode', 'OrderCommand']:

```
Block:423 line:800~801   int32_t CHeckMarketOrderCommand(CStockBizContext* lpContext, USTExchangeIndex nExchIndex, HSOrderCommand OrderCommand, CStockCode* pStockCode)
Block:425 line:801~803   {
                         //创业板前五日不可以市价申报
Block:426 line:803~804   if( CNST_UST_ORDERCOMMAND_LIMIT != OrderCommand && CNST_STOCK_SECOND_BOARD == pStockCode->m_stCodeField.StockProperty && 0 == pStockCode->m_stCodeField.Up
                             T F   T F   T
                             T F
Block:427 line:804~807   {
                           AMUST_LOGBIZERR_RETURN(lpContext, ERR_CHECK_MAKKET_ORDER_COMMAND, wrap("nExchIndex", nExchIndex),
                           wrap("OrderCommand", OrderCommand), wrap("StockCode", pStockCode->m_stCodeField.StockCode), "UpLimitedRatio: ", wrap(pStockCode->m_stCodeField.UpLimitedRatio) );
                         }
Block:429 line:807~810   return ERR_OK;
Block:430 line:810~810   }
```

N_18 : ['StockCode', 'OrderPrice']:

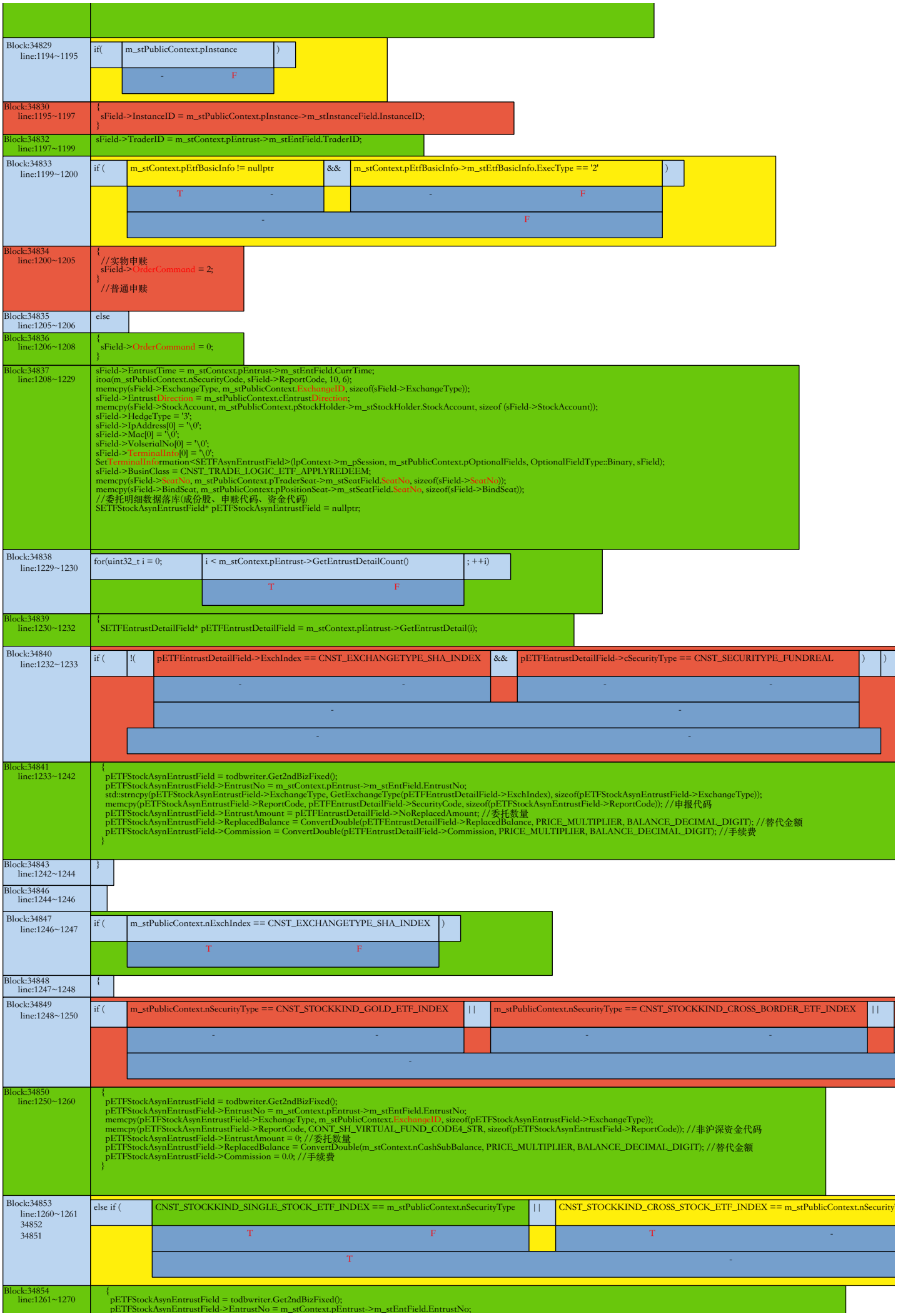
```
Block:322 line:628~629   int32_t CheckOrderPriceLimit(CStockBizContext* lpContext, USTPrice1 OrderPrice, USTPrice1 UpPrice, USTPrice1 DownPrice, USTStockCodeStr StockCodeStr)
Block:324 line:629~630   {
Block:325 line:630~631   if ( unlikely(UpPrice == 0 || DownPrice == 0) )
                             - F
Block:326 line:631~633   {
                           return 0;
                         }
Block:328 line:633~634
Block:329 line:634~635   if ( unlikely(OrderPrice>UpPrice || OrderPrice < DownPrice) )
                             T F
Block:330 line:635~640   {
                           ///报错返回，超过涨跌停范围
                           AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_OUTOFBOUND,wrap("StockCode", StockCodeStr), wrap("OrderPrice", ConvertDouble(OrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGI
                           wrap("UpPrice", ConvertDouble(UpPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT),
                           wrap("DownPrice", ConvertDouble(DownPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), PRICE_DECIMAL_DIGIT));
                         }
Block:332 line:640~642   return 0;
Block:333 line:642~642   }
```

```
Block:467 line:872~873   )
Block:469 line:873~877   {
                           uint64_t nKey = ((1LL * pStockCode->m_stCodeField.StockCode << 32) | (1LL * ExchIndex << 28) | SeatIndex);
                           CStockHolder* pStockHolder = lpContext->m_pDataImpl->GetStkHolderByIndex(nStockHoldIndex);
Block:470 line:877~878   if ( likely(pStockHolder) )
                             - -
Block:471 line:878~881   {
                           auto tStockHoldReal = pStockHolder->m_mStockHoldReal.find(nKey);
Block:472 line:881~882   if ( tStockHoldReal != pStockHolder->m_mStockHoldReal.end() )
                             - -
Block:473 line:882~884   {
                           auto v = tStockHoldReal->second;
Block:474 line:884~885   for (auto it = v.begin(); it != v.end() ; ++it)
                             - -
Block:475 line:885~891   {
                           EnableAmount = CalcHoldEnableValue<CStockHoldReal>(lpContext, *it, (*it)->m_stHoldReal.ExchangeType,
                           (*it)->m_pStockCode->m_stCodeField.StockType,
                           (*it)->m_pStockCode->m_stCodeField.HzjyFlag, EntrustDirection);
                           nSumEnableAmount += EnableAmount;
                         }
Block:478 line:891~893   }
Block:480 line:893~894   }
Block:482 line:894~895
Block:483 line:895~895   }
```

N_26 : ['StockCode', 'Direction']:

N_3059 : ['IPAddress', 'VolSerialNo', 'TerminalInfo', 'ExchangeID', 'SeatNo', 'Direction', 'OrderPrice', 'OrderVolume', 'OrderCommand']:

```
Block:34826 line:1172~1173   int32_t CETFOrderInsertFlow::WriteBusinToDBRedo()
Block:34828 line:1173~1194   {
                               CRedoLdpMsgWriterHelperTwo<SETFAsynEntrustField, SETFStockAsynEntrustField> todbwriter(lpContext);
                               todbwriter.TagTodb(m_stContext.pEntrust->m_stEntField.EntrustNo);
                               todbwriter.GetHead()->FunctionID = UST_FUNC_ASYN_ETF_INSERT_REDO;
                               //ETF委托落库数据
                               SETFAsynEntrustField* sField = (SETFAsynEntrustField*)todbwriter.Get1stBizFixed();
                               sField->EntrustAmount = m_stPublicContext.nOrderVolume;
                               sField->EntrustPrice = ConvertDouble(m_stPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);
                               sField->EntrustBalance = ConvertDouble(m_stContext.nEntrustBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT);
                               sField->Commission = ConvertDouble(m_stContext.nCommission, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT);
                               sField->EntrustDate = lpContext->m_pDataImpl->GetSysarg()->m_nInitDate;
                               sField->EntrustNo = m_stContext.pEntrust->m_stEntField.EntrustNo;
                               sField->BatchNo = m_stContext.pEntrust->m_stEntField.BatchNo;
                               sField->ReportNo = m_stContext.pEntrust->m_stEntField.EntrustNo;
                               sField->FundID = m_stPublicContext.pCombi->m_pFund->m_stFundField.FundID;
                               sField->AssetID = m_stPublicContext.pAsset->m_stAssetField.AssetID;
                               sField->CombiID = m_stPublicContext.pCombi->m_stCombiField.CombiID;
                               sField->InstanceID = 0;
```


Block:34829 line:1194~1195
if(m_stPublicContext.pInstance)
Block:34830 line:1195~1197
{ sField->InstanceID = m_stPublicContext.pInstance->m_stInstanceField.InstanceID; }
Block:34832 line:1197~1199
sField->TraderID = m_stContext.pEntrust->m_stEntField.TraderID;
Block:34833 line:1199~1200
if (m_stContext.pEtfBasicInfo != nullptr && m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.ExecType == '2')
Block:34834 line:1200~1205
{ //实物申赎 sField->OrderCommand = 2; } //普通申赎
Block:34835 line:1205~1206
else
Block:34836 line:1206~1208
{ sField->OrderCommand = 0; }
Block:34837 line:1208~1229
sField->EntrustTime = m_stContext.pEntrust->m_stEntField.CurrTime;
itoa(m_stPublicContext.nSecurityCode, sField->ReportCode, 10, 6);
memcpy(sField->ExchangeType, m_stPublicContext.ExchangeID, sizeof(sField->ExchangeType));
sField->EntrustDirection = m_stPublicContext.cEntrustDirection;
memcpy(sField->StockAccount, m_stPublicContext.pStockHolder->m_stStockHolder.StockAccount, sizeof (sField->StockAccount));
sField->HedgeType = '3';
sField->IpAddress[0] = '\0';
sField->Mac[0] = '\0';
sField->VolserialNo[0] = '\0';
sField->TerminalInfo[0] = '\0';
SetTerminalInformation<SETFAsynEntrustField>(lpContext->m_pSession, m_stPublicContext.pOptionalFields, OptionalFieldType::Binary, sField);
sField->BusinClass = CNST_TRADE_LOGIC_ETF_APPLYREDEEM;
memcpy(sField->SeatNo, m_stPublicContext.pTraderSeat->m_stSeatField.SeatNo, sizeof(sField->SeatNo));
memcpy(sField->BindSeat, m_stPublicContext.pPositionSeat->m_stSeatField.SeatNo, sizeof(sField->BindSeat));
//委托明细数据落库(成份股、申赎代码、资金代码)
SETFStockAsynEntrustField* pETFStockAsynEntrustField = nullptr;
Block:34838 line:1229~1230
for(uint32_t i = 0; i < m_stContext.pEntrust->GetEntrustDetailCount() ; ++i)
Block:34839 line:1230~1232
{ SETFEntrustDetailField* pETFEntrustDetailField = m_stContext.pEntrust->GetEntrustDetail(i);
Block:34840 line:1232~1233
if (!(pETFEntrustDetailField->ExchIndex == CNST_EXCHANGETYPE_SHA_INDEX && pETFEntrustDetailField->cSecurityType == CNST_SECURITYPE_FUNDREAL))
Block:34841 line:1233~1242
{ pETFStockAsynEntrustField = todbwriter.Get2ndBizFixed(); pETFStockAsynEntrustField->EntrustNo = m_stContext.pEntrust->m_stEntField.EntrustNo; std::strncpy(pETFStockAsynEntrustField->ExchangeType, GetExchangeType(pETFEntrustDetailField->ExchIndex), sizeof(pETFStockAsynEntrustField->ExchangeType)); memcpy(pETFStockAsynEntrustField->ReportCode, pETFEntrustDetailField->SecurityCode, sizeof(pETFStockAsynEntrustField->ReportCode)); //申报代码 pETFStockAsynEntrustField->EntrustAmount = pETFEntrustDetailField->NoReplacedAmount; //委托数量 pETFStockAsynEntrustField->ReplacedBalance = ConvertDouble(pETFEntrustDetailField->ReplacedBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT); //替代金额 pETFStockAsynEntrustField->Commission = ConvertDouble(pETFEntrustDetailField->Commission, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT); //手续费 }
Block:34843 line:1242~1244
}
Block:34846 line:1244~1246
Block:34847 line:1246~1247
if (m_stPublicContext.nExchIndex == CNST_EXCHANGETYPE_SHA_INDEX)
Block:34848 line:1247~1248
{
Block:34849 line:1248~1250
if (m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX)
Block:34850 line:1250~1260
{ pETFStockAsynEntrustField = todbwriter.Get2ndBizFixed(); pETFStockAsynEntrustField->EntrustNo = m_stContext.pEntrust->m_stEntField.EntrustNo; memcpy(pETFStockAsynEntrustField->ExchangeType, m_stPublicContext.ExchangeID, sizeof(pETFStockAsynEntrustField->ExchangeType)); memcpy(pETFStockAsynEntrustField->ReportCode, CONT_SH_VIRTUAL_FUND_CODE4_STR, sizeof(pETFStockAsynEntrustField->ReportCode)); //非沪深资金代码 pETFStockAsynEntrustField->EntrustAmount = 0; //委托数量 pETFStockAsynEntrustField->ReplacedBalance = ConvertDouble(m_stContext.nCashSubBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT); //替代金额 pETFStockAsynEntrustField->Commission = 0.0; //手续费 }
Block:34853 line:1260~1261 34852 34851
else if (CNST_STOCKKIND_SINGLE_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType || CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurity
Block:34854 line:1261~1270
{ pETFStockAsynEntrustField = todbwriter.Get2ndBizFixed(); pETFStockAsynEntrustField->EntrustNo = m_stContext.pEntrust->m_stEntField.EntrustNo;

```
        memcpy(pETFStockAsynEntrustField->ExchangeType, m_stPublicContext.ExchangeID, sizeof(pETFStockAsynEntrustField->ExchangeType));
        memcpy(pETFStockAsynEntrustField->ReportCode, CONT_SH_VIRTUAL_FUND_CODE1_STR, sizeof(pETFStockAsynEntrustField->ReportCode)); //沪市资金代码
        pETFStockAsynEntrustField->EntrustAmount = 0; //委托数量
        pETFStockAsynEntrustField->ReplacedBalance = ConvertDouble(m_stContext.nSHCashSubBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT); //替代金额
        pETFStockAsynEntrustField->Commission = 0.0; //手续费
```

Block:34855 line:1270~1271

```
if ( CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType )
```

T F

Block:34856 line:1271~1280

```
    {
     pETFStockAsynEntrustField = todbwriter.Get2ndBizFixed();
     pETFStockAsynEntrustField->EntrustNo = m_stContext.pEntrust->m_stEntField.EntrustNo;
     memcpy(pETFStockAsynEntrustField->ExchangeType, m_stPublicContext.ExchangeID, sizeof(pETFStockAsynEntrustField->ExchangeType));
     memcpy(pETFStockAsynEntrustField->ReportCode, CONT_SH_VIRTUAL_FUND_CODE2_STR, sizeof(pETFStockAsynEntrustField->ReportCode)); //深市资金代码
     pETFStockAsynEntrustField->EntrustAmount = 0; //委托数量
     pETFStockAsynEntrustField->ReplacedBalance = ConvertDouble(m_stContext.nSZCashSubBalance, PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT); //替代金额
     pETFStockAsynEntrustField->Commission = 0.0; //手续费
    }
```

Block:34858 line:1280~1281

```
    }
```

Block:34861 line:1281~1282

```
}
```

Block:34863 line:1282~1285

```
//写落库redo
```

Block:34864 line:1285~1286

```
if ( unlikely(todbwriter.Done() != ERR_OK) )
```

- F

Block:34865 line:1286~1289

```
{
 ERROR_LOG(lpContext, ERR_WRITEREDO_FAIL,"【股票ETF】写异步落库redo失败\n", wrap("AssetID", m_stPublicContext.pAsset->m_stAssetField.AssetID),
 wrap("EntrustNo", m_stContext.pEntrust->m_stEntField.EntrustNo));
}
```

Block:34867 line:1289~1292

```
return m_nErrorNo;
```

Block:34868 line:1292~1292

```
}
```

N_3058 : ['ExchangeID', 'SeatNo', 'Direction', 'OrderPrice', 'OrderVolume']:

Block:34822 line:1107~1108

```
int32_t CETFOrderInsertFlow::OutMarketOfferOrder()
```

Block:34824 line:1108~1166

```
{
 /*
 USTNumID BusinFlag = CNST_BUSINFLAG_ETF_APPLYREDEEM;
 if(CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType)
 {
  BusinFlag = CNST_BUSINFLAG_ETF_OUTCROSSMARKET;
 }
 //报盘结构体
 STOCK_ETF_OUT_MARKET_MSG sReportMsg;
 sReportMsg.MsgHead.FunctionID = FUNC_OFFER_REPORT;
 sReportMsg.MsgHead.UserDefined = 0; //普通交易写死0，报盘根据这个识别什么业务
 sReportMsg.MsgHead.Token = m_stPublicContext.pAsset->m_stAssetField.GlobalIndex;
 memcpy(sReportMsg.OrderWthdrw.ExchangeType, m_stPublicContext.ExchangeID, sizeof (sReportMsg.OrderWthdrw.ExchangeType));
 sReportMsg.OrderWthdrw.BusinFlag = BusinFlag;
 memcpy(sReportMsg.OrderWthdrw.StockAccount, m_stPublicContext.pStockHolder->m_stStockHolder.StockAccount, sizeof (sReportMsg.OrderWthdrw.StockAccount));
 sReportMsg.OrderWthdrw.EntrustType = CNST_ENTRUSTTYPE_NORMAL;
 memcpy(sReportMsg.OrderWthdrw.SeatNo, m_stPublicContext.pSeat->m_stSeatField.SeatNo, sizeof (sReportMsg.OrderWthdrw.SeatNo));
 memcpy(sReportMsg.OrderWthdrw.ETFCode, m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.SecurityCode, sizeof(sReportMsg.OrderWthdrw.ETFCode)); //二级代码
 sReportMsg.OrderWthdrw.ReportPrice = ConvertDouble(m_stPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);
 sReportMsg.OrderWthdrw.EntrustBs = m_stPublicContext.cEntrustDirection + '0' - 2; //申购传'1' 赎回传'2'
 sReportMsg.OrderWthdrw.ReportAmount = m_stPublicContext.nOrderVolume;
 sReportMsg.OrderWthdrw.EntrustNo = m_stContext.pEntrust->m_stEntField.EntrustNo;
 memcpy(sReportMsg.OrderWthdrw.AffiliatedStockAccount, m_stPublicContext.pAffiliatedStockHolder->m_stStockHolder.StockAccount, sizeof (sReportMsg.OrderWthdrw.AffiliatedStockAccount));
 memcpy(sReportMsg.OrderWthdrw.AffiliatedSeatNo, m_stPublicContext.pAffiliatedSeat->m_stSeatField.SeatNo, sizeof (sReportMsg.OrderWthdrw.AffiliatedSeatNo));
 sReportMsg.OrderWthdrw.ExternOfferReportBufcount = m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size();
 //遍历成份股
 for(size_t i = 0; i < m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size() && i < 1024 ; ++i)
 {
  CEtfStocklist* pEtfStockFiled = m_stContext.pEtfBasicInfo->m_vecEtfStocklist[i];
  memcpy(sReportMsg.OrderWthdrw.ExternOfferReportBuf[i].SecurityCode, pEtfStockFiled->mstEtfStocklist.SecurityCodeElement, sizeof(sReportMsg.OrderWthdrw.ExternOfferReportBuf[i].SecurityCode));
  memcpy(sReportMsg.OrderWthdrw.ExternOfferReportBuf[i].ExchangeType, pEtfStockFiled->mstEtfStocklist.ExchangeTypeElement, sizeof(sReportMsg.OrderWthdrw.ExternOfferReportBuf[i].ExchangeType));
  sReportMsg.OrderWthdrw.ExternOfferReportBuf[i].ApplicationVol = m_stContext.vStockAmount[i] - m_stContext.vStockRepAmount[i]; //未替代数量BALANCE_DECIMAL_DIGIT
  sReportMsg.OrderWthdrw.ExternOfferReportBuf[i].ApplicationAmount = ConvertDouble(m_stContext.vStockRepBalance[i], PRICE_MULTIPLIER, BALANCE_DECIMAL_DIGIT); //替代金额
 }
 if (unlikely((m_stPublicContext.pSeat->GetTransParam(2) == nullptr) ||
 (m_stPublicContext.pSeat->GetTransParam(2)->ReportStatus != CNST_TRANSPARAM_REPORTSTATUS)))
 {
  INFO_LOG(lpContext, "未开报盘", wrap("EntrustNo", sReportMsg.OrderWthdrw.EntrustNo));
  return ERR_OK;
 }
 //int32_t nRet = lpContext->PostMsg2Offer(m_stPublicContext.pSeat->GetTransParam(1)->TransIndex, (void *) &sReportMsg, sizeof(STOCK_OFFER_ORDER_WTHDRW_MSG));
 int32_t nRet = lpContext->PostMsg2Offer(m_stPublicContext.pSeat->GetTransParam(2)->TransIndex, (void *) &sReportMsg, sizeof(STOCK_ETF_OUT_MARKET_MSG));
 if (nRet == ERR_OK)//组播新方案，不论成功还是失败都置成待报
 {
  m_stContext.pEntrust->m_stEntField.EntrustStatus = CNST_ENTRUSTSTATUS_WAITREPORT;
 }
 else
 {
  ERROR_LOG(lpContext, ERR_OFFER_INTERATION, "报单失败", wrap("EntrustNo", sReportMsg.OrderWthdrw.EntrustNo));
 }
 */
 return ERR_OK;
```

Block:34825 line:1166~1166

```
}
```

N_3070 : ['StockCode', 'OrderVolume']:

Block:35041 line:1988~1989

```
int32_t CETFOrderInsertFlow::InsertEntrustDeatil()
```

Block:35043 line:1989~1992

```
{
 //成份股委托明细
 SETFEntrustDetailField* pETFEntrustDetail = nullptr;
```

Block:35044 line:1992~1993

```
for(size_t i = 0; i < m_stContext.pEtfBasicInfo->m_vecEtfStocklist.size() ; ++i)
```

T F

Block:35045 line:1993~1996

```
 {
  CEtfStocklist* pEtfStockFiled = m_stContext.pEtfBasicInfo->m_vecEtfStocklist[i];
```

Block:35046 line:1996~1998

```
if ( m_stPublicContext.nSecurityType == CNST_STOCKKIND_GOLD_ETF_INDEX || m_stPublicContext.nSecurityType == CNST_STOCKKIND_CROSS_BORDER_ETF_INDEX ||
```


Block:35047 line:1998~2001
{
//黄金 货币 跨境ETF不需要展示成份股信息
continue;
}
Block:35049 line:2001~2003
//深圳跨市场ETF 场内申赎 非深市成分股不需要插入
Block:35050 line:2003~2006
if(
JudgeSZVirtualFundCode(m_stContext.vStockCode[i])
||(
CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType
&&
m_stPublicContext
Block:35051 line:2006~2009
{
}
Block:35052 line:2009~2010
else
Block:35053 line:2010~2018
{
//查找股票代码
USTStockCode nStockCode = m_stContext.vStockCode[i];
pETFEntrustDetail = m_stContext.pEntrust->NewEntrustDetail(nStockCode, m_stContext.vStockIndex[i]);
CHECK_BIZ_OBJ_ADD_FAIL(pETFEntrustDetail, m_nErrorNo, "新增ETF委托明细记录失败",
wrap("AccountIndex", m_stPublicContext.nAssetIndex));
CStockCode* pStockCode = m_stPublicContext.pExcharg->GetStockCode(nStockCode);
Block:35054 line:2018~2019
if (
nullptr == pStockCode
&&
CNST_STOCKKIND_CROSS_STOCK_ETF_INDEX == m_stPublicContext.nSecurityType
)
T
F
T
T
F
Block:35055 line:2019~2021
{
pStockCode = m_stPublicContext.pOtherExcharg->GetStockCode(nStockCode);
}
Block:35057 line:2021~2042
AMUST_LOGBIZERR_CHECK_OBJ_NOT_FOUND_INNER_ASSIGN(pStockCode, "找不到成份股代码信息 ", wrap("StockCode", nStockCode));
pETFEntrustDetail->ExchIndex = m_stContext.vStockIndex[i];
itoa(pStockCode->m_stCodeField.StockCode, pETFEntrustDetail->SecurityCode, 10, 6); //成份股代码
pETFEntrustDetail->NoReplacedAmount = m_stContext.vStockAmount[i] - m_stContext.vStockRepAmount[i];
pETFEntrustDetail->ReplacedAmount = m_stContext.vStockRepAmount[i];
pETFEntrustDetail->ReplacedBalance = m_stContext.vStockRepBalance[i];
pETFEntrustDetail->Commission = m_stContext.vCommission[i];
pETFEntrustDetail->CashSubstitueType = pEtfStockFiled->mstEtfStocklist.CashSubstitueType;
pETFEntrustDetail->cSecurityType = CNST_SECURITYPE_COMPOSITION;
INFO_LOG(lpContext, "成分股委托明细", wrap("ExchIndex", pETFEntrustDetail->ExchIndex),
wrap("SecurityCode", pETFEntrustDetail->SecurityCode),
wrap("cSecurityType", pETFEntrustDetail->cSecurityType),
wrap("ReplacedBalance", ConvertDouble(pETFEntrustDetail->ReplacedBalance, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
wrap("ReplacedAmount", pETFEntrustDetail->ReplacedAmount),
wrap("NoReplacedAmount", pETFEntrustDetail->NoReplacedAmount),
wrap("CashSubstitueType", pETFEntrustDetail->CashSubstitueType),
wrap("Commission", pETFEntrustDetail->Commission));
}
Block:35058 line:2042~2043
}
Block:35061 line:2043~2068
//申赎代码委托明细(一级市场代码)
pETFEntrustDetail = m_stContext.pEntrust->NewEntrustDetail(atoi(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode1), GetExchangeIndex(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.ExchangeT
CHECK_BIZ_OBJ_ADD_FAIL(pETFEntrustDetail, m_nErrorNo, "新增ETF委托明细记录失败",
"AccountIndex:", wrap(m_stPublicContext.nAssetIndex));
pETFEntrustDetail->ExchIndex = m_stPublicContext.nExchIndex;
memcpy(pETFEntrustDetail->SecurityCode, m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode1, sizeof(pETFEntrustDetail->SecurityCode)); //申赎代码
pETFEntrustDetail->NoReplacedAmount = m_stPublicContext.nOrderVolume;
pETFEntrustDetail->ReplacedAmount = 0;
pETFEntrustDetail->ReplacedBalance = 0;
pETFEntrustDetail->Commission = 0;
pETFEntrustDetail->CashSubstitueType = '\0';
pETFEntrustDetail->cSecurityType = CNST_SECURITYPE_APPLYREDEEM;
INFO_LOG(lpContext, "申赎代码委托明细", wrap("ExchIndex", pETFEntrustDetail->ExchIndex),
wrap("SecurityCode", pETFEntrustDetail->SecurityCode),
wrap("cSecurityType", pETFEntrustDetail->cSecurityType),
wrap("ReplacedBalance", ConvertDouble(pETFEntrustDetail->ReplacedBalance, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
wrap("ReplacedAmount", pETFEntrustDetail->ReplacedAmount),
wrap("NoReplacedAmount", pETFEntrustDetail->NoReplacedAmount),
wrap("CashSubstitueType", pETFEntrustDetail->CashSubstitueType),
wrap("Commission", pETFEntrustDetail->Commission));
//资金代码委托明细
Block:35062 line:2068~2069
if (
m_stPublicContext.nSecurityType == CNST_STOCKKIND_SINGLE_STOCK_ETF_INDEX
)
T
F
Block:35063 line:2069~2070
{
Block:35064 line:2070~2071
if (
m_stPublicContext.nExchIndex == CNST_EXCHANGETYPE_SHA_INDEX
)
T
F
Block:35065 line:2071~2085
{
pETFEntrustDetail = m_stContext.pEntrust->NewEntrustDetail(atoi(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode1), GetExchangeIndex(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.Exchang
CHECK_BIZ_OBJ_ADD_FAIL(pETFEntrustDetail, m_nErrorNo, "新增ETF委托明细记录失败",
"AccountIndex:", wrap(m_stPublicContext.nAssetIndex));
pETFEntrustDetail->ExchIndex = m_stPublicContext.nExchIndex;
memcpy(pETFEntrustDetail->SecurityCode, m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode1, sizeof(pETFEntrustDetail->SecurityCode));
pETFEntrustDetail->NoReplacedAmount = 0;
pETFEntrustDetail->ReplacedAmount = 0;
pETFEntrustDetail->ReplacedBalance = m_stContext.nCashSubBalance; //总替代金额
pETFEntrustDetail->Commission = 0;
pETFEntrustDetail->CashSubstitueType = '\0';
pETFEntrustDetail->cSecurityType = CNST_SECURITYPE_FUNDREAL;
}
Block:35067 line:2085~2087
}
Block:35068 line:2087~2088
else
Block:35069 line:2088~2089
{

Block:35070 line:2089~2090

```
if ( CNST_EXCHANGETYPE_SHA_INDEX == m_stPublicContext.nExchIndex )
```

T F

Block:35071 line:2090~2107

```
{
pETFEntrustDetail = m_stContext.pEntrust->NewEntrustDetail(atoi(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode1), GetExchangeIndex(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.Exchang
CHECK_BIZ_OBJ_ADD_FAIL(pETFEntrustDetail, m_nErrorNo, "新增ETF委托明细记录失败",
"AccountIndex:", wrap(m_stPublicContext.nAssetIndex));
pETFEntrustDetail->ExchIndex = m_stPublicContext.nExchIndex;
memcpy(pETFEntrustDetail->SecurityCode, m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfCode1, sizeof(pETFEntrustDetail->SecurityCode)); //资金代码
pETFEntrustDetail->NoReplacedAmount = 0;
pETFEntrustDetail->ReplacedAmount = 0;
pETFEntrustDetail->ReplacedBalance = m_stContext.nCashSubBalance; //总替代金额
pETFEntrustDetail->Commission = 0;
pETFEntrustDetail->CashSubstitueType = '\0';
pETFEntrustDetail->cSecurityType = CNST_SECURITYPE_FUNDREAL;
}
```

Block:35072 line:2107~2108

```
else
```

Block:35073 line:2108~2109

```
{
```

Block:35074 line:2109~2110

```
if ( m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.ExecType == CNST_SZ_ETF_APPLY_REDEEM_INFIELD )
```

T -

Block:35075 line:2110~2129

```
{
pETFEntrustDetail = m_stContext.pEntrust->NewEntrustDetail(CONT_SZ_VIRTUAL_FUND_CODE, GetExchangeIndex(m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.ExchangeType));
CHECK_BIZ_OBJ_ADD_FAIL(pETFEntrustDetail, m_nErrorNo, "新增ETF委托明细记录失败",
"AccountIndex:", wrap(m_stPublicContext.nAssetIndex));
pETFEntrustDetail->ExchIndex = m_stPublicContext.nExchIndex;
memcpy(pETFEntrustDetail->SecurityCode, CONT_SZ_VIRTUAL_FUND_CODE_STR, sizeof(pETFEntrustDetail->SecurityCode)); //资金代码
pETFEntrustDetail->NoReplacedAmount = 0;
pETFEntrustDetail->ReplacedAmount = 0;
//pETFEntrustDetail->ReplacedBalance = m_stContext.nCashSubBalance; //总替代金额
pETFEntrustDetail->Commission = 0;
pETFEntrustDetail->CashSubstitueType = '\0';
pETFEntrustDetail->cSecurityType = CNST_SECURITYPE_FUNDREAL;
//如果是虚拟代码159900，则替代金额表示替代非深市成分股替代金额
pETFEntrustDetail->ReplacedBalance = m_stContext.nVirtualCodeSubBalance; //非深市成分股替代金额
}
```

Block:35077 line:2129~2130

```
}
```

Block:35078 line:2130~2132

```
}
```

Block:35079 line:2132~2134

Block:35080 line:2134~2135

```
if ( likely(pETFEntrustDetail != nullptr) )
```

T -

Block:35081 line:2135~2144

```
{
INFO_LOG(lpContext, "资金代码委托明细", wrap("ExchIndex", pETFEntrustDetail->ExchIndex),
wrap("SecurityCode", pETFEntrustDetail->SecurityCode),
wrap("cSecurityType", pETFEntrustDetail->cSecurityType),
wrap("ReplacedBalance", ConvertDouble(pETFEntrustDetail->ReplacedBalance, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT), BALANCE_DECIMAL_DIGIT),
wrap("ReplacedAmount", pETFEntrustDetail->ReplacedAmount),
wrap("NoReplacedAmount", pETFEntrustDetail->NoReplacedAmount),
wrap("CashSubstitueType", pETFEntrustDetail->CashSubstitueType),
wrap("Commission", pETFEntrustDetail->Commission));
}
```

Block:35083 line:2144~2146

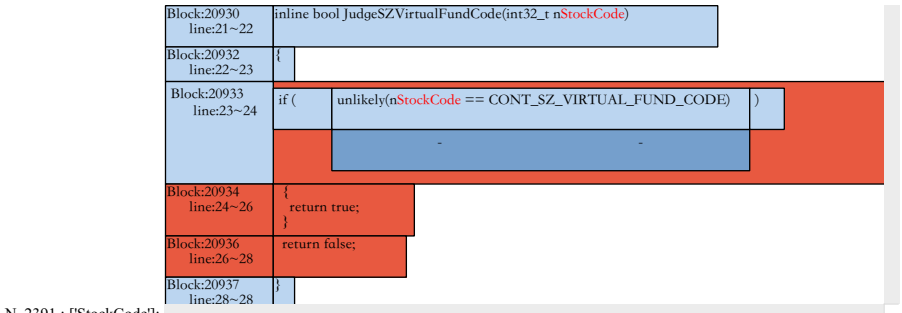
```
return m_nErrorNo;
```

Block:35084 line:2146~2146

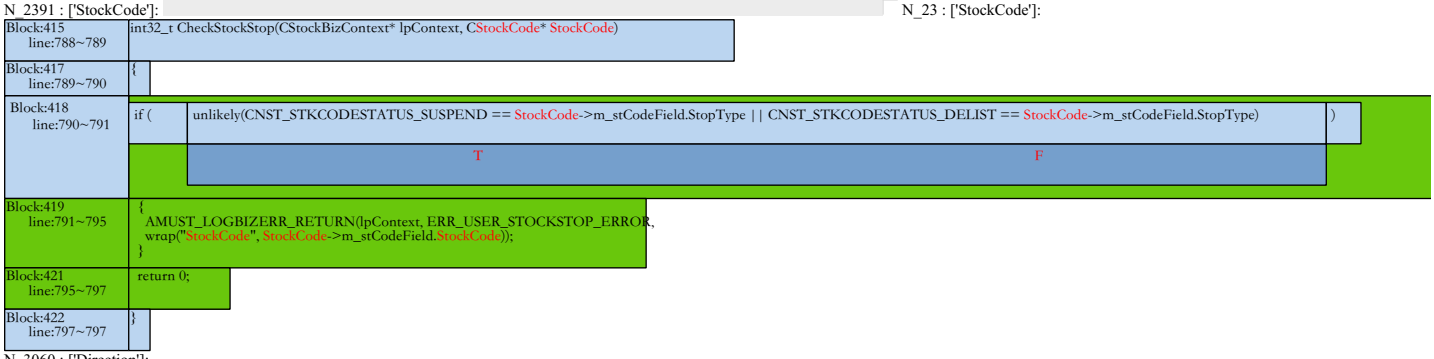
```
}
```

N_3064 : ['StockCode', 'Direction', 'OrderVolume']:

```
Block:34922 line:1425~1426
int32_t CETFOrderInsertFlow::RollBackRiskData()
Block:34924 line:1426~1431
{
 RollBackBusinData();
 //return ERR_OK;
 //未进行风控更新不回滚
Block:34925 line:1431~1432
if ( ! m_stPublicContext.bIsRiskUpdate )
 -
 -
Block:34926 line:1432~1434
 return ERR_OK;
}
Block:34928 line:1434~1445
 //ETF数量更新
 SRiskUpdateField tRiskUpdateField;
 tRiskUpdateField.m_lpContext = lpContext;
 tRiskUpdateField.m_pCombi = m_stPublicContext.pCombi;
 tRiskUpdateField.m_pTrader = m_stPublicContext.pTrader;
 //USTNumID  ReportUnitAmount = m_stContext.pEtfBasicInfo->m_stEtfBasicInfo.EtfReportUnit;
 //申购
Block:34929 line:1445~1446
if ( m_stPublicContext.cEntrustDirection == CNST_ENTRUSTDIR_APPLY )
 -
 -
Block:34930 line:1446~1460
 {
  tRiskUpdateField.m_pSecurityInfo = m_stContext.pEtfBasicInfo->m_pSecurityInfo;
  tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_BUY + '0';
  tRiskUpdateField.m_iEntrustAmount = -m_stPublicContext.nOrderVolume;
  tRiskUpdateField.m_nEntrustBalance = - m_stPublicContext.pRiskEntrust->m_stEntField.OrderBalance;
  tRiskUpdateField.m_nCommission = -m_stContext.nCommission;
  tRiskUpdateField.m_pRiskEntrust = m_stPublicContext.pRiskEntrust;
  //ETF风控数据更新
  CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
  //成分股风控数据更新
  int32_t i = 0;
Block:34931 line:1460~1461
  for (auto& pEtfStocklistField : m_stContext.pEtfBasicInfo->m_vecEtfStocklist)
Block:34932 line:1461~1462
  {
Block:34933 line:1462~1463
if ( pEtfStocklistField->m_pSecurityInfo == NULL )
 -
 -
Block:34934 line:1463~1465
   {
    continue;
   }
Block:34936 line:1465~1480
   tRiskUpdateField.m_pSecurityInfo = pEtfStocklistField->m_pSecurityInfo;
   tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_SALE + '0';
   //tRiskUpdateField.m_iEntrustAmount = m_stPublicContext.nOrderVolume * pEtfStocklistField->mstEtfStocklist.StockAmount / ReportUnitAmount;
   USTNumID iEntrustAmount = (m_stContext.vStockAmount[i] - (USTNumID)m_stContext.vStockRepAmount[i]);
   tRiskUpdateField.m_iEntrustAmount = -iEntrustAmount;
   tRiskUpdateField.m_nEntrustBalance = 0;
   tRiskUpdateField.m_nCommission = 0;
   //成分股风控代码更新
   CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
   ++i;
  }
Block:34939 line:1480~1485
  //AddEtfOrderCashInfo(m_stPublicContext.pCombi->m_pAsset, -(m_stContext.nCashSubBalance), -(m_stContext.nCommission), -(m_stContext.nEstimateCash));
 }
  //赎回
Block:34940 line:1485~1486
 else
Block:34941 line:1486~1500
 {
  //获取ETF信息
  tRiskUpdateField.m_pSecurityInfo = m_stContext.pEtfBasicInfo->m_pSecurityInfo;
  //ETF数量更新
  tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_SALE + '0';
  tRiskUpdateField.m_iEntrustAmount = -m_stPublicContext.nOrderVolume;
  tRiskUpdateField.m_nCommission = -m_stContext.nCommission;
  //ETF风控数据更新
  CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
  int32_t i = 0;
  //成分股风控数据更新
Block:34942 line:1500~1501
  for (auto& pEtfStocklistField : m_stContext.pEtfBasicInfo->m_vecEtfStocklist)
Block:34943 line:1501~1511
  {
   CExcharg *pExcharg = &(lpContext->m_pDataImpl->GetExcharg()[GetExchangeIndex(pEtfStocklistField->mstEtfStocklist.ExchangeTypeElement)]);
   tRiskUpdateField.m_pSecurityInfo = pExcharg->GetSecurityInfo(pEtfStocklistField->mstEtfStocklist.SecurityCodeElement);
   CHECK_OBJ_NOT_FOUND(tRiskUpdateField.m_pSecurityInfo, "找不到成分股证券代码信息 ", "SecurityCodeElement:", wrap(pEtfStocklistField->mstEtfStocklist.SecurityCodeElement));
   tRiskUpdateField.m_cEntrustBs = CNST_ENTRUSTDIR_BUY + '0';
   USTNumID iEntrustAmount = (m_stContext.vStockAmount[i] - (USTNumID)m_stContext.vStockRepAmount[i]);
   tRiskUpdateField.m_iEntrustAmount = -iEntrustAmount;
   CStockCode* pStockCode = pExcharg->GetStockCode(atoi(pEtfStocklistField->mstEtfStocklist.SecurityCodeElement));
Block:34944 line:1511~1512
if( nullptr != pStockCode )
 -
 -
Block:34945 line:1512~1515
   {
    //ETF赎回，成分股相当于买入，委托金额 = 成分股未替代数量数量*昨日结算价
    tRiskUpdateField.m_nEntrustBalance = tRiskUpdateField.m_iEntrustAmount * pStockCode->m_lpStockQuote->YesterdayClosePrice;
   }
Block:34947 line:1515~1521
   //成分股风控代码更新
   CStockRiskUpdate::RiskOrderUpdate(tRiskUpdateField);
   ++i;
  }
Block:34950 line:1521~1522
 }
Block:34951 line:1522~1525
 return ERR_OK;
Block:34952 line:1525~1525
}
```



N_2391 : ['StockCode']:



N_23 : ['StockCode']:

N_3060 : ['Direction']:

N_2418 : ['ExchangeID']:

```
int nOrderCount = 0;
int nSuccessCount = 0;
USTBalance nTotalBalance = 0;
USTBalance nSuccessBalance = 0;
this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordBegin(this->m_lpContext->GetThreadNo(), UST_FUNC_BATCH_ORDER_INSERT);
```

Block:21402 line:1182~1183

```
if ( unlikely(this->m_lpContext->GetLogLevel() <= LEVEL_INFO) )
```

- F

Block:21403 line:1183~1185

```
{
 CLdpMsgReaderHelperTwo<REQFIELD1_T, REQFIELD2_T> readerTemp(this->m_lpContext);
}
```

Block:21405 line:1185~1219

```
//风控触警信息个数请零
this->m_lpContext->m_pRiskContext->SetZeroRiskCount();
SBatchEntrustInfo* pBatchInfo = nullptr;
//系统状态校验
INFO_LOG(this->m_lpContext, "【批量订单服务】系统状态校验");
this->m_lpContext->m_nErrorNo = this->CheckSystemStatus();
CHECK_BATCHORDER_ERR_GOTO_SVREND(this->m_lpContext->m_nErrorNo);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("CheckSystemStatus");
//会话信息获取
INFO_LOG(this->m_lpContext, "【批量订单服务】会话信息获取");
this->m_lpContext->m_nErrorNo = this->GetSession(this->m_lpReqReader->GetHead()->Token);
CHECK_FUNC_ERRNO_GOTO_SVREND(this->m_lpContext->m_nErrorNo);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("GetSession");
//第一个结果集是否为空
INFO_LOG(this->m_lpContext, "【批量订单服务】解析请求第一个");
this->m_lpContext->m_nErrorNo = this->Get1stReqField(true);
CHECK_FUNC_ERRNO_GOTO_SVREND(this->m_lpContext->m_nErrorNo);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("Get1stReqField");
//获取最小委托比例
nMinEntrustRatio = this->GetMinEntrustRatio();
//获取公用数据
INFO_LOG(this->m_lpContext, "【批量订单服务】获取公用数据");
this->m_lpContext->m_nErrorNo = this->GetPublicData();
CHECK_FUNC_ERRNO_GOTO_SVREND(this->m_lpContext->m_nErrorNo);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("GetPublicData");
this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 1);
```

Block:21406 line:1219~1220

```
while (1)
```

Block:21407 line:1220~1226

```
{
 //解析请求入参
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】解析请求",wrap("i",i));
 int32_t &nErrNo = this->m_lpContext->m_nBizErrNo[i] = ERR_OK;
 nErrNo = this->Get2ndReqField(true);
 //取不到为止，退出循环
```

Block:21408 line:1226~1227

```
if ( unlikely(this->m_lpReqField2 == nullptr || nOrderCount >= MAX_BATCH_ORDER_COUNT) )
```

T F

Block:21409 line:1227~1229

```
 {
  break;
 }
```

Block:21411 line:1229~1235

```
 m_arrReqFiled2[i] = (REQFIELD2_T*)(this->m_lpReqField2);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("Get2ndReqField");
 nOrderCount++;
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】创建具体订单处理业务流");
 pOrderImpl = this->m_orderFactory.CreateOrderImpl((REQFIELD2_T*)(this->m_lpReqField2));
```

Block:21412 line:1235~1236

```
if ( unlikely(pOrderImpl == nullptr) )
```

T F

Block:21413 line:1236~1240

```
 {
  WARNING_LOG(this->m_lpContext, "【批量订单服务-单笔】不支持的业务",wrap("i",i), wrap("ExchangeID", ((REQFIELD2_T*)(this->m_lpReqField2))->ExchangeID));
  nErrNo = ERR_ASSET_FUND_BUSINESS_UNKNOWN;
  SET_CONTEXT_BIZ_ERRMSG(this->m_lpContext, nErrNo, wrap("ExchangeID", ((REQFIELD2_T*)(this->m_lpReqField2))->ExchangeID));
 }
```

Block:21415 line:1240~1270

```
 m_arrOrderImpl[i] = pOrderImpl;
 CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("CreateOrderImpl");
 //m_arrOrderImpl[i]->m_nOrderIndex = i;
 //设置委托下标
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】设置委托下标", wrap("订单下标", i));
 pOrderImpl->SetOrderIndex(i);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("SetOrderIndex");
 //初始化
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】初始化", wrap("订单下标", i));
 nErrNo = pOrderImpl->Init((void*)(this->m_lpReqField2), (void*)&m_stPublicContext, BATCH_ORDER, this->m_pSession, this->m_reqOptionalFields);
 CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("Init");
 //业务区分
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】业务区分", wrap("订单下标", i));
 nErrNo = pOrderImpl->DistinguishBusin();
 CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("DistinguishBusin");
 //基础数据获取
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】基础数据获取", wrap("订单下标", i));
 nErrNo = pOrderImpl->GetBasicData();
 CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("GetBasicData");
 //联合风控推送
```

Block:21416 line:1270~1271

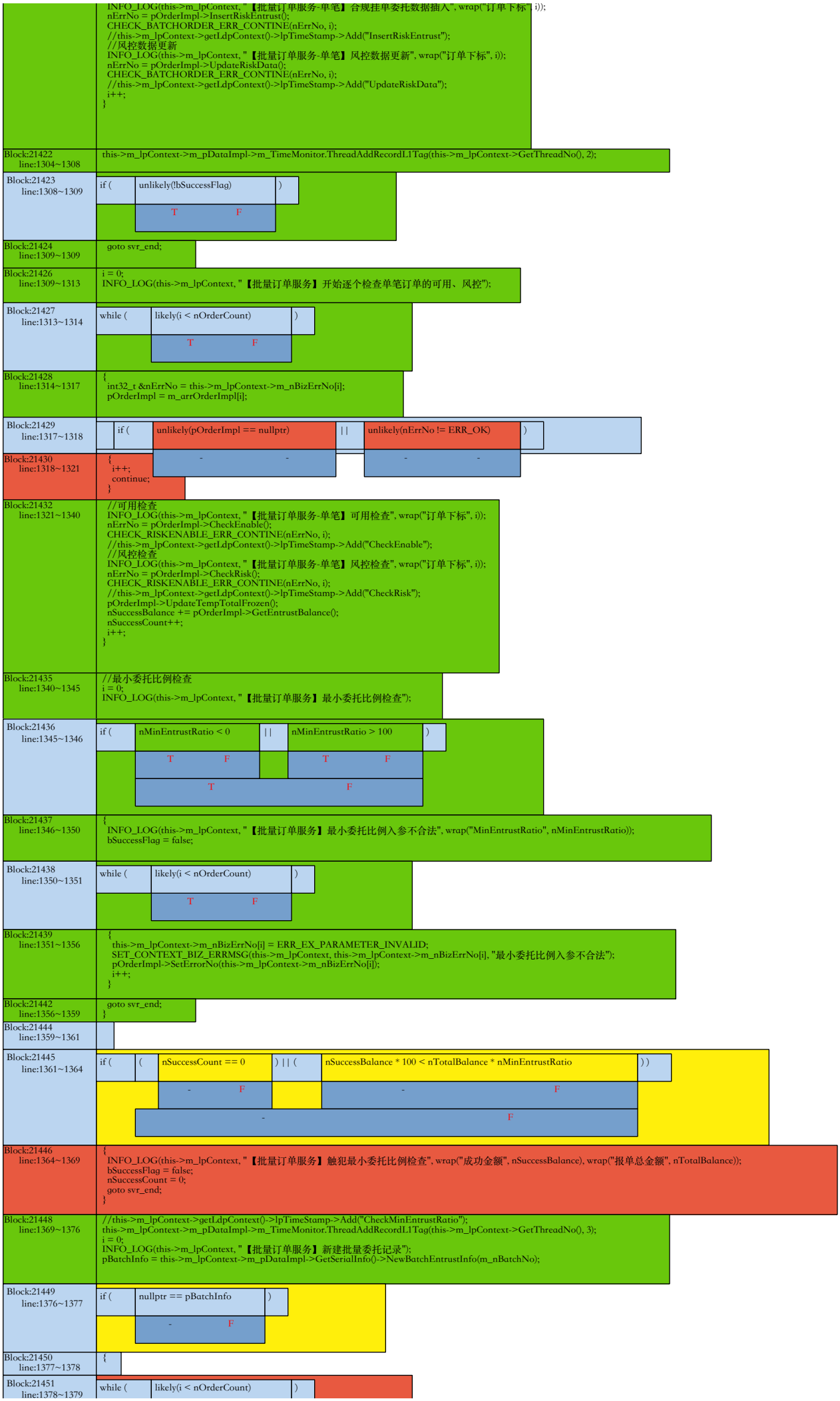
```
if (CNST_TO_UNION_RISK_SWITCH_ON == this->m_lpContext->GetOpenToUniRisk())
```

Block:21417 line:1271~1276

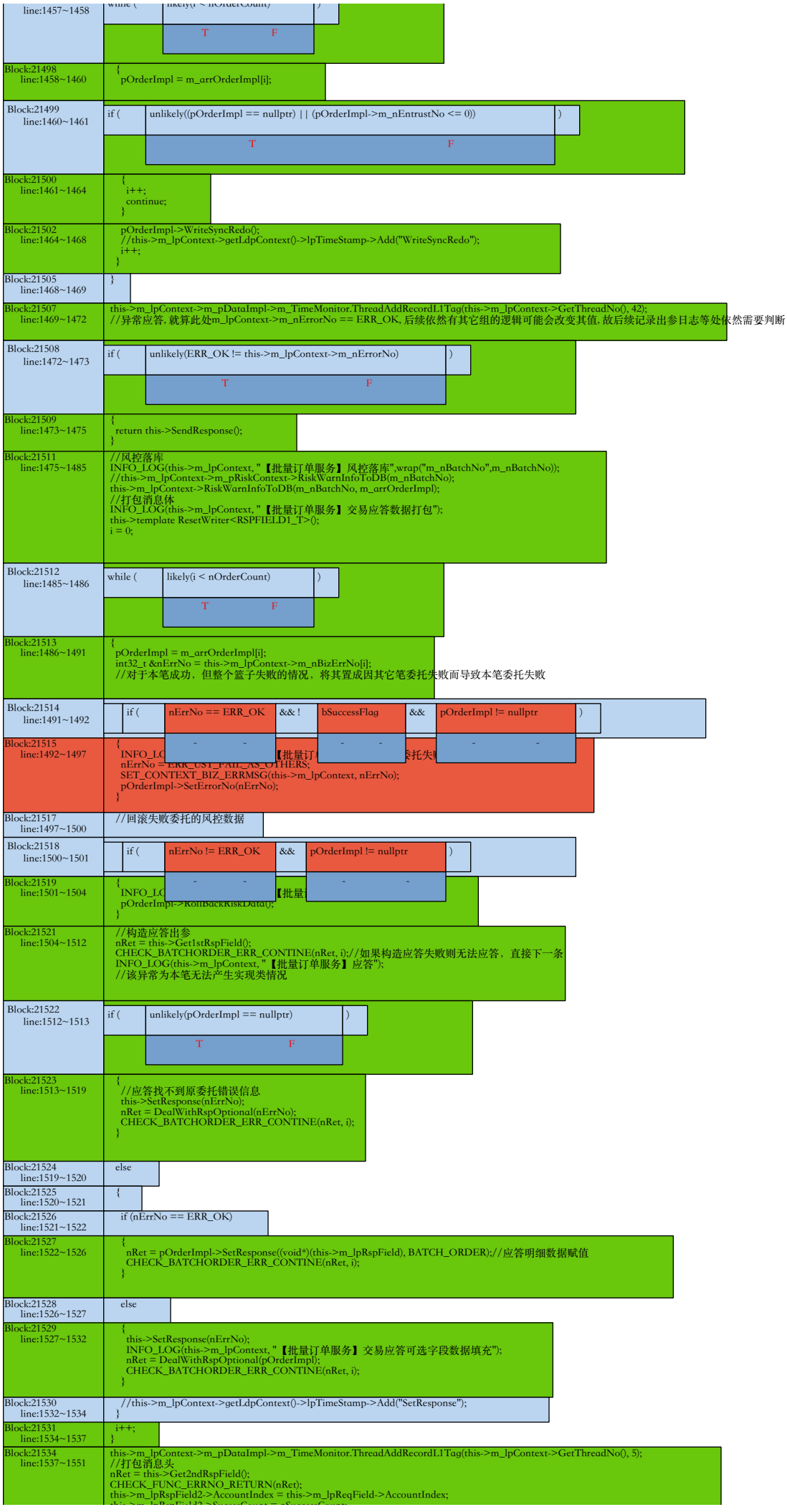
```
 {
  INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】联合风控推送", wrap("订单下标", i));
  nErrNo = pOrderImpl->SendToUnionRisk();
  CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
  //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("SendToUnionRisk");
 }
```

Block:21419 line:1276~1304

```
 //业务校验
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】业务校验", wrap("订单下标", i));
 nErrNo = pOrderImpl->CheckBusiness();
 CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("CheckBusiness");
 //业务数据计算
 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】业务数据计算", wrap("订单下标", i));
 nErrNo = pOrderImpl->CalcBusinData();
 CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
 //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("CalcBusinData");
 nTotalBalance += pOrderImpl->GetEntrustBalance();
 //合规挂单委托数据插入
```

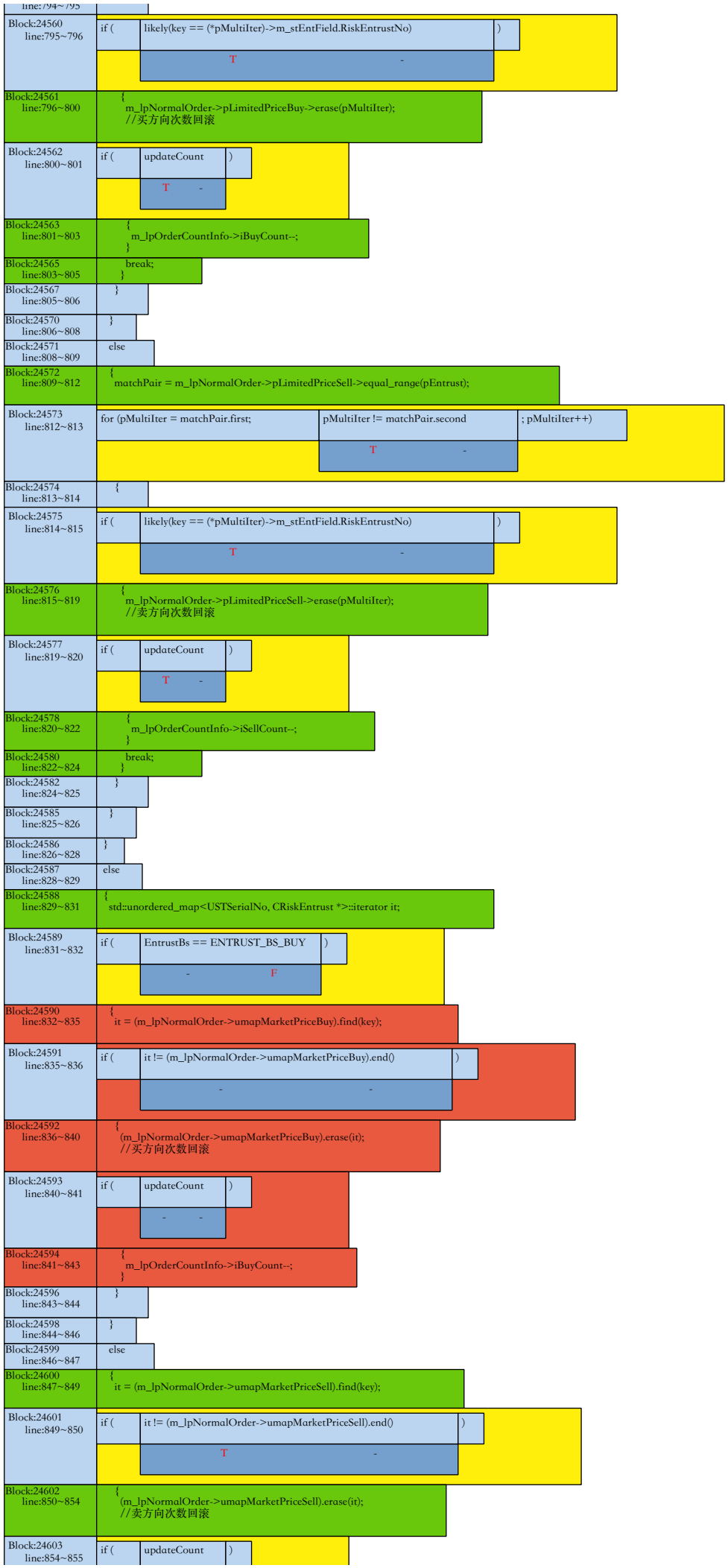

INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】合规挂单委托数据插入", wrap("订单下标", i));
nErrNo = pOrderImpl->InsertRiskEntrust();
CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("InsertRiskEntrust");
//风控数据更新
INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】风控数据更新", wrap("订单下标", i));
nErrNo = pOrderImpl->UpdateRiskData();
CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("UpdateRiskData");
i++;
}
Block:21422 line:1304~1308
this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 2);
Block:21423 line:1308~1309
if (
unlikely(!bSuccessFlag)
)
T
F
Block:21424 line:1309~1309
goto svr_end;
Block:21426 line:1309~1313
i = 0;
INFO_LOG(this->m_lpContext, "【批量订单服务】开始逐个检查单笔订单的可用、风控");
Block:21427 line:1313~1314
while (
likely(i < nOrderCount)
)
T
F
Block:21428 line:1314~1317
{
int32_t &nErrNo = this->m_lpContext->m_nBizErrNo[i];
pOrderImpl = m_arrOrderImpl[i];
Block:21429 line:1317~1318
if (
unlikely(pOrderImpl == nullptr)
||
unlikely(nErrNo != ERR_OK)
)
-
-
-
-
Block:21430 line:1318~1321
{
i++;
continue;
}
Block:21432 line:1321~1340
//可用检查
INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】可用检查", wrap("订单下标", i));
nErrNo = pOrderImpl->CheckEnable();
CHECK_RISKENABLE_ERR_CONTINE(nErrNo, i);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("CheckEnable");
//风控检查
INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】风控检查", wrap("订单下标", i));
nErrNo = pOrderImpl->CheckRisk();
CHECK_RISKENABLE_ERR_CONTINE(nErrNo, i);
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("CheckRisk");
pOrderImpl->UpdateTempTotalFrozen();
nSuccessBalance += pOrderImpl->GetEntrustBalance();
nSuccessCount++;
i++;
}
Block:21435 line:1340~1345
//最小委托比例检查
i = 0;
INFO_LOG(this->m_lpContext, "【批量订单服务】最小委托比例检查");
Block:21436 line:1345~1346
if (
nMinEntrustRatio < 0
||
nMinEntrustRatio > 100
)
T
F
T
F
T
F
Block:21437 line:1346~1350
{
INFO_LOG(this->m_lpContext, "【批量订单服务】最小委托比例入参不合法", wrap("MinEntrustRatio", nMinEntrustRatio));
bSuccessFlag = false;
Block:21438 line:1350~1351
while (
likely(i < nOrderCount)
)
T
F
Block:21439 line:1351~1356
{
this->m_lpContext->m_nBizErrNo[i] = ERR_EX_PARAMETER_INVALID;
SET_CONTEXT_BIZ_ERRMSG(this->m_lpContext, this->m_lpContext->m_nBizErrNo[i], "最小委托比例入参不合法");
pOrderImpl->SetErrorNo(this->m_lpContext->m_nBizErrNo[i]);
i++;
}
Block:21442 line:1356~1359
goto svr_end;
}
Block:21444 line:1359~1361
Block:21445 line:1361~1364
if (
(
nSuccessCount == 0
) || (
nSuccessBalance * 100 < nTotalBalance * nMinEntrustRatio
))
-
F
-
F
-
F
Block:21446 line:1364~1369
{
INFO_LOG(this->m_lpContext, "【批量订单服务】触犯最小委托比例检查", wrap("成功金额", nSuccessBalance), wrap("报单总金额", nTotalBalance));
bSuccessFlag = false;
nSuccessCount = 0;
goto svr_end;
}
Block:21448 line:1369~1376
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("CheckMinEntrustRatio");
this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 3);
i = 0;
INFO_LOG(this->m_lpContext, "【批量订单服务】新建批量委托记录");
pBatchInfo = this->m_lpContext->m_pDataImpl->GetSerialInfo()->NewBatchEntrustInfo(m_nBatchNo);
Block:21449 line:1376~1377
if (
nullptr == pBatchInfo
)
-
F
Block:21450 line:1377~1378
{
Block:21451 line:1378~1379
while (
likely(i < nOrderCount)
)

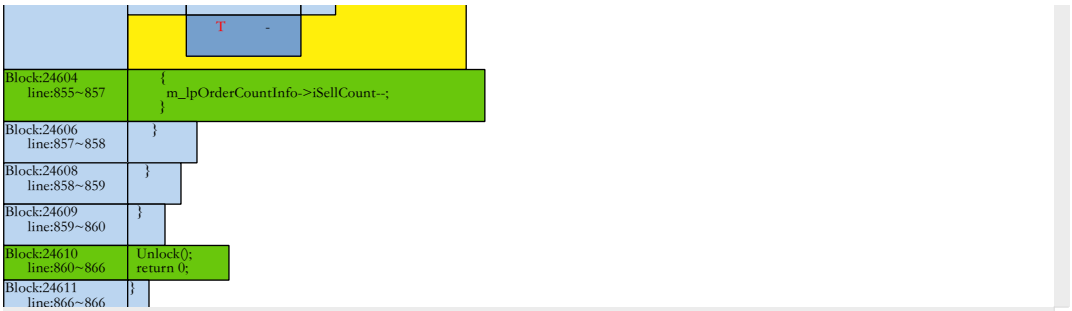
	- -
Block:21452 line:1379~1385	{ pOrderImpl = m_arrOrderImpl[i]; this->m_lpContext->m_nBizErrNo[i] = ERR_USER_ADD_TABLERECORD_FAIL; SET_CONTEXT_BIZ_ERRMSG(this->m_lpContext, this->m_nBizErrNo[i], "新建批量委托记录失败"); pOrderImpl->SetErrorNo(this->m_lpContext->m_nBizErrNo[i]); i++; }
Block:21455 line:1385~1387	goto svr_end; }
Block:21457 line:1387~1392	//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("NewBatchEntrustInfo"); INFO_LOG(this->m_lpContext, "【批量订单服务】交易数据更新、报盘",wrap("nOrderCount", nOrderCount)); i = 0;
Block:21458 line:1392~1393	while (likely(i < nOrderCount)) T F
Block:21459 line:1393~1395	{ int32_t &nErrNo = this->m_lpContext->m_nBizErrNo[i];
Block:21460 line:1395~1396	if (unlikely(nErrNo != ERR_OK)) - F
Block:21461 line:1396~1399	{ i++; continue; }
Block:21463 line:1399~1410	pOrderImpl = m_arrOrderImpl[i]; //交易数据更新 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】交易数据更新", wrap("订单下标", i)); this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(this->m_lpContext->GetThreadNo(), 85); nErrNo = pOrderImpl->UpdateBusinData(m_nBatchNo); this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL2Tag(this->m_lpContext->GetThreadNo(), 86); CHECK_BATCHORDER_ERR_CONTINE(nErrNo, i); //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("UpdateBusinData");
Block:21464 line:1410~1411	if (CNST_TO_UNION_RISK_SWITCH_OFF == this->m_lpContext->GetOpenToUniRisk())
Block:21465 line:1411~1416	{ //报盘 INFO_LOG(this->m_lpContext, "【批量订单服务-单笔】报盘", wrap("订单下标", i)); pOrderImpl->OfferOrder(); //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("OfferOrder"); }
Block:21467 line:1416~1418	i++; }
Block:21470 line:1418~1424	this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 4); //写落库redo文件 INFO_LOG(this->m_lpContext, "【批量订单服务】写落库redo文件",wrap("m_iTodbWriteRedo", ((CBizContext*)this->m_lpContext)->m_iTodbWriteRedo));
Block:21471 line:1424~1425	if (CNST_ASYNWRITEREDO_YES == ((CBizContext*)this->m_lpContext)->m_iTodbWriteRedo)
Block:21472 line:1425~1427	{ i = 0;
Block:21473 line:1427~1428	while (likely(i < nOrderCount)) T F
Block:21474 line:1428~1429	{
Block:21475 line:1429~1430	if (unlikely(this->m_lpContext->m_nBizErrNo[i] != ERR_OK)) - F
Block:21476 line:1430~1433	{ i++; continue; }
Block:21478 line:1433~1439	pOrderImpl = m_arrOrderImpl[i]; pOrderImpl->WriteBusinToDBRedo(); //this->m_lpContext->getLdpContext()->lpTimeStamp->Add("WriteBusinToDBRedo"); i++; }
Block:21481 line:1439~1440	}
Block:21483 line:1440~1442	this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 41);
Block:21484 line:1442~1442	svr_end:
Block:21485 line:1442~1444	//置委托处理结束
Block:21486 line:1444~1445	for (i = 0; i < nOrderCount; i++)
Block:21487 line:1445~1447	{ pOrderImpl = m_arrOrderImpl[i];
Block:21488 line:1447~1448	if (likely((pOrderImpl != nullptr) && (pOrderImpl->m_nEntrustNo > 0))) T F
Block:21489 line:1448~1450	{ this->m_lpContext->m_pDataImpl->GetSerialInfo()->SetEntrustDone(pOrderImpl->m_nEntrustNo); }
Block:21491 line:1450~1451	}
Block:21494 line:1451~1454	//写主备redo文件 INFO_LOG(this->m_lpContext, "【批量订单服务】写主备redo文件", wrap("m_iSyncWriteRedo", ((CBizContext*)this->m_lpContext)->m_iSyncWriteRedo));
Block:21495 line:1454~1455	if (CNST_WRITEREDO_YES == ((CBizContext*)this->m_lpContext)->m_iSyncWriteRedo)
Block:21496 line:1455~1457	{ i = 0;
Block:21497	while (likely(i < nOrderCount))


line:1457~1458
while (likely(i < nOrderCount))
T F
Block:21498
line:1458~1460
{
pOrderImpl = m_arrOrderImpl[i];
Block:21499
line:1460~1461
if (unlikely((pOrderImpl == nullptr) || (pOrderImpl->m_nEntrustNo <= 0)))
T F
Block:21500
line:1461~1464
{
i++;
continue;
}
Block:21502
line:1464~1468
pOrderImpl->WriteSyncRedo();
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("WriteSyncRedo");
i++;
}
Block:21505
line:1468~1469
}
Block:21507
line:1469~1472
this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 42);
//异常应答, 就算此处m_lpContext->m_nErrorNo == ERR_OK, 后续依然有其它组的逻辑可能会改变其值, 故后续记录出参日志等处依然需要判断
Block:21508
line:1472~1473
if (unlikely(ERR_OK != this->m_lpContext->m_nErrorNo))
T F
Block:21509
line:1473~1475
{
return this->SendResponse();
}
Block:21511
line:1475~1485
//风控落库
INFO_LOG(this->m_lpContext, "【批量订单服务】风控落库",wrap("m_nBatchNo",m_nBatchNo));
//this->m_lpContext->m_pRiskContext->RiskWarnInfoToDB(m_nBatchNo);
this->m_lpContext->RiskWarnInfoToDB(m_nBatchNo, m_arrOrderImpl);
//打包消息体
INFO_LOG(this->m_lpContext, "【批量订单服务】交易应答数据打包");
this->template ResetWriter<RSPFIELD1_T>();
i = 0;
Block:21512
line:1485~1486
while (likely(i < nOrderCount))
T F
Block:21513
line:1486~1491
{
pOrderImpl = m_arrOrderImpl[i];
int32_t &nErrNo = this->m_lpContext->m_nBizErrNo[i];
//对于本笔成功，但整个篮子失败的情况，将其置成因其它笔委托失败而导致本笔委托失败
Block:21514
line:1491~1492
if (nErrNo == ERR_OK && ! bSuccessFlag && pOrderImpl != nullptr)
Block:21515
line:1492~1497
{
INFO_LO 【批量订
nErrNo = ERR_UST_FAIL_AS_OTHERS;
SET_CONTEXT_BIZ_ERRMSG(this->m_lpContext, nErrNo);
pOrderImpl->SetErrorNo(nErrNo);
}
Block:21517
line:1497~1500
//回滚失败委托的风控数据
Block:21518
line:1500~1501
if (nErrNo != ERR_OK && pOrderImpl != nullptr)
Block:21519
line:1501~1504
{
INFO_LO 【批量
pOrderImpl->RollBackRiskData();
}
Block:21521
line:1504~1512
//构造应答出参
nRet = this->Get1stRspField();
CHECK_BATCHORDER_ERR_CONTINE(nRet, i);//如果构造应答失败则无法应答，直接下一条
INFO_LOG(this->m_lpContext, "【批量订单服务】应答");
//该异常为本笔无法产生实现类情况
Block:21522
line:1512~1513
if (unlikely(pOrderImpl == nullptr))
T F
Block:21523
line:1513~1519
{
//应答找不到原委托错误信息
this->SetResponse(nErrNo);
nRet = DealWithRspOptional(nErrNo);
CHECK_BATCHORDER_ERR_CONTINE(nRet, i);
}
Block:21524
line:1519~1520
else
Block:21525
line:1520~1521
{
Block:21526
line:1521~1522
if (nErrNo == ERR_OK)
Block:21527
line:1522~1526
{
nRet = pOrderImpl->SetResponse((void*)(this->m_lpRspField), BATCH_ORDER);//应答明细数据赋值
CHECK_BATCHORDER_ERR_CONTINE(nRet, i);
}
Block:21528
line:1526~1527
else
Block:21529
line:1527~1532
{
this->SetResponse(nErrNo);
INFO_LOG(this->m_lpContext, "【批量订单服务】交易应答可选字段数据填充");
nRet = DealWithRspOptional(pOrderImpl);
CHECK_BATCHORDER_ERR_CONTINE(nRet, i);
}
Block:21530
line:1532~1534
//this->m_lpContext->getLdpContext()->lpTimeStamp->Add("SetResponse");
}
Block:21531
line:1534~1537
i++;
}
Block:21534
line:1537~1551
this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 5);
//打包消息头
nRet = this->Get2ndRspField();
CHECK_FUNC_ERRNO_RETURN(nRet);
this->m_lpRspField2->AccountIndex = this->m_lpReqField->AccountIndex;

```
this->m_lpRspField2->SucessCount = nSuccessCount;
this->m_lpRspField2->BatchNo = m_nBatchNo;
this->m_lpRspField2->ProductIndex = this->m_lpReqField->ProductIndex;
this->m_lpRspField2->ProjectIndex = this->m_lpReqField->ProjectIndex;
this->m_lpRspField2->StrategyIndex = this->m_lpReqField->StrategyIndex;

Block:21535 line:1551~1552
if (ERR_OK == this->m_lpContext->m_nErrorNo)

Block:21536 line:1552~1555
{
 this->m_lpRspField2->SessionID = this->m_pSession->SessionIndex;
}

Block:21537 line:1555~1556
else

Block:21538 line:1556~1558
{
 this->m_lpRspField2->SessionID = 0;
}

Block:21539 line:1558~1561
//风控触警信息打包

Block:21540 line:1561~1562
if (this->m_lpContext->m_pRiskContext->RiskCount() > 0)

Block:21541 line:1562~1563
{

Block:21542 line:1563~1564
 for (int i = 0; i < this->m_lpContext->m_pRiskContext->RiskCount(); i++)

Block:21543 line:1564~1570
 {
  nRet = this->GetRiskInfoRspField(2);
  CHECK_FUNC_ERRNO_RETURN(nRet);
  //风控触警信息赋值
  this->m_lpContext->m_pRiskContext->RiskInfoSet(this->m_lpRiskInfo, i);
 }

Block:21546 line:1570~1572
}

Block:21547 line:1572~1573
else

Block:21548 line:1573~1575
{
 this->template WriteNewDataSet<RISKINFO_T>(2);
}

Block:21549 line:1575~1576

Block:21550 line:1576~1577
if ( | unlikely(this->m_lpContext->GetLogLevel() <= LEVEL_INFO) | )
       - F

Block:21551 line:1577~1578
{

Block:21552 line:1578~1579
 if (ERR_OK == this->m_lpContext->m_nErrorNo)

Block:21553 line:1579~1586
 {
  ILdpMsgReader* ldpMsgReader = (ILdpMsgReader*)this->m_lpContext->GetGlobalData()->GetLdpHost()->NewObject(ObjectID::LdpMsgReader);
  CLdpMsgReaderHelperOne<RSPFIELD1_T> tempRspReader(this->m_lpContext, this->m_lpContext->GetMsgWriter()->GetBuffer(), this->m_lpContext->GetMsgWriter()->GetLength(), false, ldpMsgReac
  memcpy(ldpMsgReader->GetHead(), this->m_lpContext->GetMsgWriter()->GetHead(), sizeof(LdpMsg::Head));
  tempRspReader.LogMsg("应答消息体数据————————————————————————");
  this->m_lpContext->GetGlobalData()->GetLdpHost()->FreeObject(ldpMsgReader);
 }

Block:21554 line:1586~1587
 else

Block:21555 line:1587~1589
 {
  INFO_LOG(this->m_lpContext, wrap("功能号", this->m_lpContext->GetMsgWriter()->GetHead()->FunctionID), wrap("应答-错误号", this->m_lpContext->m_nErrorNo), "——————————————
 }

Block:21556 line:1589~1590
}

Block:21558 line:1590~1595
this->SendResponse();
// this->m_lpContext->getLdpContext()->lpTimeStamp->Add("SendResponse");
this->m_lpContext->m_pDataImpl->m_TimeMonitor.ThreadAddRecordL1Tag(this->m_lpContext->GetThreadNo(), 6);
return ERR_OK;

Block:21559 line:1595~1595
}
```

```
Block:24547 line:768~769
int32_t CSelfDealCheckMgr::Remove(CRiskEntrust*pEntrust, bool updateCount /*= true*/)

Block:24549 line:769~770
{

Block:24550 line:770~771
if ( | unlikely(pEntrust == nullptr) | )
       - F

Block:24551 line:771~773
{
 return 0;
}

Block:24553 line:773~787
Lock(); //上锁,函数执行途中禁止直接return
std::pair<std::multiset<CRiskEntrust*, PFCMP>::iterator, std::multiset<CRiskEntrust*, PFCMP>::iterator> matchPair;
std::multiset<CRiskEntrust*, PFCMP>::iterator pMultiIter;
USTSerialNo key = pEntrust->m_stEntField.RiskEntrustNo;
USTOrderCommand OrderCommand = pEntrust->m_stEntField.OrderCommand;
USTType EntrustBs = pEntrust->m_stEntField.EntrustBs;
//普通单
//DEBUG_LOG_TRACE("报单指令 %d\n", OrderCommand);

Block:24554 line:787~788
if ( | likely(OrderCommand == 1) | )
       T F

Block:24555 line:788~789
 {

Block:24556 line:789~790
if ( | EntrustBs == ENTRUST_BS_BUY | )
       T F

Block:24557 line:790~793
 {
  matchPair = m_lpNormalOrder->pLimitedPriceBuy->equal_range(pEntrust);
  //精确找到要删除的委托

Block:24558 line:793~794
for (pMultiIter = matchPair.first; | pMultiIter != matchPair.second | ; pMultiIter++)
                                     T -

Block:24559
  {
```


Block:24560 line:795~796
if (likely(key == (*pMultiIter)->m_stEntField.RiskEntrustNo))
T -
Block:24561 line:796~800
{
m_lpNormalOrder->pLimitedPriceBuy->erase(pMultiIter);
//买方向次数回滚
Block:24562 line:800~801
if (updateCount)
T -
Block:24563 line:801~803
{
m_lpOrderCountInfo->iBuyCount--;
}
Block:24565 line:803~805
break;
}
Block:24567 line:805~806
}
Block:24570 line:806~808
}
Block:24571 line:808~809
else
Block:24572 line:809~812
{
matchPair = m_lpNormalOrder->pLimitedPriceSell->equal_range(pEntrust);
Block:24573 line:812~813
for (pMultiIter = matchPair.first; pMultiIter != matchPair.second ; pMultiIter++)
T -
Block:24574 line:813~814
{
Block:24575 line:814~815
if (likely(key == (*pMultiIter)->m_stEntField.RiskEntrustNo))
T -
Block:24576 line:815~819
{
m_lpNormalOrder->pLimitedPriceSell->erase(pMultiIter);
//卖方向次数回滚
Block:24577 line:819~820
if (updateCount)
T -
Block:24578 line:820~822
{
m_lpOrderCountInfo->iSellCount--;
}
Block:24580 line:822~824
break;
}
Block:24582 line:824~825
}
Block:24585 line:825~826
}
Block:24586 line:826~828
}
Block:24587 line:828~829
else
Block:24588 line:829~831
{
std::unordered_map<USTSerialNo, CRiskEntrust *>::iterator it;
Block:24589 line:831~832
if (EntrustBs == ENTRUST_BS_BUY)
- F
Block:24590 line:832~835
{
it = (m_lpNormalOrder->umapMarketPriceBuy).find(key);
Block:24591 line:835~836
if (it != (m_lpNormalOrder->umapMarketPriceBuy).end())
- -
Block:24592 line:836~840
{
(m_lpNormalOrder->umapMarketPriceBuy).erase(it);
//买方向次数回滚
Block:24593 line:840~841
if (updateCount)
- -
Block:24594 line:841~843
{
m_lpOrderCountInfo->iBuyCount--;
}
Block:24596 line:843~844
}
Block:24598 line:844~846
}
Block:24599 line:846~847
else
Block:24600 line:847~849
{
it = (m_lpNormalOrder->umapMarketPriceSell).find(key);
Block:24601 line:849~850
if (it != (m_lpNormalOrder->umapMarketPriceSell).end())
T -
Block:24602 line:850~854
{
(m_lpNormalOrder->umapMarketPriceSell).erase(it);
//卖方向次数回滚
Block:24603 line:854~855
if (updateCount)



N_2544 : ['OrderCommand']:

N_3061 : ['Direction']:

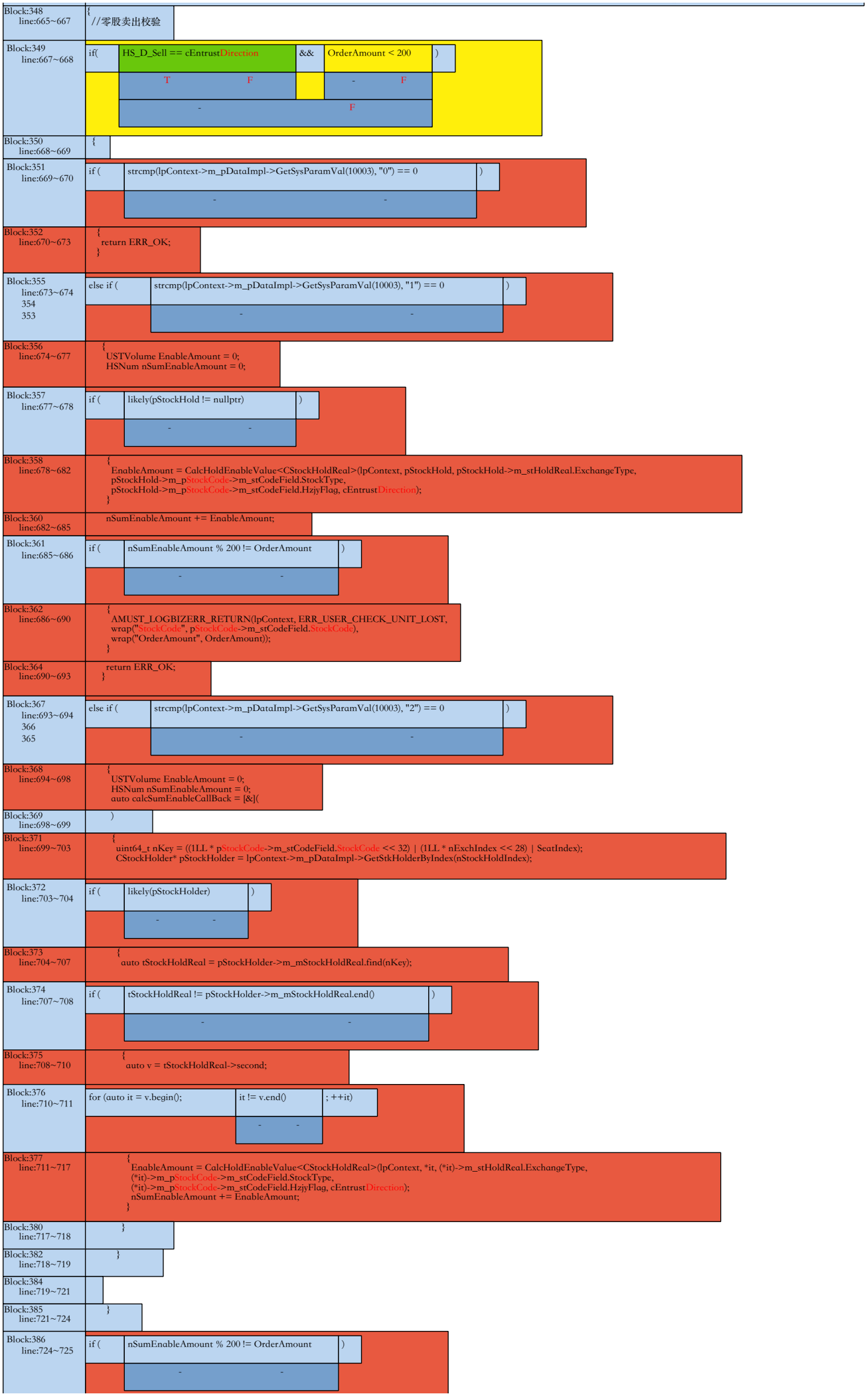
```
Block:34893 line:1346~1347
int32_t CETFOrderInsertFlow::CheckSingleFundEnable(USTType EstimateType, USTBalance CheckBalance)
Block:34895 line:1347~1352
{
 USTBalance nEnableValue = 0;
 nEnableValue = CalcCashEnableValue(lpContext, m_stPublicContext.pAsset, m_stPublicContext.nExchIndex, m_stPublicContext.nSecurityType,
 m_stPublicContext.cEntrustDirection, CNST_BUSINPROTYPE_ENTRUST, EstimateType);
 CHECK_BIZ_FUNC_ERR_RETURN;
Block:34896 line:1352~1353
if ( nEnableValue < CheckBalance )
 - F
Block:34897 line:1353~1358
 {
  ///报错返回
  AMUST_LOGBIZERR_ASSIGN(lpContext, m_nErrorNo, ERR_FUND_CANUSEFUND_NOTENOUGH, wrap("nEnableValue", nEnableValue / 10000),
  wrap("CheckBalance", CheckBalance));
  return m_nErrorNo;
 }
Block:34899 line:1358~1362
 //[资金更新] 现金差额
 SCashEnableFactorField* pCashEnableFactor = UpdateCashEnableValueAndCashFactor(lpContext, m_stPublicContext.pAsset, m_stPublicContext.nExchIndex, m_stPublicContext.nSecurityType,
 m_stPublicContext.cEntrustDirection, CNST_BUSINPROTYPE_ENTRUST, EstimateType, CheckBalance);
Block:34900 line:1362~1363
if ( unlikely(pCashEnableFactor == nullptr) )
 - F
Block:34901 line:1363~1366
 {
  m_nErrorNo = ERR_ETF_NOT_FIND_CASH_ENABLE_CFG;
 }
Block:34902 line:1366~1367
 else
Block:34903 line:1367~1369
 {
  m_stContext.vCashEnableFactorField.push_back(pCashEnableFactor);
 }
Block:34904 line:1369~1372
 return m_nErrorNo;
Block:34905 line:1372~1372
}
```

N_22 : ['StockCode', 'OrderCommand']:

```
Block:398 line:750~751
int32_t CheckOrderAmountLimit(CStockBizContext* lpContext, USTVolume OrderAmount, USTNumID OrderCommand, CStockCode* pStockCode)
Block:400 line:751~753
{
 //限价申报
Block:401 line:753~754
if( CNST_UST_ORDERCOMMAND_LIMIT == OrderCommand )
 T F
Block:402 line:754~755
 {
Block:403 line:755~758
 if ( unlikely((pStockCode->m_stCodeField.LimitHighAmount > 0 && OrderAmount > pStockCode->m_stCodeField.LimitHighAmount) || (pStockCode->m_stCodeField.LimitLowAmount >= 0 && (
  T
Block:404 line:758~766
  {
   ///报错返回，超过数量上下限范围
   AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_ENTRUSTAMOUNT_OVERFLOW,
   wrap("OrderCommand", OrderCommand),
   wrap("OrderAmount", OrderAmount),
   wrap("LimitHighAmount", pStockCode->m_stCodeField.LimitHighAmount),
   wrap("LimitLowAmount", pStockCode->m_stCodeField.LimitLowAmount));
  }
Block:406 line:766~768
 }
Block:407 line:768~769
 else
Block:408 line:769~770
 {
Block:409 line:770~773
 if ( unlikely((pStockCode->m_stCodeField.MktHighAmount > 0 && OrderAmount > pStockCode->m_stCodeField.MktHighAmount) || (pStockCode->m_stCodeField.MktLowAmount >= 0 && Orde
  T
Block:410 line:773~781
  {
   ///报错返回，超过数量上下限范围
   AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_ENTRUSTAMOUNT_OVERFLOW,
   wrap("OrderCommand", OrderCommand),
   wrap("OrderAmount", OrderAmount),
   wrap("MktHighAmount", pStockCode->m_stCodeField.MktHighAmount),
   wrap("MktLowAmount", pStockCode->m_stCodeField.MktLowAmount));
  }
Block:412 line:781~782
 }
Block:413 line:782~785
 return 0;
Block:414 line:785~785
}
```

N_20 : ['StockCode', 'Direction', 'OrderCommand']:

```
Block:346 line:663~665
int32_t CheckSTBoardOrderAmountAndUnit(CStockBizContext *lpContext, USTVolume OrderAmount, USTExchangeIndex nExchIndex, USTNumID nStockHoldIndex,
USTNumID OrderCommand, USTEntrustDirection cEntrustDirection, CStockCode* pStockCode, USTUNumID32 SeatIndex, CStockHoldReal* pStockHold)
```


Block:348 line:665~667
{
//零股卖出校验
Block:349 line:667~668
if(HS_D_Sell == cEntrustDirection && OrderAmount < 200)
T F
- F
- F
Block:350 line:668~669
{
Block:351 line:669~670
if (strcmp(lpContext->m_pDataImpl->GetSysParamVal(10003), "0") == 0)
- -
Block:352 line:670~673
{
return ERR_OK;
}
Block:355 line:673~674 354 353
else if (strcmp(lpContext->m_pDataImpl->GetSysParamVal(10003), "1") == 0)
- -
Block:356 line:674~677
{
USTVolume EnableAmount = 0;
HSNum nSumEnableAmount = 0;
Block:357 line:677~678
if (likely(pStockHold != nullptr))
- -
Block:358 line:678~682
{
EnableAmount = CalcHoldEnableValue<CStockHoldReal>(lpContext, pStockHold, pStockHold->m_stHoldReal.ExchangeType,
pStockHold->m_pStockCode->m_stCodeField.StockType,
pStockHold->m_pStockCode->m_stCodeField.HzjyFlag, cEntrustDirection);
}
Block:360 line:682~685
nSumEnableAmount += EnableAmount;
Block:361 line:685~686
if (nSumEnableAmount % 200 != OrderAmount)
- -
Block:362 line:686~690
{
AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_CHECK_UNIT_LOST,
wrap("StockCode", pStockCode->m_stCodeField.StockCode),
wrap("OrderAmount", OrderAmount));
}
Block:364 line:690~693
return ERR_OK;
}
Block:367 line:693~694 366 365
else if (strcmp(lpContext->m_pDataImpl->GetSysParamVal(10003), "2") == 0)
- -
Block:368 line:694~698
{
USTVolume EnableAmount = 0;
HSNum nSumEnableAmount = 0;
auto calcSumEnableCallBack = [&](
Block:369 line:698~699
)
Block:371 line:699~703
{
uint64_t nKey = ((1LL * pStockCode->m_stCodeField.StockCode << 32) | (1LL * nExchIndex << 28) | SeatIndex);
CStockHolder* pStockHolder = lpContext->m_pDataImpl->GetStkHolderByIndex(nStockHoldIndex);
Block:372 line:703~704
if (likely(pStockHolder))
- -
Block:373 line:704~707
{
auto tStockHoldReal = pStockHolder->m_mStockHoldReal.find(nKey);
Block:374 line:707~708
if (tStockHoldReal != pStockHolder->m_mStockHoldReal.end())
- -
Block:375 line:708~710
{
auto v = tStockHoldReal->second;
Block:376 line:710~711
for (auto it = v.begin(); it != v.end() ; ++it)
- -
Block:377 line:711~717
{
EnableAmount = CalcHoldEnableValue<CStockHoldReal>(lpContext, *it, (*it)->m_stHoldReal.ExchangeType,
(*it)->m_pStockCode->m_stCodeField.StockType,
(*it)->m_pStockCode->m_stCodeField.HzjyFlag, cEntrustDirection);
nSumEnableAmount += EnableAmount;
}
Block:380 line:717~718
}
Block:382 line:718~719
}
Block:384 line:719~721
Block:385 line:721~724
}
Block:386 line:724~725
if (nSumEnableAmount % 200 != OrderAmount)
- -

Block:387 line:725~729
```
{
 AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_CHECK_UNIT_LOST,
 wrap("StockCode:", pStockCode->m_stCodeField.StockCode),
 wrap("OrderAmount", OrderAmount));
}
```

Block:389 line:729~732
```
return ERR_OK;
}
```

Block:390 line:732~733
```
else
```

Block:391 line:733~740
```
{
 AMUST_LOGBIZERR_RETURN(lpContext, ERR_USER_CHECK_UNIT_LOST,
 wrap("系统参数不在值域范围内， StockCode", pStockCode->m_stCodeField.StockCode),
 wrap("OrderAmount", OrderAmount),
 wrap("系统参数:"),
 wrap(lpContext->m_pDataImpl->GetSysParamVal(10003))
 );
}
```

Block:394 line:740~743
```
// Note: 华泰一期项目不控制零股卖出
//CStockHoldReal* pHoldReal = NULL;
}
```

Block:396 line:743~746
```
return CheckOrderAmountLimit(lpContext, OrderAmount, OrderCommand, pStockCode);
```

Block:397 line:746~746
```
}
```

N_2786 :
['OrderRef', 'ExchangeID', 'StockCode', 'Direction', 'OrderPrice', 'OrderVolume', 'OrderCommand']:

Block:29388 line:2432~2433
```
int32_t CNewStockOrderInsertFlow::SetResponse(void* pRspField, ORDER_SVERICE_TYPE svrType)
```

Block:29390 line:2433~2435
```
{
// 应答写入
```

Block:29391 line:2435~2436
```
if ( BATCH_ORDER == svrType )
```
T -

Block:29392 line:2436~2451
```
{
 CHSInternalRspOrderInsertField* rsp = static_cast<CHSInternalRspOrderInsertField*>(pRspField);
 rsp->ErrorNo = m_nErrorNo;
 itoa(m_stStockPublicContext.EntrustReference, rsp->OrderRef);
 itoa(m_stStockPublicContext.nStockCode, rsp->StockCode, 10, 6);
 std::strncpy(rsp->ExchangeID, GetExchangeType(m_stStockPublicContext.nExchIndex), sizeof(rsp->ExchangeID));
 rsp->ExchangeID[sizeof(rsp->ExchangeID) - 1] = '\0';
 rsp->Direction = m_stStockPublicContext.cEntrustDirection;
 rsp->OrderPrice = ConvertDouble(m_stStockPublicContext.nOrderPrice, PRICE_MULTIPLIER, PRICE_DECIMAL_DIGIT);
 rsp->OrderVolume = m_stStockPublicContext.nOrderVolume;
 rsp->OrderCommand = m_stStockPublicContext.nOrderCommand;
 rsp->ExternReportNo = m_stStockPublicContext.nExternReportNo;
 rsp->RiskOrderNo = m_stStockPublicContext.nRiskEntrustNo;
```

Block:29393 line:2451~2452
```
if ( rsp->ErrorNo == ERR_OK )
```
T -

Block:29394 line:2452~2458
```
{
 //nBatchNo = m_stStockPublicContext->pEntrust->m_stEntField.BatchNo;
 itoa(m_stStockPublicContext.pEntrust->m_stEntField.EntrustNo, rsp->OrderSysID);
 rsp->OrderStatus = m_stStockPublicContext.pEntrust->m_stEntField.EntrustStatus;
 rsp->InsertTime = m_stStockPublicContext.pEntrust->m_stEntField.CurrTime;
}
```

Block:29395 line:2458~2459
```
else
```

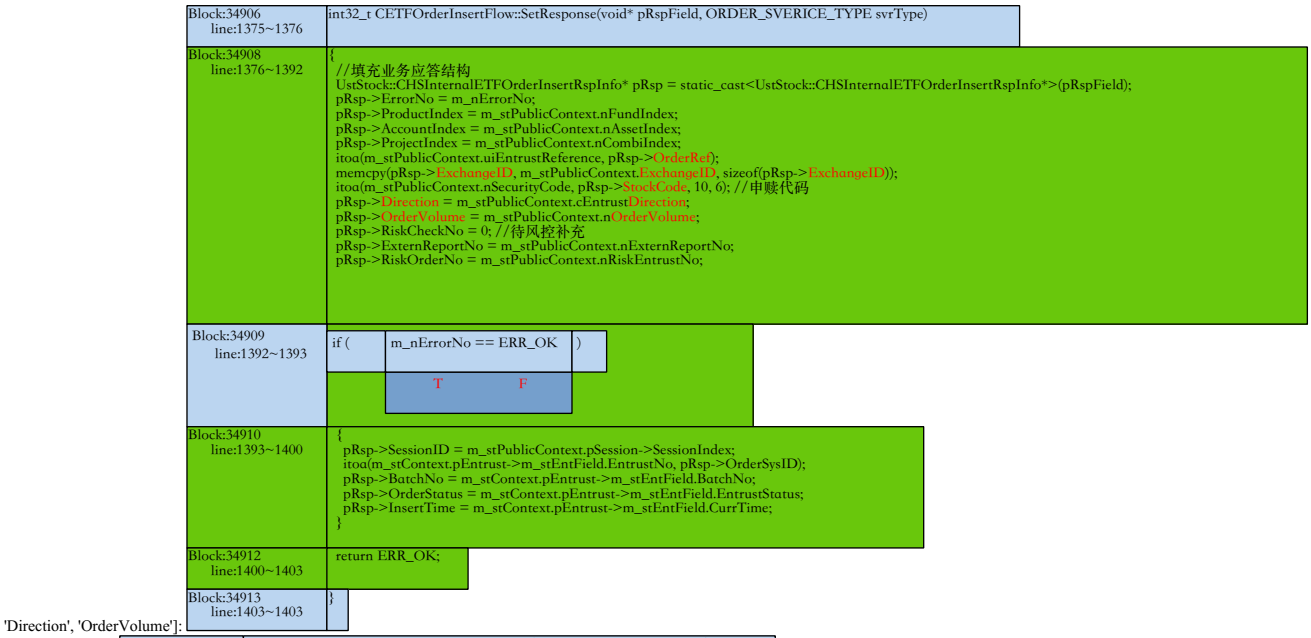
Block:29396 line:2459~2473
```
{
 std::strncpy(rsp->OrderSysID, "", sizeof(rsp->OrderSysID));
 rsp->OrderStatus = '\0';
 rsp->InsertTime = 0;
 //打错误信息
 //HSErrorMsg ErrorMsg;
 //lpContext->m_lpMsgFormat->Format(rsp->ErrorNo);
 //strncpy(ErrorMsg, (const char*)lpContext->m_lpMsgFormat->GetUTF8Message(), sizeof(ErrorMsg) - 1);
 //ErrorMsg[sizeof(ErrorMsg) - 1] = '\0';
 ////LdpMsg::SetBizOptional(writer.GetHead());//可选字段则必须打标记.调用过Reset则m_lpMsgWriter->GetHead()不为空
 //IOptionalFieldsWriter* optWriter = writer.GetOptionalWriter();
 //optWriter->WriteBinary(0, ErrorMsg, strlen(ErrorMsg) + 1);
 //optWriter->Finish();
}
```

Block:29397 line:2473~2474
```
}
```

Block:29399 line:2474~2477
```
return 0;
```

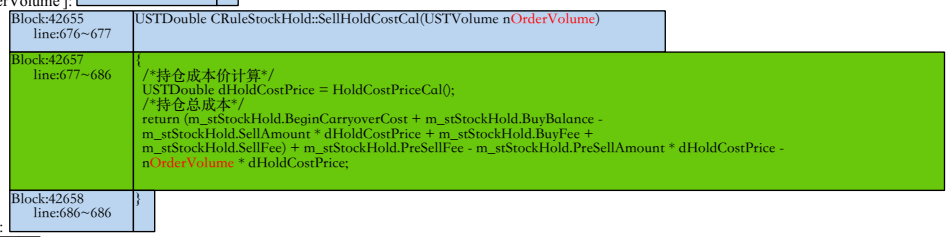
Block:29400 line:2477~2477
```
}
```

N_3062 : ['OrderRef', 'ExchangeID', 'StockCode',



'Direction', 'OrderVolume']:

N_3710 :



['OrderVolume']:

N_21 : ['StockCode', 'Direction']: