第一层if是判断gold\_copy\_trade表中source\_trade\_status\_cd的值，（’TOMS NEW’, ‘TOMS CNCLO’）

if (!DispatchSTX29Contants.VALUE\_SOURCE\_TRADE\_STATUS\_CD\_NEW.equals(sourceTradeStatusCd)) {

//cancel的时候 count计算，只有GTM cancel 才是NEW\_TO\_CANC%开头的scenario,gold\_copy\_trade表中SOURCE\_TARDE\_STATUS\_CD字段会更新成‘TOMS CNCLO’, CANCELLED\_AT字段会被更新; accounting cancel 只是match code的变化

if (DispatchSTX29Contants.NEW\_TO\_CANC\_WITH\_MATCH\_STATUS\_NO\_CHANGE\_AS\_UNMATCHED.equals(countScenario)) {

//之前new了一条没有match,然后来了一条GTM cancel，需要把new的时候记算的pending减去，TSM\_GUID表删除

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.DELETE);

//TDPPendingCount和totalCount都是记录当前区间的值，所以需要比较cancelAt和PreFundRollTime

if (tradeCreatedCancelledAt >= preFundRollTime) {

incrementalCounts.setTotalCount(1);

//TDPPending Count是new的时候加的，所以需要判断createdAt 和PreRollTime

if (newCreatedTime >= preFundRollTime) {

if (!isTPlusOneTrade) {

incrementalCounts.setTdpPendingCount(-1);

}

}

}

//new的trade之后再startCountDate之后才会被计算pending的值，所以减的时候要判断createdAt 和 startCountDate, new的时间在startCountDate之后才需要减。

if (newCreatedTime >= currentTsmCount.getStartCountDate().getTime()) {

//现在T+1 也是记录所有区间的T+1,所以和totalPending是等价的

if (isTPlusOneTrade) {

incrementalCounts.setNonTdpPendingCount(-1);

targetTsmGuid.setTdpTradeFlg(FLAG\_N);

} else {

incrementalCounts.setTotalPendingCount(-1);

}

}

} else if (DispatchSTX29Contants.NEW\_TO\_CANC\_WITH\_MATCHED\_TO\_UNMATCHED.equals(countScenario)) {

//之前new了一条之后被match ,然后来了一条GTM cancel，需要重新计算pending, 加入TSM\_GUID表

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.INSERT);

//totalCount, TdpPendingCount 是当前区间，和上个逻辑一样

if (tradeCreatedCancelledAt >= preFundRollTime) {

incrementalCounts.setTotalCount(1);

if (!isTPlusOneTrade) {

incrementalCounts.setTdpPendingCount(1);

}

}

if (isTPlusOneTrade) {

incrementalCounts.setNonTdpPendingCount(1);

targetTsmGuid.setTdpTradeFlg(FLAG\_N);

} else {

incrementalCounts.setTotalPendingCount(1);

}

} else if (DispatchSTX29Contants.UNMATCHED\_TO\_MATCHED\_TYPE.equals(countScenario)) {

//之前new了一条，然后来了一条GTM cancel, 最后来了一条 accounting cancel 和GTM cancel match上

//validatedCount ,和completedCount 都是记录当前区间的值

if (tradeCreatedCancelledAt >= preFundRollTime) {

if (DispatchSTX29Contants.isValidatedMatchCode(matchCode)) {

incrementalCounts.setValidatedCount(1);

}

if (DispatchSTX29Contants.VALUE\_TRADE\_COMPLETED\_MATCH\_CD.equals(matchCode)) {

incrementalCounts.setCompletedCount(1);

}

}

//有以下两种情况会走到这个分支：

1. 来一条new 没有被match,然后来一条GTM cancel ,这个时候pending已经被减掉，TSM\_GUID表中删除了，最后来一条 accounting cancel，这时不需要做任何操作
2. 来一条new被match,然后来一条GTM cancel ,这个时候 pending又会加1，TSM\_GUID表插入，最后来一条 accounting cancel,这个时候需要把pending减掉，TSM\_GUID表中需要删除。

所以需要先判断TSM\_GUID表中是不是存在

if (tsmGuidService.isExist(targetTsmGuid)) {

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.DELETE);

if (tradeCreatedCancelledAt >= preFundRollTime) {

if (!isTPlusOneTrade) {

incrementalCounts.setTdpPendingCount(-1);

}

}

if (isTPlusOneTrade) {

incrementalCounts.setNonTdpPendingCount(-1);

targetTsmGuid.setTdpTradeFlg(FLAG\_N);

} else {

incrementalCounts.setTotalPendingCount(-1);

}

} else {

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.NOACTION);

}

} else {

//cancel 情况下其余Scenario

if (tradeCreatedCancelledAt >= preFundRollTime) {

//除了MATCHED\_TO\_UNMATCHED和UNMATCHED\_TO\_MATCHED，其余的scenario需要计total Count

if ((!DispatchSTX29Contants.MATCHED\_TO\_UNMATCHED\_TYPE.equals(countScenario))

&& (!DispatchSTX29Contants.UNMATCHED\_TO\_MATCHED\_TYPE.equals(countScenario))) {

incrementalCounts.setTotalCount(1);

}

if (DispatchSTX29Contants.isValidatedMatchCode(matchCode)){

incrementalCounts.setValidatedCount(1);

}

}

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.NOACTION);

}

} else {

//这个分支gold\_copy\_trade表中source\_trade\_status\_cd为’TOMS NEW’

//一条new 的trade

if (DispatchSTX29Contants.NEW.equals(countScenario)) {

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.INSERT);

if (tradeCreatedCancelledAt >= preFundRollTime) {

incrementalCounts.setTotalCount(1);

if (!isTPlusOneTrade) {

incrementalCounts.setTdpPendingCount(1);

}

}

if (isTPlusOneTrade) {

incrementalCounts.setNonTdpPendingCount(1);

targetTsmGuid.setTdpTradeFlg(FLAG\_N);

} else {

incrementalCounts.setTotalPendingCount(1);

}

} else if (DispatchSTX29Contants.MATCHED\_TO\_UNMATCHED\_TYPE.equals(countScenario)) {

//有以下两种情况会走这个分支：

1. 先来一条new的trade 被match, 然后被manual unmatch.
2. 先来一条new的trade 被match,然后来一条accounting cancel,原来的new被break

//today pending 和completed 都是当前区间的值

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.INSERT);

if (tradeCreatedCancelledAt >= preFundRollTime) {

if (!isTPlusOneTrade) {

incrementalCounts.setTdpPendingCount(1);

}

incrementalCounts.setCompletedCount(-1);

}

if (isTPlusOneTrade) {

incrementalCounts.setNonTdpPendingCount(1);

targetTsmGuid.setTdpTradeFlg(FLAG\_N);

} else {

incrementalCounts.setTotalPendingCount(1);

}

} else if (DispatchSTX29Contants.UNMATCHED\_TO\_MATCHED\_TYPE.equals(countScenario)) {

//来一条new的trade,　auto match或者manual match

targetTsmGuid.setTsmGuidAction(TsmGuid.TmsGuidActions.DELETE);

if (tradeCreatedCancelledAt >= preFundRollTime) {

if (DispatchSTX29Contants.isValidatedMatchCode(matchCode)) {

incrementalCounts.setValidatedCount(1);

}

if (DispatchSTX29Contants.VALUE\_TRADE\_COMPLETED\_MATCH\_CD.equals(matchCode)) {

incrementalCounts.setCompletedCount(1);

}

if (!isTPlusOneTrade) {

incrementalCounts.setTdpPendingCount(-1);

}

}

if (isTPlusOneTrade) {

incrementalCounts.setNonTdpPendingCount(-1);

targetTsmGuid.setTdpTradeFlg(FLAG\_N);

} else {

incrementalCounts.setTotalPendingCount(-1);

}

}

}