Observations

1. [routing].[uspInsertRoutingJob\_v4] returns a jobID
2. Add EntityTypeCode to @TaskList table in routing.uspStagingSourceProcess\_v4, persist into routing.BatchControl table

MappingProcess – inserts into mapping.Mapping table. Also returns new mappings created, #MappedOutputPrimary, #MappedOutputAssociation

1. AssociationProcess – uses #Mapped tables and updates/enddates/inserts new into entity.Association
2. RoutingFlowProcess – inserts into entity i.e. entity.BPFFPAccountAttr table from table-function against staging table

Bugs:

1. EXEC [routing].[uspUpdateSuccessRoutingJob\_v4] @p\_JobId = @JobId in CATCH block of routing.uspDataFlowProcess\_v4. Should be [routing].[uspUpdateFailRoutingJob\_v4?
2. INSERT INTO @JobTaskTable

SELECT \*

FROM [routing].[JobTask]

WHERE JobId = @JobId

again in same CATCH block as 1. Does not seem to make use of @JobTaskTable? Can it be removed

NO its OK below. So ignore

[mapping].[uspMappingProcess\_v4] and INSERT INTO #MappedOutputPrimary

EXEC [mapping].[uspMappingProcess\_v4] are out of kilter in Perforce when populating #MappedOutputPrimary. I don’t think the bug has ever been hit

Atomisation

ref.uspAtomTypeProcess

routing.uspUpdateSuccessRoutingJob\_v4

MAIN \*

(@LoopId <= @BatchesCount

mapping.uspMappingProcess\_v4. Then same calls for ‘Association’ and ‘Routing’ JobTasks

rout.uspInsertRoutingJobTask\_v4,

uspMappingDataPrepare\_v4,

uspInsertRoutingJobTask\_v4

Insert into #SourceData (dynamic from tfnx(COBDate)

MAIN \*

(@LoopId <= @BatchesCount

Insert into routing.BatchControl, TempEntityToDelete

BATCH CONTROL \*-@BatchId <= @BatchesCount

[routing].[uspInsertRoutingJob\_v4]

Schema=Entity,Name=tfn(cobdate). ETypeCode=CounterPartyIDAttr ChunkSize=10000 EntityTypeCode ChunkSize

1 entity tfnVntgCounterpartyIdAttr(@COBDATE) CounterpartyIdAttr 100000

routing.uspDataFlowProcess\_v4

routing].[uspStagingSour ceProcess\_v4. \*

START