**Create General Journal for “control account” through code in D365 F&O**

In AX 2012, to create the general journal through code we were using the helper class called **AxLedgerJournalTable** and **AxLedgerJournalTrans.**

In D365FO for Journal creation (General Journal, Invoice Journal, Payment Journal etc) is the abstract class: **MCRLedgerJournal,** since an abstract class cannot be instantiated, you will have to instantiate with the suitable class based on the journal type you want to create. Here is the code, to create the General Journal as below, in D365 finance and operations:

Ledger (202087) Dr.

To Vendor Cr.

When creating above journal, if the ledger 202087 marked as control account (i.e. Main Account form >> Ledger – 202087 >> marked on “Do not allow manual entry” then system will not allow us to post the journal. Though if it is required to generate the journal then this mark is to be “unmarked” before creating the journal and is to be “marked” after posting the journal.

Follow below steps to create the general journal for control account

1. Create Runnable Class (job) to create the general journal
2. Create Common function to get “LedgerDim” of MainAccount

Step-1: Runnable class (job) - “CreateGenJournal”

class CreateGenJournal

{

*/// <summary>*

*/// Runs the class with the specified arguments.*

*/// </summary>*

*/// <param name = "\_args">The specified arguments.</param>*

public static void main(Args \_args)

{

try

{

ttsbegin;

MCRLedgerJournal journalTable;

LedgerJournalTable ledgerJournalTable;

Counter recordsInserted;

str 20 strVendAccount, strLedger;

DimensionAttributeValueCombination objDimensionAttributeValueCombination;

LedgerdimensionAccount LedgerDimVender, LedgerDimLedger;

GetLedgerDimension getledgerdimension;

MainAccount mainAccount;

DimensionAttributevalue dimensionAttributeValue, dimensionAttributeValue\_updt;

container \_ConData;

strLedger = '202076';

strVendAccount = 'D3T-000011';

\_ConData = [strLedger]; *//container can be added with CostCenter,Department,Worker ..etc.*

getledgerdimension = new GetLedgerDimension();

LedgerDimLedger =getledgerdimension.generateLedgerDimension(\_ConData, strLedger);

select \* from objDimensionAttributeValueCombination where objDimensionAttributeValueCombination.DisplayValue == strVendAccount;

LedgerDimVender = objDimensionAttributeValueCombination.RecId;

*// unmark the control account*

select \* from mainAccount join dimensionAttributeValue where

dimensionAttributeValue.EntityInstance == mainAccount.RecId

&& mainAccount.MainAccountId == strLedger; *// pass the mainaccount of General Journal*

if (dimensionAttributeValue.IsBlockedForManualEntry == NoYes::Yes)

{

info(strFmt("%1 is control account", mainAccount.MainAccountId));

select forupdate dimensionAttributeValue\_updt where dimensionAttributeValue\_updt.RecId == dimensionAttributeValue.RecId;

dimensionAttributeValue\_updt.IsBlockedForManualEntry = NoYes::No;

dimensionAttributeValue\_updt.update();

}

journalTable = new MCRLedgerJournal\_Daily(LedgerJournalType::Daily,"Gen\_JRNL");

*//Creates the journal header table*

ledgerJournalTable = journalTable.createLedgerJournalTable("Gen\_JRNL");

journalTable.parmLedgerJournalTable(ledgerJournalTable);

journalTable.parmMCRCCGeneralLedgerId();

journalTable.parmLedgerAccountType(LedgerJournalACType::Ledger);

journalTable.parmLedgerOffsetAccountType(LedgerJournalACType::Vend);

journalTable.parmExchRate(100);

journalTable.parmCurrencyCode('INR');

journalTable.parmLineNum();

journalTable.parmLedgerAccount(LedgerDimLedger);

journalTable.parmledgerOffsetAccount(LedgerDimVender);

*//journalTable.parmLedgerAccount(68719575252);*

*//journalTable.parmledgerOffsetAccount(68719576002);*

journalTable.parmTransDate(31\12\2018);

journalTable.parmTransTxt('');

*//create the journal trans table*

journalTable.createLedgerJournalTrans(0,1100,LedgerJournalACType::Ledger);

journalTable.post(LedgerJournalTable);

info(strFmt("Journal Num: %1 has been created", ledgerJournalTable.JournalNum));

*// Mark the control account*

select forupdate dimensionAttributeValue\_updt where dimensionAttributeValue\_updt.RecId == dimensionAttributeValue.RecId;

dimensionAttributeValue\_updt.IsBlockedForManualEntry = NoYes::Yes;

dimensionAttributeValue\_updt.update();

ttscommit;

}

catch (Exception::CLRError)

{

info(AifUtil::getClrErrorMessage());

}

catch (Exception::Error)

{

info("Exception::Error caught");

}

}

}

Step-2: Class to get LedgerDim of Main Account = “GetLedgerDimension”

public class GetLedgerDimension

{

public DimensionDynamicAccount generateLedgerDimension(container \_conData, str 20 strMainAccount)

{

int hierarchyCount;

int hierarchyIdx;

RecId dimAttId\_MainAccount;

LedgerRecId ledgerRecId;

MainAccount mainAccount;

RefRecId recordvalue;

DimensionAttribute dimensionAttribute;

DimensionAttributeValue dimensionAttributeValue;

DimensionSetSegmentName DimensionSet;

DimensionStorage dimStorage;

LedgerAccountContract LedgerAccountContract = new LedgerAccountContract();

DimensionAttributeValueContract ValueContract;

List valueContracts = new List(Types::Class);

dimensionAttributeValueCombination dimensionAttributeValueCombination;

*//CostCenter,Department,Worker ..etc..*

*//container \_conData =[strMainAccount];*

mainAccount = MainAccount::findByMainAccountId(strMainAccount);

recordvalue = DimensionHierarchy::getAccountStructure(mainAccount.RecId,Ledger::current());

hierarchyCount = DimensionHierarchy::getLevelCount(recordvalue);

DimensionSet = DimensionHierarchyLevel::getDimensionHierarchyLevelNames(recordvalue);

for(hierarchyIdx = 1;hierarchyIdx<=hierarchyCount;hierarchyIdx++)

{

if(hierarchyIdx == 1)

continue;

dimensionAttribute = DimensionAttribute::findByLocalizedName(DimensionSet[hierarchyIdx],false,"en-us");

if(dimensionAttribute)

{

dimensionAttributeValue = DimensionAttributeValue::findByDimensionAttributeAndValue(dimensionAttribute,conPeek(\_conData,hierarchyIdx));

if(dimensionAttributeValue)

{

ValueContract = new DimensionAttributeValueContract();

ValueContract.parmName(dimensionAttribute.Name) ;

ValueContract.parmValue(dimensionAttributeValue.CachedDisplayValue);

valueContracts.addEnd(ValueContract);

}

}

}

LedgerAccountContract.parmMainAccount(strMainAccount);

LedgerAccountContract.parmValues(valueContracts);

dimStorage = DimensionServiceProvider::buildDimensionStorageForLedgerAccount(LedgerAccountContract);

dimensionAttributeValueCombination = DimensionAttributeValueCombination::find(dimStorage.save());

ledgerRecId = dimensionAttributeValueCombination.RecId;

return ledgerRecId;

}

}