Class : dev\_ImportJournal

**Method : Class declaration**

**class** dev\_ImportJournal **extends** RunBaseBatch

{

LedgerJournalTable ledgerJournalTable;

Filename FileName;

NoYes IncludeFirstRow, DeleteExistingLines;

DialogField dlgFileName, dlgIncludeFirstRow, dlgDeleteExistingLines;

SysQueryRun queryrun;

PaymMode paymentmode;

BankChequeNum bankChequeNumber;

CustVendPaymStatus paymentStatus, CustVendPaymStatus;

BankNegInstRecipientName recipientName;

**str** paymentStatus1;

BankTransactionType transType;

**int** enumElement;

CustVendPaymReconciliationSetStatus reStatus;

DictEnum diEnum;

Voucher voucher;

TransDate documentDate, transDate;

DataAreaId companyAccount;

SysDim departement, costCenter1, purpose;

LedgerJournalACType accountType, offsetAccountType;

AccountNum accountNum, offsetAccountNum;

InvoiceId invoiceId;

LedgerJournalTransTxt txt, offsetTxt;

AmountCur amountDebit, amountCredit;

**str** accountTypeStr, offsetAccountTypeStr;

documentNum documentNum;

DataAreaId OffsetCompanyAccount;

SysDim OffsetDepartement, OffsetCostCenter, OffsetPurpose;

DimensionDynamicAccount ledgerDim,ledgerDimOffset;

CurrencyCode currencyCode;

#define.CurrentVersion(**2**)

#define.Version1(**1**)

#localmacro.CurrentList

FileName,

IncludeFirstRow,

DeleteExistingLines

#endmacro

}

**Method : AnyType\_to\_str**

**public** **str** AnyType\_to\_str(**int** row\_1, **int** coloumn\_1,SysExcelCells cells\_1)

{

**str** Return\_Value;

SysExcelCells cells\_2;

**int** row\_2;

**int** coloumn\_2;

row\_2=row\_1;

coloumn\_2=coloumn\_1;

cells\_2=cells\_1;

**switch**(cells\_2.item(row\_2,coloumn\_2).value().variantType())

{

**case** COMVariantType::VT\_BSTR:

Return\_Value = **strFmt**("%1", cells\_2.item(row\_2,coloumn\_2).value().bStr());

**break**;

**case** COMVariantType::VT\_DECIMAL, COMVariantType::VT\_R4, COMVariantType::VT\_R8:

Return\_Value = **strFmt**("%1", **any2int**(cells\_2.item(row\_2,coloumn\_2).value().double()));

**break**;

**case** COMVariantType::VT\_I1, COMVariantType::VT\_I2, COMVariantType::VT\_I4:

Return\_Value = **strFmt**("%1", cells\_2.item(row\_2,coloumn\_2).value().int());

**break**;

**case** COMVariantType::VT\_UI1, COMVariantType::VT\_UI2, COMVariantType::VT\_UI4:

Return\_Value = **strFmt**("%1", cells\_2.item(row\_2,coloumn\_2).value().uLong());

**break**;

**case** COMVariantType::VT\_DATE: *//, COMVariantType::VT\_UI2, COMVariantType::VT\_UI4:*

Return\_Value = **strFmt**("%1", cells\_2.item(row\_2,coloumn\_2).value().date());

**break**;

**case** COMVariantType::VT\_EMPTY:

Return\_Value = '';

**break**;

**default**:

**throw** error(**strfmt**('Unhandled variant type (%1).', cells\_2.item(row\_2,coloumn\_2).value().variantType()));

}

**return** Return\_Value;

}

**Method : canGoBatch**

**public** **boolean** canGoBatch()

{

**return** **true**;

}

**Method : CheckDimensions**

**public** **boolean** CheckDimensions(Name \_nameDimensionAttribute,Sysdim \_dim,**boolean** \_Update,**boolean** \_IfNecessary )

{

**boolean** isValid;

isValid = **true**;

**if**(!DimensionAttributeValue::findByDimensionAttributeAndValueNoError(DimensionAttribute::findByName(\_nameDimensionAttribute), \_dim,\_Update,\_IfNecessary) && \_dim != '')

{

warning(**strFmt**("%1 : %2 does not exist.",\_nameDimensionAttribute, \_dim));

isValid=**false**;

}

**return** isValid;

}

**Method : ClearVariables**

**public** **void** ClearVariables()

{

Voucher="";

DocumentNum="";

DocumentDate=Global::dateNull();

TransDate=Global::dateNull();

CompanyAccount="";

Departement="";

CostCenter1="";

Purpose="";

accountType=**0**;

AccountNum="";

InvoiceId="";

Txt ="";

OffsetTxt="";

amountDebit=**0**;

amountDebit=**0**;

amountCredit=**0**;

amountCredit=**0**;

offsetAccountType=**0**;

offsetAccountNum="";

OffsetCompanyAccount="";

OffsetDepartement="";

OffsetCostCenter="";

OffsetPurpose="";

Departement = "";

CostCenter1 = "";

Purpose = "";

ledgerDim = **0**;

txt = "";

paymentmode = "";

bankChequeNumber = "";

paymentStatus1 = "";

recipientName = "";

transType = "";

currencyCode = "";

}

**Method : Delete\_UsageData**

**public** **void** Delete\_UsageData()

{

SysLastValue objSysLastValue;

**delete\_from** objSysLastValue **where** objSysLastValue.userId == **curUserId**() && objSysLastValue.company == **curext**()

&& objSysLastValue.elementName=='ABMAPJournalImport' && objSysLastValue.recordType==UtilElementType::Class;

}

**Method : Dialog**

**public** Object dialog()

{

#Excel

DialogRunBase dialog = **super**();

;

dialog.caption("Import AP Journal Invoice");

dialog.addText("Import AP Journal Invoice from the following path. Do you want to continue?");

*// dlgFileName = dialog.addFieldValue(typeid(FileNameOpen),fileName,"@SYS69909"); -- Original code*

dlgFileName = dialog.addFieldValue(**extendedTypeStr**(FileNameOpen),fileName,"@SYS69909");

dialog.filenameLookupFilter(["@SYS28576",#XLSX,"@SYS28576",#XLS]);

*// dlgIncludeFirstRow = dialog.addFieldValue(typeId(NoYes), IncludeFirstRow, "Include first row"); -- Original code*

dlgIncludeFirstRow = dialog.addFieldValue(**extendedTypeStr**(NoYesId), IncludeFirstRow, "Include first row");

dlgDeleteExistingLines = dialog.addFieldValue(**extendedTypeStr**(NoYesId), DeleteExistingLines, "Delete existing lines");

**return** dialog;

}

**Method : getFromDialog**

**public** **boolean** getFromDialog()

{

**boolean** ret;

ret = **super**();

fileName = dlgFilename.value();

IncludeFirstRow = dlgIncludeFirstRow.value();

DeleteExistingLines = dlgDeleteExistingLines.value();

**return** ret;

}

**Method : initParmDefault**

**public** **void** initParmDefault()

{

this.initQuery();

**super**();

}

**Method : initQuery**

**void** initQuery()

{

Query query = **new** Query();

;

query.addDataSource(**tablenum**(InventTable));

queryrun = **new** SysQueryRun(query);

}

**Method : new**

**protected** **void** **new**()

{

**super**();

}

**Method : Pack**

**public** **container** pack()

{

**return** [#CurrentVersion,#CurrentList,queryrun.pack()];

}

**Method : parmLedgerJournalTable**

LedgerJournalTable parmLedgerJournalTable(ledgerJournalTable \_ledgerJournalTable = ledgerJournalTable)

{

;

ledgerJournalTable = \_ledgerJournalTable;

**return** ledgerJournalTable;

}

**Method : QueryRun**

QueryRun queryrun()

{

**return** queryrun;

}

**Method : run**

**public** **void** run()

{

*//added for length of excel*

*//dev\_getexcelcount objdev\_getexcelcount;*

**int** records;

SysExcelApplication application;

SysExcelWorkbooks workbooks;

SysExcelWorkbook workbook;

SysExcelWorksheets worksheets;

SysExcelWorksheet worksheet;

SysExcelCells cells;

COMVariantType type, type1, type2, type3;

**int** row, success;

axLedgerJournalTrans axLedgerJournalTrans;

LedgerJournalTrans ledgerJournalTrans;

*//Added by NCollins - AKA\_LG007\_ImportJournal\_Currency on 16 July 2015 - Start*

LedgerParameters ledgerParameters;

*//Added by NCollins - AKA\_LG007\_ImportJournal\_Currency on 16 July 2015 - END*

*// Added by SBS on 10 Feb 2014 - Vishal - Start*

Dimension dimension;

DimensionDefault DimensionDefault,DimensionDefaultOffset;

Dimension OffsetDimension;

**int** cnt,cntoffset;

*//-->*

DataArea dataArea;

*//DEL\_Dimensions dimensions;*

**boolean** ok,DimError;

**container** acctPattern;

**container** offSetAcctPattern;

HcmApprover \_HcmApprover;

**container** cont1,cont2,cont1Offset,cont2Offset;

**container** ledgerDimension,ledgerDimensionOffset;

**boolean** Offsetcon2;

DimensionAttributeValueSetStorage dimensionAttributeValueSetStorage,offsetdimensionAttributeValueSetStorage;

*//<--*

Name nameDimensionAttribute1;

Name nameDimensionAttribute2;

Name nameDimensionAttribute3;

BankChequeTable objBankChequeTable\_del,objBankChequeTable\_Chk;

LedgerJournalTrans objledgerJournalTrans;

**int** intRefresh;

dev\_ABMAPJournalImport objdev\_ABMAPJournalImport\_Insert,objdev\_ABMAPJournalImport,objdev\_ABMAPJournalImport\_del;

ExecutePermission permission = **new** ExecutePermission();

;

permission.assert();

**super**();

nameDimensionAttribute1 = "Department";

nameDimensionAttribute2 = "Center";

nameDimensionAttribute3 = "Purpose";

DimError = **false**;

application = SysExcelApplication::construct();

workbooks = application.workbooks();

**try**

{

workbooks.open(filename);

}

**catch** (Exception::Error)

{

**throw** error("File cannot be opened.");

}

**if** (!ledgerJournalTable)

{

**throw** error("Journal table not found!");

}

row = **0** ;

**try**

{

**ttsbegin**;

**if** (deleteExistingLines == NoYes::Yes)

{

**while** **select** \* **from** objledgerJournalTrans **where** objledgerJournalTrans.JournalNum == ledgerJournalTable.JournalNum

{

**if**(objledgerJournalTrans.RecId != **0**)

{

**select** \* **from** objBankChequeTable\_Chk **where** objBankChequeTable\_Chk.Voucher==objledgerJournalTrans.Voucher;

**if**(objBankChequeTable\_Chk.ChequeNum=="")

{

**delete\_from** ledgerJournalTrans **where** ledgerJournalTrans.JournalNum == ledgerJournalTable.JournalNum

&& ledgerJournalTrans.Voucher==objledgerJournalTrans.Voucher;

}

**else**

{

info(**strFmt**("Voucher: %1 can not be deleted because cheque number %2 is created for this voucher.",objledgerJournalTrans.Voucher,objBankChequeTable\_Chk.ChequeNum));

}

}

}

}

**if** (includeFirstRow == NoYes::No)

row++;

workbook = workbooks.item(**1**);

worksheets = workbook.worksheets();

worksheet = worksheets.itemFromNum(**1**);

cells = worksheet.cells();

intRefresh = **0**;

**delete\_from** objdev\_ABMAPJournalImport\_del **where** objdev\_ABMAPJournalImport\_del.JournalNo == ledgerJournalTable.JournalNum

&& objdev\_ABMAPJournalImport\_del.Journal\_UserID == **curUserId**();

*// ttsBegin;*

**try**

{

**do**

{

this.ClearVariables();

*/\*if(intRefresh == 100)*

*{*

*this.Catche\_Refresh();*

*intRefresh = 0;*

*}\*/*

row++;

ok = **true**;

Voucher = this.AnyType\_to\_str(row,**1**,cells);*//cells.item(row, 1).value().bStr();*

DocumentNum = this.AnyType\_to\_str(row,**2**,cells);

*//cells.item(row, 2).value().variantType() == COMVariantType::VT\_R8 ?*

*// strfmt("%1", num2str(cells.item(row, 2).value().double(), 0, 0, 0, 0)) :*

*//cells.item(row, 2).value().bStr();*

DocumentDate = cells.item(row, **3**).value().date();

*//date2str(cells.item(row, 4).value().date(),DateFormat::Auto,2,1,2,1,4)*

TransDate = cells.item(row, **4**).value().date();

CompanyAccount = this.AnyType\_to\_str(row,**5**,cells);*//cells.item(row, 5).value().bStr();*

*//Added by NCollins - AKA\_LG007\_ImportJournal\_Currency on 16 July 2015 - Start*

**select** **firstOnly** dataArea

**where** dataArea.Id == CompanyAccount;

**if**(!dataArea.RecId)

{

warning(**strfmt**("@LEG78", CompanyAccount, row-**1**));

*//CompanyAccount = ledgerParameters.AKAImportCompanyId;*

}

*//Added by NCollins - AKA\_LG007\_ImportJournal\_Currency on 16 July 2015 - END*

Departement = this.AnyType\_to\_str(row,**6**,cells);

**if**(Departement!='')

{

**if**(this.CheckDimensions(nameDimensionAttribute1,Departement,**false**,**true**)==**false**)

{

DimError=**true**;

**break**;

}

}

*/\* cells.item(row, 6).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 6).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 6).value().bStr();\*/*

CostCenter1 = this.AnyType\_to\_str(row,**7**,cells);

**if**(CostCenter1!='')

{

**if**(this.CheckDimensions(nameDimensionAttribute2,CostCenter1,**false**,**true**)==**false**)

{

DimError=**true**;

**break**;

}

}

*/\*cells.item(row, 7).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 7).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 7).value().bStr();\*/*

Purpose = this.AnyType\_to\_str(row,**8**,cells);

**if**(purpose!='')

{

**if**(this.CheckDimensions(nameDimensionAttribute3,Purpose,**false**,**true**)==**false**)

{

DimError=**true**;

**break**;

}

}

*/\*cells.item(row, 8).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 8).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 8).value().bStr();\*/*

accountType = **str2enum**(accountType,this.AnyType\_to\_str(row,**9**,cells));*//str2enum(accountType, cells.item(row, 9).value().bStr());*

AccountNum = this.AnyType\_to\_str(row,**10**,cells);

**if**(AccountNum!='')

{

**if**(accountType == ledgerJournalACType::Vend)

{

**changecompany**(companyAccount)

{

**if**(!vendTable::exist(AccountNum))

{

warning(**strfmt**("Vendor account %1 not exists in company %2", accountnum,companyAccount));

DimError=**true**;

**break**;

}

}

}

**else** **if**(accountType == ledgerJournalACType::Ledger)

{

**changecompany**(companyAccount)

{

**if**(!MainAccount::findByMainAccountId(AccountNum))

{

warning(**strfmt**("Ledger account %1 not exists in company %2", accountnum,companyAccount));

DimError=**true**;

**break**;

}

}

}

**else** **if**(accountType == ledgerJournalACType::Cust)

{

**changecompany**(companyAccount)

{

**if**(!CustTable::exist(AccountNum))

{

warning(**strfmt**("Customer account %1 not exists in company %2", accountnum,companyAccount));

DimError=**true**;

**break**;

}

}

}

}

*/\*cells.item(row, 10).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 10).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 10).value().bStr();\*/*

InvoiceId = this.AnyType\_to\_str(row,**12**,cells);

*/\*cells.item(row, 12).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 12).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 12).value().bStr();\*/*

Txt = this.AnyType\_to\_str(row,**13**,cells);

*/\*cells.item(row, 13).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 13).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 13).value().bStr(); \*/*

OffsetTxt = this.AnyType\_to\_str(row,**14**,cells);

*/\*cells.item(row,14).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 14).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 14).value().bStr(); \*/*

amountDebit = cells.item(row, **15**).value().double();

amountDebit = Currency::amount(amountDebit);

amountCredit = cells.item(row, **16**).value().double();

amountCredit = Currency::amount(amountCredit);

offsetAccountType = **str2enum**(offsetAccountType,this.AnyType\_to\_str(row,**17**,cells));*//str2enum(offsetAccountType, cells.item(row, 17).value().bStr());*

offsetAccountNum = this.AnyType\_to\_str(row,**18**,cells);

*/\*cells.item(row, 18).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 18).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 18).value().bStr();\*/*

OffsetCompanyAccount = this.AnyType\_to\_str(row,**19**,cells);*//cells.item(row, 19).value().bStr();*

**if**(offsetAccountNum!='')

{

**if**(offsetAccountType == ledgerJournalACType::Vend)

{

**changecompany**(OffsetcompanyAccount)

{

**if**(!vendTable::exist(OffsetAccountNum))

{

warning(**strfmt**("Offset Vendor account %1 not exists in Offset company %2", Offsetaccountnum,OffsetcompanyAccount));

DimError=**true**;

**break**;

}

}

}

**else** **if**(offsetAccountType == ledgerJournalACType::Ledger)

{

**changecompany**(OffsetcompanyAccount)

{

*// if(!LedgerTable::exist(OffsetAccountNum)) -- Original code*

**if**(!MainAccount::findByMainAccountId(OffsetAccountNum))

{

warning(**strfmt**("Offset Ledger account %1 not exists in Offset company %2", Offsetaccountnum,OffsetcompanyAccount));

DimError=**true**;

**break**;

}

}

}

**else** **if**(offsetAccountType == ledgerJournalACType::Cust)

{

**changecompany**(OffsetcompanyAccount)

{

**if**(!CustTable::exist(OffsetAccountNum))

{

warning(**strfmt**("Offset Customer account %1 not exists in Offset company %2", Offsetaccountnum,OffsetcompanyAccount));

DimError=**true**;

**break**;

}

}

}

}

OffsetDepartement = this.AnyType\_to\_str(row,**20**,cells);

**if**(OffsetDepartement!='')

{

**if**(this.CheckDimensions(nameDimensionAttribute1,OffsetDepartement,**false**,**true**)==**false**)

{

DimError=**true**;

**break**;

}

}

*/\*cells.item(row, 20).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 20).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 20).value().bStr();\*/*

OffsetCostCenter = this.AnyType\_to\_str(row,**21**,cells);

**if**(OffsetCostCenter!='')

{

**if**(this.CheckDimensions(nameDimensionAttribute2,OffsetCostCenter,**false**,**true**)==**false**)

{

DimError=**true**;

**break**;

}

}

*/\*cells.item(row, 21).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 21).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 21).value().bStr();\*/*

OffsetPurpose = this.AnyType\_to\_str(row,**22**,cells);

**if**(OffsetPurpose!='')

{

**if**(this.CheckDimensions(nameDimensionAttribute3,OffsetPurpose,**false**,**true**)==**false**)

{

DimError=**true**;

**break**;

}

}

*/\*cells.item(row, 22).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 22).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 22).value().bStr();\*/*

**if** (ledgerJournalTable.JournalType == ledgerJournalType::Payment)

{

paymentmode = this.AnyType\_to\_str(row,**23**,cells);

*/\*cells.item(row, 23).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 23).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 23).value().bStr();\*/*

bankChequeNumber = this.AnyType\_to\_str(row,**24**,cells);

*/\*cells.item(row, 24).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 24).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 24).value().bStr();\*/*

paymentStatus1 = this.AnyType\_to\_str(row,**25**,cells);*//cells.item(row, 25).value().bStr();*

recipientName = this.AnyType\_to\_str(row,**26**,cells);*//cells.item(row, 26).value().bStr();*

transType = this.AnyType\_to\_str(row,**27**,cells);

}

currencyCode = this.AnyType\_to\_str(row,**28**,cells);

*/\*cells.item(row,28).value().variantType() == COMVariantType::VT\_R8 ?*

*strfmt("%1", num2str(cells.item(row, 28).value().double(), 0, 0, 0, 0)) :*

*cells.item(row, 28).value().bStr();\*/*

*//Added by NCollins - AKA\_LG007\_ImportJournal\_Currency on 30 June 2015 - Start*

**if**(!currencyCode)

{

currencyCode = CompanyInfo::standardCurrency();

}

**else**

**if**(!Currency::exist(currencyCode))

{

**throw** error(**strfmt**("Currency %1 does not exist (line number %1)", currencyCode, row));

}

*//Insert in dev\_ABMAPJournalImport\_Insert table from excel*

objdev\_ABMAPJournalImport\_Insert.clear();

objdev\_ABMAPJournalImport\_Insert.JournalNo = ledgerJournalTable.JournalNum;

objdev\_ABMAPJournalImport\_Insert.Voucher = voucher;

objdev\_ABMAPJournalImport\_Insert.DocumentNum = documentNum;

objdev\_ABMAPJournalImport\_Insert.DocumentDate = documentDate;

objdev\_ABMAPJournalImport\_Insert.TransDate = transDate;

objdev\_ABMAPJournalImport\_Insert.CompanyAccount = companyAccount;

objdev\_ABMAPJournalImport\_Insert.Departement = departement;

objdev\_ABMAPJournalImport\_Insert.CostCenter = costCenter1;

objdev\_ABMAPJournalImport\_Insert.Purpose = purpose;

objdev\_ABMAPJournalImport\_Insert.accountType = accountType;

objdev\_ABMAPJournalImport\_Insert.AccountNum = accountNum;

objdev\_ABMAPJournalImport\_Insert.InvoiceId = invoiceId;

objdev\_ABMAPJournalImport\_Insert.Txt = txt;

objdev\_ABMAPJournalImport\_Insert.OffsetTxt = offsetTxt;

objdev\_ABMAPJournalImport\_Insert.amountDebit = amountDebit;

objdev\_ABMAPJournalImport\_Insert.amountCredit = amountCredit;

objdev\_ABMAPJournalImport\_Insert.offsetAccountType = offsetAccountType;

objdev\_ABMAPJournalImport\_Insert.offsetAccountNum = offsetAccountNum;

objdev\_ABMAPJournalImport\_Insert.OffsetCompanyAccount = OffsetCompanyAccount;

objdev\_ABMAPJournalImport\_Insert.OffsetDepartement = OffsetDepartement;

objdev\_ABMAPJournalImport\_Insert.OffsetCostCenter = OffsetCostCenter;

objdev\_ABMAPJournalImport\_Insert.OffsetPurpose = OffsetPurpose;

objdev\_ABMAPJournalImport\_Insert.paymentmode = paymentmode;

objdev\_ABMAPJournalImport\_Insert.bankChequeNumber = bankChequeNumber;

objdev\_ABMAPJournalImport\_Insert.paymentStatus =paymentStatus1;

objdev\_ABMAPJournalImport\_Insert.recipientName=recipientName;

objdev\_ABMAPJournalImport\_Insert.transType=transType;

objdev\_ABMAPJournalImport\_Insert.currencyCode = currencyCode;

objdev\_ABMAPJournalImport\_Insert.Journal\_UserID =**curUserId**();

objdev\_ABMAPJournalImport\_Insert.UserId = Global::currentWorker();

objdev\_ABMAPJournalImport\_Insert.insert();

*//info(strFmt("row no %1",row));*

type2 = cells.item(row+**1**, **10**).value().variantType(); *//9*

}

**while** (type2 != COMVariantType::VT\_EMPTY);

workbook.close(**false**);

application.quit();

application.finalize();

application = **null**;

} *//end Try*

**catch** (Exception::Error)

{

info("Unexpected error");

workbook.close(**false**);

application.quit();

application.finalize();

application = **null**;

}

**ttscommit**;

**ttsBegin**;

row= **0**;

**if**(DimError==**false**)

{

**while** **select** \* **from** objdev\_ABMAPJournalImport **where** objdev\_ABMAPJournalImport.JournalNo == ledgerJournalTable.JournalNum

&& objdev\_ABMAPJournalImport.Journal\_UserID == **curUserId**()

{

this.ClearVariables();

voucher=objdev\_ABMAPJournalImport.Voucher;

documentNum = objdev\_ABMAPJournalImport.DocumentNum;

documentDate = objdev\_ABMAPJournalImport.DocumentDate;

transDate = objdev\_ABMAPJournalImport.TransDate;

companyAccount = objdev\_ABMAPJournalImport.CompanyAccount;

departement = objdev\_ABMAPJournalImport.Departement;

costCenter1 = objdev\_ABMAPJournalImport.CostCenter;

purpose = objdev\_ABMAPJournalImport.Purpose;

accountType = objdev\_ABMAPJournalImport.accountType;

accountNum = objdev\_ABMAPJournalImport.AccountNum;

invoiceId = objdev\_ABMAPJournalImport.InvoiceId;

txt =objdev\_ABMAPJournalImport.Txt;

offsetTxt = objdev\_ABMAPJournalImport.OffsetTxt;

amountDebit = objdev\_ABMAPJournalImport.amountDebit;

amountCredit = objdev\_ABMAPJournalImport.amountCredit;

offsetAccountType = objdev\_ABMAPJournalImport.offsetAccountType;

offsetAccountNum = objdev\_ABMAPJournalImport.offsetAccountNum;

OffsetCompanyAccount = objdev\_ABMAPJournalImport.OffsetCompanyAccount;

OffsetDepartement = objdev\_ABMAPJournalImport.OffsetDepartement;

OffsetCostCenter = objdev\_ABMAPJournalImport.OffsetCostCenter;

OffsetPurpose = objdev\_ABMAPJournalImport.OffsetPurpose;

currencyCode =objdev\_ABMAPJournalImport.currencyCode;

axLedgerJournalTrans = **new** axLedgerJournalTrans();

dimensionAttributeValueSetStorage = **new** DimensionAttributeValueSetStorage();

offsetdimensionAttributeValueSetStorage = **new** DimensionAttributeValueSetStorage();

axLedgerJournalTrans.parmJournalNum(ledgerJournalTable.JournalNum);

axLedgerJournalTrans.parmDocumentDate(documentDate);

axLedgerJournalTrans.parmTransDate(TransDate);

axLedgerJournalTrans.parmCompany(companyAccount);

**if**(CompanyAccount != **curext**())

{

**changeCompany**(CompanyAccount)

{

**if**(AccountNum != "")

{

cont1=**conNull**();

cont2=**conNull**();

cnt=**0**;

**if**(Departement != '')

{

cnt++;

cont2+=['Department',Departement];

dimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute1), Departement));

}

**if**(CostCenter1 != '')

{

cnt++;

cont2+=['Center',CostCenter1];

dimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute2), CostCenter1));

}

**if**(Purpose != '')

{

cnt++;

cont2+=['Purpose',Purpose];

dimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute3), Purpose));

}

**if**(accountType == LedgerJournalACType::Ledger)

{

cont1+=['MainAccount',AccountNum,cnt];

cont1+=cont2;

ledgerDim = AxdDimensionUtil::getLedgerAccountId(cont1);

**if**(ledgerDim==**0**)

{

acctPattern = [AccountNum,AccountNum];

ledgerDim = AxdDimensionUtil::getLedgerAccountId( acctPattern);

}

axLedgerJournalTrans.parmLedgerDimension(ledgerDim);

}

**else**

{

ledgerDim = DimensionStorage::getDynamicAccount( AccountNum,accountType);

ledgerDimension +=cnt;

ledgerDimension +=cont2;

DimensionDefault = AxdDimensionUtil::getDimensionAttributeValueSetId(ledgerDimension);

axLedgerJournalTrans.parmLedgerDimension(ledgerDim);

axLedgerJournalTrans.parmDefaultDimension(dimensionAttributeValueSetStorage.save());*//DimensionDefault);*

}

} *//end if(objdev\_JournalsTrans\_Import.AccountNum != "")*

**else**

{

axLedgerJournalTrans.parmLedgerDimension(**0**);

axLedgerJournalTrans.parmDefaultDimension(**0**);

}

} *//changeCompany*

}*//InterCompany Transactions*

**else**

{

*//Account type-Financial Dimension*

**if**(AccountNum != "")

{

cont1=**conNull**();

cont2=**conNull**();

cnt=**0**;

**if**(Departement != '')

{

cnt++;

cont2+=['Department',Departement];

dimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute1), Departement));

}

**if**(CostCenter1 != '')

{

cnt++;

cont2+=['Center',CostCenter1];

dimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute2), CostCenter1));

}

**if**(Purpose != '')

{

cnt++;

cont2+=['Purpose',Purpose];

dimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute3), Purpose));

}

**if**(accountType == LedgerJournalACType::Ledger)

{

cont1+=['MainAccount',AccountNum,cnt];

cont1+=cont2;

ledgerDim = AxdDimensionUtil::getLedgerAccountId(cont1);

**if**(ledgerDim==**0**)

{

acctPattern = [AccountNum,AccountNum];

ledgerDim = AxdDimensionUtil::getLedgerAccountId( acctPattern);

}

axLedgerJournalTrans.parmLedgerDimension(ledgerDim);

}

**else**

{

ledgerDim = DimensionStorage::getDynamicAccount( AccountNum,accountType);

ledgerDimension +=cnt;

ledgerDimension +=cont2;

DimensionDefault = AxdDimensionUtil::getDimensionAttributeValueSetId(ledgerDimension);

axLedgerJournalTrans.parmLedgerDimension(ledgerDim);

axLedgerJournalTrans.parmDefaultDimension(dimensionAttributeValueSetStorage.save());

}

} *//end if(objdev\_JournalsTrans\_Import.AccountNum != "")*

**else**

{

axLedgerJournalTrans.parmLedgerDimension(**0**);

axLedgerJournalTrans.parmDefaultDimension(**0**);

}

}*//end if(CompanyAccount != curext())*

*//Added by NCollins - AKA\_LG007\_ImportJournal\_Currency on 16 July 2015 - END*

dimension[**1**] = departement;

dimension[**2**] = costCenter1;

dimension[**3**] = purpose;

axLedgerJournalTrans.parmAccountType(accountType);

axLedgerJournalTrans.parmInvoice(invoiceId);

*//check if account exist and give warning*

**switch**(accounttype)

{

**case** ledgerJournalACType::Vend :

**changecompany**(companyAccount)

{

**if**(!vendTable::exist(AccountNum))

{

warning(**strfmt**("Vendor account %1 not exists in company %2", accountnum,companyAccount));

}

}

axLedgerJournalTrans.parmPayment(vendTable::find(accountNum).PaymTermId);

axLedgerJournalTrans.parmPostingProfile(vendParameters::find().PostingProfile);

**if**(axLedgerJournalTrans.parmPayment())

axLedgerJournalTrans.parmDue(paymTerm::find(AxledgerJournalTrans.parmPayment()).due(axLedgerJournalTrans.parmTransDate()));

**else**

axLedgerJournalTrans.parmDue(axLedgerJournalTrans.parmTransDate());

**break**;

**case** ledgerJournalACType::Ledger :

**changecompany**(companyAccount)

{

**if**(!MainAccount::findByMainAccountId(AccountNum))

{

warning(**strfmt**("Ledger account %1 not exists in company %2", accountnum,companyAccount));

}

}

**break**;

**case** ledgerJournalACType::Cust :

**changecompany**(companyAccount)

{

**if**(!CustTable::exist(AccountNum))

{

warning(**strfmt**("Customer account %1 not exists in company %2", accountnum,companyAccount));

}

}

axLedgerJournalTrans.parmPaymMode(custTable::find(AccountNum).PaymMode);

axLedgerJournalTrans.parmPayment(custTable::find(accountNum).PaymTermId);

axLedgerJournalTrans.parmPostingProfile(CustParameters::find().PostingProfile);

**if**(axLedgerJournalTrans.parmPayment())

axLedgerJournalTrans.parmDue(paymTerm::find(AxledgerJournalTrans.parmPayment()).due(axLedgerJournalTrans.parmTransDate()));

**else**

axLedgerJournalTrans.parmDue(axLedgerJournalTrans.parmTransDate());

**break**;

}

**if** (!txt && accountType == LedgerJournalACType::Vend)

{

**if** (companyAccount != **curExt**())

{

**changecompany**(companyAccount)

{

txt = vendTable::find(accountNum).name();

}

}

**else**

{

txt = vendTable::find(accountNum).name();

}

}

*// Added by SBS - VT on 10 Feb 2014 - Start*

**if** (ledgerJournalTable.JournalType == ledgerJournalType::Payment)

{

paymentmode = objdev\_ABMAPJournalImport.paymentmode;

bankChequeNumber = objdev\_ABMAPJournalImport.bankChequeNumber;

paymentStatus1 = objdev\_ABMAPJournalImport.paymentStatus;

recipientName = objdev\_ABMAPJournalImport.recipientName;

transType = objdev\_ABMAPJournalImport.transType;

**changeCompany**(companyAccount)

{

**if** (!bankTransType::exist(transType))

{

warning(**strfmt**("Bank transaction type %1 doesnot exist in company %2", transType,companyAccount));

}

}

axLedgerJournalTrans.parmPaymMode(paymentmode);

*//axLedgerJournalTrans.parmEmployeeName(recipientName);*

axLedgerJournalTrans.parmBankTransType(transType);

diEnum = **new** dictenum(**enumnum**(CustVendPaymStatus));

enumElement = diEnum.name2Value(paymentStatus1);

**if** (enumElement == **255**)

{

paymentStatus = CustVendPaymStatus::None;

warning(**strfmt**("Payment status '%1' doesnot exist.So, it has been defaulted to 'None'", paymentStatus1));

}

**else**

{

paymentStatus = **str2enum**(CustVendPaymStatus,diEnum.value2Name(enumElement));

}

axLedgerJournalTrans.parmPaymentStatus(paymentStatus);

}

axLedgerJournalTrans.parmCurrencyCode(currencyCode);

axLedgerJournalTrans.parmExchRate(Currency::exchRate(currencyCode, TransDate));

*// Added by NCollins - AKA\_LG007\_ImportJournal\_Currency on 30 June 2015 - END*

axLedgerJournalTrans.parmTxt(txt);

axLedgerJournalTrans.parmAmountCurDebit(amountDebit);

axLedgerJournalTrans.parmAmountCurCredit(amountCredit);

axLedgerJournalTrans.parmDocumentNum(documentNum);

axLedgerJournalTrans.parmOffsetCompany(OffsetCompanyAccount);

axLedgerJournalTrans.parmApproved(NoYes::Yes);

axLedgerJournalTrans.parmApprover(HcmWorker::userId2Worker(**curUserId**()));*//parmApprovedBy(curuserid());*

*//assign offset dimension if the offset company account is filled in and different with company account.*

**if**(axLEdgerJournalTrans.parmOffsetCompany() && axLedgerJournalTrans.parmCompany()!= axLedgerJournalTrans.parmOffsetCompany())

*// axLedgerJournalTrans.parmInterCoDimension(OffsetDimension);*

*//check vendor/customer/ledger exist if the offset account num is filled.*

axLedgerJournalTrans.save();

axLedgerJournalTrans.parmOffsetAccountType(offsetAccountType);

**if**(OffsetCompanyAccount != **curext**() && OffsetCompanyAccount != "")

{

**changeCompany**(OffsetCompanyAccount)

{

**if**(offsetAccountNum != "")

{

cont1Offset=**conNull**();

cont2Offset=**conNull**();

Offsetcon2 = **false**;

cntOffset=**0**;

**if**(OffsetDepartement != '')

{

cntOffset++;

cont2Offset+=['Department',OffsetDepartement];

Offsetcon2 = **true**;

offsetdimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute1), OffsetDepartement));

}

**if**(OffsetCostCenter != '')

{

cntOffset++;

cont2Offset+=['Center',OffsetCostCenter];

offsetdimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute2), OffsetCostCenter));

Offsetcon2 = **true**;

}

**if**(OffsetPurpose != '')

{

cntOffset++;

cont2Offset+=['Purpose1',OffsetPurpose];

offsetdimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute3), OffsetPurpose));

Offsetcon2 = **true**;

}

**if**(offsetAccountType == LedgerJournalACType::Ledger)

{

cont1Offset+=['MainAccount',offsetAccountNum,cntOffset];

**if**(Offsetcon2 == **true**)

{

cont1Offset+=cont2Offset;

}

ledgerDimOffset = AxdDimensionUtil::getLedgerAccountId(cont1Offset);

**if**(ledgerDimOffset==**0**)

{

OffsetacctPattern = [offsetAccountNum,offsetAccountNum];

ledgerDimOffset = AxdDimensionUtil::getLedgerAccountId(OffsetacctPattern);

}

axLedgerJournalTrans.parmoffsetLedgerDimension(ledgerDimOffset);

}

**else**

{

ledgerDimOffset = DimensionStorage::getDynamicAccount( offsetAccountNum,offsetAccountType);

ledgerDimensionOffset +=cntOffset;

**if**(Offsetcon2 == **true**)

{

ledgerDimensionOffset +=cont2Offset;

}

DimensionDefaultOffset = AxdDimensionUtil::getDimensionAttributeValueSetId(ledgerDimensionOffset);

axLedgerJournalTrans.parmOffsetDefaultDimension(offsetdimensionAttributeValueSetStorage.save());

axLedgerJournalTrans.parmoffsetLedgerDimension(ledgerDimOffset);

}

} *//end if(objdev\_JournalsTrans\_Import.AccountNum != "")*

**else**

{

axLedgerJournalTrans.parmoffsetLedgerDimension(**0**);

axLedgerJournalTrans.parmOffsetDefaultDimension(**0**);

}

}*//end changeCompany(OffsetCompanyAccount)*

}*//end if(OffsetCompanyAccount != curext() && OffsetCompanyAccount != "")*

**else**

{

**if**(offsetAccountNum != "")

{

cont1Offset=**conNull**();

cont2Offset=**conNull**();

Offsetcon2 = **false**;

cntOffset=**0**;

**if**(OffsetDepartement != '')

{

cntOffset++;

cont2Offset+=['Department',OffsetDepartement];

Offsetcon2 = **true**;

offsetdimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute1), OffsetDepartement));

}

**if**(OffsetCostCenter != '')

{

cntOffset++;

cont2Offset+=['Center',OffsetCostCenter];

Offsetcon2 = **true**;

offsetdimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute2), OffsetCostCenter));

}

**if**(OffsetPurpose != '')

{

cntOffset++;

cont2Offset+=['Purpose1',OffsetPurpose];

Offsetcon2 = **true**;

offsetdimensionAttributeValueSetStorage.addItem(DimensionAttributeValue::

findByDimensionAttributeAndValue(DimensionAttribute::findByName(nameDimensionAttribute3), OffsetPurpose));

}

**if**(offsetAccountType == LedgerJournalACType::Ledger)

{

cont1Offset+=['MainAccount',offsetAccountNum,cntOffset];

info(offsetAccountNum);

**if**(Offsetcon2 == **true**)

{

cont1Offset+=cont2Offset;

}

ledgerDimOffset = AxdDimensionUtil::getLedgerAccountId(cont1Offset);

**if**(ledgerDimOffset==**0**)

{

OffsetacctPattern = [offsetAccountNum,offsetAccountNum];

ledgerDimOffset = AxdDimensionUtil::getLedgerAccountId(OffsetacctPattern);

}

axLedgerJournalTrans.parmoffsetLedgerDimension(ledgerDimOffset);

}

**else**

{

ledgerDimOffset = DimensionStorage::getDynamicAccount( offsetAccountNum,offsetAccountType);

ledgerDimensionOffset +=cntOffset;

**if**(Offsetcon2 == **true**)

{

ledgerDimensionOffset +=cont2Offset;

}

DimensionDefaultOffset = AxdDimensionUtil::getDimensionAttributeValueSetId(ledgerDimensionOffset);

axLedgerJournalTrans.parmDefaultDimension(offsetdimensionAttributeValueSetStorage.save());

axLedgerJournalTrans.parmoffsetLedgerDimension(ledgerDimOffset);

}

} *//end if(objdev\_JournalsTrans\_Import.AccountNum != "")*

**else**

{

axLedgerJournalTrans.parmoffsetLedgerDimension(**0**);

axLedgerJournalTrans.parmOffsetDefaultDimension(**0**);

}

}*//end If*

**if**(offsetAccountNum)

{

OffsetDimension[**1**] = OffsetDepartement;

OffsetDimension[**2**] = OffsetCostCenter;

OffsetDimension[**3**] = OffsetPurpose;

**switch**(Offsetaccounttype)

{

**case** ledgerJournalACType::Vend :

**changecompany**(OffsetcompanyAccount)

{

**if**(!vendTable::exist(OffsetAccountNum))

{

warning(**strfmt**("Offset Vendor account %1 not exists in Offset company %2", Offsetaccountnum,OffsetcompanyAccount));

}

}

**break**;

**case** ledgerJournalACType::Ledger :

**changecompany**(OffsetcompanyAccount)

{

*// if(!LedgerTable::exist(OffsetAccountNum)) -- Original code*

**if**(!MainAccount::findByMainAccountId(OffsetAccountNum))

{

warning(**strfmt**("Offset Ledger account %1 not exists in Offset company %2", Offsetaccountnum,OffsetcompanyAccount));

}

}

**break**;

**case** ledgerJournalACType::Cust :

**changecompany**(OffsetcompanyAccount)

{

**if**(!CustTable::exist(OffsetAccountNum))

{

warning(**strfmt**("Offset Customer account %1 not exists in Offset company %2", Offsetaccountnum,OffsetcompanyAccount));

}

}

**break**;

}

}

*//if offset account is empty then take make the offset txt same with txt.*

**if**(!offsetTxt && axledgerJournalTrans.parmOffsetTxt())offsetTxt = txt;*//.parmOffsetAccount())offsetTxt = txt;*

axLedgerJournalTrans.parmOffsetTxt(offsetTxt);

axLedgerJournalTrans.save();

**if** (ledgerJournalTable.JournalType == ledgerJournalType::Payment)

{

reStatus = **new** CustVendPaymReconciliationSetStatus();

reStatus.parmRecid(axLedgerJournalTrans.ledgerJournalTrans().RecId);

reStatus.parmStatus(paymentStatus);

reStatus.run();

**if** (amountDebit != **0**

&& offsetAccountType == ledgerjournalactype::Bank

&& bankChequeNumber != "")

{

axLedgerJournalTrans.parmBankChequeNum(bankChequeNumber);

dev\_ImportJournal::createCheque(bankChequeNumber, OffsetAccountNum,

axLedgerJournalTrans.ledgerJournalTrans());

axLedgerJournalTrans.parmPaymReference(bankChequeNumber);

}

}

axLedgerJournalTrans.save();

success++;

row++;

}*//end of objdev\_ABMAPJournalImport table*

}

**ttsCommit**;

info(**strfmt**("%1 row(s) succesfully imported", row));

} *//end Main Try*

**catch** (Exception::Error)

{

**ttsabort**;

error("Import failed");

}

}

**Method : runsImpersonated**

**public** **boolean** runsImpersonated()

{

**return** **true**;

}

**Method : showbatchtab**

**boolean** showBatchTab(**boolean** \_showBatchTab = \_showBatchTab)

{

**boolean** ret;

ret = **false**;*//super(\_showBatchTab);*

**return** ret;

}

**Method : showqueryvalue**

**public** **boolean** showQueryValues()

{

**return** **false**;

}

**Method : unpack**

**public** **boolean** unpack(**container** packedClass)

{

Version version = RunBase::getVersion(packedClass);

**container** queryCon;

**switch** (version)

{

**case** #CurrentVersion:

[version,#CurrentList,queryCon] = packedClass;

**if** (SysQuery::isPackedOk(queryCon))

queryrun = **new** QueryRun(queryCon);

**else**

this.initQuery();

**break**;

**default**:

**return** **false**;

}

**return** **true**;

}

**Method : construct**

**static** dev\_ImportJournal construct()

{

**return** **new** dev\_ImportJournal();

}

**Method : createCheque**

*// Added by SBS - VT on 10 Feb 2014 - Start*

**server** **static** **public** **void** createCheque(BankChequeNum \_chequeNum,

CompanyBankAccountId \_accountID,

LedgerJournalTrans \_ledgerJournalTrans)

{

BankChequeTable bankChequeTable;

;

bankChequeTable.clear();

bankChequeTable.ChequeNum = \_chequeNum;

bankChequeTable.ChequeStatus = ChequeStatus::Created;

bankChequeTable.AccountID = \_accountID;

**if** (\_ledgerJournalTrans.AccountType == ledgerjournalACType::Vend)

{

bankChequeTable.RecipientType = BankChequeRecipientType::Vend;

}

**else** **if** (\_ledgerJournalTrans.AccountType == ledgerjournalACType::Cust)

{

bankChequeTable.RecipientType = BankChequeRecipientType::Cust;

}

**else** **if** (\_ledgerJournalTrans.AccountType == ledgerjournalACType::Ledger)

{

bankChequeTable.RecipientType = BankChequeRecipientType::Ledger;

}

bankChequeTable.ChequeStatus = chequeStatus::Payment;

bankChequeTable.TransDate = \_ledgerJournalTrans.TransDate;

bankChequeTable.Voucher = \_ledgerJournalTrans.Voucher;

bankChequeTable.RecipientCompany = **curext**();

*// bankChequeTable.RecipientAccountNum = \_ledgerJournalTrans.DEL\_AccountNum;*

*// bankChequeTable.BankNegInstRecipientName = \_ledgerJournalTrans.EmployeeName;*

bankChequeTable.RecipientTransVoucher = \_ledgerJournalTrans.Voucher;

bankChequeTable.CurrencyCode = \_ledgerJournalTrans.CurrencyCode;

bankChequeTable.AmountCur = \_ledgerJournalTrans.AmountCurDebit;

**if** (BankChequeTable::exist(\_accountID, \_chequeNum))

{

checkFailed(**strfmt**("Unable to create the check %1, check already exists", bankChequeTable.ChequeNum));

**throw** error(**strfmt**("@SYS18447"));

}

bankChequeTable.insert();

}

*// Added by SBS - VT on 10 Feb 2014 - End*

**Method : main**

**static** **void** main(Args args)

{

dev\_ABMAPJournalImport objdev\_ABMAPJournalImport\_delAll;

dev\_ImportJournal dev\_ImportJournal = dev\_ImportJournal::construct();

dev\_ImportJournal.Delete\_UsageData();

**delete\_from** objdev\_ABMAPJournalImport\_delAll **where** objdev\_ABMAPJournalImport\_delAll.Journal\_UserID == **curUserId**();

**if** (args.record())

**if** (args.record().TableId == **tablenum**(LedgerJournalTable))

dev\_ImportJournal.parmLedgerJournalTable(args.record());

**if** (dev\_ImportJournal.prompt())

dev\_ImportJournal.run();

}