# Check All Accounts

## Check Single Account

Requires the Trader Account code.

Clear down the History for this account and retrieve all the transactions (excluding SQU, PQU, WOR, WIN, JCT, JST, JPT, JSA, JPA, SRN, PRN), using stored procedure isp\_GetAccountTransactions\_ResetHistory.

For each transaction retrieved:

For all transactions except SOR, POR, SDN, PDN:

Determine the correct exchange rate to use:

If the Syss.TotalConv field (in EXCHQSS, against Prefix ‘SYS’) holds ‘V’, and the transaction daily rate (DOCUMENT.thDailyRate) is not zero, use the transaction daily rate.

If the Syss.TotalConv field holds ‘V’, and the transaction daily rate is zero, use the Daily Rate against the System Currency record matching the transaction currency.

If the Syss.TotalConv field holds ’C’, and the transaction company rate (DOCUMENT.thCompanyRate) is not zero, use the transaction company rate.

If the Syss.TotalConv field holds ‘C’, and the transaction company rate is zero, use the Company Rate against the System Currency record matching the transaction currency.

Calculate the Balance, converting to base currency [CurrncyU.ConvCurrITotal]()].

BasicBalance [Real48] :=

Round(Conv\_TCurr(DOCUMENT.thNetValue,

ExchangeRate,

DOCUMENT.thCurrency,

0,

False), 2)

+ Round(Conv\_TCurr(DOCUMENT.thTotalVAT,

DOCUMENT.thDailyRate,

DOCUMENT.thCurrency,

DOCUMENT.thUseOriginalRates,

False), 2)

- Round(Conv\_TCurr(DOCUMENT.thTotalLineDiscount,

ExchangeRate,

DOCUMENT.thCurrency,

0,

False), 2)

+ Round(DOCUMENT.thVariance, 2)

+ Round(DOCUMENT.thReValueAdj, 2)

- Round(Conv\_TCurr(DOCUMENT.thSettleDiscAmount,

ExchangeRate,

DOCUMENT.thCurrency,

0,

False) \* Ord(DOCUMENT.thSettleDiscTaken), 2)

+ Round(DOCUMENT.PostDiscAm, Dp);

Balance := BasicBalance \* DocCnst[DOCUMENT.thDocType] \* DocNotCnst;

Calculate the Cost, converting to base currency.

BasicCost [Real48] := Round(Conv\_TCurr(DOCUMENT.thTotalCost,

ExchangeRate,

DOCUMENT.Currency,

0,

False),2);

Cost := BasicCost \* DocCnst[DOCUMENT.thDocType] \* DocNotCnst;

Calculate the Net Value, converting to base currency.

BasicNet [Real48] := DOCUMENT.thNetValue

- DOCUMENT.thTotalLineDiscount

- (DOCUMENT.thSettleDiscAmount\*Ord(DOCUMENT.thSettleDiscTaken));

BasicNet := Round(Conv\_TCurr(BasicNet, ExchangeRate, DOCUMENT.thCurrency, 0, False), 2);

Net := BasicNet \* DocCnst[DOCUMENT.thDocType] \* DocNotCnst;

Post the Balance, Cost, and Net to History

BalanceToPost [Real48] := Balance

CostToPost [Real48] := Cost

NetToPost [Real48] := Net

if (BalanceToPost < 0):

Sales [Real48] := ABS(BalanceToPost);

Purchases [Real48] := 0;

else

Sales := 0;

Purchases := BalanceToPost;

// For direct payments write to the other side as well

If (DOCUMENT.thDocType In [SRF,SRI,PRF,PPI]):

if BalanceToPost < 0:

Purchases := ABS(BalanceToPost)

else

Sales := ABS(BalanceToPost);

If the transaction is posted (thRunNo > 0) :

Post Purchases and Sales to History and YTD History, against ‘U’ + Trader Code

If the transaction is one of SIN, SCR, SRF, SRI, SJI, SJC, SDN, PIN, PDN, PCR, PRF, PPI, PJI, PJC:

Post Net Value and Cost to History against ‘W’ + Trader Code

Post Net Value and Cost to YTDNCF History against ‘W’ + Trader Code

Post Purchases and Sales to History and YTD History, against ‘V’ + Trader Code

Post Purchases and Sales to History and YTD History, agaisnt ‘V’ + Trader Code + Control G/L

If the transaction is unposted (thRunNo <= 0) :

Post Purchases and Sales to History and YTD History, against ‘U’ + Trader Code

Post Purchases and Sales to History and YTD History, against ‘U’ + Trader Code + Control G/L

If the transaction is one of SIN, SCR, SRF, SRI, SJI, SJC, SDN, PIN, PDN, PCR, PRF, PPI, PJI, PJC:

Post Net Value and Cost to History against ‘W’ + Trader Code

Post Net Value and Cost to YTDNCF History against ‘W’ + Trader Code

Add the Control G/L Code to a list of Control G/L Codes so that we can update the affected G/L records at the end of the routine

If this transaction is one of SRF, SRI, PRF, or PPI, and there is an outstanding balance (less any settled amount on this transaction), OR if thOutstanding is CHAR(0), if thUntilDate holds CHAR(255), clear thUntilDate.

if ((LInv.InvDocHed in DirectSet) and (BalanceToPost – DOCUMENT.thAmountSettled <> 0)) or

((Linv.AllocStat = #0) and (LInv.UntilDate = #255)) then

begin

if LInv.UntilDate = #255 then

LInv.UntilDate := '';

end;

Keep track of the date of the oldest due Transaction for Document types SIN, SJI, SRI, SQU, SOR, SDN, SBT, PIN, PJI, PPI, PQU, POR, PDN, PBT

If ((DOCUMENT.thDueDate <> '') and (DOCUMENT.thDueDate < OldestDueDate) and

(BalanceToPost \* DocCnst[DOCUMENT.thDocType] \* DocNotCnst > 0)) and

(DOCUMEMT.thDocType In SalesSplit+PurchSplit-RecieptSet-CreditSet) then

OldestDueDate := DOCUMENT.thDueDate;

For Sales/Purchase Order or Delivery Transactions (SOR,POR,SDN,PDN), where the Run Number is -40 or -50, post the outstanding balance to History:

Calculating the outstanding balance:

If the System flag IncludeVATInCommittedBalance is False:

Use DOCUMENT. thTotalOrderOS, coverted to base currency.

Otherwise:

If DOCUMENT.thManualVAT is True:

Calculate the total, as DOCUMENT.thNetValue – DOCUMENT.thTotalLineDiscount. If there is a settlement discount value (DOCUMENT.thSettleDiscAmount) then subtract this as well.

If this is not zero, and DOCUMENT.thTotalOrderOS is not zero, the outstanding balance is DOCUMENT.thTotalOrderOS divided by this value (round to 2 decimal places), otherwise the outstanding balance is simply DOCUMENT.thTotalOrderOS.

Otherwise:

Add up the line totals of all the lines against the transaction.

Calculating the line total:

1. Quantity Outstanding: Calculate the quantity outstanding on the line by adding DETAILS.tlQtyDel to DETAILS.QtyWOff, and then subtracting the result from DETAILS.tlQty. If DETAILS.tlUsePack is True then multiply the quantity outstanding by DETAILS.tlQtyMul.
2. Net Value: If DETAILS.tlPriceMultiplier is not zero, then the Net Value is DETAILS.tlNetValue multiplied by DETAILS.tlPriceMultiplier, otherwise it is simply DETAILS.tlNetValue.
3. Calculate the Price Per Unit:
   1. If DETAILS.tlPrxPack is True, and DETAILS.tlQtyPack is not zero, and DETAILS.tlQtyMul is not zero:
      1. If DETAILS.tlShowCase is True, then the Price Per Unit is the Net Value (as calculated by step 2 above), and the Quantity Outstanding (as calculated by step 1 above) should be recalculated by adding DETAILS.tlQtyDel to DETAILS.QtyWOff, subtracting the result from DETAILS.tlQty, then dividing the result by DETAILS.tlQtyPack.
      2. If DETAILS.tlShowCase is False, then the Price Per Unit is DETAILS.tlQtyMul divided by DETAILS.tlQtyPack, multiplied by the Net Value (as calculated by step 2 above).
   2. If DETAILS.tlPrxPack is False, or DETAILS.tlQtyPack is zero, or DETAILS.tlQtyMul is zero, then the Price Per Unit is the same as the Net Value (as calculated by step 2 above).

Once the outstanding balance has been calculated, multiply it by the Document Type multiplier, then multiply the result by -1.0 (to reverse the sign). [Real48]

If the transaction actually has a non-empty Trader Code, post the outstanding balance to the ‘U’ History and YTD History for this Trader Code.

If CUST.acInvoiceTo is non-empty, locate the matching Trader Account, and if this account has an acOfficeType of 1 then similarly post the outstanding balance to the ‘U’ History and YTD History for this Trader Code.

For JCT transactions with a DOCUMENT.thRunNo of -110, and for JST transactions with an empty DOCUMENT.thDeliveryNoteRef and a DOCUMENT.thRunNo of -111, post the outstanding balance to History.

Calculate the outstanding balance by subtracting DOCUMENT.tlTotalOrdered from DOCUMENT.tlTotalCost.

Once the outstanding balance has been calculated, multiply it by the Document Type multiplier, then multiply the result by -1.0 (to reverse the sign).

If the transaction actually has a non-empty Trader Code, post the outstanding balance to the ‘U’ History and YTD History for this Trader Code.

If CUST.acInvoiceTo is non-empty, locate the matching Trader Account, and if this account has an acOfficeType of 1 then similarly post the outstanding balance to the ‘U’ History and YTD History for this Trader Code.

Calculate and store the maximum weeks overdue for this Customer/Supplier, based on the oldest due date:

If EXCHQSS.DebtLMode is 1, 3 or 5, the value is actually the number of days overdue, otherwise it is the number of weeks.

Store the result in CUST.acCreditStatus, truncated to a maximum value of 32767.

Recalculate and post the History balances for all the G/L Codes that were found:

For each G/L Code that was found, call the Profit to Date stored procedure to calculate the YTD figures for the last Audit Year (EXCHQSS.AuditYr), and post the results to the History for ‘U’, ‘V’, and ‘W’ against the G/L Code, Audit Year and YTD period.

## XRate Function for Determining the Correct Exchange Rates

Function XRate(TransactionExchangeRates: CurrTypes; UseSystemDailyRate: Boolean; Currency: Byte): Double;

Var

Trate: Double;

Begin

Trate := 0;

If (UseSystemDailyRate) then

Trate := SyssCurr^.Currencies[Currency].CRates[True]

Else

Begin

UseDailyRate := (Syss.TotalConv = ’V’);

If (TransactionExchangeRates[UseDailyRate] <> 0) then

Trate := TransactionExchangeRates[UseDailyRate]

else

Trate := SyssCurr^.Currencies[Currency].CRates[False];

End;

Result := Trate;

end;