# Allocation Transaction List

*Last edited: 17 April 2015*

## Allocation Wizard Overview

When the user adds a new Sales Receipt or Purchase Payment via the Add button from the Sales/Purchase Daybook, the Allocation Wizard is launched to guide them through the process (note that there is a user-profile setting which disables this, but by default the use of the wizard is enabled), and to create a list of transactions that can be allocated against. This list is stored in EXSTKCHK.

The Wizard allows the user to enter the basic details required, in particular the Trader code, the Amount available for allocation, and the currency. It also allows them to specify the order in which the available transactions are listed (by default they are listed by transaction date).

The creation of the list is mainly done from TScanAlloc.Alloc\_Scan in AllocS2U.pas.

This basically searches for transactions against the selected Trader and adds an entry into EXSTKCHK, against a prefix and subtype to ‘X’ and ‘A’, for each transaction that can be allocated to, assigning the appropriate values to the fields, based on the transaction and the details supplied by the user.

## Generating the Transaction List

The Allocation Wizard creates an allocation header record in the AllocWizardSession table, copying the details from the dialog. This is not part of creating the transaction list, but the header record is subsequently used to obtain details which are required when generating the list. It can be located using the arcCustSupp flag (‘C’ or ‘S’) plus the Trader Code in arcCustCode – there should only every be one matching record.

1. If there was a previous Allocation list against this Trader, and the user has asked to create a new list, all the records in EXSTKCHK which are against a Prefix and Subtype of ‘X’ and ‘A’, and against the matching arcCustSupp and arcCustCode in the AllocWizardSession table, are located and deleted.
2. (*TScanAlloc.Alloc\_CalcDoc*) Find all the Transactions against the Trader, where the thOutstanding flag is 'S' (if the Trader is a supplier) or 'C' (if the Trader is a Customer). If we are not allocating in consolidated currency (currency 0), only include transactions which are in the same currency.
   1. Only include a record if:
      1. There is an outstanding balance on this transaction.
         1. For SQU and PQU transactions, always assume there is no outstanding balance.
         2. If DOCUMENT.thNomAuto is False, always assume there is no outstanding balance.
         3. For all other transactions, use the same Balance calculation that is used by Check All Accounts and accept non-zero results.
      2. The document is not SRC or PPY, OR the balance is negative on an SRC or positive on a PPY.
      3. This transaction is not the PPY being matched to. This is only relevant if we are allocating from a Trader ledger (and thus using an already-existing transaction), rather than through the Allocations Wizard.
   2. Prepare a new Misc (EXSTKCHK) record, with a Prefix and Subtype of XA.
   3. Assign the basic details:
      1. Assign CUSTSUPP.acCustCode to EXSTKCHK.ariCustCode
      2. Assign CUSTSUPP.acCustCode to EXSTKCHK.ariCustCode
      3. Assign CUSTSUPP.acCustSupp to EXSTKCHK.ariCustSupp
      4. Assign DOCUMENT.thCurrency to EXSTKCHK.ariOrigCurr
   4. Calculate and assign the total value in transaction currency to EXSTKCHK.ariOrigVal:
      1. Calculate basic value:
         1. DOCUMENT.thInvNetVal + DOCUMENT.thInvVat - DOCUMENT.thDiscAmount - (DOCUMENT.thDiscSetAm \* Ord(DOCUMENT.thDiscTaken))
      2. If DOCUMENT.thDiscTaken is true, subtract DOCUMENT.thDiscSetAm
      3. Multiply the result by the multiplier for this transaction type (DOCUMENT.thDocType) and then multiply by -1.0 to invert the sign.
      4. Assign the result to EXSTKCHK.arOrigVal
   5. Calculate and assign the transaction total value in base currency:
      1. ariBaseEquiv = ConvCurrItotal(LInv, BOff, BOn, BOn) \* DocCnst[DOCUMENT.thDocType] \* DocNotCnst
   6. Apply any settlement discount (*AllocForceInvSDisc(LInv, EInv, AllocCtrl, 0*):
      1. If the discount has been taken (DOCUMENT.thDiscTaken is True), there is nothing to be done.
      2. If AllocWizardSession.arcIncSDisc is False, there is nothing to be done.
      3. Otherwise:
         1. Calculate the due date, based on DOCUMENT.thTransDate, DOCUMENT.thSettleDiscDays, and AllocWizardSesssion.arcArcSDDaysOver.
         2. If the due date is later than today then take DOCUMENT. thSettleDiscAmount as the Discount amount.
         3. Using the Discount amount, calculate DOCUMENT.PostDiscAm
         4. Note that some of the following actions are based on the Transaction details *before* the discount is applied.
   7. Calculate and assign the discounted outstanding amount in base currency to EXSTKCHK.ariBaseOS:
      1. ariBaseOS = BaseTotalOS(EInv)
   8. For multi-currency versions:
      1. Calculate and assign the discounted outstanding amount in the transaction currency:
         1. ariCurrOS = CurrencyOS(EInv, BOff, BOff, BOff)
      2. If the currency (ariOrigCurr) matches the allocation header currency (arcPayCurr), store the outstanding amount in the transaction currency (as calculated above)
         1. ariOutStanding = ariCurrOS
      3. If the currency (ariOrigCurr) does not match the allocation header currency (arcPayCurr), convert the total from base to the prospective Sales Receipt currency
         1. ariOutStanding = Currency\_ConvFT(ariBaseOS, 0, arcPayCurr, UseCoDayRate)
      4. Determine the value to use in the sort key:
         1. If the currency on the transaction matches the currency on the prospective Sales Receipt, use the amount converted to the transaction currency (ariCurrOS)
         2. If the currency on the transaction does not match the currency on the prospective Sales Receipt, use the outstanding amount in the previously-determined currency (ariOutStanding)
      5. Assign ariOrigReValAdj, as DOCUMENT.thReValueAdj \* DocCnst[DOCUMENT.thDocType] \* -1.0
   9. For single-currency versions, assign EXSTKCHK.arBaseOS to EXSTKCHK.ariOutStanding, and use EXSTKCHK.arOutStanding as the value to use in the sort key.
   10. Set the sign multiplier based on whether this is a Debit or a Credit:
       1. If the DOCUMENT.thDocType is one of SRC,PPY,SRF,PRF,SCR,PCR,SJC,PJC and the line total before accounting for settlement discount is less than zero, use a multiplier of -1.0.
       2. Otherwise:
          1. Calculate the Balance on the transaction (see Appendix 1) and subtract DOCUMENT.thAmountSettled.
          2. If we are dealing with Sales Receipts, and the Balance is less than zero, or if we are dealing with Purchase Payments, and the Balance is greater than zero, use a multiplier of -1.0.
       3. If neither of the above sets the multiplier, set the multiplier to 1.0.
   11. Determine the sort key and assign it to EXSTKCHK.ariKey:
       1. The sort key starts with EXSTKCHK.ariCustSupp followed by EXSTKCHK.ariCustCode padded with spaces to 6 characters.
       2. This is followed by one or two sort types, as selected by the user on the second page of the Allocations Wizard dialog. The sort types are stored in AllocWizardSesssion.arcSortBy and AllocWizardSesssion.arcSort2By. If the two types are the same, then the second one is ignored. Construct a string based on the sort type (if there are two different types then create two strings, one for each type, and concatenate them):
          1. Sort Type 0: use DOCUMENT.thTransDate. If AllocWizardSesssion.arcSortD is true, then invert the ASCII value of the string to force the correct sort order.
          2. Sort Type 1: use DOCUMENT.thDueDate. If AllocWizardSesssion.arcSortD is true, then invert the ASCII value of the string to force the correct sort order.
          3. Sort Type 2: Use the sort value calculated above (2.8.4 and 2.9), multiplied by the Document Type multiplier, and then multiplied by -1.0 to invert the sign. If AllocWizardSession.arcSortD is true, then invert the value to force the correct sort order. ***Note****: the existing Exchequer routine for creating this type of sort key is flawed and does not produce the correct results. For SQL we will determine a correct way of doing it, possibly based on the system used by Sort Views.*
          4. Sort Type 3: use DOCUMENT.thYourRef. If AllocWizardSesssion.arcSortD is true, then invert the ASCII value of the string to force the correct sort order.
          5. Sort Type 4: use DOCUMENT.Currency, converting it to a string. If AllocWizardSesssion.arcSortD is true, then invert the ASCII value of the string to force the correct sort order.
          6. Sort Type 5: use DOCUMENT.OurRef. If AllocWizardSesssion.arcSortD is true, then invert the ASCII value of the string to force the correct sort order.
       3. Assign the resulting string to ariKey (note that this is actually the exstchkvar1 field).
   12. Concatenate EXSTKCHK.ariCustSupp, EXSTKCHK.ariCustCode (padding to a length of 6), and DOCUMENT.thOurRef, as assign the result to Spare2K (this is actually the exstchkvar2 field).
   13. Assign the remaining details:
       1. Assign DOCUMENT.thOurRef to EXSTKCHK.ariOurRef
       2. Assign DOCUMENT.thYourRef to EXSTKCHK.ariYourRef
       3. Assign DOCUMENT.thDueDate to EXSTKCHK.ariDueDate
       4. Assign DOCUMENT.thTransDate to EXSTKCHK.ariTransDate
       5. Assign DOCUMENT.thAmountSettled to EXSTKCHK.ariOrigSettle
       6. Assign DOCUMENT.thCurrSettled to EXSTKCHK.ariOrigOCSettle
       7. Assign DOCUMENT.thCompanyRate to EXSTKCHK.ariCompanyRate
       8. Assign DOCUMENT.thDailyRate to EXSTKCHK.ariDailyRate
       9. Multiply DOCUMENT.thBatchDiscAmount by the sign multiplier calculated by 2.10 above, and assign the result to EXSTKCHK.arOrigSetDisc
       10. Assign DOCUMENT.thDocType to EXSTKCHK.ariDocType
       11. Multiply EXSTKCHK.ariOutstanding by the sign multiplier calculated by 2.10 above.
       12. Multiply EXSTKCHK.ariBaseOS by the sign multiplier calculated by 2.10 above.
       13. Multiply EXSTKCHK.ariCurrOS by the sign multiplier calculated by 2.10 above.
   14. Store the EXSTKCHK record
   15. Get the next Transaction