



TEMENOS™



TEMENOS T24

Subroutine Guide

User Guide

Information in this document is subject to change without notice.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of TEMENOS Holdings NV.

Copyright 2005 TEMENOS Holdings NV. All rights reserved.

**Table of Contents**

Introduction	5
Application Overview	5
Date Routines	6
CDD	6
CDT	6
CFQ	7
DIETER.DATE	7
JULDATE	8
Exchanges Rates	9
CALC.ERATE.LOCAL	9
CUSTRATE	9
EXCHRATE	11
FCY.CONVERT.CHECK	12
MIDDLE.RATE.CONV.CHECK	13
OFS.GLOBUS.MANAGER	15
File Handling	16
OPF	16
EB.CLEAR.FILE	16
F.DELETE	17
F.MATREAD	17
F.MATREADU	18
F.MATWRITE	19
F.LIVE.MATWRITE	19
F.READ	20
F.READU	21
F.WRITE	22
F.LIVE.WRITE	22
LCK	23
TRANSACTION.ABORT	23
Validation	24
DUP	24
FT.NULLS.CHK	24
STORE.END.ERROR	25
LIMIT.GET.PRODUCT	25
Template Programming	26
REBUILD.SCREEN	26



Overrides	27
STORE.OVERRIDE	27
Translation	28
APP.STATIC.TEXT	28
TXT	28
Messages.....	29
ERR	29
OVE	29
Generic	30
B.UPDATE.BATCH	30
CALCULATE.CHARGE.....	30
EB.GET.ACCT.BALANCE.....	32
E.GET.LOCAL.AMT	34
E.GET.STMT.NARRATIVE	34
EB.FORMAT.RATE.....	35
EB.LOCREF.SETUP	35
EB.READLIST	36
EB.ROUND.AMOUNT	36
FATAL.ERROR	37
GET.NARRATIVE	37
GET.SETTLEMENT.DEFAULTS	38
GET.STANDARD.SELECTION.DETS	39
LIMIT.CURR.CONV	40
LOAD.COMPANY	40
MSK.....	41
PRO.....	41
SC.CALC.YIELD	42
EB.GETFIELD	42
Financial Update Routines.....	44
ACCOUNT.SUSPENSE	44
LIMIT.CHECK.....	44
Printing.....	49
PST.....	49
PRINTER.ON	49
PRINTER.OFF	49
PRINTER.CLOSE	50
RG.GET.LOCAL.TEXT.....	50



SPOOL.REPORT	51
IN2 Routines	52
Introduction.....	52



Introduction

Application Overview

This guide contains a list of the subroutines used most frequently by T24 applications and those which are most likely to be of use to developers on client sites. Wherever a subroutine is provided to perform a specified task, the subroutine provided should be used.

Temenos reserves the right to modify the following routines as and when required. However, where possible, any such changes and the impact upon the use of the subroutines will be documented.



Date Routines

CDD

Description Calculates the difference, in days, between two supplied dates

Arguments (REGION, DATE1, DATE2, DAYS)

Incoming

REGION	Region code
DATE1	Start date, YYYYMMDD
DATE2	End date, YYYYMMDD
DAYS	'W' to indicate working days 'C' to indicate calendar days

Returned

DAYS	The number of days difference, + or -
------	---------------------------------------

CDT

Description Calculates forward or previous date from a start date using the number of days.

Arguments (REGION, DATE, DAYS)

Incoming

REGION	Region code
DATE	Start date, YYYYMMDD
DAYS	The number of days forward in the format + or -, the number of days, working or calendar eg. +99C, -6W, +66W+3C

Returned

DATE	The calculated date
ETEXT	Error message if unsuccessful

**CFQ**

Description: Calculates the next date from today's date and a frequency code.

Arguments: COMI from I_COMMON

Incoming:
COMI YYYYYMMDDXXXXX where XXXXX is the frequency code.

Returned:

COMI YYYYYMMDDXXXXX the calculated date and the frequency code.

DIETER.DATE

Description: This routine converts dates from the format used throughout GLOBUS (YYYYMMDD) to the standard jBASE format (either internal or external. External dates formatted as per jBASE CONVERSIONs)

Arguments: GLOBUS.DATE, jBASE.DATE, CONVERSION

Incoming:

GLOBUS.DATE Date in Standard T24 Format (yyyymmdd)

JBASE.DATE The date in internal jBASE date format, or formatted according to CONVERSION

CONVERSION Any valid jBASE Date conversion

Note that both GLOBUS.DATE and JBASE.DATE cannot be passed. One date is mandatory.

Returned:

GLOBUS.DATE Date in standard T24 format (yyyymmdd)

JBASE.DATE The date in internal jBASE date format, or formatted according to CONVERSION

**JULDATE****Description:**

Convert from Julian to Gregorian date format and vice versa

Arguments:

GREGORIAN.DATE,JULIAN.DATE

Incoming:

GREGORIAN.DATE

Standard T24 Date in the format yyymmdd

JULIAN.DATE

Julian Date in the format yyyyjjj

One of the above dates must be passed to the subroutine; the other date must be null.

Returned:

GREGORIAN.DATE

Standard T24 Date in the format yyymmdd if passed as null

JULIAN.DATE

Julian Date in the format yyyyjjj if passed as null



Exchanges Rates

CALC.ERATE.LOCAL

Description: This routine calculates an exchange rate given a foreign amount and local currency equivalent.

Arguments: YLCL.AMT, YFOR.CCY, YFOR.AMT, YEXCH.RATE

Incoming:

YLCL.AMT	Local equivalent amount
YFOR.CCY	Foreign currency
YFOR.AMT	Foreign Currency amount

Returned:

YEXCH.RATE	The derived exchange rate
------------	---------------------------

CUSTRATE

Description: Perform all the tasks involved in foreign exchange of two currencies on a buy/sell basis

Arguments: CCY.MARKETS, BUY.CCY,BUY.AMT,SELL.CCY,SELL.AMT, BASE.CCY,TREASURY.RATE,CUST.RATE,CUST.SPREAD, SPREAD.PCT,LOCAL.CCY.BUY,LOCAL.CCY.SELL, RETURN.CODE

Incoming:

CCY.MARKETS	Is a multi-valued field containing the markets for the buy and sell side. This parameter needs be multi-valued only for Multi Market. For all cases where the buy and sell markets are the same, CCY.MARKETS should be a single value only. In case of two markets, the format should be: BUY.CCY.MKT:VM:SELL.CCY.MKT.
BUY.CCY and SELL.CCY	These are the currencies that will be involved in the transaction
BUY.AMT and SELL.AMT	These are the amounts that will be involved in the transaction. Only one is passed to the routine. The other is calculated from it and the



	customer rate
BASE.CCY	This is the currency in terms of which the RATES will be expressed
TREASURY.RATE	This is the rate at which the TREASURY will exchange the currencies
CUSTOMER.RATE	The rate at which the currencies will be exchanged for the customer
CUSTOMER.SPREAD	The difference between the TREASURY.RATE and the CUSTOMER.RATE
SPREAD.PCT	The percentage of the default CUSTOMER.SPREAD to be used in calculating the CUSTOMER.RATE

Of the above parameters, only CCY.MARKETS, BUY/SELL CCY and one amount are mandatory. The other amount must be blank and only 1 of the CUSTOMER fields may have a value. NOTE: Only one CURRENCY.MARKET is mandatory.

Returned:

LOCAL BUY/SELL AMTS	The local equivalents of the buy and sell amounts involved in this transaction.
RETURN.CODE	This holds the status of various test performed during the course of the routine. RETURN.CODE is a dynamic array composed of 4 fields, which are as follows: 1) Invalid input parameter - This is a fatal error and the routine will immediately terminate. In such a case, this field is set to one and an explanation of the error returned in ETEXT 2) Negotiable amount exceeded - The field is set to the amount by which the amount entered exceeds the negotiable amount for the relevant currency 3) Fixing date error - This field indicates which of the currencies has a FIXING DATE rate and the current date does not match the date the rate was last fixed. If this field is set then 1 indicates BUY.CCY is in error, 2 that it is the SELL.CCY and 3 that both are in error 4) Suspended market - This indicates that one of the currencies has had quotes suspended whether it be the LOCAL, BUY or SELL CCY. If the LOCAL is suspended, this automatically terminates the routine. For BUY and SELL, the routine is terminated only if the suspension indicator is not set to 'BLOCKED'. If the routine is terminated, a relevant message is returned in ETEXT. If the indicator is set, then its value is returned in RETURN.CODE. Note: Suspension of currencies is not currently available.



EXCHRATE

Description: Performs all the tasks involved in foreign exchange of two currencies

Arguments: CCY.MKT,BUY.CCY,BUY.AMT,SELL.CCY,
SELL.AMT,BASE.CCY,EXCHANGE.RATE,
DIFFERENCE,LCY.AMT,RETURN.CODE

Incoming:

CCY.MKT	The currency market in which the transaction will take place
BUY.CCY and SELL.CCY	The currencies that will be involved in the transaction
BUY.AMT and SELL.AMT	The amounts that will be involved in the transaction
BASE.CCY	The currency in terms of which the EXCHANGE.RATE will be expressed
EXCHANGE.RATE	The rate of exchange between the two currencies

If any of these fields, with exception of BUY/SELL CCY and CCY.MKT are left blank, then they will be filled in by the routine if at all possible

Returned:

DIFFERENCE	The difference between the figures given above and the internally generated BASE.CCY amount
------------	---

LCY.AMT	The amount of the transactions in terms of the local currency
---------	---

RETURN.CODE	The status of various tests performed during the course of the routine. RETURN.CODE is a dynamic array composed of six fields, which are as follows: -
-------------	--

1) Invalid input parameter - This is a fatal error and the routine will immediately terminate. In such a case, this field is set to 1 and an explanation of the error returned in ETEXT

2) Tolerance error - This is the signed percentage value by which the input rate differs from the calculated rate to 2 decimal places. This is only set if the error is greater than that permitted for the BASE.CCY.

3) Negotiable amount exceeded - The field is set to 1 if the BUY.AMT exceeds the NEGOTIABLE AMOUNT for the BUY.CCY. It is set to 2 if this is true for the SELL.CCY and 3 if true for both currencies

**FCY.CONVERT.CHECK****Description:**

Performs the following functions:

Converts a foreign amount to a local amount when a foreign amount is passed but not a local.

Converts a local amount to a foreign amount when a local amount is passed but not a foreign.

Calculates an exchange rate when both a foreign and local amount is passed.

Recalculates foreign and local amounts and returns the difference.

Calculates a percentage exchange rate difference when a rate is passed.

Arguments:

(FAMT, FCY, RATE, RATE.TYPE, LAMT, DIF.AMT, DIF.RATE)

Incoming:

FAMT	Foreign amount
FCY	Foreign currency code, USD etc
RATE	Exchange rate
RATE.TYPE	Middle, buy or sell and the rate number 1, 2 or 3 in the format M1, B2, M3 etc. (default M1)
LAMT	Local amount

Returned:

FAMT	When calculated
LAMT	When calculated



RATE	When calculated
DIF.AMT	The amount difference when calculated
DIF.RATE	The percentage rate difference when calculated

MIDDLE.RATE.CONV.CHECK

Description: Performs the following functions:

Converts a foreign amount to a local amount when a foreign amount is passed but not a local.

Converts a local amount to a foreign amount when a local amount is passed but not a foreign.

Calculates an exchange rate when both a foreign and local amount is passed.

Recalculates foreign and local amounts and returns the difference.

Calculates a percentage exchange rate difference when a rate is passed.

Note: The middle rate is used for all calculations unless a rate is explicitly passed.

Arguments: (FAMT, FCY, RATE, MARKET, LAMT, DIF.AMT, DIF.RATE)

Incoming:

FAMT	Foreign amount
FCY	Foreign currency code, USD etc
RATE	Exchange rate
MARKET	Currency market (default 1)



Subroutine

LAMT Local amount

Returned:

FAMT When calculated

LAMT When calculated

RATE When calculated

DIF.AMT The amount difference when calculated

DIF.RATE The percentage rate difference when calculated



OFS.GLOBUS.MANAGER

Previously when we called the OFS.GLOBUS.MANAGER routine the SENSITIVITY variable is set so if there is some thing in the write cache then the core will not take notice.

Now the SENSITIVITY Variable has been removed from the OFS.GLOBUS.MANAGER routine.

So that if it is called when there is something in the Write cache then the core will force a transaction abort and call a new subroutine CHECK.CACHE.WRITE in the Initialise Para.

New Routine

1.CHECK.CACHE.WRITE

Description:

This is a new routine. This routine returns an argument back. This routine should scan through the entire CACHE i.e. FWF and find if there is any WRITE operation pending on it. If so it should return back immediately with a positive argument

Arguments:

WRITE.PRESENT

Passing Null Value from the OFS.GLOBUS.MANAGER Routine

Incoming:

WRITE.PRESENT- Null Value and Standard Common Variables

Returned

If there is any WRITE operation pending on it, if so it should return back immediately with a positive argument otherwise Null value returned.



File Handling

OPF

Description: File open routine. Handles company modification of file name hence, it must be called for all file opens.

Arguments: (FILE.NAME, FILE.VAR)

Incoming:

FILE.NAME Name of the file. If a company mnemonic is already present then that file is opened. Otherwise the correct mnemonic is determined.

Returned:

FILE.NAME The full file name e.g. FXXX.CUSTOMER.

FILE.VAR File variable.

EB.CLEAR.FILE

Description: Clear a file

This routine will clear a file or deleted selected records from the specified file.

Arguments: (FILE.NAME, FILE.VAR)

Incoming:

FILE.NAME Field 1 = File name, field 2 (optional) = selection arguments

FILE.VAR Open file variable

Returned:



The file name must include the correct prefix so an OPF should previously been used to determine this. The selection arguments should conform to those expected by jBASE, i.e. WITH CUSTOMER.NO = "10001"

F.DELETE

Description:

Deletes a record from a file

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments:

(FL, ID)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key

Returned:

F.MATREAD

Description:

Read a dimensioned array from a file

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments:

(FL, ID, MAT ARY, SIZE, F.FL, ER)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
ARY	Matrix for MATREAD/WRITE

Subroutine

SIZE	Size of array
F.FL	File variable

Returned:

ARY	Dimensioned array (data record)
ER	Error message

F.MATREADU**Description:**

Read a dimensioned array record from a file with a lock

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments:

(FL, ID, MAT ARY, SIZE, F.FL, ER, RETRY)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
ARY	Matrix for MATREAD/WRITE
SIZE	Size of array
F.FL	File variable
RETRY	Retry arguments for lock

Returned:

ARY	Dimensioned array (data record)
ER	Error message

Format of RETRY argument

P msg	Prompt the user with 'msg' if the record is locked. The default if 'msg' is null is 'xxxxx FILE id RECORD LOCKED - RETRY Y/N'
-------	---



R nn xx	Retry xx times with a sleep interval of nn seconds. If xx is not specified the record is continually retried.
I	Ignore the lock (REC or ARY is left as null)
E	Return immediately with an error message
null	Retry continuously with a sleep interval of one second

F.MATWRITE

Description:

Writes a dimensioned array (record) to a file

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments:

(FL, ID, MAT ARY, SIZE)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
ARY	Dimensioned array (record)
SIZE	Size of array

Returned:

F.LIVE.MATWRITE

Description:

Writes a dimensioned array (record) to a file as normal like the F.MATWRITE routine, but the difference being that Audit fields are updated on the LIVE file record and any History updates are done automatically on the \$HIS file record. These updates are performed because this routine is wrapped around a CALL to an external subroutine called AUTH.AND.HIST.WRITE, which does the actual updating.

The other difference being that the INSERT for IO routine processing is no longer in this routine like it is in the F.MATWRITE routine. The INSERT for IO processing is in the routine F.MATWRITE which is called by AUTH.AND.HIST.WRITE and the later routine is called by

**F.LIVE.MATWRITE.**

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines, buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments: (FL, ID, MAT ARY, SIZE)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
ARY	Dimensioned array (record)
SIZE	Size of array

Returned:**F.READ**

Description: Read a record from a file

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments: (FL, ID, REC, F.FL, ER)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
REC	Record buffer
F.FL	File variable

**Returned:**

REC	Data record
ER	Error message

F.READU

Description: Read a record with a lock

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments: (FL, ID, REC, F.FL, ER, RETRY)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
REC	Record buffer
F.FL	File variable
RETRY	Retry arguments for lock

Returned:

REC	Data record
ER	Error message

Format of RETRY argument

P msg	Prompt the user with 'msg' if the record is locked. The default if 'msg' is null is 'xxxxx FILE id RECORD LOCKED - RETRY Y/N'
R nn xx	Retry xx times with a sleep interval of nn seconds. If xx is not specified the record is continually retried
I	Ignore the lock (REC or ARY is left as null)
E	Return immediately with an error message
null	Retry continuously with a sleep interval of one second



F.WRITE

Description:

Writes a record to a file

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.

Arguments:

(FL, ID, REC)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
REC	Record buffer

Returned:

REC	Data record
-----	-------------

F.LIVE.WRITE

Description:

Writes a record to a file as normal like the F.WRITE routine, but the difference being that Audit fields are updated and any History updates are done automatically. These updates are performed because this routine is wrapped around a CALL to an external subroutine called AUTH.AND.HIST.WRITE, which does the actual updating.

The other difference being that the INSERT for IO routine processing is no longer in this routine like it is in the F.WRITE routine. The INSERT for IO processing is in the routine F.MATWRITE which is called by AUTH.AND.HIST.WRITE and the later routine is called by F.LIVE.WRITE.

Application software interacts with the journal processor through a series of IO routines, essentially replacing the standard Data/Basic verbs READ/READU/WRITE etc. These routines buffer the transactions IO until a 'flush' point at the end of the transaction, when the journal is updated and the file writes performed. They must be used in place of 'read and write' statements for all on-line processing.



Arguments: (FL, ID, REC)

Incoming:

FL	File name 'F.xxxx.xxxxx'
ID	Key
REC	Record buffer

Returned:

REC	Data record
-----	-------------

LCK

Description: Displays record locked message

Arguments: TEXT (from I_COMMON)

Incoming:

TEXT	Message to be displayed
------	-------------------------

TRANSACTION.ABORT

Description: Subroutine to cancel the current transaction and clear deferred writes and releases locks.

Arguments: None

Incoming:

Standard variables	common
--------------------	--------

Returned:



Cleared system write cache and lock table (except current transaction id)

Validation

DUP

Description: Checks for duplicate occurrences of values in multi and sub-valued fields at cross-validation.

Arguments: None

Incoming:

R.NEW(x) The current record

AF The field number (x) to be validated

F(x) The array containing the field names and definition as multi and/or sub-valued fields

Returned:

END.ERROR Error flag indicating cross-validation error

ETEXT, T.ETEXT Error message: 'INPUT OR LINE-DELETION MISSING' or 'DUPLICATE'

FT.NULLS.CHK

Description: This routine will check for a particular field, if the field has been expanded with extra multi- or sub-values, and if there are any null values.

Arguments: None

Incoming:

R.NEW(x) The current record

AF The field number (x) to be validated



F(x) The array containing the field names and definition as multi and/or sub valued fields

Returned:

END.ERROR Error flag indicating cross-validation error

ETEXT, T.ETEXT Set if null values- or sub-values are found

STORE.END.ERROR

Description: Routine to store error messages by field, to be displayed at the end of validation processing.

Arguments: AF – from I_COMMON
AV – from I_COMMON
AS – from I_COMMON
ETEXT – from I_COMMON

Incoming:

AF Field number in error
AV Value number in error
AS Sub-value in error
ETEXT Error message

LIMIT.GET.PRODUCT

Description This subroutine is used by applications to validate and default the Limit Reference used in a given transaction. It is called as part of the cross-validation processing.

Arguments (YR.SYSTEM, YCUST.NO, YCCY, YPRODUCT)

Inward

YR.SYSTEM The Limit Parameter record for the application (Optional)



YCUST.NO	The customer number of the transaction (Mandatory)
YCCY	The currency of the transactions (Mandatory)
YPRODUCT	The limit reference and serial number if supplied in the transaction Optional additional fields are: <2> "DEPOSIT" denoting this is a deposit contract or "INFO" denoting this is an information type contract <3> Any hard coded suffix passed from the application. This will be combined with the application name when searching LIMIT.PARAMETER for relevant conditions if supplied
APPLICATION	The common variable contains the name of the application to be looked up in the LIMIT.PARAMETER record.
R.NEW	The content of the current record to be used in the look up of LIMIT PARAMETER

Returned

YR.SYSTEM	The limit parameter applicable to the APPLICATION
YR.PRODUCT	Null – no product found or the value of the limit reference for the Product
ETEXT	Description of the error found

Calling Limit Get Product:

The routine should be called in the cross-validation phase of the contract only.

Template Programming

REBUILD.SCREEN

Description: Invokes MATRIX.ALTER to reset screen pointers for automatically expanded multi-values.

Arguments: None

Incoming:

R.NEW()	The current record
T.FIELDNO<x>	The array containing the current fields used in screen display
T.ENRI<x>	The array containing the associated field enrichments for T.FIELDNO



T.ETEXT<x> The array containing the associated field error messages for T.FIELDNO

Returned:

T.FIELDNO<x> The array containing the current fields used in screen display after the rebuild of screen display

T.ENRI<x> The array containing the associated field enrichments for T.FIELDNO

T.ETEXT<x> The array containing the associated field error messages for T.FIELDNO

Overrides**STORE.OVERRIDE**

Description: Prompts for an override and stores the message in the main file record if the user replies 'Y'.

Arguments: (CURR.NO)
AF – from I_COMMON
AV - from I_COMMON
AS - from I_COMMON
TEXT - from I_COMMON

Incoming:

CURR.NO Current record number (0 clears all previous overrides)

AF Field number to return to if user replies 'NO'

AV Value number to return to if user replies 'NO'

AS Sub-Value to return to if user replies 'NO'

TEXT Override message

Returned:



TEXT User's reply, 'Y' or 'NO'

Translation

APP.STATIC.TEXT

Description: Returns array of application messages from the static text translation file. The messages are defined by the application and their translations (entered by the user) are read in at run time.

Arguments: (MAT ARRAY, KEY)

Incoming:

KEY ID of F.STATIC.TEXT record.

Returned:

ARRAY Array of translated messages.

TXT

Description: Translates dynamic messages. The message can contain a variable portion e.g. RATE 99.9999 EXCEEDS TOLERANCE BY 10%.

Arguments: (MESSAGE)

Incoming:

MESSAGE Message to be translated. If it contains a variable portion then it should be a dynamic array in the following format:

 <1> 'RATE & EXCEEDS TOLERANCE BY &'
 <2> 99.9999 <vm> 10%

Returned:

MESSAGE Translated message



Messages

ERR

Description:

Displays error message on the same line as the input field or on line 22 if running under a multi-line version. Can be used only within the confines of a template program.

Arguments:

E From I_COMMON

ECOMI From I_COMMON

Incoming:

E Error message to be displayed (will be translated)

ECOMI Should contain the user input if it is to be displayed with the error message.

OVE

Description:

Prompts for an override and validates Y/N input.

Arguments:

TEXT (from I_COMMON)

Incoming:

TEXT Prompt to be displayed (will be translated)

Returned:

TEXT 'Y' or 'NO'



Generic

B.UPDATE.BATCH

Description: Adds a batch job to an existing batch process. Enables application programs to initiate ad-hoc jobs in the overnight run.

Arguments: (PROCESS, RUN.DATE, FREQ, PRINTER, DATA, JOB.NAME)

Incoming:

PROCESS	Name of process in batch file
RUN.DATE	Date to run job, YYYYMMDD format
FREQ	Frequency code or null (see the <i>Batch Processing</i> chapter in the <i>System Administration Guide</i>)
PRINTER	Printer for output or null
DATA	Data to be passed to job
JOB.NAME	Name of job to be initiated

Returned:

ETEXT	Error message if unsuccessful
-------	-------------------------------

CALCULATE.CHARGE

Description: Calculates charges, commission and tax amounts and returns enough information to raise the accounting entries.

Arguments: (CUSTOMER, DEAL.AMOUNT, DEAL.CURRENCY,
CURRENCY.MARKET, CROSS.RATE, CROSS.CURRENCY,
DRAWDOWN.CCY, T.DATA, CUST.COND, TOT.CHARGE.LCCY,
TOT.CHARGE.FCCY)

Incoming:



CUSTOMER	The customer who has to pay the charge
DEAL.AMOUNT	The deal amount used for percentage charge calculations
DEAL.CURRENCY	The currency of the deal amount
CURRENCY.MARKET	The market to use for currency conversions
CROSS.RATE	The deal rate used in the transaction (optional)
CROSS.CURRENCY	The other currency involved (optional)
DRAWDOWN.CCY	The currency to return the charges in (default deal currency)
T.DATA<1,x>	A list of charge, commission and tax codes
T.DATA<2,x>	An optional list of code types, CHG, COM, TAX
CUST.COND	Conditions overriding the default charge tables
CUST.COND<1,x>	A percentage of the charge to be levied
CUST.COND<2,x>	A list of currencies and charge amounts to be used instead of that
CUST.COND<3,x>	Defined on the charge tables
CUST.COND<4,x>	The customer to be used instead of CUSTOMER

Returned:

T.DATA<2,x>	The code types CHG, COM, TAX
T.DATA<3,x>	Category or internal a/c to post the charges to
T.DATA<4,x>	The charge amount in local
T.DATA<5,x>	The charge amount in foreign currency
T.DATA<6,x>	The mid rate used between local and foreign
T.DATA<7,x>	The transaction code for the entry
TOT.CHARGE.FCCY	The total charge to debit the drawdown account
TOT.CHARGE.LCCY	Its local equivalent



ETEXT Any errors encountered

Notes:

The term 'charge' is used to denote charges, commissions and tax. The charge codes should be defined on [FT.CHARGE.TYPE](#), [FT.COMMISSION.TYPE](#) and [TAX](#).

If the charge type is not passed then the routine will endeavour to determine the type by reading the CHARGE, COMMISSION and TAX files in sequence.

Tax can be applied on a charge, hence you could get more charges returned than were passed. All identical charge codes are consolidated before returning, i.e. if all four charges had VAT applied then you would only get one VAT amount returned.

When performing ANY currency conversion between the DEAL.CURRENCY and the CROSS.CURRENCY, the CROSS.RATE will be used; otherwise the mid-rate will be used.

The customer conditions are optional and allow you to debit an individual customer a percentage of the normal charge or an agreed flat rate, overriding the charge table definitions. e.g. Specific customers may be charged 50% commission on certain deals.

User input/modified amounts can also be passed in CUSTOMER.CONDITION values 2 and 3, just the tax will be recalculated. Most applications allow the user to override the default charge. This mechanism provides the facility to 'return' the charge amounts back to CALCULATE.CHARGE, which will then recalculate any charge tax.

EB.GET.ACCT.BALANCE**Description:**

This routine will return an account balance of a given type on a given date and can return the balance as at a requested system date too (ie. Without adjustments).

The routine obtains the balances by using the correct ACCT.ACTIVITY record for the date requested.

If a BOOKING.DATE balance is requested for a date less than or equal to the BK.BAL.START.DATE from the DATES file, the routine will return the booking dated balance for the BK.BAL.START.DATE. Since the routine returns the opening balance for a date one calendar day must be added to the requested date to pass into the routine so that the closing balance is returned:

Arguments:

ACCT.ID, ACCT.REC, BALANCE.TYPE, BALANCE.DATE,
SYSTEM DATE BALANCE CREDIT M/M/T CREDIT M/M/T



SYSTEM.DATE, BALANCE, CREDIT.MVMT, CREDIT.MVMT,
DEBIT.MVMT, ERR.MSG

Incoming:

ACCT.ID	Account Number
ACCT.REC	Account record
BALANCE.TYPE	'VALUE', 'TRADE' or 'BOOKING'
BALANCE.DATE	Date balance is required for (YYYYMMDD), will default to Today if left blank
SYSTEM.DATE	Optional date as of system date

Returned:

BALANCE	Returned balance
CREDIT.MVMT	Returned net credit movement for the date
DEBIT.MVMT	Returned net debit movement for the date
ERR.MSG	Any error messages

**E.GET.LOCAL.AMT**

Description: Enquiry Subroutine to calculate the local currency equivalent using MIDDLE.RATE.CONV.CHECK.

Arguments: O.DATA from I_ENQUIRY.COMMON

Incoming:

O.DATA In the format CCY nnnnn where:
CCY is the currency
nnnnn is the amount

Outgoing:

O.DATA Local Equivalent of the incoming amount, calculated using the middle rate

E.GET.STMT.NARRATIVE

Description: Subroutine to return a calculated statement entry narrative for enquiry purposes. The narrative is calculated according the definitions in TRANSACTION, STMT.NARR.FORMAT and STMT.NARR.PARAM

Arguments: None

Incoming:

O.DATA Statement entry id

Returned:

R.RECORD<42> The calculated narrative as a multi-valued field

VM.COUNT The number of multi-values in the narrative

**EB.FORMAT.RATE**

Description: This routine formats rate values depending upon the system settings. If extended precision is on, the maximum no. of decimals allowed is 9 else it is 6. Total no of digits allowed for a rate value is 11.

Arguments: RATE

Incoming:

RATE Rate to be formatted.

Returned:

RATE Formatted rate.

EB.LOCREF.SETUP

Description: This routine will return the local reference data of a specified application.

Arguments: APP.NAME, LOC.REF.ARRAY

Incoming:

APP.NAME The application for which local reference details are required

Returned:

LOC.REF.ARRAY	The array of local reference items for the required application:
LOC.REF.ARRAY<x,1>	Name of local reference field
LOC.REF.ARRAY<x,2>	Maximum length
LOC.REF.ARRAY<x,3>	Local reference type
LOC.REF.ARRAY<x,4>	Vetting table applications
LOC.REF.ARRAY<x,5>	Vetting table enrichments
LOC.REF.ARRAY<x,6>	Default possible
LOC.REF.ARRAY<x,7>	NOCHANGE or NOINPUT indicator

**EB.READLIST**

Description: Routine to return a list of keys from a select statement.

Arguments: (SELECT.STATEMENT, KEY.LIST, LIST.NAME, SELECTED, SYSTEM.RETURN.CODE)

Incoming:

SELECT.STATEMENT Execute statement (optional - select can be active)

KEY.LIST Id list returned, null if error

LIST.NAME Used as part of SAVE.LIST id

Returned:

KEY.LIST Selected list returned

SELECTED Number of records selected

SYSTEM.RETURN.CODE @SYSTEM.RETURN.CODE after select statement. Status codes returned by system processes

EB.ROUND.AMOUNT

Description: Formats amount according to currency definition, by:

- using, by default, two decimal places, OR,
- using the defined number of decimal places for the currency, OR,
- using the lowest unit of currency (e.g. 10, 0.05 etc.) AND rounding according to the lowest unit

Arguments: CURRENCY.ID, AMOUNT, CAL, CUSTOMER.ID

Incoming:

CURRENCY.ID Currency code

AMOUNT Amount

Subroutine

CAL	""	use lowest currency unit rounding
	1	non-cash rounding
	2	cash rounding

CUSTOMER.ID	for future use
-------------	----------------

Returned:

AMOUNT	The amount rounded and formatted according to the currency and CAL specified
--------	--

FATAL.ERROR

Description: Application fatal abort: displays an error message, exits from 'EX' and updates the PROTOCOL file. Should be called for all 'fatal' errors in both on-line mode and batch.

Arguments: (ROUTINE)
TEXT from I_COMMON

Incoming:

ROUTINE	Name of the routine which invoked FATAL.ERROR
---------	---

TEXT	Message to be displayed (will be translated).
------	---

Returned:

The routine does not return to the program: the program aborts.

GET.NARRATIVE

Description: Subroutine to return calculated statement narrative using [TRANSACTION](#), [STMT.NARR.PARAM](#) and [STMT.NARR.FORMAT](#) applications.



Arguments: STMT.ID, STMT.REC, NARR

Incoming:

STMT.ID Id to statement entry record

STMT.REC Statement entry record

Returned:

NARR Derived narrative (multi-value)

GET.SETTLEMENT.DEFAULTS

Description: This routine is used to get the default account and settlement details for a particular customer and application. Defaults may be obtained from Portfolio SEC.ACC.MASTER, AGENCY, CUSTOMER.CCY.ACCT or NOSTRO.ACCOUNT files.

Arguments: YCUST, YCCY, YCCY.MKT, YAPPLN, YPORTFOLIO.NO, YORDER, YACCT, YCB.CUST, YCB.ACCT, YIB.CUST

Incoming:

YCUST The customer for which defaults are required (mandatory)

YCCY The currency for which defaults are required (mandatory)

YCCY.MKT Currency market for which defaults are required (default value 1)

YAPPLN The application for which defaults are required. (Mandatory). This can be passed as a two character application identifier (e.g. MM, LD) and with an optional suffix, separated by "-" (e.g. MM-PRIN)

YPORTFOLIO.NO The portfolio in which the deal resides, a number in the range 1 to 999 (optional)

YORDER The order in which the defaults have to be returned: Any combination of the letters "C"ustomer, "V"ostro, "N"ostro or "P"ortfolio in the order in which the calling



application requires it. For example PVCN will default first from the portfolio, then vostro from agency, then customer ccy acct then nostro Account.

YACCT If incoming argument is null then the account number based on the ORDER.FOR.DEFAULTS is returned. Null if no default is found. If the incoming argument contains an account number, the account is checked to see if it is a Nostro and the beneficiary details are returned and "N" in ORDER.FOR.DEFAULTS. If it is not a Nostro, Null is returned in this argument, ORDER.FOR.DEFAULTS and in the Beneficiary details.

Returned:

YORDER The value from which the account was defaulted. May be P, V, C or N. For example if the defaulted account is obtained from the portfolio, the value P will be returned

YACCT The default account number obtained

YCB.CUST Correspondent bank of customer if settling through a Nostro account customer number

YCB.ACCT Account number of customer's account at correspondent bank when settling using a Nostro account

YIB.CUST Intermediary bank number required where there is no direct relationship between the correspondent bank and the bank when settlement is made via a Nostro account (customer number)

GET.STANDARD.SELECTION.DETS

Description: The routine reads the [STANDARD.SELECTION](#) record and merges the SYS and USR fields together. The routine contains a dimensioned cache of Standard Selection records T.SS.REC and dynamic list of Standard Selection ids T.SS.IDS.

Arguments: STANDARD.SEL.ID, SS.REC

Incoming:

STANDARD.SEL.ID Key to STANDARD.SELECTION record

Returned:



SS.REC STANDARD.SELECTION record with USR and SYS fields merged into the SYS fields.

LIMIT.CURR.CONV

Description: Program to convert a currency amount to another currency, assuming that the call parameters have previously been validated. The currency from is assumed to be the base currency. Xrate is calculated as to.rate/from.rate by this program. This routine will cater for conversion to a currency not defined in the current company.

Arguments: PCCY.FROM, PAMT.FROM, PCCY.TO, YAMT.TO, YPROC.FLAG

Incoming:

PCCY.FROM	Currency to convert from
PAMT.FROM	Currency amount in from currency
PCCY.TO	Currency to convert to
YPROC.FLAG	"SETUP" initialises a common area for the currency "" calculates amount

Returned:

YAMT.TO	Amount in PAMT.TO currency
---------	----------------------------

LOAD.COMPANY

Description: Builds and loads a company profile into I_COMMON. This routine must be called for a change of company.

Arguments: (COMPANY)

Incoming:

COMPANY	The company to be loaded. Must be a key to F.COMPANY
---------	--

**Returned:**

ETEXT	Error message if unsuccessful
-------	-------------------------------

MSK

<u>Description:</u>	Masks input to a specified format.
----------------------------	------------------------------------

<u>Arguments:</u>	(N, T) COMI from I_COMMON
--------------------------	------------------------------

Incoming:

N	'N' parameter from field definition
---	-------------------------------------

T	'T' parameter from field definition
---	-------------------------------------

COMI	Data to reformat
------	------------------

Returned:

DISPLAY	Formatted data
---------	----------------

PRO

<u>Description:</u>	Outputs a message to the protocol file for audit or exception purposes.
----------------------------	---

<u>Arguments:</u>	(MESSAGE)
--------------------------	-----------

Incoming:

MESSAGE	Message to be written to file
---------	-------------------------------



SC.CALC.YIELD

Description: Calculates yields, yield to maturity, duration and modified duration for all bonds and optionally updates SECURITY.SUPP file.

Arguments: SECURITY.ID, CALC.ONLY, and R.SECURITY.SUPP

SECURITY.ID indicates the security to which the calculation applies. If CALC.ONLY and R.SECURITY.SUPP are set to null then the price used to calculate the above will be taken as the last price from the relevant [SECURITY.MASTER](#) record, the relevant [SECURITY.SUPP](#) record will be read and updated. This is the normal situation.

If CALC.ONLY is not set to null then R.SECURITY.SUPP must be populated. This is used to calculate a yield to maturity based on a trade price. The price used to calculate the above will be taken from the **PRICE** field relating to the type "M" **CALL.PUT.MATURITY** field in the passed record if populated, otherwise it will default to the SECURITY.MASTER price. The passed record will be updated and returned but will not be written to the database.

EB.GETFIELD

Description: Retrieve a field or a set of fields from the current application record, another record within the same file or from another file using literal field names.

Arguments: (REF.APPLICATION,REF.ID,FIELD.NAME,FIELD.DATA,CACHE,RESERVED02,RESERVED01,FIELD.ERR)

Incoming:

REF.APPLICATION The company to be loaded. Must be a key to F.COMPANY

REF.ID Key of record from which to retrieve. If left blank and REF.APPLICATION is blank, field is retrieved from current record; if left blank and REF.APPLICATION is specified, current record key is used to retrieve from REF.APPLICATION

FIELD.NAME Literal field name as string or dynamic array of field names as strings, FM VM or SVM delimited

CACHE Flag is set to 1 if cache-read is preferred

Returned:

FIELD.DATA Dynamic array of data from FIELD.NAME fields



Subroutine

FIELD.ERR Dynamic array of error codes. Set to 1 for each field which could not be read.

ETEXT Error message if unsuccessful



Financial Update Routines

ACCOUNT.SUSPENSE

Description: Routine to suspend or remove the suspension from an account.

Arguments: ACCOUNT.ID, TYPE,ERR

Incoming:

ACCOUNT.ID Account number

TYPE 'SUSPENSE' or 'SETTLE'

Returned:

ERR Error code if unsuccessful

LIMIT.CHECK

Description: This subroutine is used by applications both online and end of day to verify and update the limits system. The application is required to supply the correct values from the transaction in order to give the correct update to the system.

Arguments: (PLIAB.ORIG, PCUST.NO, PREF.NO, PSER.NO, PTXN.REF.MNE, PTXN.DATE, PTXN.CCY, PTXN.AMT, POTH.CCY.OR.COMMITM, PDEAL.DESK, PACC.CO, PACC.NO, PACC.BAL, PACC.CCY, PCURR.NO, PFIND.REC, PCALL.TIME, PCALL.ID, PRETURN.CODE)

Incoming:

PLIAB.ORIG Contains the liability number of the customer of the limit found in the field **CUSTOMER.LIABILITY** in the [CUSTOMER](#) record. A Customer may belong to a liability group, where the limit will be held both at customer and liability level. This should always be passed, it may be null in the customer record.

PCUST.NO The actual customer number of the transaction that requires the limit to be checked and updated. This is a mandatory field.

PREF.NO This is the 4-7 character [LIMIT.REFERENCE](#) key held on the contract. This is usually stored in the format 4-7N. 2N in the



contract, only the first element should be passed. The content of the LIMIT.REFERENCE should be validated / defaulted by calling the routine LIMIT.GET.PRODUCT.

PSER.NO The 2-digit serial number of the limit record between 01 and 99. This is held in the contract in the **LIMIT.REF** field as the second element of the data.

PTXN.REF.MNE The transaction id of the contract with the company mnemonic appended in a multi-company environment in the format:

Trans Ref | Mnemonic

If no mnemonic is passed the current company is assumed. When called for Account limits, the reference should be passed as Account number.

PTXN.DATE The maturity date of the contract should be passed here. For call / notice contracts a number of days can also be passed. For accounts do not pass any date.

PTXN.CCY The currency of the movement / balance of the transaction to be checked and update the limits system. This should always be the currency of the contract or account.

PTXN.AMT This is the amount of the transaction in PTXN.CCY. It should be a signed amount. For foreign exchange transactions a second field is present containing the other amount of the deal.

POTH.CCY.OR.COMMITM

This argument is only used for Forex or Commitment contracts. For FX deals the other (sell) currency of the transaction should be passed, this corresponds to the amount in PTXN.AMT<2>. For loan transactions linked to a commitment this describes how the limit is to be impacted with regard to the commitment.

Possible values are:

Y – Commitment contract

YR – Revolving Commitment contract

YN – Non revolving commitment contract

NR – Do not reduce online limit

PDEAL.DESK The dealer desk of the transaction, for future use.

PACC.CO When called from accounting to check the impact of an overdraft the company of the account passed in PACC.NO should be supplied. If not present, the current



company is assumed.

PACC.NO When processing limits for accounts the account number must be passed that the movement to be checked is to be applied to.

PACC.BAL When processing for an account, i.e. PACC.NO is supplied, this should contain the account balance PRIOR to the transaction. The transaction amount is passed in PTXN.AMT

PACC.CCY When processing for an account, this should contain the currency of the account number supplied in PACC.NO.

PCURR.NO This should be passed with the number of overrides in the Override field in the application. The process will update R.NEW(V-9) in the next position. A value of zero will reset the overrides.

PFIND.REC This should be left as null if the application is required to allow the creation of default limit records. If only predefined limits are allowed, a value of Y should be passed.

PCALL.TIME This controls the method of processing overrides on-line and Close of Business;

Possible values:

Null	Normal on-line process, override requires response
U	Used in BATCH mode, always update overrides
E	Used in Batch, return if error is found. Error is returned in ETEXT
V	Verify limit online - no update (not used)
B	Verify limit in batch mode – no update (not used)

PCALL.ID This describes to the limit check routine, the type of processing made to the application. It can have the following values:

VAL New values are being passed and must be used to update limits

DEL The values passed are the old values that should be removed

PERC REVAL Special mode used in the Close of Business to signify revaluation processing.

R.NEW()
[Common] Should contain the contract. The overrides will be updated here.

ID.NEW
[Common] The transaction id.



V [Common] The size of the current application record

Returned:

PRETURN.CODE	Return code describing the limit process
5	No line allocated
6	Line not available
7	Line unavailable
8	Line unavailable to this customer
9	Line unavailable to this currency
10	Line unavailable to this company
11	Not used
12	Excess
13	Not used
14	Clean Risk Excess
15	Insufficient time bands for transaction
16	Commitments are revolving with a non revolving limit or vice versa

How to call LIMIT.CHECK

The limit check subroutine should be called at the unauthorised stage of transaction process (i.e. in the BEFORE.UNAUTH.WRITE section of the template). It should be called AFTER any call to the accounting process (EB.ACCOUNTING) to avoid any potential locking order conflicts.

New transactions

The new values of currency, amount and maturity date should be passed with a PCALL.ID value of "VAL". This will add the new unauthorised values of the transaction.

Deletion of unauthorised transactions

The old values of currency and maturity date should be passed with a PCALL.ID value of "DEL". Take care to establish the correct old values, these can be found in the following common arrays:

Unauthorised new contract	R.NEW
Unauthorised change authorised	R.NEW.LAST
Unauthorised reversal	R.NEW

**Reversal of authorised transaction**

The old values of currency and maturity date should be passed with a PCALL.ID value of "DEL". Take care to establish the correct old values, these can be found in the following common arrays:

Previously authorised

R.OLD

Change of unauthorised contracts

A call of type DEL must be made first to remove the previous values (see deletion of unauthorised transactions). New values should be supplied with a call type of VAL as per new transactions.

HOLD processing

The HLD function should be disabled (apart from 1st Input of the transaction) to avoid corruption of the limits database. This should be done by adding the value .NOH to the **ADDITIONAL.INFO** field of the application [PGM.FILE](#) record.

Principal Increase

The limits system will usually reflect the worst case scenario, so any principal increase, forward dated or not, should impact the limits system immediately and not during the Close of Business process.

Close of Business processing - Unauthorised

Where Close of Business processing deletes / places in HLD unauthorised transactions, the process must also update the limit system by calling LIMIT.CHECK with DEL.

Close of Business Processing – Maturity

Maturity of contracts should result in a call to LIMIT.CHECK with DEL to remove the transaction from the limit system.

Close of Business Processing – Principal Decrease / Repayment

When the decrease takes place in the Close of Business, the system should call LIMIT.CHECK first to delete the previous values, then to replace with the new principal.



Printing

PST

Description: Opens/closes printer for output and defines headers.

Arguments: (ACTION)

Incoming:

ACTION 'OPEN' to initialise
"" to close
or a dynamic array in the following format:
<1> = 'HEADER' or 'HEADER.80' to define a header for 132 or 80 columns
<2> = Report ID and title (separated by a space)

Returned:

ACTION For header specifications returns heading lines in field one to three of a dynamic array

PRINTER.ON

Description: Sets up the printer environment for the report and directs output to the print unit. This routine must be used instead of the Data/Basic PRINTER ON statement.

Arguments: (NAME, UNIT)

Incoming:

NAME Name of the report as defined on the report control file

UNIT Print unit to which output should be directed. 0 - 254, default 0

PRINTER.OFF

Description: Redirects output from the print unit to the screen. This routine must be used instead of the Data/Basic PRINTER OFF statement.



Arguments: None

PRINTER.CLOSE

Description: Spools report as directed by the report control system. This subroutine must be used in place of the Data/Basic PRINTER CLOSE statement.

Arguments: (NAME, UNIT, USER.NAME)

Incoming:

NAME	Name of the report as defined on the report control file
UNIT	The print unit number. 0 - 254 , default 0
USER.NAME	Used to retrieve addressing information for the report banner. Must be set to blank.

RG.GET.LOCAL.TEXT

Description: Repgen subroutine will get the description from **REMARK** field of a [LOCAL.TABLE](#) record.
[PGM.FILE](#) entry to be used in REPGEN is **RG.LOCAL.TABLE.TEXT**.

Arguments: LOCAL.TABLE.NO, VET.TABLE.ENTRY, LOCAL DESC

Incoming:

LOCAL.TABLE.NO:	The ID of the record in LOCAL.TABLE
VET.TABLE.ENTRY:	The data from which text is required from the REMARK field.

Returned:

LOCAL.DESC	The description as defined in REMARK field Or Null if VET . TABLE . ENTRY is not available in the LOCAL.TABLE
------------	---



SPOOL.REPORT

Description: Spools a report, which has not been created, by the report control system. This is only intended for use with COMO files. This routine must be used instead of executing a spool command.

Arguments: ('' , NAME, TREE.NAME, '', '', '')

Incoming:

NAME Name of the report as defined on the report control file.

TREE.NAME The path name of the COMO file. This must be in the format 'filename>recordname' e.g. &COMO&>TEST.COMO. The filename must be a valid VOC entry.

'' All other arguments must be specified as null (internal use only).



IN2 Routines

Introduction

The IN2 routines are described fully in the *IN2 ROUTINES* user guide.