

FINACLE *7.0*

DBA Training Manual

SCRIPTING EVENTS

ExpertEdge Software

Info.training@cwlggroup.com

Document number:		VersionRev:	1.1
Authorized by:	Olawuwo Shade	Signature/Date:	

Document revision list

Ver.Rev	Date	Author	Description
1.00	01-02-2011	Arije Abayomi	<i>Original version</i>

All rights reserved by

*Expertedge Software Limited,
14c Kayode Abraham Street,
Off Ligali Ayorinde,
Victoria Island,
Lagos.*

No part of this volume may be reproduced or transmitted in any form or by any means electronic or mechanical including photocopying and recording or by any information storage or retrieval system except as may be expressly permitted.

Expertedge believes that the information in this publication is accurate as of its publication date. This document could include typographical errors, omissions or technical inaccuracies. Expertedge reserves the right to revise the document and to make changes without notice.

TABLE OF CONTENTS

1	INTRODUCTION.....	3
1.1	WHAT IS A FINACLE™ ‘SCRIPT EVENT’	4
1.2	REPOSITORIES AND CLASSES.....	5
1.3	RETURN STATUS OF THE SCRIPT	5
2	FORM LOAD EVENTS	8
2.1	IDENTIFYING THE FIELD NAMES.....	8
2.2	PRE LOAD FORM EVENT.....	9
2.3	POST LOAD FORM EVENT.....	10
2.4	POST EXECUTE FORM EVENT	11
2.5	PRE KEY REPLAY EVENT	12
2.6	POST KEY REPLAY EVENT.....	15
3	FINANCIAL TRANSACTION EVENTS.....	17
3.1	ACCOUNT DETAILS DISPLAY IN TM MENU OPTION	17
3.2	CLEARING TRANSACTIONS.....	19
3.3	TRANSACTIONS THROUGH TM.....	24
3.4	OTHER FINANCIAL TRANSACTIONS	33
4	TRANSACTION UPLOAD SCRIPTS.....	35
5	EVENT CHARGES RELATED SCRIPTING EVENTS.....	39
5.1	SCRIPTS FOR CALCULATION OF CHARGES FOR DEMAND DRAFTS	41
6.1	SCRIPTS FOR CALCULATION OF MICR CHARGES	42
6.2	SCRIPTS FOR CALCULATION OF STOP PAYMENT CHARGES.....	43
6.3	SCRIPTS FOR CALCULATION OF ACCOUNT CLOSURE CHARGES.....	44
6.4	MAHA TRANSACTION TEMPLATES.....	46
7	SECTION OBJECTIVE.....	47
8	PROCESSES INVOLVED	47
9	MTTUSERHOOKS.....	47
9.1	DEFINETRANSACTION	47
9.2	DEFINEPARTTRAN.....	49
9.3	POPULATEINPUTDETAILS	55
9.4	ENDDEFINITION	55
9.5	PROGRAMMING HINTS FOR MTT.....	56

10	<i>APPENDIX</i>	56
10.1	FIELDS IN STDIN CLASS IN BANCS REPOSITORY	56

1 INTRODUCTION

Finacle™ is a Core Banking package which supports a number of deployment topologies (Fully Distributed to Fully Centralized) and Implementation scenarios (Single currency, Multiple currency, Various Retail and Trade finance banking products and Interest methods etc.). In order to support these multitude of features, it is essential that Finacle™ is open to 'Customization' by the Bank so that it can implement the features that it requires and in the way that it desires. Also, the Bank will need to interface some of their 'peripheral applications' (like Treasury, Consumer Finance etc.) which are not supported directly by Finacle™, to Finacle™ in various ways. Another area of 'Customizability' is in the interface to various special purpose equipment like ATM switches, VRUs, MICR encoders etc. which are supplied by many vendors with some differences in actual implementation.

So far, Finacle™ had addressed this issue by defining 'parameters' which was set up by the Bank depending upon its requirements. However, this has the limitation of requiring that the developers of Finacle™ know all the possible business logic before hand and then implement them using parameters. However, a far more powerful approach is to allow the Bank to 'take control' at certain 'events', apply their own logic on the data associated with that event and be able to either influence the outcome of that event or communicate the occurrence of the event to other 'Custom' applications. The Finacle™ 'extensibility toolkit' allows the bank to do just this.

The Script Engine based Customization (SEBC) consists of 3 parts:

- 1□ The Script Engine – This is an 'interpretive language' processor which allows a wide range of key programming constructs, while allowing the Bank to 'custom develop' additional functions that can be called from the 'scripts'. Please refer to Scripting Syntax document for more details.
- 2□ The Finacle™ functions – Certain standard functions and certain 'module specific' functions have been developed by the Finacle™ developers which can be called from the 'scripts'. Please refer to Scripting Hooks document for more details.
- 3□ The Finacle™ 'script events' – Right across Finacle™, there are several events that have been 'opened up' by defining script events. What it means is that for a 'script-enabled event', the Finacle™ application, checks for the existence of a defined script and if present, transfers control to the Script Engine to execute that script after populating all the data required by that event in 'input fields'. It is the script's responsibility to apply whatever logic it wants and provide the necessary 'output fields' at the end of the script for Finacle™ to continue its processing. This document contains more details about this.

1.1 WHAT IS A FINACLE™ ‘SCRIPT EVENT’

Scripting events are supported in FINACLE™ for the Bank to enable the following kinds of customization:

- 1□ Manipulate the data that is given as input (event specific ones and not the ones in STDIN class) and provide the output as needed by Finacle™ to continue processing. Basically, these events occur at the interface between ‘applications’ or ‘vendor-specific’ devices. These events provide an easy way to take care of ‘formatting’ differences between different versions/applications. The Account formatting events are an example of this, in which the Bank is expected to apply its specific logic to convert an externally known account number to the internally known one and vice versa. Also, events that allow massaging of messages to and from **BancsConnect** fall into this category.
- 2□ Manipulate menu options, screen variables, and key-based screen events to customize the screens and workflow. Note that for the full functionality to be available, the Bank has to install the Workflow module of Finacle™. The Form Load and Post-Execute events are examples of this.
- 3□ At various points during different business transactions, Transaction creation events (MTT event) are generated which allows the Bank to customize the transaction that needs to be posted. These would typically be used for defining charge transactions, CDCI-related transactions, interest provisioning and interest effecting transactions etc.
- 4□ Apart from this there are a number of events that have been provided with ‘extensibility’ in mind. These events allow the bank to ‘communicate’ data of Finacle™ events to other applications and vice-versa. For example the ‘Customer creation/modification’ event allows the Bank to keep the Customer data in Finacle™ in synch with a ‘Central Customer master’ if any.

Some examples:

- 1□ The bank decides the account number format. But this format may contain some format that is common for all the accounts in the Service Outlet and the user wants the same to be appended to the account number entered in some particular format by default. It might be the Bank code, Branch code, Currency code, currency alias, Data center alias etc. which are common to all accounts.

The account number in the database might be 01-02-03-04-0001

Where 01 - corresponds to Bank code, 02 - corresponds to Branch code, 03 - corresponds to the currency alias, 04 - corresponds to the scheme code and 0001- is the actual account number.

The format of the account is such that the user has to enter ‘-’ in between the account number which is very cumbersome. By making use of script, this can be done by default and the user needs to enter only the actual account number ‘010203040001’ and it will be converted to the format by the script written by the user before it is further processed.

- 2□ The Branch may be connected to an ATM that supports only 10 character account number and the actual account number size in FINACLE™ may be more. In this case, the FINACLE™ account number format is to be converted to ATM account number format using some logic. This can be done in scripting. The bank will decide on the logic to derive the unique account number for ATM from FINACLE™ format and will code the same in the Script.

The account number in FINACLE™ may be 01-02-03-04-0001.

Where 01 - corresponds to Bank code 02 - corresponds to Branch code 03 - corresponds to the currency alias 04 - corresponds to the scheme code 0001 is the actual account number.

To convert to 10 character ATM account number format, the bank may decide to leave out the bank code portion as the ATM is attached to the branches of the same BANK and leave out the formatting of account number with '- '.

The ATM account number that is derived can be 0203040001 that should be unique in the data center.

1.2 REPOSITORIES AND CLASSES

Please refer to Scripting Syntax Document for a general discussion on Repositories and classes.

All Finacle™ 'script events' (except for Workflow related events and MTT related events) interface with the scripts using a standard repository called "BANCS". There are 3 classes that have been predefined in this repository, STDIN, INPUT and OUTPUT, each of which hold 'String' type of fields.

STDIN class is the set of common fields that is available in any event.

The fields available in the standard CLASS STDIN are listed in the appendix on page 56

Using the following syntax in the scripts, one can access the field in the repositories –

ex : BANCS.STDIN.dcAlias.

The INPUT CLASS contains fields' specific to the events, which are the data items that are made available to the script defined for that event. The list of fields available to individual events is described in the appropriate section below.

The OUTPUT CLASS contains fields' specific to the events, which are the data items that will be accessed by Finacle™ after the execution of the script to determine its logic of further execution. The list of fields available to individual events is described in the appropriate section below.

1.3 RETURN STATUS OF THE SCRIPT

One of the standard output fields from the script is successOrFailure. In case a FAILURE is encountered in the script then the OUTPUT CLASS variable successOrFailure can be set to FAILURE as shown

Ex : BANCS.OUTPUT.successOrFailure = "F"

The valid values are "S" for success and "F" Failure.

The default return status from the script is SUCCESS.

.

2 FORM LOAD EVENTS

Facility to customise any Finacle™ screen is provided using FINACLE™ script engine. This is facilitated using a combination of Script events and Scripting user hooks (refer to Scripting hooks document for more details). The kinds of customisation allowed through scripting are:

- 1❑ Set the values of screen data items
- 2❑ Set the attributes of screen data items
- 3❑ Automatically execute key-based events defined (by Infosys) in the form, without the user having to hit the key
- 4❑ There are five predefined events that will be executed whenever a form is accessed. They are 'Pre Load', 'Post Load' and 'Post Execute', 'Pre Key Replay' and 'Post Key Replay'.

2.1 IDENTIFYING THE FIELD NAMES.

The name of a field is not the same as the literal that is displayed on screen in a form. For e.g. in CUMM option, the customer NRE flag is internally identified by a field name by the application. In order to write a form event, it is essential that the user refers to this field name only in the script.

In order to achieve this objective, an utility known as '*dispfields*' is provided.

This executable must be executed at the unix prompt along with the form name.

The syntax of the utility is as described below.

dispfields

Usage : dispfields -1 crtfile -2 forms dir. path -3 formfilename|-4 listfilename|-5 (all forms) [-6 printfilename] [-7 errorfilename]

1. the crt file name must be mentioned. (if the file is not in the current directory the entire path must be mentioned)
2. Forms dir. – here the forms directory must be mentioned for the utility to pick the form. (\$TBA_PROD_ROOT/ cust/INFENG/forms directory)
3. Formfilename – Here we mention the form name. (e.g bafe3012)
4. Listfilename – The list file name can also be mentioned (optional)
5. All forms - If -5 is specified then all the forms are displayed. (optional. If option 5 is specified then 3 must not be specified)
6. Printfilename - the output can be directed to a print file name (optional)
7. Errorfilename – piping of the error to an error file name (optional)

EXAMPLE:

```
dispfields -1 $TBAF_DEFAULT/P.crt -2 $TBA_PROD_ROOT/cust/INFENG/forms -3 baff0009
```

displays the following screen.

```

baff0009          xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx          xxxxxxxxxxxx
                    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

-----
Function          _   xxxxxxxxxxxxxxxxxxxxxxxx
Customer Id       _____

-----
Name              _____          Short Name          _____
Cust.Type         _____ xxxxxxxxxxxxxxxxxxxxxxxx * Account Manager _____
Cust.Status       _____ xxxxxxxxxxxxxxxxxxxxxxxx Sex          _ xxxxxxxxx
Cust.Group        _____ xxxxxxxxxxxxxxxxxxxxxxxx Status Date _____
Occupation        _____ xxxxxxxxxxxxxxxxxxxxxxxx Cust Non Resident? _
Staff No.         _____ xxxxxxxxxxxxxxxxxxxxxxxx * Customer Staff ? _
Constitution      _____ xxxxxxxxxxxxxxxxxxxxxxxx Customer Minor ? _
Bank Code         _____ xxxxxxxxxxxxxxxxxxxxxxxx First A/c Date xxxxxxxxxxxx
                  Introduder's Details:
                  Nat.Id.Card No _____
Customer Id.      _____ xxxxxxxxxxxxxxxxxxxxxxxx * Tot Mod Times xxx
Name              _____ * Suspended ? x
Introd.Status     _____ xxxxxxxxxxxxxxxxxxxxxxxx Date Of Birth _____
Frequency of Pass Sheet _/_/_/_/_
Enter Option     _ xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
+----- ( ==>> ) --+

0001/0001          funcblk.header_title          baff0009 01/10

```

In the above screen, the CUMM option first screen, the form name baff0009 is shown and from here we can find out the field name of Cust Non Resident? is datablk1.cust_nre_flg.

2.2 PRE LOAD FORM EVENT.

This event will be executed before the form is loaded. Please note that this event happens only once per form load and not once per transaction that the form supports. This is because a user can complete multiple transactions in the same form before exiting. If some customisation is required to be done on a per-transaction basis, then the 'Pre Key Replay' event on ACCEPT key of Function block should be used.

SCRIPT:

The script **<formname> PreLoad.scr** should exist in the TBA_SCRIPTS directory. Script will be executed only if script exist in TBA SCRIPTS directory.

INPUTS AVAILABLE IN THE SCRIPT:

There are no fields available in the INPUT class for this event. However **urhk_TBAF_InquireFieldValue** can be used to get values of fields in the form that is calling this form.

OUTPUTS EXPECTED:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

This will be assigned the value got by processing the input.

SuccessOrFailure	"S" or "F". If not "S" then fatal error will occur.
------------------	---

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

There is no default logic that is applied.

TYPICAL USAGE SCENARIO(S):

This event would be required to be scripted by the Bank for those forms in which default field value or default field attribute default needs to be changed from the Infosys setting. The user hooks `urhk_TBAF_SetValue` and `urhk_TBAF_SetAttrib` should be used in this event.

It can also be used to define 'Replay key' events for the form using the userhook `urhk_TBAF_SetReplayKey`.

SAMPLE SCRIPT:

For an involved example, see 3.4. The following is a simple example.

```
# The following script sets mandatory attribute for each of the fields below
# It uses a user hook called urhk_TBAF_SetAttrib
<--start
sv_a = urhk_TBAF_SetAttrib("tdff0009.d1.lien_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d1.nominee_print_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d1.printing_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d2.auto_renewal_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d2.tds_exemp_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d2.prtg_rmks|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d2.int_accrual_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d2.close_on_maturity_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d3.sulabh_flg|M")
sv_a = urhk_TBAF_SetAttrib("tdff0009.d3.safe_custody_flg|M")

exitscript
end-->
```

2.3 POST LOAD FORM EVENT.

This event will be executed after the form is loaded, but before executing any logic specified in the form. Please note that this event happens only once per form load and not once per transaction that the form supports. This is because a user can complete multiple transactions in the same form before exiting. If some customisation is required to be done on a per-transaction basis, then the 'Pre Key Replay' event on ACCEPT key of Function block should be used.

SCRIPT:

The script **<formname>PostLoad.scr** should exist in the TBA_SCRIPTS directory.

Script will be executed only if script exist in TBA_SCRIPTS directory.

INPUTS AVAILABLE IN THE SCRIPT:

There are no fields available in the INPUT class for this event. However **`urhk_TBAF_InquireFieldValue`** can be used to get the default values of fields in the form.

OUTPUTS EXPECTED:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

This will be assigned the value got by processing the input.

SuccessOrFailure	"S" or "F". If not "S" then fatal error will occur.
------------------	---

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

There is no default logic that is applied.

TYPICAL USAGE SCENARIO(S):

This event would be required to be scripted by the Bank for those forms in which default field value or default field attribute default needs to be changed from the Infosys setting. The user hooks **urhk_TBAF_ChangeFieldValue** and **urhk_TBAF_ChangeFieldAttrib** should be used.

It can also be used to define 'Replay key' events for the form using the userhook **urhk_TBAF_SetReplayKey**.

Though both the PreLoad and PostLoad event are very similar in their usage scenario, the actual event used will depend upon whether the logic is dependent on field values of the calling form or the called form or not dependent at all. If logic is dependent on field values of the calling form then PreLoad event should be used. If logic is dependent on field values of the called form then PostLoad event should be used. If logic is not dependent upon the field values at all then either one can be used.

SAMPLE SCRIPT:

Similar to 3.1

2.4 POST EXECUTE FORM EVENT

This event will be executed after the form is unloaded, i.e. when a user quits a form. Please note that this event happens only once per form and not once per transaction that the form supports. This is because a user can complete multiple transactions in the same form before exiting. If some customisation is required to be done on a per-transaction basis, then the 'Pre Key Replay' event on ACCEPT or QUIT keys should be used.

SCRIPT:

The script **<formname> PostExecute.scr** should exist in the TBA_SCRIPTS directory.

Script will be executed only if script exist in TBA_SCRIPTS directory.

INPUTS AVAILABLE IN THE SCRIPT:

There are no fields available in the INPUT class for this event. However **urhk_TBAF_InquireFieldValue** can be used to get the values of the fields in the form that called this form.

OUTPUTS EXPECTED:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

This will be assigned the value got by processing the input.

successOrFailure	"S" or "F". If not "S" then fatal error will occur.
------------------	---

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

There is no default logic that is applied.

TYPICAL USAGE SCENARIO(S):

This event allows the Bank to implement some minimum workflow capabilities. This can be done because, when a form that is called from menu is exited, the context of the post-execute event becomes the menu form and the script can then examine menu fields and execute key-based events in the menu. This will allow for the automatic calling of another menu option after exiting from one.

SAMPLE SCRIPT:

Contact Infosys in case usage of this event is warranted.

2.5 PRE KEY REPLAY EVENT

This event will occur just before the Infosys logic associated with the specified key (using **urhk_TBAF_SetReplayKey**) is executed. When this event occurs and a Pre Replay script has been specified in the **urhk_TBAF_SetReplayKey** call, then that script is executed.

SCRIPT:

The script **<xxxxxxx.scr>** (specified in the **urhk_TBAF_SetReplayKey**) should exist in the TBA_SCRIPTS directory.

Script will be executed only if script exist in TBA_SCRIPTS directory.

INPUTS AVAILABLE IN THE SCRIPT:

There are no fields available in the INPUT class for this event. However **urhk_TBAF_InquireFieldValue** can be used to get the values of the fields in the form.

OUTPUTS EXPECTED:

No outputs are expected by Finacle™.

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

There is no default logic that is applied.

TYPICAL USAGE SCENARIO(S):

This event allows the Bank to implement some minimum workflow capabilities like automating the visiting of 'mandatory screens', protecting/unprotecting of certain fields in datablock based on values entered in other blocks etc.

SAMPLE SCRIPT:

In this example, a sample script has been written which can be used to invoke all the mandatory forms to be visited while opening a deposit account automatically.

First we will code the PreLoad script of the deposit form to execute an 'accept-key' when the option block is visited.

Then we will code the pre-replay script of the above key to populate the appropriate option code into the option field.

Then we will code the post-replay script of the above key to reset the replay key so that when the option block is visited again the next option is chosen.

SCRIPT NAME :-

tdfe3201PreLoad.scr

```
<--start
sv_a = urhk_TBFAF_SetReplayKey("optnblk.key-f2|tdoptpre.scr|0|tdoptpost.scr|0")
exitscript
end-->
```

SCRIPT NAME :-

tdoptpre.scr

```

<--start
sv_b = cint(INTBAF.INTBAFC.TbafEventStep)
if (sv_b == 0) then
GOTO STEP0
endif
if (sv_b == 1) then
GOTO STEP1
endif
if (sv_b == 2) then
GOTO STEP2
endif
if (sv_b == 3) then
GOTO STEP3
endif
exitscript
# Populate option as G to visit general details.
STEP0:
sv_a = urhk_TBAF_ChangeFieldValue("optnblk.acct_opn_option|G")
exitscript
# Populate option as N to visit Nominee details
STEP1:
sv_a = urhk_TBAF_ChangeFieldValue("optnblk.acct_opn_option|N")
exitscript
# Populate option as F to visit Flow details.
STEP2:
sv_a = urhk_TBAF_ChangeFieldValue("optnblk.acct_opn_option|F")
exitscript
# Populate option as V to visit Advance details.
STEP3:
sv_a = urhk_TBAF_ChangeFieldValue("optnblk.acct_opn_option|V")
exitscript
end-->
Script Name :- tdoptpost.scr
<--start
sv_b = cint(INTBAF.INTBAFC.TbafEventStep)
if (sv_b == 0) then
GOTO STEP0
endif
if (sv_b == 1) then
GOTO STEP1
endif
if (sv_b == 2) then
GOTO STEP2
endif

exitscript
# Populate Replay key to execute Step 1 or Step 2 of pre-script
STEP0:
sv_a = urhk_TBAF_InquireFieldValue (datablk1.nom_available_flg)
if (B2KTEMP.TEMPSTD.TBAFRESULT = "Y")
sv_a = urhk_TBAF_SetReplayKey("optnblk.key-f2|tdoptpre.scr|1|tdoptpost.scr|1")
else
sv_a = urhk_TBAF_SetReplayKey("optnblk.key-f2|tdoptpre.scr|2|tdoptpost.scr|2")
exitscript
# Populate Replay key to execute Step 2 of pre-script
STEP1:
sv_a = urhk_TBAF_SetReplayKey("optnblk.key-f2|tdoptpre.scr|2|tdoptpost.scr|2")
exitscript
# Populate Replay key to execute Step 3 of pre-script
STEP2:
sv_a = urhk_TBAF_SetReplayKey("optnblk.key-f2|tdoptpre.scr|3")
exitscript

end-->

```


2.6 POST KEY REPLAY EVENT

This event will occur just after the Infosys logic associated with the specified key (using **urhk_TBAF_SetReplayKey**) is executed. When this event occurs and a Post Replay script has been specified in the **urhk_TBAF_SetReplayKey** call, then that script is executed. Please note that when the Replay key is the COMMIT key, then this script is executed only if the Commit actually happens. I.e. if the user has hit the commit key but the system displays an error because of whatever reason then this script will not be executed.

SCRIPT:

The script **<xxxxxxxx.scr>** (specified in the **urhk_TBAF_SetReplayKey**) should exist in the TBA_SCRIPTS directory.

Script will be executed only if script exist in TBA_SCRIPTS directory.

INPUTS AVAILABLE IN THE SCRIPT:

There are no fields available in the INPUT class for this event. However **urhk_TBAF_InquireFieldValue** can be used to get the values of the fields in the form.

OUTPUTS EXPECTED:

No outputs are expected by Finacle™.

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

There is no default logic that is applied.

TYPICAL USAGE SCENARIO(S):

This event allows the Bank to implement some minimum workflow capabilities like automating the visiting of 'mandatory screens', protecting/unprotecting of certain fields in datablock based on values entered in other blocks etc. Typically used in tandem with Pre Key Replay event.

This event can also be used to integrate Finacle™ with some other applications. By scripting this event for the COMMIT key, the Bank can pass the data already entered in Finacle™ to some other applications.

SAMPLE SCRIPT:

See example in Pre Key Replay event

HOT-KEY EVENT

This event occurs whenever a function key which is defined as a 'Scripting Hot Key' in the CRT definition file is pressed by a user on any Finacle™ screen.

SCRIPT:

The script **HotKeyScript.scr** should exist in the TBA_SCRIPTS directory.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. One can access the inputs by referring to the fields in the following manner.

BANCS.INPUT.<fieldName>

LogicalKeyName	Logical name assigned to the function key in the CRT file
ThisFormName	The formname in which the function key was hit
ThisBlockName	The block of the form in which the function key was hit
ThisFieldName	The field of the form in which the function key was hit

OUTPUTS EXPECTED:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

This will be assigned the value got by processing the input.

SuccessOrFailure	"S" or "F". If not "S" then fatal error will occur.
------------------	---

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

There is no default logic that is applied.

TYPICAL USAGE SCENARIO(S):

This event allows the Bank to interface Finacle™ with their other applications. By defining a 'hot key' (which may be applicable on certain pre-defined Finacle™ screens), the bank can instruct their users to use that key in certain situations where control needs to be transferred to another application for some additional processing before returning to the Finacle™ screen. Then this script can be used as the 'binder' between Finacle™ and the other application, in terms of transferring certain data from the Finacle™ screen to the other application and vice-versa.

Note that for this event to occur, the appropriate 'crd' file needs to contain an entry for the hotkey as shown below:

```
;          Invoke a script          - CTRL-E
key-call-script-1    |5|Tba_TrgInvokeScript|
```

In this example 'key-call-script-1' will be passed as the logical key to the HotKeyScript.scr whenever the user presses CTRL-E on any Finacle™ screen.

SAMPLE SCRIPT:

Please see sample/scripts directory

3 FINANCIAL TRANSACTION EVENTS

3.1 ACCOUNT DETAILS DISPLAY IN TM MENU OPTION

Whenever a new part-transaction is entered or displayed in the 'part-transaction' block of the 'Transaction Maintenance' screen, the 'account block' at the bottom displays the details of the account being debited or credited in the part-transaction.

By default a number of account details (like customer status, mode of operation, balance break-up etc.) are being displayed. This event occurs whenever such a display occurs.

The account details for the account can be obtained by calling the user hook `ushk_getAcctDetailsInRepository` by passing account number as input to the user hook. (refer document on Scripting User hooks for the functionality details of user hook).

TBAF_ChangeFieldAttrib user hook can be used to display or hide the account details (refer document on Scripting User hooks for the functionality details of the user hook) being shown in the 'account/customer information block' of the 'transaction maintenance screen.

SCRIPT

The script `CheckAndDispAcctBal.scr` should be available in `TBA_SCRIPTS` directory.

INPUT AVAILABLE IN THE SCRIPT

The defined inputs are available in a repository called `BANCS`, a class called `INPUT` and with the field names mentioned below. One can access the inputs by referring to the fields in the following manner.

`BANCS.INPUT.<fieldName>`

AcctId	Account number (Foracid) of the account currently being displayed in the 'part transaction block' of the screen.
--------	--

OUTPUTS EXPECTED:

The script is expected to set the attributes of the required fields appropriately. No other outputs are expected in the `OUTPUT` class of `BANCS` repository.

DEFAULT LOGIC IN CASE OF SCRIPT IS NOT AVAILABLE.

All account details as defined in the transaction maintenance screen will be displayed.

TYPICAL USAGE SCENARIO(S):

This event is typically used for controlling the kinds of information about the customer and account that is to be shown in the 'transaction maintenance' screen. The bank can decide which of the fields are to be 'hidden' and which are to be displayed based on the account number passed to this script.

SAMPLE SCRIPT:

This example shows how to hide account balance fields(except shadow balance) in TM if the account belongs to an employee

```
Script Name - CheckAndDispAcctBal.scr
<--start
sv_a = BANCS.INPUT.AcctId

# -----
# - sv_a (account id) is the input to user hook getAcctDetailsInRepository
# - Call user hook getAcctDetailsInRepository to put get the Account details
# -----

sv_b = urhk_getAcctDetailsInRepository(sv_a)
# if getAcctDetailsInRepository function returns failure exit out of script.
if( sv_b == 1 ) then
  exitscript
endif
sv_c="E"

# -----
# If the acctOwnership flag of OUTPARAM class is equal to "E"
# then hide the appropriate fields
# -----

if (BANCS.OUTPARAM.acctOwnership == sv_c) then
  sv_z = "dispblk2.acct_bal|H"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.acct_bal_dr_cr_ind|H"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.avail_amt|H"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.avail_amt_dr_cr_ind|H"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.eff_avail_amt|H"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.eff_avail_amt_dr_cr_ind|H"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
else
  sv_z = "dispblk2.acct_bal|U"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.acct_bal_dr_cr_ind|U"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.avail_amt|U"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.avail_amt_dr_cr_ind|U"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.eff_avail_amt|U"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
  sv_z = "dispblk2.eff_avail_amt_dr_cr_ind|U"
  sv_z = urhk_TBAF_ChangeFieldAttrib(sv_z)
endif
exitscript
end-->
```

The above script will hide or display the account balances in TM screen of Finacle™.

3.2 CLEARING TRANSACTIONS

LODGING OF INSTRUMENTS

The script will be called in OCTM menu option in this mode when <KEY-COMMIT> is pressed after entering the outward clearing part tran and instrument details. The script will be invoked once for each part tran entry with the event as "OWCLG_LODGE".

SCRIPT:

The script **FinTran.scr** should exist in the TBA_SCRIPTS directory. Also, the parameter "Use Fin. Transactions script?" must have been set as 'Y' in SRGPM menu option for the scheme of the account involved.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. The inputs can be accessed by referring to the field in the following manner :

BANCS.INPUT.<fieldName>

The input fields available are:

Event	The value of this will be "OWCLG_LODGE"
AddtlDetails	The additional user defined data associated with the account
AcctNum	The account number used for instrument lodging.
AcctCrncy	The account currency code.
SchemeCode	The scheme code of the account.
SchemeType	The scheme type of the account.
CustomerId	The customer id of the account.
ZONE_SolId	The sol id of the clearing zone.
ZONE_Code	The clearing zone code.
ZONE_ClearingDate	The clearing zone date (DD-MM-YYYY 00:00:00)
ZONE_CreditAcctNum	The zone credit account number.
ZONE_DebitAcctNum	The zone debit account number.
ZONE_HomeClgFlg	The zone Home clearing flag.
ZONE_ShadowBalFlg	The shadow balance flag.
ZONE_MicrClgFlg	The MICR clearing flag.
ZONE_MicrAbbreviation	The abbreviation of the MICR code.
ZONE_MicrChrgCode	The MICR charge code.
ZONE_Status	The status of the zone.
ZONE_TotalDrPTranAmt	The total debit part tran amount.
ZONE_TotalCrPTranAmt	The total credit part tran amount.
ZONE_TotalInstAmt	The total instrument amount.
ZONE_TotalShadowBal	The total shadow balance amount.
ZONE_TotalPendingAmt	The total pending amount.
ZONE_RemainingShadowBal	The remaining amount in the shadow balance.
ZONE_BarInstAmt	The bar instrument amount.

ZONE_BarNumber	The bar number.
ZONE_BarDate	The bar date.
ZONE_SetsEntered	The number of sets entered.
ZONE_SetsVerified	The sets which are verified.
ZONE_FrgnCrcyClgFlg	The foreign currency clearing flag.
ZONE_SttlmntInInstCrcyFlg	The settlement in the instrument currency flag.
ZONE_SttlmntCrcyCode	The settlement currency code.
ZONE_CrcyCode	The currency code of the zone.
ZONE_ToSttlmntCrcyRate	The currency rate used for the settlement of the zone.
ZONE_RateCode	The rate code used for the zone transaction.
ZONE_ClgSecBankCode	The clearing section bank code.
ZONE_SetRelRegFlg	The flag which shows the zone is regularised or not.
ZONE_SetRelDrAcct	The account used for debiting when the zone is released.
ZONE_Latency	The buffer number of days for releasing the zone and value date clearing transactions.
TranAmt	The Instrument amount in the account currency.
ShdBalRemAmt	The remaining amount which still remains to be regularised.
PendingAmt	The amount which cannot be regularised.
PtranAmtInInstCrcy	The part tran amount in the instrument currency.
SttlInstToHomeCrcyRate	The rate of the settlement from the Instrument to the Home currency.
HomeToAcctCrcyRate	The rate of the settlement from the Home to the account currency.
SttlInstToAcctCrcyRate	The rate of the settlement from instrument to the account currency.
AcctSolId	The sol_id of the account.
DrCrInd	The debit or credit indicator.
EntryDate	The date of entry for the zone.
VerifiedDate	The date of the verification.
StatusFlg	The status flag.
SttlmntToHomeRateCode	The rate code for the settlement from the Instrument to the Home currency.
HomeToAcctRateCode	The rate code for the settlement from the Home to the account currency.
SttlmntToAcctRateCode	The rate code for the settlement from instrument to the account currency.
SetRelRegStatus	The set released and regularised status.
TDRefNum	The TD reference number.
SetNumber	The set number.
NumOfInstruments	Number of instruments lodged.

In addition to the above details, the following instrument wise details are also available. Here, the field name will include the instrument number also. Hence, for example, INST3_Amount stands for the amount of the 3rd instrument. In general, the instrument number is given as a wild card ('?'). This can actually be any number from 1 to n where n is the number of instruments involved (available in NumOfInstruments field as given above).

INST?_SrlNum	Instrument serial number
INST?_ShadowBalCode	Instrument shadow balance code
INST?_TranCode	Instrument transaction code
INST?_ID	Instrument ID
INST?_Alpha	Instrument alpha
INST?_Date	Instrument date
INST?_Amount	Instrument amount
INST?_BankCode	Instrument bank code
INST?_BranchCode	Instrument branch code
INST?_SortCode	Instrument sort code
INST?_Status	Instrument status
INST?_EntryDate	Instrument entry date
INST?_VerifiedDate	Instrument Verified date
INST?_ValueDate	Instrument value date

OUTPUTS EXPECTED FROM THE SCRIPT:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

SuccessOrFailure	To check whether the script returned success or failure. Values "S" for success and "F" for failure. In case of "F", the lodging will not proceed.
ErrorDesc	In case of failure, the error message that ought to be displayed on the screen.
AddtlDetails	The additional user defined data. This will be stored in the database.
DeleteUAD	The flag to indicate whether to delete the UAD record (User additional detail) or not. Values "Y" for delete and "N" for do not delete. In this mode, this output is not significant

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In this mode user can do additional user specific validations on the part transaction and instrument details, raise exceptions or errors for exception conditions and accept additional user defined data. More than one exception can be raised from within the script. The additional data can be accepted by bringing up the general parameter acceptance screen in which user can define his own fields and accept relevant values for them. If an error is returned by the script, the lodging of instruments will fail and successful lodging will go through only on correcting the error condition. If one or more exception is raised from the script, then all such exceptions will be shown to the user in the exception handler form which can be overridden if required. If any additional data is accepted by user, then the same will be saved and will be provided as input to the script in the regularization mode.

SAMPLE SCRIPT:

Please refer to the script FinTran.sscr in sample/scripts directory.

REGULARIZATION OF THE INSTRUMENTS WHERE SHADOW BALANCE FLAG IS 'Y'

The script will be called in OCTG, AUTOREG and MCLZOH menu option in this mode when <KEY-COMMIT> is pressed after request for zone regularization and where the shadow balance flag is set as 'Y'.

SCRIPT:

The script **FinTran.scr** should exist in the TBA_SCRIPTS directory. Also, the parameter "Use Fin. Transactions script?" must have been set as 'Y' in SRGPM menu option for the scheme of the account involved.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. The inputs can be accessed by referring to the field in the following manner.

BANCS.INPUT.<fieldName>

The input fields available are :

Event	The value of this will be "OWCLG_REGULARIZE"
AddtlDetails	The additional user defined data associated with the account
AcctNum	The account number used for instrument lodging.
AcctCrncy	The account currency code.
SchemeCode	The scheme code of the account.
SchemeType	The scheme type of the account.
CustomerId	The customer id of the account.
ZONE_SolId	The sol id of the clearing zone.
ZONE_Code	The clearing zone code.
ZONE_ClearingDate	The clearing zone date (DD-MM-YYYY 00:00:00)
ZONE_CreditAcctNum	The zone credit account number.
ZONE_DebitAcctNum	The zone debit account number.
ZONE_HomeClgFlg	The zone Home clearing flag.
ZONE_ShadowBalFlg	The shadow balance flag.
ZONE_MicrClgFlg	The MICR clearing flag.
ZONE_MicrAbbreviation	The abbreviation of the MICR code.
ZONE_MicrChrgCode	The MICR charge code.
ZONE_Status	The status of the zone.
ZONE_TotalDrPTranAmt	The total debit part tran amount.

ZONE_TotalCrPTranAmt	The total credit part tran amount.
ZONE_TotalInstAmt	The total instrument amount.
ZONE_TotalShadowBal	The total shadow balance amount.
ZONE_TotalPendingAmt	The total pending amount.
ZONE_RemainingShadowBal	The remaining amount in the shadow balance.
ZONE_BarInstAmt	The bar instrument amount.
ZONE_BarNumber	The bar number.
ZONE_BarDate	The bar date.
ZONE_SetsEntered	The number of sets entered.
ZONE_SetsVerified	The sets which are verified.
ZONE_FrgnCrcncyClgFlg	The foreign currency clearing flag.
ZONE_SttlmntInInstCrcncyFlg	The settlement in the instrument currency flag.
ZONE_SttlmntCrcncyCode	The settlement currency code.
ZONE_CrcncyCode	The currency code of the zone.
ZONE_ToSttlmntCrcncyRate	The currency rate used for the settlement of the zone.
ZONE_RateCode	The rate code used for the zone transaction.
ZONE_ClgSecBankCode	The clearing section bank code.
ZONE_SetRelRegFlg	The flag which shows the zone is regularised or not.
ZONE_SetRelDrAcct	The account used for debiting when the zone is released.
ZONE_Latency	The buffer number of days for releasing the zone and value date clearing transactions.
TranAmt	The Instrument amount in the account currency.
ShdBalRemAmt	The remaining amount which still remains to be regularised.
PendingAmt	The amount which cannot be regularised.
PtranAmtInInstCrcncy	The part tran amount in the instrument currency.
SttlInstToHomeCrcncyRate	The rate of the settlement from the Instrument to the Home currency.
HomeToAcctCrcncyRate	The rate of the settlement from the Home to the account currency.
SttlInstToAcctCrcncyRate	The rate of the settlement from instrument to the account currency.
AcctSolId	The sol_id of the account.
DrCrInd	The debit or credit indicator.
EntryDate	The date of entry for the zone.
VerifiedDate	The date of the verification.
StatusFlg	The status flag.
SttlmntToHomeRateCode	The rate code for the settlement from the Instrument to the Home currency.
HomeToAcctRateCode	The rate code for the settlement from the Home to the account currency.
SttlmntToAcctRateCode	The rate code for the settlement from instrument to the account currency.
SetRelRegStatus	The set released and regularised status.
TDRefNum	The TD reference number.
SetNumber	The set number.

OUTPUTS EXPECTED FROM THE SCRIPT:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

SuccessOrFailure	To check whether the script returned success or failure. Values "S" for success and "F" for failure. In this mode, this output is not checked for. Regularization will proceed.
ErrorDesc	In case of failure, any error message. Again this field is not significant in this mode.
AddtlDetails	The additional user defined data. Since this data is not modifiable in this mode, send back the input addtlDetails itself in this field.
DeleteUAD	The flag to indicate whether to delete the UAD record (User additional detail) or not. Values "Y" for delete and "N" for do not delete. If "Y", then the additional details for this clearing part tran will be deleted from the database

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In this mode user can use the additional user defined data specified at lodging time to do some additional processing (for e.g. initiate a workflow to execute some other menu option). Irrespective of the value returned by the script, the regularization process will continue. If any additional processing like initiating a work flow is specified within the script, then the corresponding action is done.

SAMPLE SCRIPT:

Please refer to script FinTran.sscr in sample/scripts directory.

3.3 TRANSACTIONS THROUGH TM

PART TRAN ENTRY

The script will be called in TM menu option in this mode when the part tran details are entered in the TM menu option and <KEY-ACCEPT> is pressed. Here the event will be "TM_ENTRY".

SCRIPT:

The script **FinTran.scr** should exist in the TBA_SCRIPTS directory. Also, the parameter "Use Fin. Transactions script?" must have been set as 'Y' in SRGPM menu option for the scheme of the account involved.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. The inputs can be accessed by referring to the field in the following manner.

BANCS.INPUT.<fieldName>

The input fields available are:

Event	The value of this field will be "TM_ENTRY"
NumLiensLifted	The number of liens lifted by the transaction.
TotLienAmtLifted	The total lien amount lifted by the transaction.
AddtlDetails	The user specified additional details.
AcctNum	The account number of the part tran
SchemeCode	The scheme code of the account used.
SchemeType	The scheme type of the account used.
TranAmt	The transaction amount in the tran currency of the part tran
RefAmt	The reference amount in the ref currency.
Rate	The rate used in the transaction for converting to the account currency.
ValMode	This will be required to be passed as input to the FTS_RaiseException user hook function if using it.
DrCrInd	"D" for debit and "C" for credit part tran
TranType	The transaction type as given in the TM menu
ExceptionIgnoreFlg	Field required to be passed as input to raise exception user hook.
TranCreationMode	The transaction creation mode.
PST_Flag	The flag of the PST table.
TranSubType	The transaction sub type.
InstrumentType	The type of instrument used like cheque etc
InstrumentDate	The date of the instrument used for the transaction.
InstrumentAlpha	The alpha number of the instrument.
InstrumentNum	The instrument number of the transaction.
ReportCode	The report code used for the transaction.
TranCurrency	The currency code of the transaction.
RefCurrency	The reference currency code.
RateCode	The rate code used for the transaction.
TranDate	The date of the transaction.
ValueDate	The value date of the transaction.
CustomerId	The customer Id of the account.
AcctSolId	The sol_id of the account.
RestrictModifyInd	The restrict or modify indicator.
InstantTodAmt	The Instant TOD amount granted for a part transaction

TodLvlIntFlg	The TOD level interest flag for the Instant TOD
--------------	---

OUTPUTS EXPECTED FROM THE SCRIPT:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

SuccessOrFailure	To check whether the script returned success or failure. "S" for success and "F" for failure. If "F", then the error message given in the next field below will be displayed and the <ACCEPT> will fail.
ErrorDesc	Any error message in case of script failure.
AddtlDetails	The additional user defined and accepted data if any. This will be saved in the database for later usage.
DeleteUAD	The flag to indicate whether to delete the UAD record (User additional detail) or not. Values "Y" for delete and "N" for do not delete. In this mode, this output is not relevant. Any value can be passed back.

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In this mode user can do additional user specific validations on the part transaction, raise exceptions or errors for exception conditions and accept additional user defined data. More than one exception can be raised from within the script. The additional data can be accepted by bringing up the general parameter acceptance screen in which user can define his own fields and accept relevant values for them. If an error is returned by the script, the part transaction entry will fail and successful entry will go through only on correcting the error condition. If one or more exception is raised from the script, then all such exceptions will be shown to the user in the exception handler form at the time of posting which can be overridden if required. If any additional data is accepted by user, then the same will be saved and will be provided as input to the script in the post mode.

SAMPLE SCRIPT:

Please refer to script FinTran.sscr in sample/scripts directory.

PART TRAN REQUEST POST

The script will be called in the TM menu option in this mode when the individual part trans are being requested for posting (i.e. when option code is given as 'P' and <KEY-ACCEPT> is pressed. Here the event will be "TM_REQUEST_POST".

SCRIPT:

The script **FinTran.scr** should exist in the TBA_SCRIPTS directory. Also, the parameter "Use Fin. Transactions script?" must have been set as 'Y' in SRGPM menu option for the scheme of the account involved.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. The inputs can be accessed by referring to the field in the following manner.

BANCS.INPUT.<fieldName>

The input fields available are :

Event	The value of this field will be "TM_REQUEST_POST" in this mode.
NumLiensLifted	The number of liens lifted by the transaction.
TotLienAmtLifted	The total lien amount lifted by the transaction.
AddtlDetails	The user specified additional details.
AcctNum	The account number of the part tran
SchemeCode	The scheme code of the account used.
SchemeType	The scheme type of the account used.
TranAmt	The transaction amount in the tran currency of the part tran
RefAmt	The reference amount in the ref currency.
Rate	The rate used in the transaction for converting to the account currency.
ValMode	This will be required to be passed as input to the FTS_RaiseException user hook function if using it.
DrCrInd	"D" for debit and "C" for credit part tran
TranType	The transaction type as given in the TM menu
ExceptionIgnoreFlg	Field required to be passed as input to raise exception user hook.
TranCreationMode	The transaction creation mode.
PST_Flag	The flag of the PST table.
TranSubType	The transaction sub type.
InstrumentType	The type of instrument used like cheque etc
InstrumentDate	The date of the instrument used for the transaction.
InstrumentAlpha	The alpha number of the instrument.
InstrumentNum	The instrument number of the transaction.
ReportCode	The report code used for the transaction.
TranCurrency	The currency code of the transaction.
RefCurrency	The reference currency code.
RateCode	The rate code used for the transaction.
TranDate	The date of the transaction.
ValueDate	The value date of the transaction.
CustomerId	The cutomer Id of the account.
AcctSolId	The sol_id of the account.
RestrictModifyInd	The restrict or modify indicator.
InstantTodAmt	The Instant TOD amount granted for a part transaction
TodLvlIntFlg	The TOD level interest flag for the Instant TOD

OUTPUTS EXPECTED FROM THE SCRIPT:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

SuccessOrFailure	To check whether the script returned success or failure. "S" for success and "F" for failure. If "F", then the error message given in the next field below will be displayed and the POSTING REQUEST will fail.
ErrorDesc	Any error message in case of script failure.
AddtlDetails	The additional user defined and accepted data if any. In this mode, the input addtdetails should be copied to this output field as no modification is allowed here.
DeleteUAD	The flag to indicate whether to delete the UAD record (User additional detail) or not. Values "Y" for delete and "N" for do not delete. In this mode, this output is not relevant. Any value can be passed back.

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In this mode user can do additional user specific validations on the part transaction, raise exceptions or errors for exception conditions and validate the additional user defined data specified in the entry mode. If an error is returned by the script, the part transaction posting request will fail and successful posting request will go through only on correcting the error condition. If one or more exception is raised from the script, then all such exceptions will be shown to the user in the exception handler form which can be overridden if required. If the Instant TOD amount has changed due to some other transaction, then the script can decide whether the user needs to visit the Instant TOD details screen again. The user hook, getAcctDetailsInRepository has to be called to obtain all the available amounts and the script can decide based on these available amounts and the 2 inputs, InstantTodAmt and TodLvlIntFlg.

SAMPLE SCRIPT:

Please refer to script FinTran.sscr in sample/scripts directory.

USER ADDITIONAL DETAIL DISPLAY

The script will be called in TM menu option in this mode when for each part tran, option code is given as 'J' and <KEY-ACCEPT> is pressed. Here the event will be "TM_USER_ADDTL_DETAILS".

SCRIPT:

The script **FinTran.scr** should exist in the TBA_SCRIPTS directory. Also, the parameter "Use Fin. Transactions script?" must have been set as 'Y' in SRGPM menu option for the scheme of the account involved.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. The inputs can be accessed by referring to the field in the following manner.

BANCS.INPUT.<fieldName>

The input fields available are :

Event	The value of this field will be "TM_USER_ADDTL_DETAILS" in this mode.
NumLiensLifted	The number of liens lifted by the transaction.
TotLienAmtLifted	The total lien amount lifted by the transaction.
AddtlDetails	The user specified additional details.
AcctNum	The account number of the part tran
SchemeCode	The scheme code of the account used.
SchemeType	The scheme type of the account used.
TranAmt	The transaction amount in the tran currency of the part tran
RefAmt	The reference amount in the ref currency.
Rate	The rate used in the transaction for converting to the account currency.
ValMode	This will be required to be passed as input to the FTS_RaiseException user hook function if using it.
DrCrInd	"D" for debit and "C" for credit part tran
TranType	The transaction type as given in the TM menu
ExceptionIgnoreFlg	Field required to be passed as input to raise exception user hook.
TranCreationMode	The transaction creation mode.
PST_Flag	The flag of the PST table.
TranSubType	The transaction sub type.
InstrumentType	The type of instrument used like cheque etc
InstrumentDate	The date of the instrument used for the transaction.
InstrumentAlpha	The alpha number of the instrument.
InstrumentNum	The instrument number of the transaction.
ReportCode	The report code used for the transaction.
TranCurrency	The currency code of the transaction.

RefCurrency	The reference currency code.
RateCode	The rate code used for the transaction.
TranDate	The date of the transaction.
ValueDate	The value date of the transaction.
CustomerId	The customer Id of the account.
AcctSolId	The sol_id of the account.
RestrictModifyInd	The restrict or modify indicator.
InstantTodAmt	The Instant TOD amount granted for a part transaction
TodLvlIntFlg	The TOD level interest flag for the Instant TOD

OUTPUTS EXPECTED FROM THE SCRIPT:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

SuccessOrFailure	To check whether the script returned success or failure. "S" for success and "F" for failure. If "F", then the error message given in the next field below will be displayed on the screen and cursor will be positioned at the option code.
ErrorDesc	Any error message in case of script failure.
AddtlDetails	The additional user defined and accepted data if any. Since this mode is only to display the data, no modifications are allowed. Hence, the input addtdetails should be copied to this output field and returned back.
DeleteUAD	The flag to indicate whether to delete the UAD record (User additional detail) or not. Values "Y" for delete and "N" for do not delete. In this mode, this output is not relevant. Any value can be passed back.

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In this mode user can view the additional details specified in the entry mode. He cannot modify the data. In case the data needs to be modified, it can be done as part of part tran modification only.

SAMPLE SCRIPT:

Please refer to script FinTran.sscr in sample/scripts directory.

CHECK PART TRAN

The script will be called in TM menu option in this mode when for each part tran, option code is given as 'K' (Check part tran exceptions) and <KEY-ACCEPT> is pressed. Here the event will be "TM_CHECK_EXCEPTION".

SCRIPT:

The script **FinTran.scr** should exist in the TBA_SCRIPTS directory. Also, the parameter "Use Fin. Transactions script?" must have been set as 'Y' in SRGPM menu option for the scheme of the account involved.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. The inputs can be accessed by referring to the field in the following manner.

BANCS.INPUT.<fieldName>

The input fields available are :

Event	The value of this field will be "TM_CHECK_EXCEPTION" in this mode.
NumLiensLifted	The number of liens lifted by the transaction.
TotLienAmtLifted	The total lien amount lifted by the transaction.
AddtlDetails	The user specified additional details.
AcctNum	The account number of the part tran
SchemeCode	The scheme code of the account used.
SchemeType	The scheme type of the account used.
TranAmt	The transaction amount in the tran currency of the part tran
RefAmt	The reference amount in the ref currency.
Rate	The rate used in the transaction for converting to the account currency.
ValMode	This will be required to be passed as input to the FTS_RaiseException user hook function if using it.
DrCrInd	"D" for debit and "C" for credit part tran
TranType	The transaction type as given in the TM menu
ExceptionIgnoreFlg	Field required to be passed as input to raise exception user hook.
TranCreationMode	The transaction creation mode.
PST_Flag	The flag of the PST table.
TranSubType	The transaction sub type.
InstrumentType	The type of instrument used like cheque etc
InstrumentDate	The date of the instrument used for the transaction.
InstrumentAlpha	The alpha number of the instrument.
InstrumentNum	The instrument number of the transaction.
ReportCode	The report code used for the transaction.
TranCurrency	The currency code of the transaction.

RefCurrency	The reference currency code.
RateCode	The rate code used for the transaction.
TranDate	The date of the transaction.
ValueDate	The value date of the transaction.
CustomerId	The customer Id of the account.
AcctSolId	The sol_id of the account.
RestrictModifyInd	The restrict or modify indicator.
InstantTodAmt	The Instant TOD amount granted for a part transaction
TodLvlIntFlg	The TOD level interest flag for the Instant TOD

OUTPUTS EXPECTED FROM THE SCRIPT:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

SuccessOrFailure	To check whether the script returned success or failure. "S" for success and "F" for failure. If "F", then the error message given in the next field below will be displayed on the screen and cursor will be positioned at the option code.
ErrorDesc	Any error message in case of script failure.
AddtlDetails	The additional user defined and accepted data if any. The input addtldetails should be copied to this output field and returned back as no modifications of addtldetails are allowed in this mode.
DeleteUAD	The flag to indicate whether to delete the UAD record (User additional detail) or not. Values "Y" for delete and "N" for do not delete. In this mode, this output is not relevant. Any value can be passed back.

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In this mode user will have to perform the same validations that he performed during entry mode for raising any exceptions. The same exceptions, if any, must be raised in this mode so that these too appear along with the built-in exceptions for this part tran in the exception display window. There is no data modification in this mode.

SAMPLE SCRIPT:

Please refer to script FinTran.sscr in sample/scripts directory.

3.4 OTHER FINANCIAL TRANSACTIONS

POSTING OF ANY FINANCIAL PART TRAN

The script will be called in this mode whenever any financial part tran is being posted (by any module) in the application. Here the event will be "PTRAN_POSTING".

SCRIPT:

The script **FinTran.scr** should exist in the TBA_SCRIPTS directory. Also, the parameter "Use Fin. Transactions script?" must have been set as 'Y' in SRGPM menu option for the scheme of the account involved.

INPUTS AVAILABLE IN THE SCRIPT:

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. The inputs can be accessed by referring to the field in the following manner.

BANCS.INPUT.<fieldName>

The input fields available are :

Event	The value of this field will be "PTRAN_POSTING" in this mode.
NumLiensLifted	The number of liens lifted by the transaction.
TotLienAmtLifted	The total lien amount lifted by the transaction.
AddtIDetails	The user specified additional details.
AcctNum	The account number of the part tran
SchemeCode	The scheme code of the account used.
SchemeType	The scheme type of the account used.
TranAmt	The transaction amount in the tran currency of the part tran
RefAmt	The reference amount in the ref currency.
Rate	The rate used in the transaction for converting to the account currency.
ValMode	This will be required to be passed as input to the FTS_RaiseException user hook function if using it.
DrCrInd	"D" for debit and "C" for credit part tran
TranType	The transaction type as given in the TM menu
ExceptionIgnoreFlg	Field required to be passed as input to raise exception user hook.
TranCreationMode	The transaction creation mode.
PST_Flag	The flag of the PST table.
TranSubType	The transaction sub type.
InstrumentType	The type of instrument used like cheque etc
InstrumentDate	The date of the instrument used for the transaction.
InstrumentAlpha	The alpha number of the instrument.
InstrumentNum	The instrument number of the transaction.
ReportCode	The report code used for the transaction.
TranCurrency	The currency code of the transaction.

RefCurrency	The reference currency code.
RateCode	The rate code used for the transaction.
TranDate	The date of the transaction.
ValueDate	The value date of the transaction.
CustomerId	The customer Id of the account.
AcctSolId	The sol_id of the account.
RestrictModifyInd	The restrict or modify indicator.
InstantTodAmt	The Instant TOD amount granted for a part transaction
TodLvlIntFlg	The TOD level interest flag for the Instant TOD

OUTPUTS EXPECTED FROM THE SCRIPT:

The output is accessed in the script in the following manner:

BANCS.OUTPUT.<fldName>

SuccessOrFailure	To check whether the script returned success or failure. "S" for success and "F" for failure. If "F" then posting will fail citing the reason given in the field below
ErrorDesc	Any error message in case of script failure – why posting should fail.
AddtlDetails	The additional user defined and accepted data if any. The input addtdetails should be copied to this output field and returned back as no modifications of addtdetails are allowed in this mode.
DeleteUAD	The flag to indicate whether to delete the UAD record (User additional detail) or not. Values "Y" for delete and "N" for do not delete. If "Y" then the additional details stored in the database will be permanently deleted.

DEFAULT LOGIC IN CASE SCRIPT IS NOT PRESENT:

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In this mode user can do additional user specific validations on the part transaction, raise exceptions or errors for exception conditions and use the additional user defined data specified at entry time to do some additional processing (for e.g. initiate a workflow to execute some other menu option).. More than one exception can be raised from within the script. If an error is returned by the script, the part transaction posting will fail and successful entry will go through only on correcting the error condition. If one or more exception is raised even then part tran posting fails. If any additional processing is specified within the script, then the corresponding action is taken.

The Instant TOD amount might have changed between entry and posting. This can happen due to the available amount having changed because of some other transaction or sanction limit/drawing power change. Based on the two inputs, InstantTodAmt and TodLvlIntFlg and


the available amounts present in the user hook getAcctDetailsInRepository, the script can decide whether the user has to visit the Instant TOD details screen again.

SAMPLE SCRIPT:

Please refer to script FinTran.sscr in sample/scripts directory.

4 TRANSACTION UPLOAD SCRIPTS

Scripts related to Menu Option TTUM/TTUMP/BRTTUM/BRTTUMP.

 (will be referred as TTUM now onwards)

EVENT NAME : TRANSFER TRANSACTION UPLOAD.

This event occurs when either of TTUM/TTUMP/BRTTUM/BRTTUMP menu option is invoked for the upload of transaction.

SCRIPT NAMES :

(All the scripts mentioned below are available in TBA_SCRIPTS directory)

1 mcf3009PreLoad.scr

This is a pre-load script for menu option DDUPLOAD.

This contain the values for the different fields of the form. Depending on the menu option different values can be populated for the fields of the form.

2 Ttumupload.scr

This is the script which will be called from the menu option . The name of the script can be mentioned in preload script of TTUM menu option (mcf3009PreLoad.scr).

e.g.

```
sv_a= urhk_TBAF_SetValue("mcf3009.datablk.pg0_script_name|Ttumupload.scr")
```

Ttumupload.scr internally calls one of the following scripts depending on the version of the upload file.

3 parseTTUM1595.scr (for vrp1.5.95)

4 parseTTUM1635.scr (for vrp1.6.35)

5 parseTTUM1638.scr (for vrp1.6.38)

By default Ttumupload.scr is making a call to parseTTUM1638.scr i.e. upload format of version vrp1.6.38 .

If upload format is being used is different than vrp1.6.38 , then corresponding change has to be made in Ttumupload.scr.

The detail of the script is as follows.

6 Ttumupload.scr :

Our objective of this menu option is to upload the transfer transaction records. The file containing the records to be uploaded will be taken as input.

This is a MTT script, which creates the upload transaction.

INPUTS TO THE SCRIPT:

The input to the script is the entire line of the record to be uploaded. This line should conform to the format specified for Menu option: TTUM in upload document.

This is available to the script in repository variable MTT.InputDetails.inputRecordString.

Repository variables	Description
MTT.InputDetails.inputRecordString	The entire line from the file containing the records to be uploaded. This line should conform to the format specified for menu option : DDUPLOAD in upload document.

OUTPUT EXPECTED FROM THE SCRIPT :

None.

7 parseTTUM1595.scr

8 parseTTUM1635.scr

9 parseTTUM1638.scr

Above scripts contain the parsing logic for the record to be uploaded.

This will parse the input record and give values of different fields.

This will be called from script Ttumupload.scr.

This script after parsing for different fields , will validate those fields. In case of failure it will populate MTT.Error.Description with error message and make a call to MTT user hook urhk_MTTS_ReportErrorCondition(""). This will also populate BANCS.OUTPUT.successOrFailure = "F" in case of failure.

INPUTS TO THE SCRIPT:

The input to the script is the entire line of the record to be uploaded.

This is available to the script in repository variable MTT.InputDetails.inputRecordString.

Repository variables	Description
MTT.InputDetails.inputRecordString	The entire line from the file containing the records to be uploaded. This line should conform to the format specified for menu option : DDUPLOAD in upload document

OUTPUT EXPECTED FROM THE SCRIPT:

Repository variables	Description
BANCS.OUTPUT.successOrFailure	The value of the field will be "F" in case of failure.
MTT.Error.Description	This will contain the error description.
MTT.PartTranDetails.Account	Customer account
MTT.PartTranDetails.ReportCode	Report code for Customer account
MTT.PartTranDetails.Refnum	Ref num for Customer account.
MTT.PartTranDetails.Particulars	Tran particulars
MTT.PartTranDetails.RefAmount	Ref amount
MTT.PartTranDetails.RefCurrency	Ref currency
MTT.PartTranDetails.InstrmntType	Instrmnt type
MTT.PartTranDetails.InstrmntAlpha	Instrmnt alpha
MTT.PartTranDetails.InstrmntDate	Instrmnt date
MTT.PartTranDetails.InstrmntNum	Instrmnt num
MTT.PartTranDetails.RateCode(not in vrp1595)	Rate code .
MTT.PartTranDetails.Rate(not in vrp1595)	Rate.
MTT.PartTranDetails.ValueDate(not in vrp1595)	Value date of the transaction.

If the part tran is HO part tran following additional fields will be delivered.

Repository variables	Description
MTT.HOAddnlDetails.HOTranType	Tran type of transaction (originating , responding , Reversal)
MTT.HOAddnlDetails.CategoryCode	Tran category code.
MTT.HOAddnlDetails.ToFromBankCode	To bank code
MTT.HOAddnlDetails.ToFromBranchCode	Branch code
MTT.HOAddnlDetails.AdvcExtnCntrCode	Extn counter code.
MTT.HOAddnlDetails.AdviceInd	Advice indicator.
MTT.HOAddnlDetails.AdviceNumber	Advice Number
MTT.HOAddnlDetails.BarAdviceDate	Date of Advice .
MTT.HOAddnlDetails.BillNumber	Bill no.
MTT.HOAddnlDetails.HeaderTextCode	Header Text code
MTT.HOAddnlDetails.HeaderFreeText	Header Free Text .

MTT.HOAddnlDetails.Particular1	Particulars 1 of HO.
MTT.HOAddnlDetails.Particular2	Particulars 2 of HO.
MTT.HOAddnlDetails.Particular3	Particulars 3 of HO.
MTT.HOAddnlDetails.Particular4	Particulars 4 of HO.
MTT.HOAddnlDetails.Particular5	Particulars 5 of HO.
MTT.HOAddnlDetails.amtLine1	Amount Line 1.
MTT.HOAddnlDetails.amtLine2	Amount Line 2.
MTT.HOAddnlDetails.amtLine3	Amount Line 3.
MTT.HOAddnlDetails.amtLine4	Amount Line 4.
MTT.HOAddnlDetails.amtLine5	Amount Line 5.
MTT.HOAddnlDetails.BatRemarks	Bar remarks
MTT.HOAddnlDetails.PayeeAcct	Payee account.
MTT.HOAddnlDetails.RecvAdvcNum	Received advc num.
MTT.InputDetails.RecvAdvcDate	Received advc date.
MTT.HOAddnlDetails.OrgTranDate	Originating tran date for reversal
MTT.HOAddnlDetails.OrgTranId	Originating tran id for reversal.
MTT.HOAddnlDetails.OrgPTranSrlNum	Origination part tran srl num.
MTT.HOAddnlDetails.free_text	Free text.

5 EVENT CHARGES RELATED SCRIPTING EVENTS

A unified charge setup, calculation infrastructure has been built into Finacle™ enabling the user for greater customisation of charges. The following are the service charges that can be setup, calculated and charged by the bank on the following events (services rendered to the customer).

- 10 Demand draft related
- 11 DD issue
- 12 DD cancellation
- 13 DD duplicate issue
- 14 HO Charges –
- 15 Originating credit
- 16 Originating reversal
- 17 Clearing functions
- 18 Inward rejections
- 19 Outward rejections through Inward Clearing
- 20 Outward rejections through CVDOR menu
- 21 MICR charges
- 22 Outward Clearing
- 23 Account upkeep
- 24 Account Opening
- 25 Account Maintenance
- 26 Account Closure
- 27 Ledger Folio charges - Transactions
- 28 Stop Payment charges
- 29 Minimum Balance charges – Minimum balance not maintained
- 30 Inactive account maintenance
- 31 Dormant account maintenance
- 32 Commitment charges – Non utilisation of credit line
- 33 Bank Guarantee Charges

- 34 BG issue
- 35 BG modification of clauses
- 36 Documentary Credit related charges
- 37 DC Issue
- 38 Advise of Documentary Credits Charges
- 39 Closure of Documentary Credits Charges
- 40 Amendment of Documentary Credits Charges
- 41 Transfer of Documentary Credits Charges
- 42 Reinstatement of Documentary Credits Charges
- 43 Utilisation Of Documentary Credits without a Bill
- 44 General Charges

These charges are categorised into Four types. Service triggered charges, Deferred charges, Batch charges and non-standard event triggered charges (general charges).

Service triggered charges are computed and charged at the time of delivery of the service. A DD Issue charge is calculated and charged from the customer who has requested for the DD. These become additional part transactions along with the part transactions for DD Issue. Each such Service (Event) is "Known" to the application and is available in PTTM menu option as "Event Type".

Deferred charges refer to service charges that are not charged at the time of delivering the service, but deferred to a later period. Charges which are calculated at periodic intervals – Account opening, Pass sheet printing. For eg: if DEFCALC is set up for daily calculation, all accounts opened that day and all accounts for which pass sheets were printed that day would be charged for the service.

Batch charges are similar except batch charges are for a set of similar services and are setup to be charged for a period of time. For eg: Ledger Folio entries for three months can be charged based on the number of entries. Charges which fall under this umbrella are:

Ledger Folio, Minimum balance, Account maintenance, Inactive account maintenance, Domant account maintenance and Commitment charges.

Over and above this, General Charges is provided as a way of calculating non-standard charges (like locker charges etc.). These are charged on-line from General charges menu option.

Scripts can be used to tailor the charges over and above the functionality provided by the existing setup menu options (Please refer to the document on Event Charges for further details). Each charge here is associated with an Event Type which is "known" to the

application. Link to a script can be established in PTTM menu option for each of the services mentioned above (and an GCHRG event for general charges). This script has a set of common repository variables which are always passed to the script.

For all scripts mentioned under Event Based charges section, the BANCS.INPUT class has a common set of fields. The output always is a currency code and an amount. The input list which is common to all the scripts set up through PTTM is given below.

The defined common inputs for all the scripts set up through PTTM are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. One can access the inputs by referring to the fields in the following manner.

BANCS.INPUT.<fieldName>

Event Type	EventType
Event Id	EventId
Part Tran Business Type	BusinessType
Charge Level Code	CustomerChargeCode
Minimum Charge Amount	MinimumChargeAmount
Minimum Charge Currency	MinimumChargeAmountCrncy
Maximum Charge Amount	MaximumChargeAmount
Maximum Charge Currency	MaximumChargeAmountCrncy
Rate Code	RateCode
Discount Percentage	DiscountPercentage
Round Off Indicator	RoundOffIndicator
Round Off Value	RoundOffValue
Charge Calculation Currency	ChrgCalcCrncy
Charge Collection Currency	ChrgCollCrncy
Percentage	AmtPcnt
Fixed Charge	FixedAmt
Context Currency	InputCrncy
Context Amount	InputAmount

5.1 SCRIPTS FOR CALCULATION OF CHARGES FOR DEMAND DRAFTS

These scripts can be used to generate the charges for DD issue, DD cancellation & DD duplicate issue.

SCRIPT

The script name can be anything but it should be present in TBA_SCRIPTS directory.

INPUT TO THE SCRIPT

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. One can access the inputs by referring to the fields in the following manner

BANCS.INPUT.<fieldName>

DD Amount	Dd_amt
-----------	--------

OUTPUT OF THE SCRIPT

The Output of the script will be available in OUTPUT class of BANCS repository. The values to be populated are

BANCS.OUTPUT.<fieldName>

Charge Amount	CollAmount
Charge Currency	CollCrncy

DEFAULT LOGIC IN CASE SCRIPT IS NOT AVAILABLE

No default logic is provided.

6 TYPICAL USAGE SCENARIO(S):

Sometimes it might be required that the discount percentage for a customer is dependent on the Demand Draft amount. In all these cases the script can be used to arrive at the charges.

SAMPLE SCRIPT

6.1 SCRIPTS FOR CALCULATION OF MICR CHARGES

These scripts can be used to generate the charges for Issue of cheque books.

SCRIPT:

The script name can be anything but it should be present in TBA_SCRIPTS directory.

INPUT TO THE SCRIPT

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. One can access the inputs by referring to the fields in the following manner

BANCS.INPUT.<fieldName>

Number of Leaves	numOfLvs
Account Number (acid)	issueAcctNum

OUTPUT OF THE SCRIPT

The Output of the script will be available in OUTPUTclass of BANCS repository. The values to be populated are

BANCS.OUTPUT.<fieldName>

Charge Amount	collAmount
Charge Currency	collCrcncy

DEFAULT LOGIC IN CASE SCRIPT IS NOT AVAILABLE

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

The event id set up in PTTM does not take into consideration the number of leaves in a chequebook while calculating the charges. So in all cases where the charge amount is dependent on number of leaves the script will have to be used.

SAMPLE SCRIPT

Please refer to micrchrg.sscr present in sample/scripts directory

6.2 SCRIPTS FOR CALCULATION OF STOP PAYMENT CHARGES

These scripts can be used to generate the charges for stop payment processing.

SCRIPT

The script name can be anything but it should be present in TBA_SCRIPTS directory.

INPUT TO THE SCRIPT

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. One can access the inputs by referring to the fields in the following manner

BANCS.INPUT.<fieldName>

Number of Leaves	numOfLvs
Account Number (acid)	issueAcctNum
Cheque Amount	chqAmt
Account Balance	issueAcctBal

OUTPUT OF THE SCRIPT

The Output of the script will be available in OUTPUT class of BANCS repository. The values to be populated are

BANCS.OUTPUT.<fieldName>

Charge Amount	collAmount
Charge Currency	collCrncy

DEFAULT LOGIC IN CASE SCRIPT IS NOT AVAILABLE

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

The event id set up in PTTM does not take into consideration the number of leaves while calculating the charges. So in all cases where the charge amount is dependent on number of leaves the script will have to be used. Another case would be to charge different amount if the cheque is stopped due to lack of balance in the account.

SAMPLE SCRIPT

Please refer to stpmthrg.sscr present in sample/scripts directory

6.3 SCRIPTS FOR CALCULATION OF ACCOUNT CLOSURE CHARGES

These scripts can be used to generate the charges for premature account closure.

SCRIPT

The script name can be anything but it should be present in TBA_SCRIPTS directory.

INPUT TO THE SCRIPT

The defined inputs are available in a repository called BANCS, a class called INPUT and with the field names mentioned below. One can access the inputs by referring to the fields in the following manner

BANCS.INPUT.<fieldName>

Scheme Code	schmCode
Account Number (foracid)	account

OUTPUT OF THE SCRIPT

The Output of the script will be available in OUTPUT class of BANCS repository. The values to be populated are

BANCS.OUTPUT.<fieldName>

Charge Amount	collAmount
Charge Currency	collCrncy

DEFAULT LOGIC IN CASE SCRIPT IS NOT AVAILABLE

No default logic is provided.

TYPICAL USAGE SCENARIO(S):

In all cases where the charge to be collected depends on the scheme code to which the account belongs the script can be used. Another case would be for the accounts which have the same charge level code but a few of the accounts have to be charged differently this script can be used.

SAMPLE SCRIPT

Please refer to close_chrg.sscr present in sample/scripts directory

6.4 MAHA TRANSACTION TEMPLATES

The accounting policies for customers differ across banks. Finacle™ provides a utility called the Maha Transaction Template (MTT), which allows you to customise the transactions generated during different events. Through this utility, you can:

- Define one or more transactions each with a user defined set of part transactions. The only restriction is that the transaction should be balanced.
- Perform mathematical operations to calculate amounts.
- Define a complete transaction or a part of the transaction. This decision is controlled in Finacle™ and is dependent on the event.
- Consolidate one or more part transactions given a set of identical transactions. All transactions will be merged into one and all part transactions where consolidation is required will be made into a single part transaction. You can skip a whole transaction while merging a set of transactions.
- Define contra or consolidated contra part transactions for one or more part transactions.
- Customise part transactions through account definition, amount derivation logic, remarks, particulars and report codes population.
- Amount derivation logic includes amount table code set-up based calculation.
- Flag error conditions with suitable error messages, which will be printed in the error report.

Example:

Consider an account where interest has been collected in advance and credited to an "Advance interest account".

During interest calculation, the interest amount is more than that collected in advance. Hence the advance amount should be recognised for profit and loss and the remaining interest should be taken from the customer's operative account.

During interest calculation, if the account balance in the operative account is less than the minimum balance for that account, then the amount above the minimum balance limit should be debited from the operative account and the remaining amount should be debited from the shortfall account.

In such cases, the MTT can be used to perform the above operations.

7 SECTION OBJECTIVE

The Objective of this section is to introduce the user to the concepts of Maha Transaction Template (Referred as MTT in short), The transaction creation events where the MTT can be invoked and the user hooks available to MTT. The pre requisite must be the user must have sufficient knowledge of Finacle™ and Scripting in Finacle™.

8 PROCESSES INVOLVED

MTT is basically a combination of the record layout concept of MRT and the functionality of PTT implemented using Script Engine. The limitation in PTT of accepting only one input amount based on which the additional set of part transactions have to be derived are overcome in MTT. In this template, the part transaction definition can be based on more than one amount and can also be conditional. The script engine in MTT makes use of special script hooks to define a complete transaction. All the flexibility of the script, like conditional statements, amount manipulation etc. is directly available in MTT.

Each script hook in MTT reads data from the script from a special class defined in the MTT repository. The field **Repository Name**, which is default populated with the value MTT and cannot be modified, is a mandatory field for all the user hooks. The user hooks that are mandatory in any MTT script are described below.

9 MTTUSERHOOKS

9.1 DEFINETRANSACTION

You can use this script hook to initiate a transaction.

SYNTAX:

sv_a = urhk_MTTs_DefineTransaction (Variable)

The variable can be a scratch pad variable (sv_b) or a string like XYZ.

FUNCTIONALITY:

This function allows the script to initiate a financial transaction by defining some of its attributes. A name is used while the transaction is being defined so that multiple transactions can be defined in the same script. Each transaction having a unique identifier string value.

INPUT:

Input String contains the identification of the transaction. This identifier should be referred to while defining the part-transaction to link the part-transaction to the transaction.

Example - "XYZ"

It accepts the following inputs from the TranDetails class:

Field Name	Description
TranDateDiff	The date of the transaction. This has to be specified as the number of days relative to the BOD date. The value for this field will be 0 if the transaction is being created as of that day.
TranType	The tran type for the particular event. This field does not support back dated transactions.
TranSubType	The tran sub type for the particular event.
TranEventType	The event type of the particular transaction.
Remarks	Any remarks about the transaction.
IgnoreExcp	Flag which indicates if exceptions can be overridden or not.
Consolidate	Indicates that some of the part transactions under the particular transaction will be consolidated.
ReversalFlg	Flag which indicates if a reversal transaction is to be generated for the current transaction or not.

Reversal DateDiff	The tran date for the reversal transaction. This has to be specified as the number of days relative to the BOD date. This field does not support back dated transactions.
-------------------	---

Output:

The return value will be 0 in case the transaction definition is accepted. If not, it will be 1.

☞ For a sample script refer to the sample scripts directory in *\$TBA_PROD_ROOT/sample/scripts/BillTran.sscr*.

9.2 DEFINEPARTTRAN

This script hook accepts all data required to define a complete part transaction.

SYNTAX:

sv_a = urhk_MTTs_DefinePartTran (Variable)

Where the variable can be a scratch pad variable (sv_b) or a string like XYZ.

FUNCTIONALITY:

This function allows the script to initiate a financial transaction by defining some of its attributes. The attributes are first made available in the MTT.PartTranDetails class. A call is made to define the part transaction after that. The class variables are identified in the table below.

INPUT:

The input string contains the identification of the part-transaction. This is used to consolidate multiple part-transactions into one part-transaction for all part-transactions where consolidate flag is set.

Example - "XYZ"

It accepts the following fields from the PartTranDetails class:

Field Name	Description
Account	The account number (foracid) of the account to be debited/credited
RefAmount	The amount to be debited/credited. Positive amount indicates credit, while a negative amount indicates a debit.
ValueDateDiff	The value date of the part-transaction in terms of number of days from BOD date (+ or -)
Refnum	Corresponds to ref_num in DTD table. Max 20 characters
RefCurrency	The currency that the amount is in
RateCode	Rate code to be used in case the account currency is not them same as the amount currency
Rate	The rate of conversion can be specified instead of a rate code. If both are specified, then the rate is used.

Field Name	Description
ReversalDateDiff	The date in terms of number of days past BOD date.
Remarks	Remarks for the part-transaction. Max of 30 characters
Particulars	Particulars for the part-transaction. Max of 50 characters
CustomerId	Customer Id of the part-transaction if other than customer who owns the account
ReportCode	Valid report code for the part-transaction
PrintAdviceInd	Flag indicating whether advice to customer needs to be printed or not.
PtranBusinessType	This is specific to the MTT event. Each event will describe the additional processing that happens for special values of this field, in screvent.doc
Consolidate	Flag indicating if this part-transaction has to be consolidated with other part-transactions which have the same part-transaction identifier
ProxyPstgInd	Flag indicating whether this part-transaction is allowed to be proxy-posted, if required.
TODRefType	Reference type of the TOD (which is set up at scheme level) that should be used in case of insufficient balance in the account. If this field is set, automatic TOD will be generated in case of shortfall.
TODAmount	The amount of TOD to be granted. The TOD amount can be calculated by applying user logic in the script and populated into this field so that the TOD will be granted for the exact amount populated into this field. To grant TOD it is not only sufficient to populate TODAmount, TODRefType should also be populated. If the TOD amount is not populated in the script but TODRefType is alone populated then system will calculate the TOD amount (this amount is nothing but insufficient balance required to post the current part transaction) that is to be granted.
CarveAmtFlg	If set to "Y", the system will automatically carve the amount during transaction creation that will be automatically removed when the part-transaction is posted. This is used to lock sufficient funds in the account, even if posting fails.
TranIdentifier	The identifier of the transaction to which this part-transaction belongs
ProxyReversalDate	In case the transaction is being proxy posted and the reversal should happen on a future date (this is also taken as the difference between BOD date).
ProxyReversalValueDate	In case the transaction is being proxy posted and the reversal transaction should have a different value date.
ProxyEventType	The proxy event type for that part transaction if the part transaction is being proxy posted.
ProxyForacid	The proxy account on which the proxy transaction has to be created.

Additional Part transaction Details for HO Accounts (**HOC Scheme Types**)

MTT accepts the following fields from the **HOAddnlDetails** class:

Field Name	Description
HOTranType	The transaction type of the HO transaction (Originating , responding)
OrgTranId	Original transaction Id (valid for HO reversal transactions)
OrgTranDate	Original transaction date (valid for HO reversal transactions)
OrgPTranSrINum	valid for HO reversal transactions
CategoryCode	Transaction category code
ToFromBankCode	To from Bank code
ToFromBranchCode	To from branch code
AdvExtnCntrCode	Extension counter code
AdviceInd	advice indicator of the HO part transaction
AdviceNumber	Advice number of the HO part transaction
BarAdviceDate	Bar /advice date of the HO part transaction.
BillNumber	Bill number of the HO part transaction
HeaderTextCode	Header text code
RecvAdvDate	Received Advice date of the HO part transaction
RecvAdvNum	Received advice number
DuplicateAdvFlg	Flag , whether the advice is duplicate or not.
Particular1	Particulars field of Ho part transaction
Particular2	Particulars field of Ho part transaction
Particular3	Particulars field of Ho part transaction
Particular4	Particulars field of Ho part transaction
Particular5	Particulars field of Ho part transaction
amtLine1	Amount
amtLine2	Amount
amtLine3	Amount
amtLine4	Amount
amtLine5	Amount
BatRemarks	Remarks of the HO part transaction Additional details.
PayeAcct	The account belonging to the other bank/branch
HeaderFreeText	Header free text of the HO part transaction.

Additional Part Transaction Details for Loan Accounts (**LAA Scheme Types**)

MTT accepts the following fields from the **LAAddnIDetails** class:

Field Name	Description
DemandFlowId	Valid loan demand flow id. This flow id should be of 'Collection' flow nature for credit part transactions. Basically this will be the collection flow id, the offset method and offset sequence associated with this flow id will be used to squaring of demand in particular order. For debit part transactions the flow id should be of 'Demand' flow nature and of 'Interest'/'Bank Charges'/'Other Charges' flow type. In this case a demand will get raised with the given flow id.

Field Name	Description
TypeOfDemand	<p>The valid values P - Principal Demands only, I - Interest Demands only, B - Bank Charge Demands Only, O - Other Charge Demands Only and A - All demands</p> <p>This field is applicable only for credit part transactions. In addition to the offset method and offset sequence associated with collection flow id, the value specified in this field will be used to identify demands that are to be squared off. This field will have significant with interest route flag of loan account.</p> <p>If interest route flag is 'L' – Keep interest liability in Loan Account, then this field can have any of the above given values. If interest route flag is 'O' – Keep interest liability in office a/c, then this field can have only 'P' as valid value.</p>
OriginOfTran	<p>The valid values 'INT', 'LDS'. This field is applicable only for credit part transactions. This field is also associated with interest route flag of Loan account. If interest route flag is 'O' for loan account then any direct credits to loan account (through TM) can be controlled by an exception. Moreover, if the credit transaction to loan account is credited by Interest calculation process or Loan Demand Satisfaction Process then this exception will not be raised. So to avoid the exception either 'INT', 'LDS' can be populated into this field.</p>

Additional Part Transaction Details for Loan Interest Accounts (**OAB Scheme Type of accounts that is partitioned by partition type 'LOANS'**)

MTT accepts the following fields from the **LAAddnlDetails** class:

Field Name	Description
DemandFlowId	<p>Valid loan demand flow id. This flow id should be of 'Collection' flow nature for credit part transactions. Basically this will be the collection flow id, the offset method and offset sequence associated with this flow id will be used to squaring of demand in particular order. For debit part transactions the flow id should be of 'Demand' flow nature and of 'Interest' flow type. In this case an interest (Normal/Penal) demand will get raised with the given flow id.</p>
TypeOfDemand	<p>The valid values I - Interest Demands only,</p> <p>This field is applicable only for credit part transactions. In addition to the offset method and offset sequence associated with collection flow id, the value specified in this field will be used to identify demands that are to be squared off.</p>

Field Name	Description
LoanAcctId	Loan account id on behalf of which transaction is taking place to Loan Interest Account. During interest calculation process, interest is calculated for this loan account but interest debit is taking place to loan interest account and during loan demand satisfaction process interest collection is taking place to this loan account and actual credit transaction is taking place to loan interest account.
OriginOfTran	The valid values 'INT', 'LDS'. This field is applicable only for credit part transactions. This field is also associated with interest route flag of Loan account. If interest route flag is 'O' for loan account then any direct credits to loan account (through TM) can be controlled by an exception. Moreover, if the credit transaction to loan account is credited by Interest calculation process or Loan Demand Satisfaction Process then this exception will not be raised. So to avoid the exception either 'INT', 'LDS' can be populated into this field.

Additional Part Transaction Details for Deposit Accounts (**TDA Scheme Types**)

MTT accepts the following fields from the **TDAddnlDetails** class:

Field Name	Description
TDFlowCode	Valid deposit flow id. The transaction that is taking place to deposit account is under this flow id. Interest calculation process can have following flow id as valid values. II - Interest Inflow IO - Interest Outflow CI - Consolidate Interest Inflow CO - Consolidate Interest Outflow

Each part transaction that is defined should also have a unique name and should be passed as an argument to the user hook. Each part transaction has a **TranIdentifier** field that contains the name of the transaction. Using this, you can define multiple transactions in a given script. A negative amount indicates a debit part transaction while a positive amount includes a credit amount transaction.

OUTPUT:

The return value will be 0 in case the transaction definition is accepted. If not, it will return 1.

Example

```
#The following defines a commission part transaction based on Bill Amount, Bill Currency
and Commission Code
if (MTT.InputDetails.CommissionCode != "") then
BANCS.INPARAM.InputAmount=MTT.InputDetails.BillAmount
BANCS.INPARAM.CurrencyCode=MTT.InputDetails.HomeCurrency
BANCS.INPARAM.AmountTableCode=MTT.InputDetails.CommissionCode
sv_r = urhk_B2k_GetASTMAmount("")
sv_z = CDOUBLE(BANCS.OUTPUTPARAM.OutputAmount)
MTT.PartTranDetails.RefAmount=CDOUBLE(BANCS.OUTPUTPARAM.OutputAmount)
MTT.PartTranDetails.Account=MTT.InputDetails.CommAcct
sv_z = MTT.InputDetails.CommAcct
MTT.PartTranDetails.RefCurrency=MTT.InputDetails.HomeCurrency
MTT.PartTranDetails.ValueDateDiff="0"
MTT.PartTranDetails.Consolidate="N"
MTT.PartTranDetails.TranIdentifier="BillCharges"
MTT.PartTranDetails.PTranBusinessType="5"
sv_r = urhk_MTTs_DefinePartTran("COMMISSION")
# "COMMISSION" is used to refer to this part transaction later.
endif
```


9.3 POPULATEINPUTDETAILS

This script hook populates the event specific fields in the InputDetails class for the next entity that needs processing. This is needed at the beginning of the script hook and in each loop, if multiple entities need to be processed at a time. This script hook requires no inputs. The return value of the hook should be checked for each of the items to be processed.

SYNTAX:

```
sv_a = urhk_MTTS_PopulateInputDetails ("")
```

FUNCTIONALITY:

This function gets all the details put out by the MTT event in repository variables. The details depend on the current event that is in process. This is equivalent to defining the transaction header details in some respects.

INPUT:

There is no input to this function.

OUTPUT:

The return value will be zero (0) in case the processing is successful. If not, it will return 1. The actual fields depend on the event.

☞ For a sample script refer to the sample scripts directory in *\$TBA_PROD_ROOT/sample/scripts/BillTran.sscr*.

9.4 ENDDEFINITION

This script hook is required if it has a loop for processing multiple entities. It signifies the end of the complete definition for a given entity and the start of the processing for the next entity. This script hook also requires no inputs.

SYNTAX:

```
sv_a = urhk_MTTS_EndDefinition ("")
```

FUNCTIONALITY:

This function signals the end of definition for a part-transaction or transaction. It is needed only if the script has a loop for processing multiple entities. This hook ensures that a part transaction is created in internal storage variables within the application based on the MTT.PartTranDetails class variables populated.

INPUT:

There is no input for this function.

OUTPUT:

The return value will be 0 in case the processing is successful. If not, it will return 1.

For a sample script refer to the sample scripts directory in \$TBA_PROD_ROOT/sample/scripts/BillTran.sscr.

9.5 PROGRAMMING HINTS FOR MTT

The following things should be ensured while using the user hook scripts in MTT:

- The PopulateInputDetails class should be called before accessing any of the variables in the InputDetails class.
- A transaction should be defined before creating any part transaction for it.
- All the required fields should be explicitly assigned values before each call to the user hook. This is because each hook clears all the variables of the corresponding class.

10 APPENDIX**10.1 FIELDS IN STDIN CLASS IN BANCS REPOSITORY**

#	FIELD NAME	VALUE CONTAINED
1.	"languageCode"	This field contains the value of the language code of the user e.g. INFENG
2.	"userId"	The userid of the user who executed the script
3.	"onlineOrBatch"	Whether this script is being executed through a batch program or online ("O" or "B")
4.	"userWorkClass"	The workclass of the userid who executed the script from UPM
5.	"menuOption"	The menu option which called this script
6.	"homeCrncyCode"	The home currency of the data center from SCFM
7.	"homeCrncyAlias"	The home currency alias

#	<i>FIELD NAME</i>	<i>VALUE CONTAINED</i>
8.	"CurrentBancsVersion"	The Finacle™ version
9.	"myBankCode"	The bank code of the database
10	"myBrCode"	The branch code of the SOL
11	"myExtCode"	The Extentison counter
12	"mySolId"	The SOLID
13	"mySolAlias"	The SOLALIAS
14	" mySolDesc"	The description for the Sol as specified in SCFM
15	" homeSolId"	The Sol to which the User belongs
16	" homeSolAlias"	The Home SOL Alias
17	" homeSolDesc"	The Home Sol description as specified in SCFM.
18	"dcAlias"	The DC ALIAS
19	"SBString"	The value of custoption for SBSTING
20	"CAString"	The value of custoption for CASTING
21	"LLString"	The value of custoption for LLSTING
22	"CCString"	The value of custoption for CCSTING
23	"sysDate"	The system date of the machine
24	"BODDate"	The current BOD date of the SOL
25	"termClass"	The terminal class from TPM
26	" moduleIdentity"	The module which is calling the Script
27	"TestFlg"	Whether the script is being invoked in test mode. E.g through menu option "SCRIPT"
28	"WFflg"	Whether the script is a workflow script
29	"ScriptName"	The name of the script