1.1

ARRANGEMENT ARCHITECTURE

Product Setup & Technical Guide

Amendment History

Version	Date	Author / Changed By	Status
0.1	25th May 2015	Dinesh BR	Initial Draft
0.1	25th May 2015	Iyyappan R	Initial Draft
0.2	23rd July 2015	Dinesh BR	Updated sections – 2.2, 3.1, 7.2 Included sections – 5, 8.1
0.2	23rd July 2015	Krishnaveni	Reviewed

Table of Contents

1. Introduction

2. Components of AA

    2.1 Flow of AA

    2.2 Product Line

    2.3 Product Group

    2.4 Product

    2.5 Product Class

    2.6 Product Conditions

3. Defining conditions

    3.1 Activity API

    3.2 Activity Presentation

    3.3 Charge

    3.4 Activity Charges

    3.5 Activity Mapping

4. Activity setup flow of AA

5. Adding Local Reference Field

6. New Arrangement Architecture

    6.1 Create New Arrangement Architecture

    6.2 Find AA loan / authorize the AA Loan

7. Commonly used Insert file and common variables in AA

    7.1 Insert files

    7.2 Common variables

    7.3 Property classes

8. AA at Subroutine level

    8.1 Core APIs

    8.2 Reading the property class

8.3 RAISE conditions

8.4 Common variables

8.5 Select command in the AA record

9. Updating Fields in AA

9.1 Authorizing the record

9.2 Updating existing record

9.3 Updated Restructure date & End date

10. T24 Browser Level debug in TAFJ for AA routine

10.1 Browser listener

10.2 Variables tab

1. Introduction

This document helps the user to understand about the functionality of Arrangement Architecture (AA) and flow of product setup and technical aspects in AA.

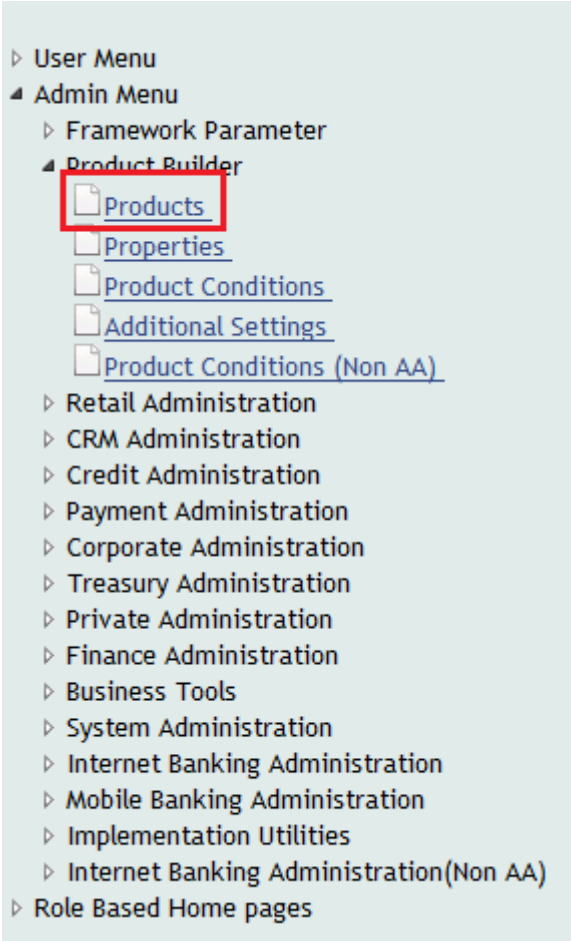
2. Components of AA

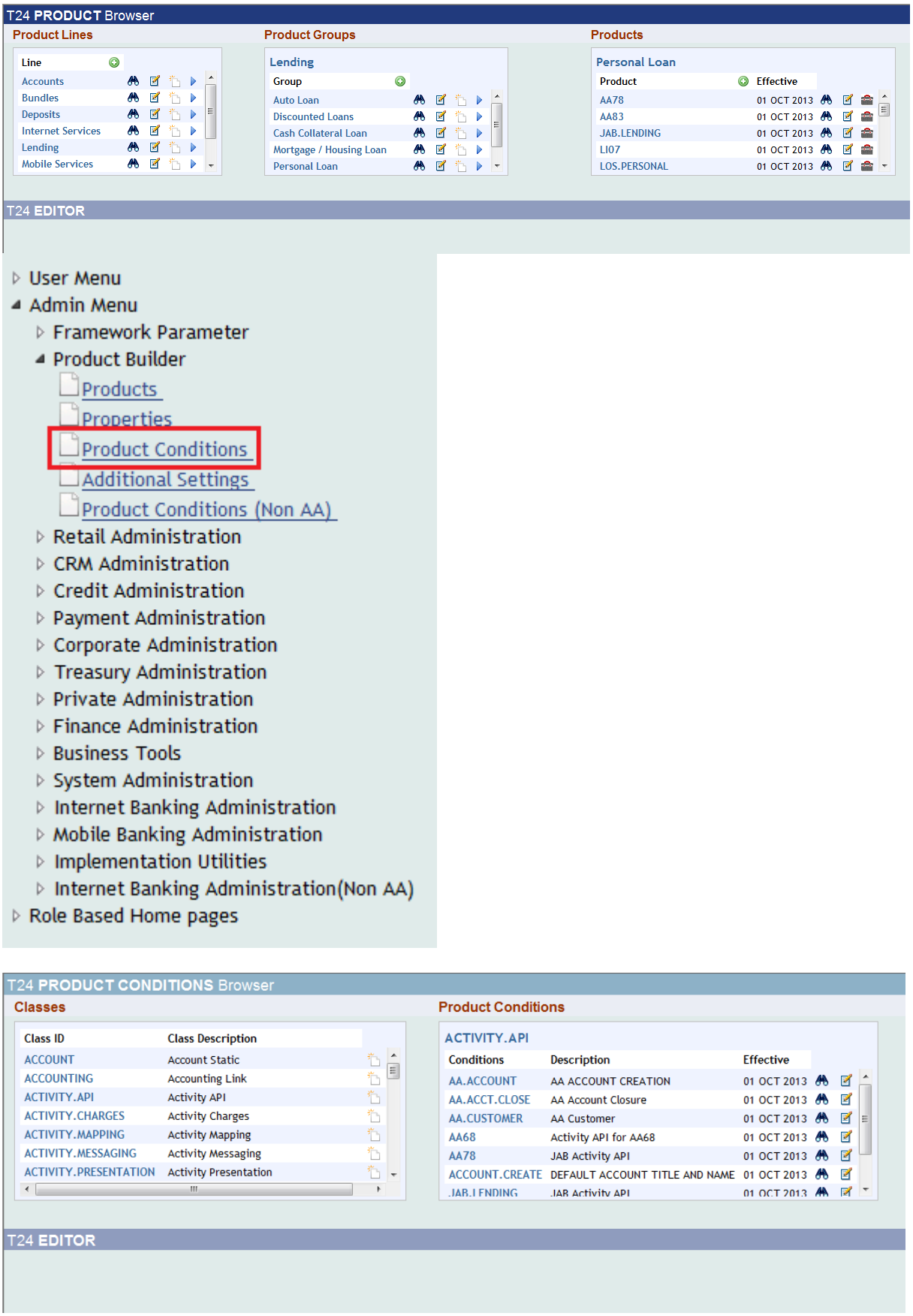
In AA, we have 5 major components are below,

Product Line
Product Group
Product
Property class
Product conditions

2.1 Flow of AA

From 5 major components, 3 are available under "**Products**" link and remaining 2 under "**Product Conditions**".




T24 PRODUCT Browser
Product Lines
Line
Accounts
Bundles
Deposits
Internet Services
Lending
Mobile Services
Product Groups
Lending
Group
Auto Loan
Discounted Loans
Cash Collateral Loan
Mortgage / Housing Loan
Personal Loan
Products
Personal Loan
Product
Effective
AA78 01 OCT 2013
AA83 01 OCT 2013
JAB.LENDING 01 OCT 2013
LI07 01 OCT 2013
LOS.PERSONAL 01 OCT 2013
T24 EDITOR
User Menu
Admin Menu
Framework Parameter
Product Builder
Products
Properties
Product Conditions
Additional Settings
Product Conditions (Non AA)
Retail Administration
CRM Administration
Credit Administration
Payment Administration
Corporate Administration
Treasury Administration
Private Administration
Finance Administration
Business Tools
System Administration
Internet Banking Administration
Mobile Banking Administration
Implementation Utilities
Internet Banking Administration(Non AA)
Role Based Home pages

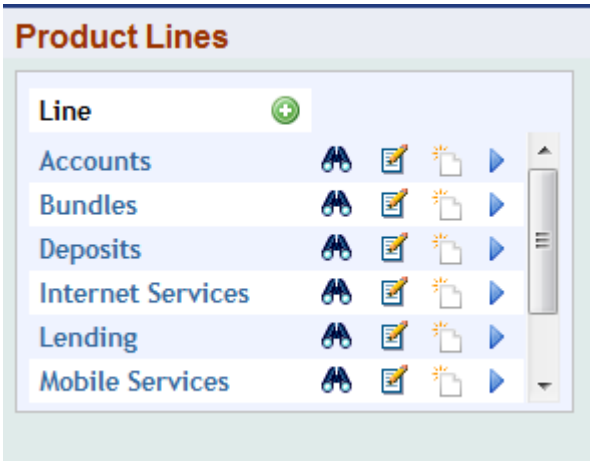
T24 PRODUCT CONDITIONS Browser
Classes
Class ID
Class Description
ACCOUNT Account Static
ACCOUNTING Accounting Link
ACTIVITY.API Activity API
ACTIVITY.CHARGES Activity Charges
ACTIVITY.MAPPING Activity Mapping
ACTIVITY.MESSAGING Activity Messaging
ACTIVITY.PRESENTATION Activity Presentation
Product Conditions
ACTIVITY.API
Conditions
Description
Effective
AA.ACCOUNT AA ACCOUNT CREATION 01 OCT 2013
AA.ACCT.CLOSE AA Account Closure 01 OCT 2013
AA.CUSTOMER AA Customer 01 OCT 2013
AA68 Activity API for AA68 01 OCT 2013
AA78 JAB Activity API 01 OCT 2013
ACCOUNT.CREATE DEFAULT ACCOUNT TITLE AND NAME 01 OCT 2013
T24 EDITOR

In the above components, Product Line and Product Group will be created only by Temenos. In other components, we will be able to define conditions.

2.2 Product Line

Product Line is a high level definition of the business component which was created by Temenos. We can't able to create or modify anything in this Product Line.

It is where we select the purpose of what we are going to do. If we wish to get a loan, then we will go with "Lending" product line. If we wish to deposit, then we select "Deposits" product line.



2.3 Product Group

Product group provides basic shape to the product. This is where we will select which loan we are going to take. For example: if we wish to take personal loan, we will select "Personal Loan" product group.

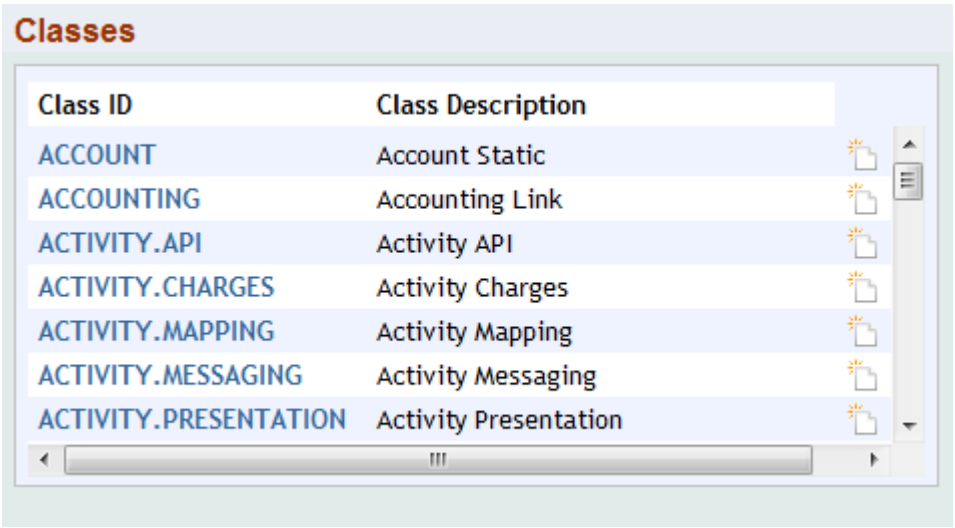
2.4 Product

This is where the user links the desired properties to appropriate conditions.

For the product group we selected, we will have different products to use. In this product, we will define the conditions about how the product should behave.

2.5 Property Class

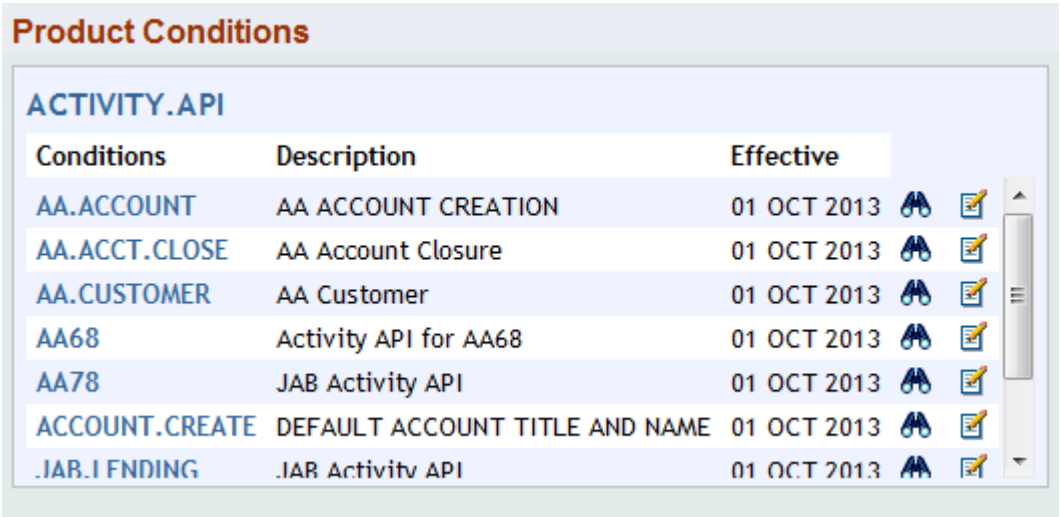
Class is where will have set of activities to behave. These can be given by Temenos and cannot be amended.



Some major used classes are ACTIVITY.API and ACTIVITY.PRESENTATION.

2.6 Product Conditions

Product Condition is where we will define the behavior.


Product Conditions
ACTIVITY.API
Conditions
Description
Effective
AA.ACCOUNT
AA ACCOUNT CREATION
01 OCT 2013
AA.ACCT.CLOSE
AA Account Closure
01 OCT 2013
AA.CUSTOMER
AA Customer
01 OCT 2013
AA68
Activity API for AA68
01 OCT 2013
AA78
JAB Activity API
01 OCT 2013
ACCOUNT.CREATE
DEFAULT ACCOUNT TITLE AND NAME
01 OCT 2013

3. Defining conditions

In the product condition, we will define conditions for classes. For the respective class, there will be some set of conditions to define the class.

For example: We are considering ACTIVITY.API, we have to create or modify to define conditions.

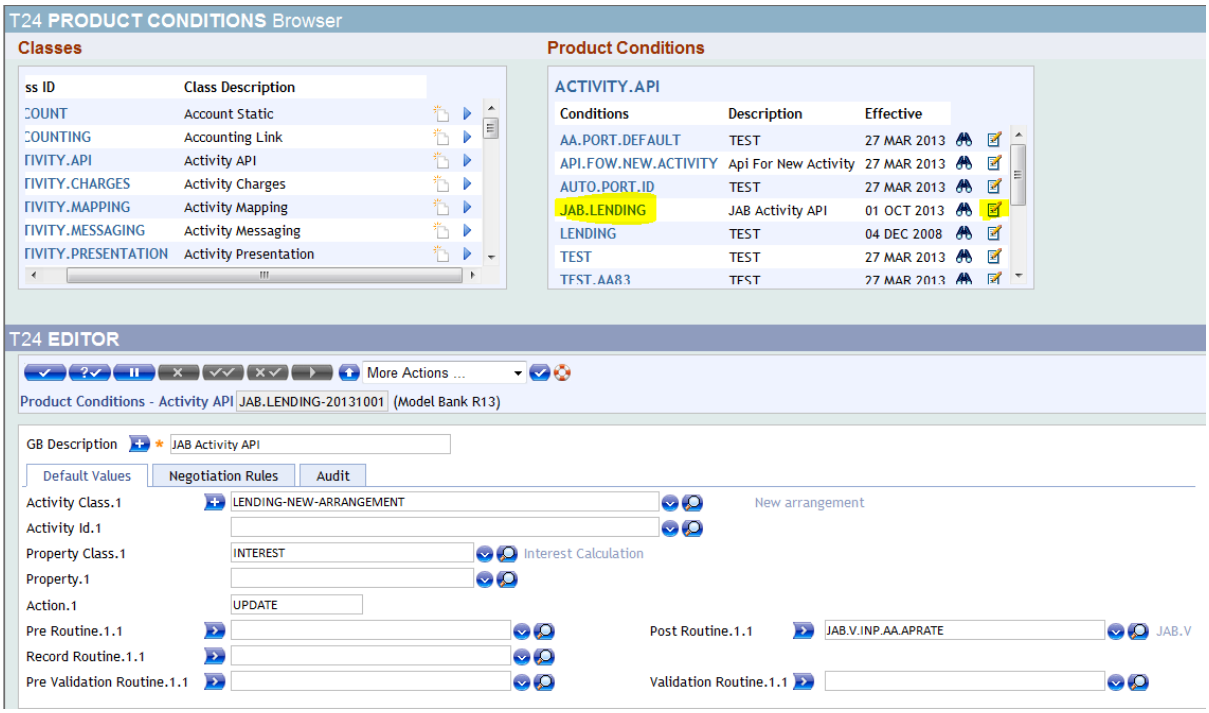
Some of the frequently used classes are given below:

3.1 ACTIVITY.API

This is the place where we play around with routines.

In this ACTIVITY.API only, we have to attach routines.

Here, user can define **ACTIVITY, PROPERTY.CLASS and ACTION.**



In order to define our local routines we have to provide valid **Activity Class / Activity ID** along with the **Property class/ Property** along with **Action.**

From the above screenshot, our routine will get triggered whenever the user updates the values in the Interest tab/section for a new a Lending contract.

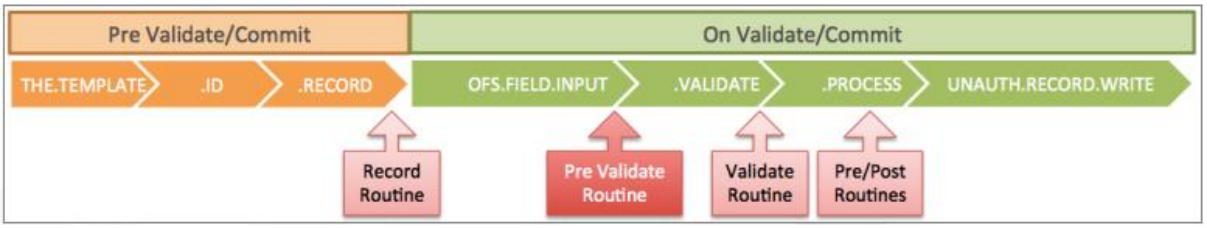
We can define routines under following sections.

- Record routine
  Triggered when record opens
- Pre validation routine
  Triggered before core validation of the record
- Validation routine
  Triggered during validation

- Pre routine

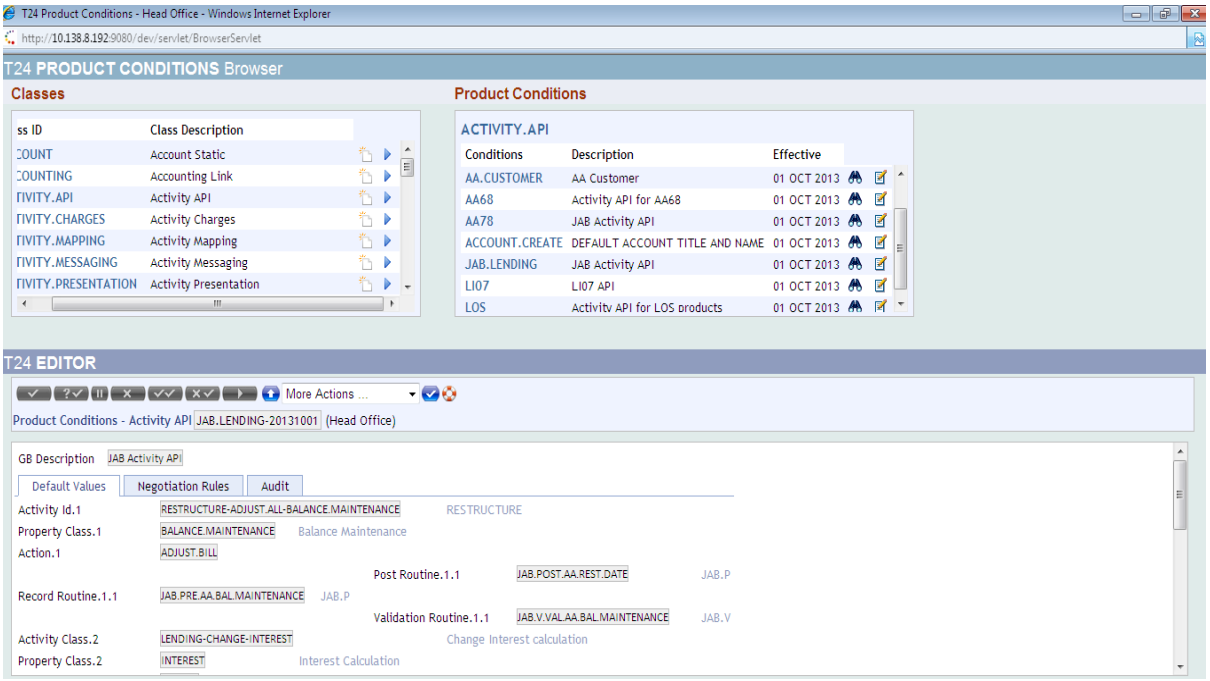
  Triggered before the defined action takes place
- Post routine

  Triggered after the defined action takes place



Above given screenshot clearly shows, in which stage which routine will get triggered.

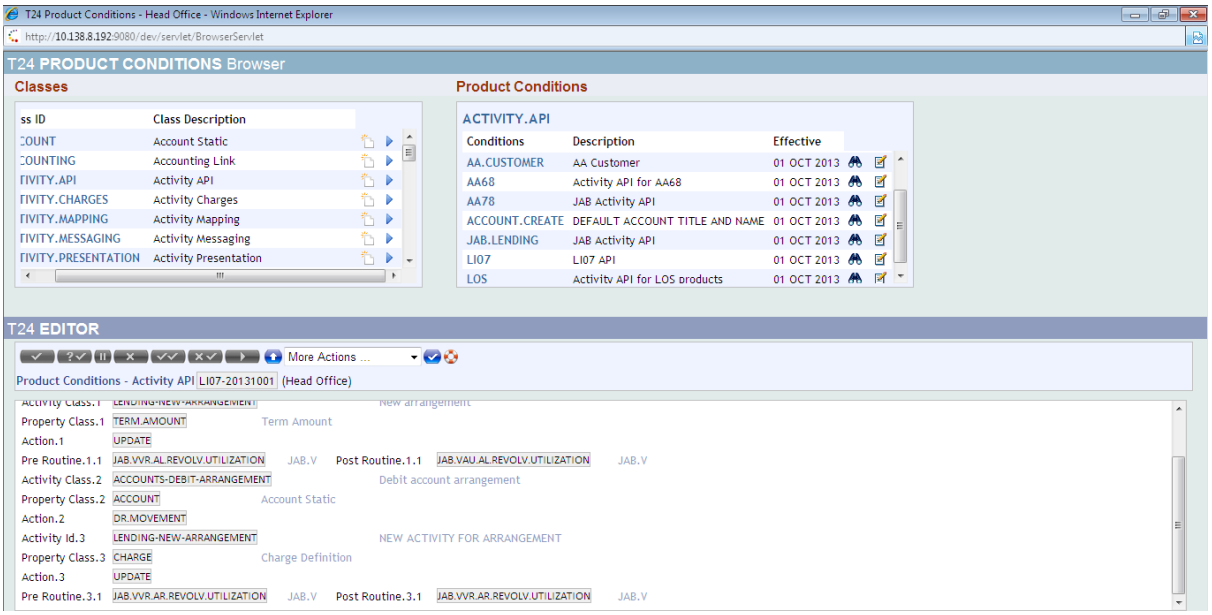
The below screenshot shows the attachment of Record, Validation and Post routine in the product JAB.LENDING.



Validation routine:

From the above screenshot our routine will get triggered whenever the user adjust the bill then validate the record for a "RESTRUCTURE-ADJUST-ALL-BALANCE-MAINTAINTENANCE" activity and Action– "ADJUST-BILL".

Pre Routine:

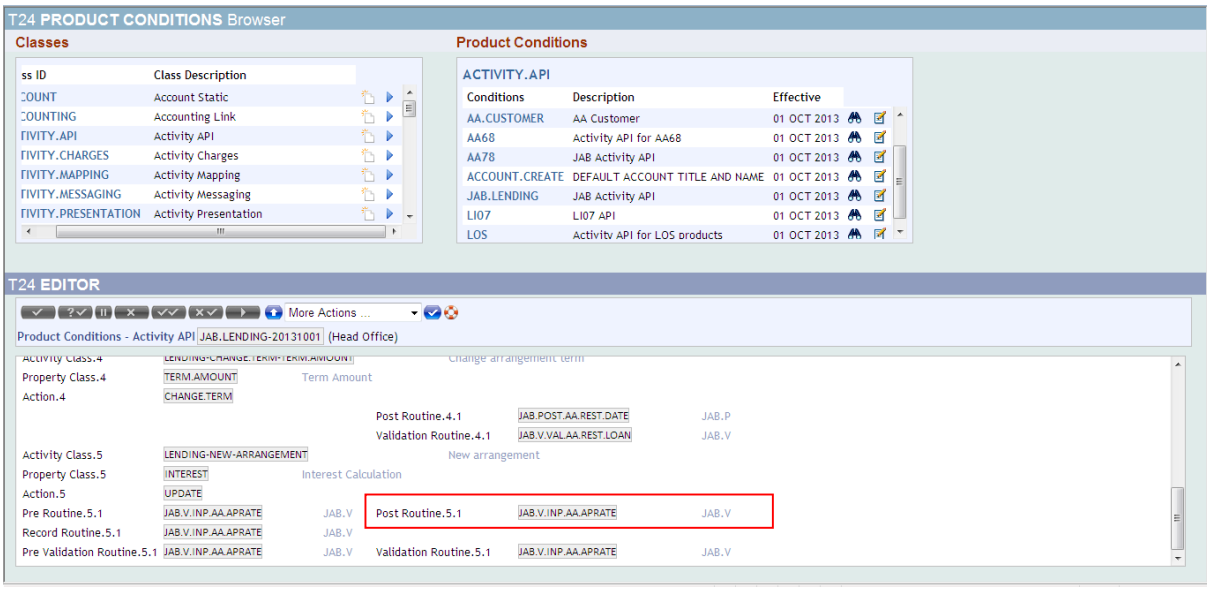


From the above screenshot our routine will get triggered whenever the user opens the record and updates in the Charge tab/section for a new a Lending contract.

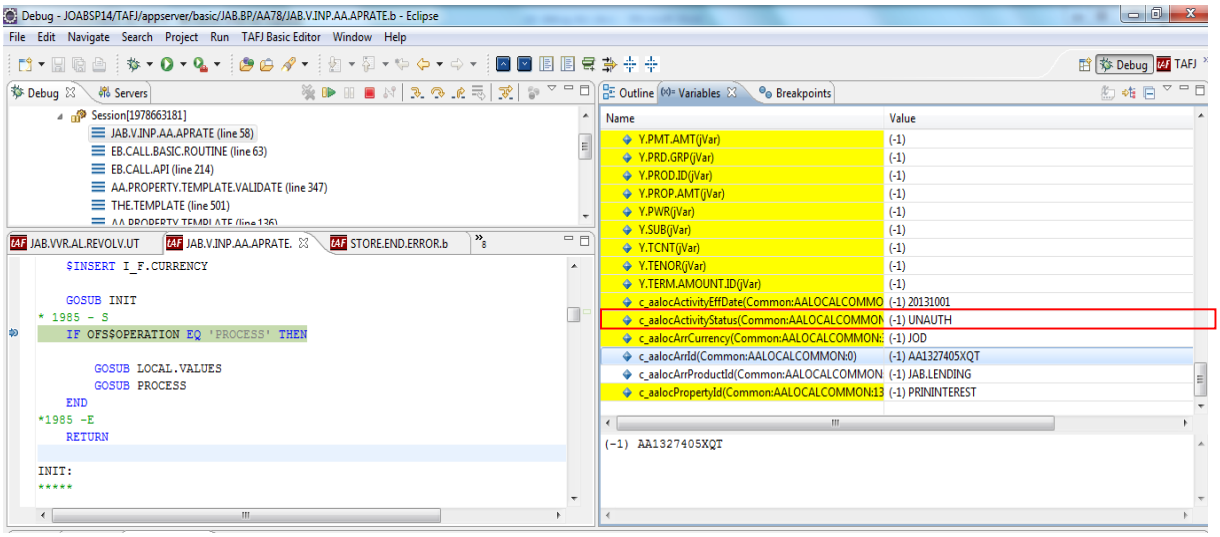
Post Routine:

Post routine is triggered both UNAUTH and AUTH status for a Lending New Arrangement.

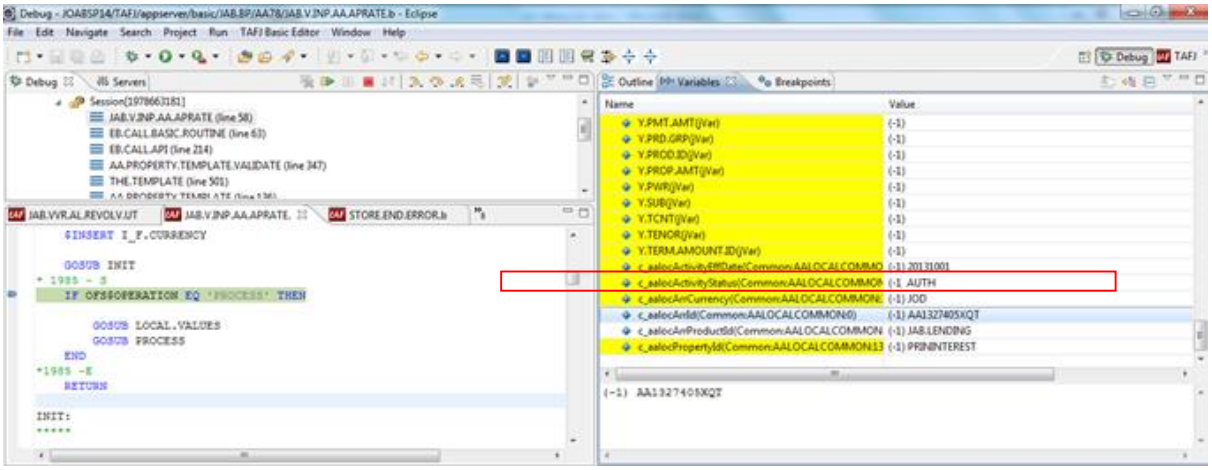
Local routine to perform the required action is attached as Post routine in JAB.LENDING product.



The below screenshot shows the post routine triggered with “UNAUTH” status,



The below screenshot shows the post routine triggered with “AUTH” status,



Note:

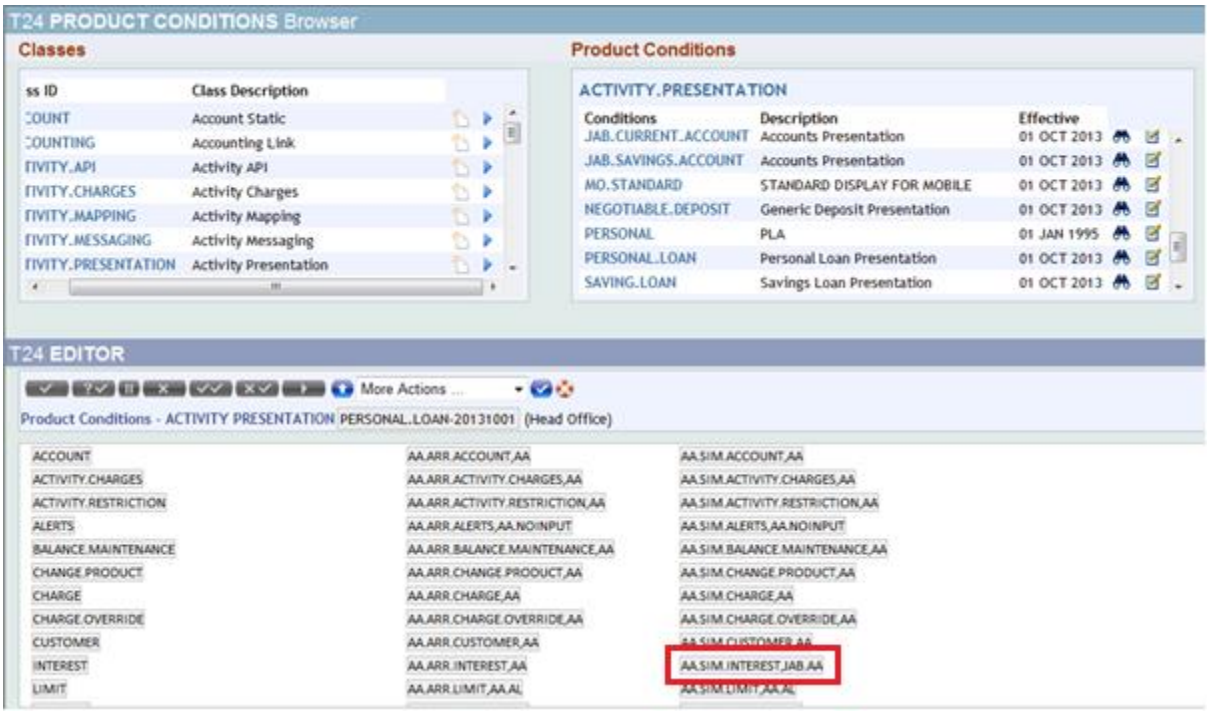
Pre and Post routines are called during all stages of template – like Input, Delete, Authorization, and Reversal. Hence the developer should always branch the control using **OFS$OPERATION**.

For ex: From the above screenshot, we have given as OFS$OPERATION EQ “PROCESS”. If condition gets satisfied, then system will process the defined code. Else, it will return. This is how we are taking control on code for pre and post routines.

3.2 ACTIVITY.PRESENTATION

It is used for defining various versions in a single screen which will be used for different Property classes/Properties/Activities during an entry.

Here, we will attach VERSIONS.

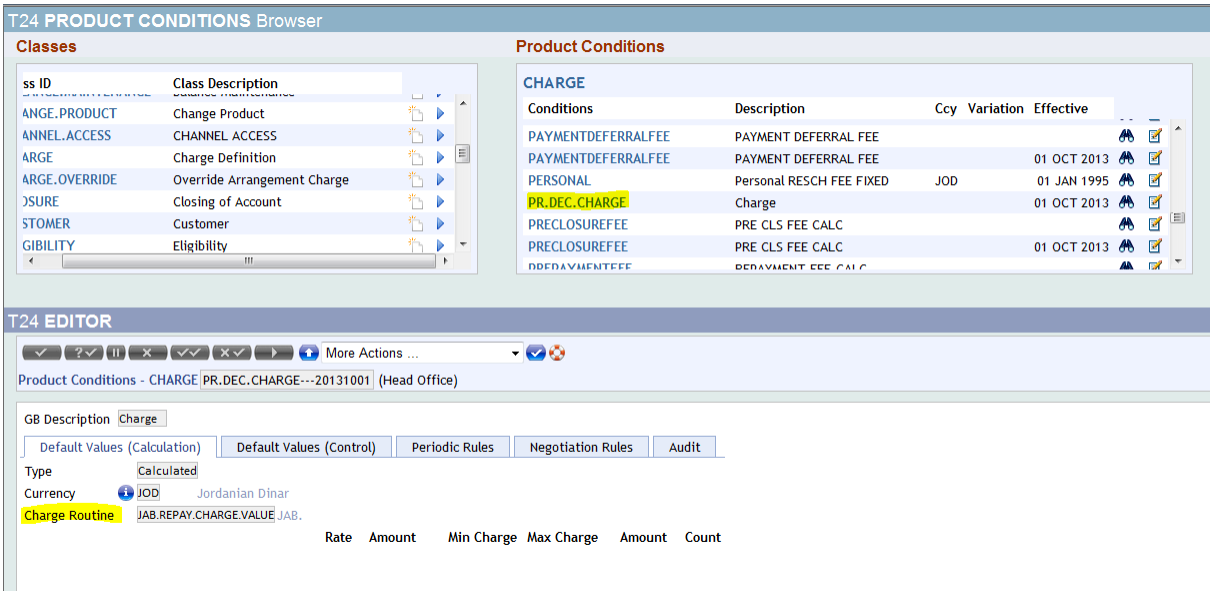


No routines should be defined in the versions defined in ACTIVITY.PRESENTATION. The versions should only define screen layout.

Adding local field to the application is covered in Section 5.

3.3 CHARGE

This property is used to define charges. This property class can be added to PAYMENT SCHEDULE property class for defining scheduled charges. To define charges related to an activity this class can be used in ACTIVITY CHARGE property class.

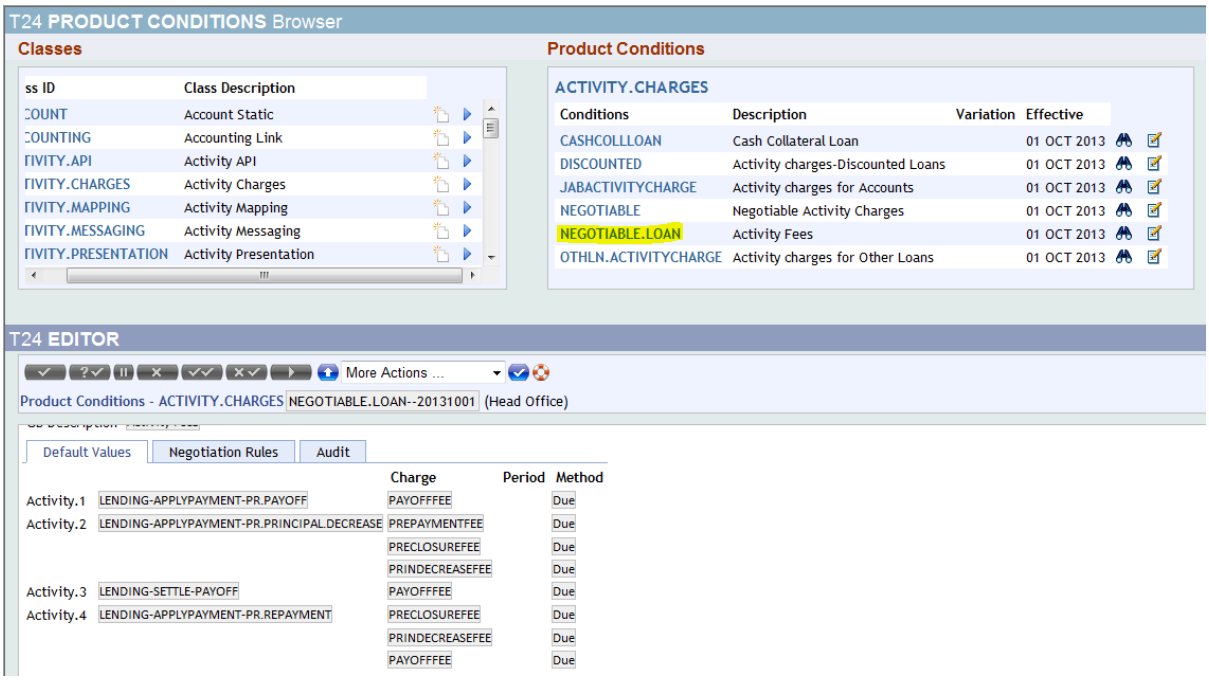


Charge routine field is used to define calculation as required by user. Usually, we will attach routine in this field to calculate charge.

3.4 ACTIVITY.CHARGES

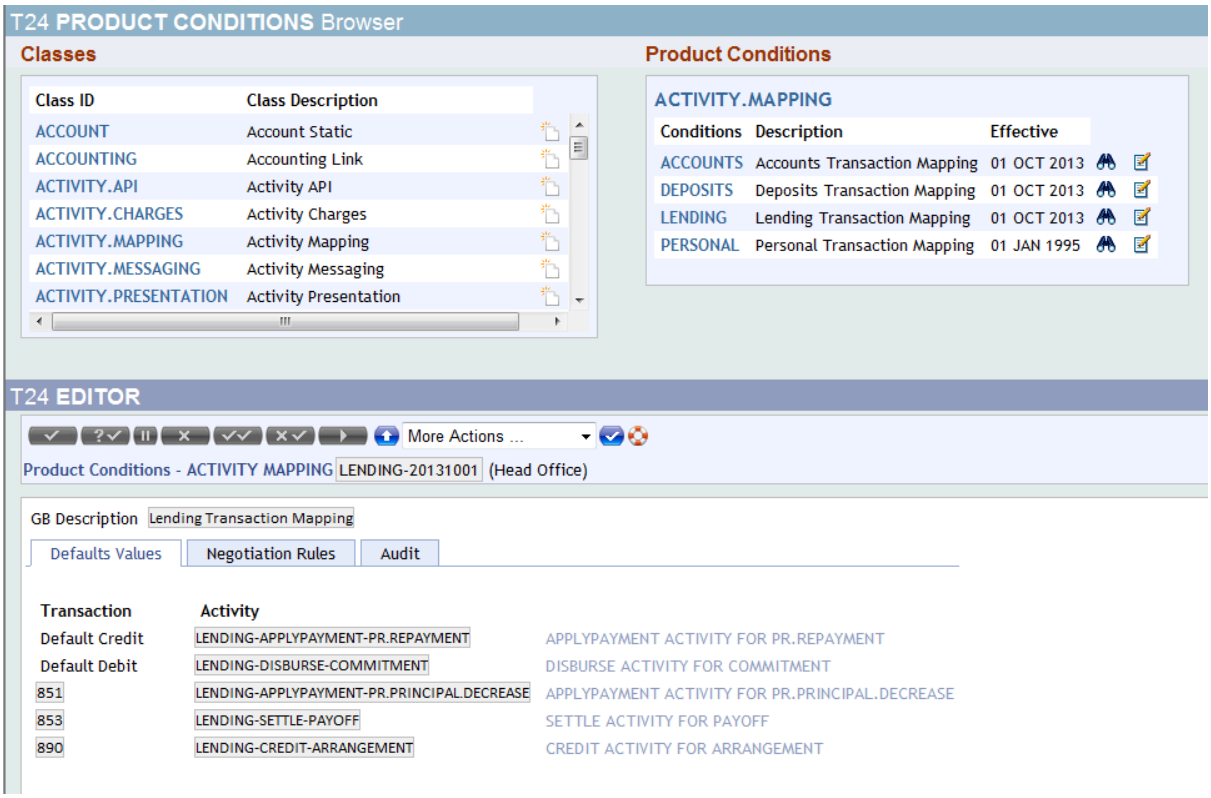
It defines the charge that needs to be applied when activity is triggered.

For example: If we increase/decrease Term amount (LENDING-INCREASE-TERM.AMOUNT/LENDING-DECREASE-TERM.AMOUNT) in an arrangement, corresponding charge property will be called which was defined in Property conditions.



3.5 ACTIVITY.MAPPING

In this property class, we will define about which transaction code to be hit during respective activities.

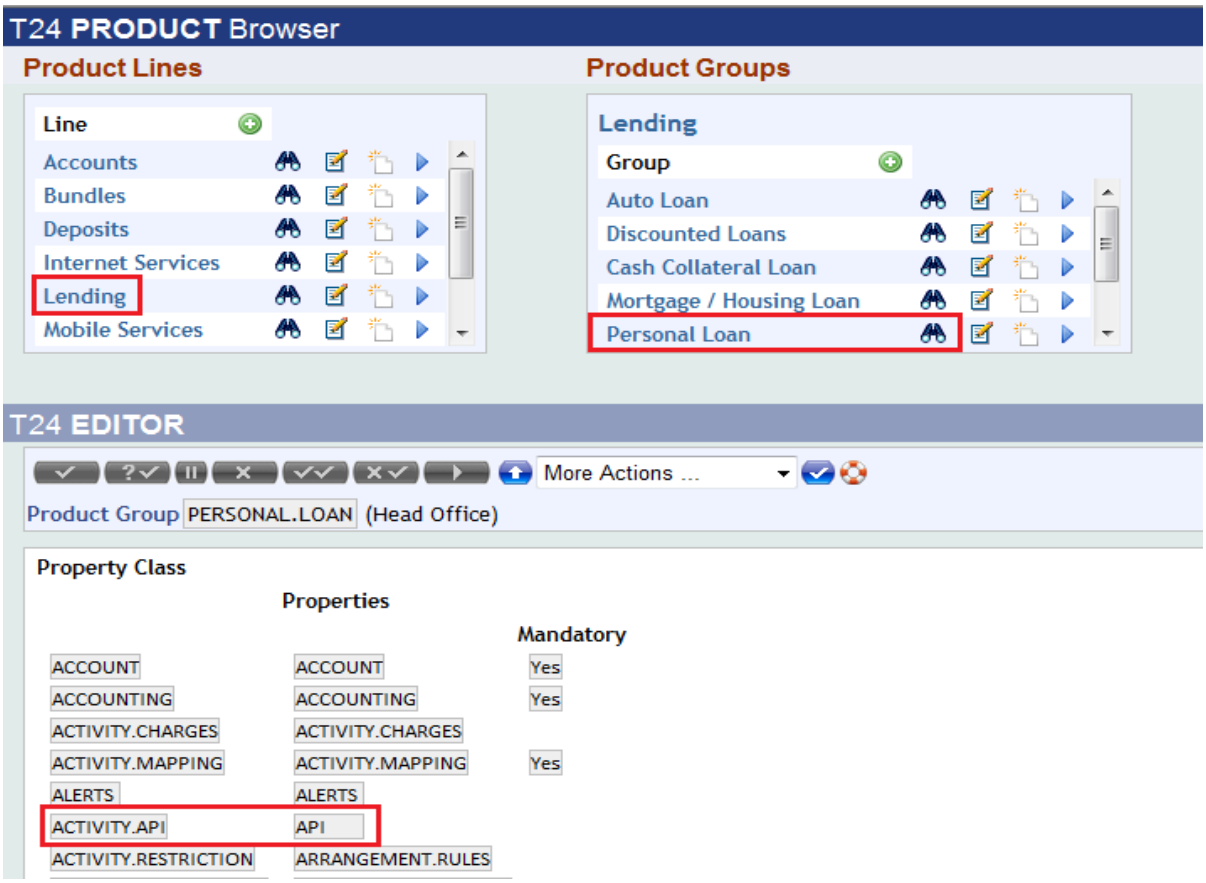


4. Activity setup flow of AA

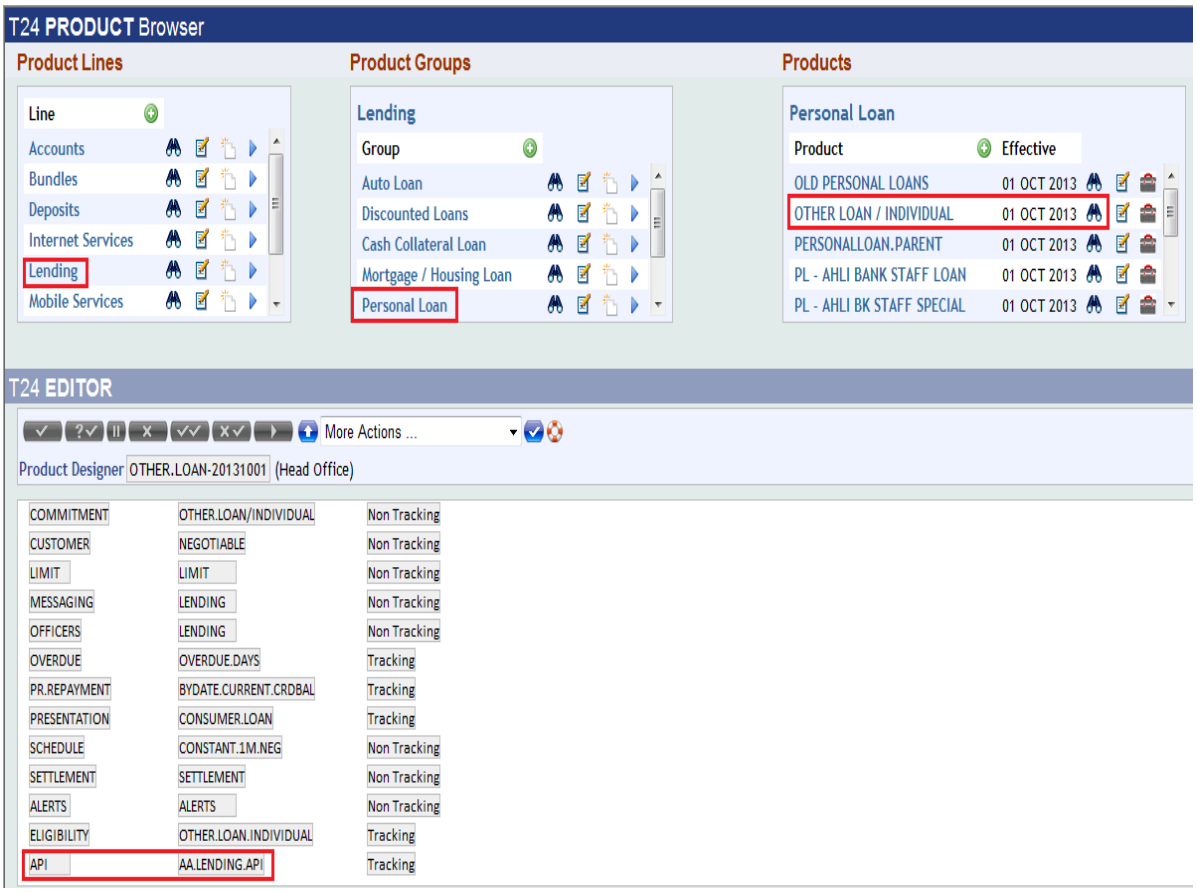
While setting up the product, we have to consider **Property classes and Property conditions** in Product Group and Product respectively.

For example: Please see the below screenshots showing flow of attaching API to product.

Lending -> Personal Loan (View) -> Activity API -> API

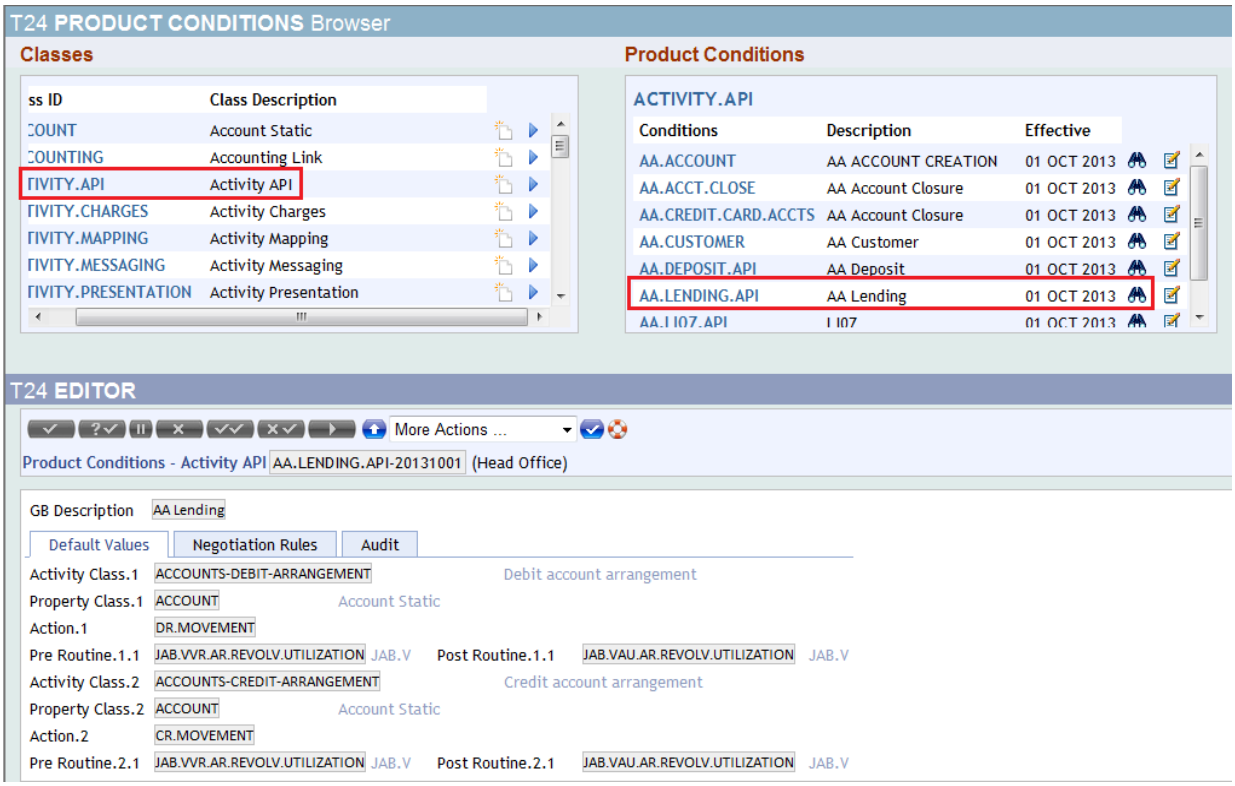


In the above screenshot, Property class is ACTIVITY.API and the id given for this property class is API. This should be given in the Property Condition.



Property class is the id of Product classes. Here, we will define the properties which will be used in Product designer.

From the above screenshot, AA.LENDING.API is the property condition defined in product. Also, it is the id of Activity.API class.



In this API, we have attached routines as Pre & Post.

The above given setup is for one property class. Likewise, we have to setup for other activities based on the requirement.

5. Adding Local Reference Field

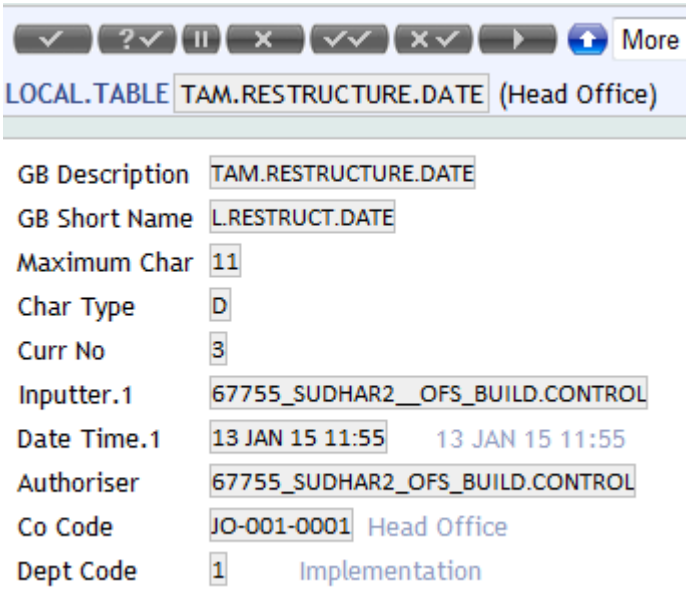
AA Properties shares the same field layout across 4 different files:

- Designer (Example: AA.PRD.DES.CUSTOMER)
- Proofing (Example: AA.PRD.PRF.CUSTOMER)
- Catalog (Example: AA.PRD.CAT.CUSTOMER)
- Arrangement (Example: AA.ARR.CUSTOMER)

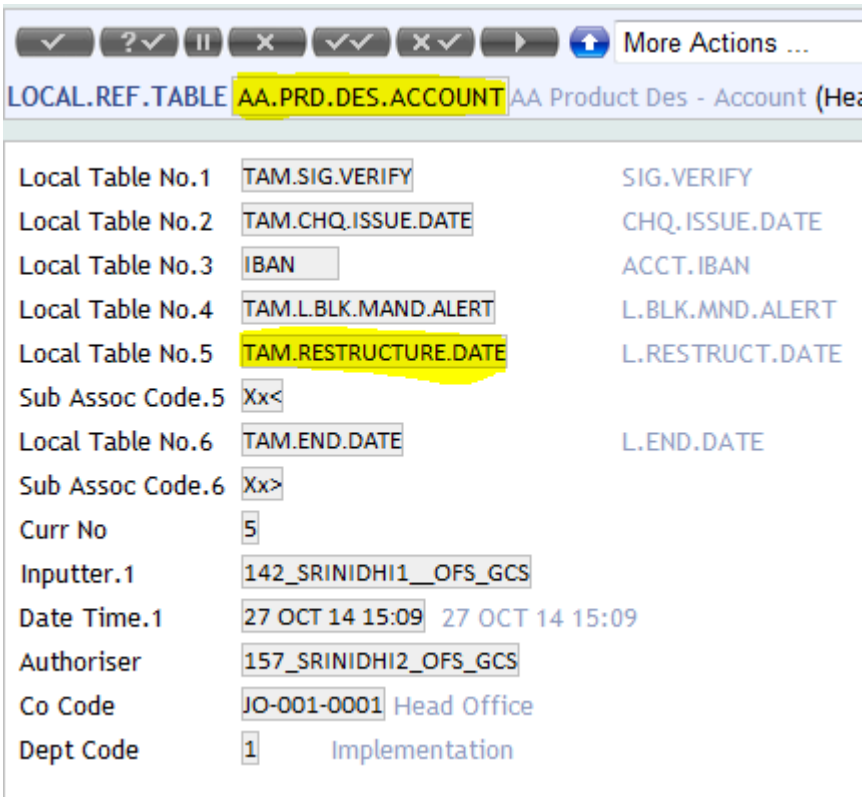
To avoid duplicity and maintain data integrity, AA allows definition of Local reference fields to just one file (the PRD.DES files) and replicates the same across other levels automatically.

We have given below the screenshots of local field and it was attached to the AA.PRD.DES.ACCOUNT table.

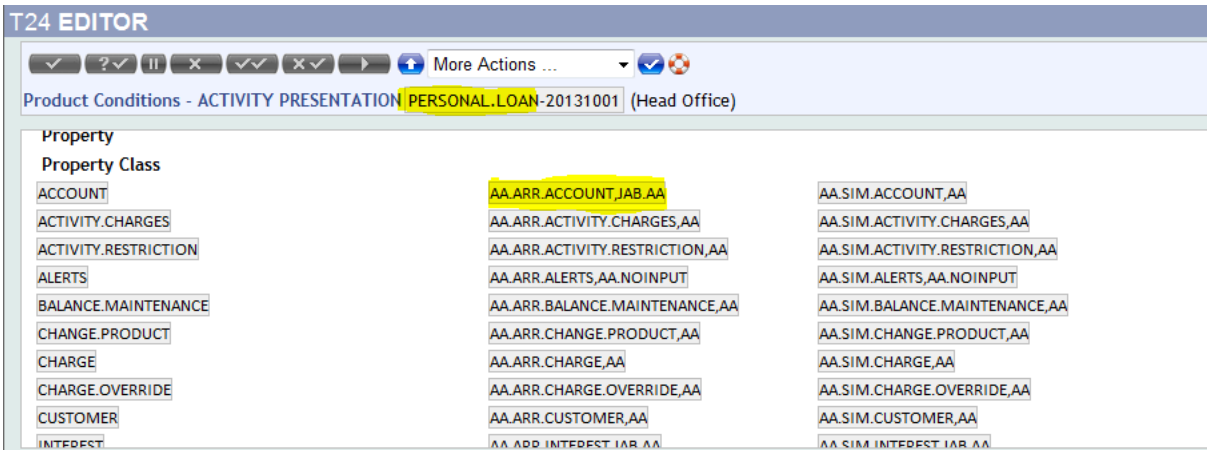
Local table:



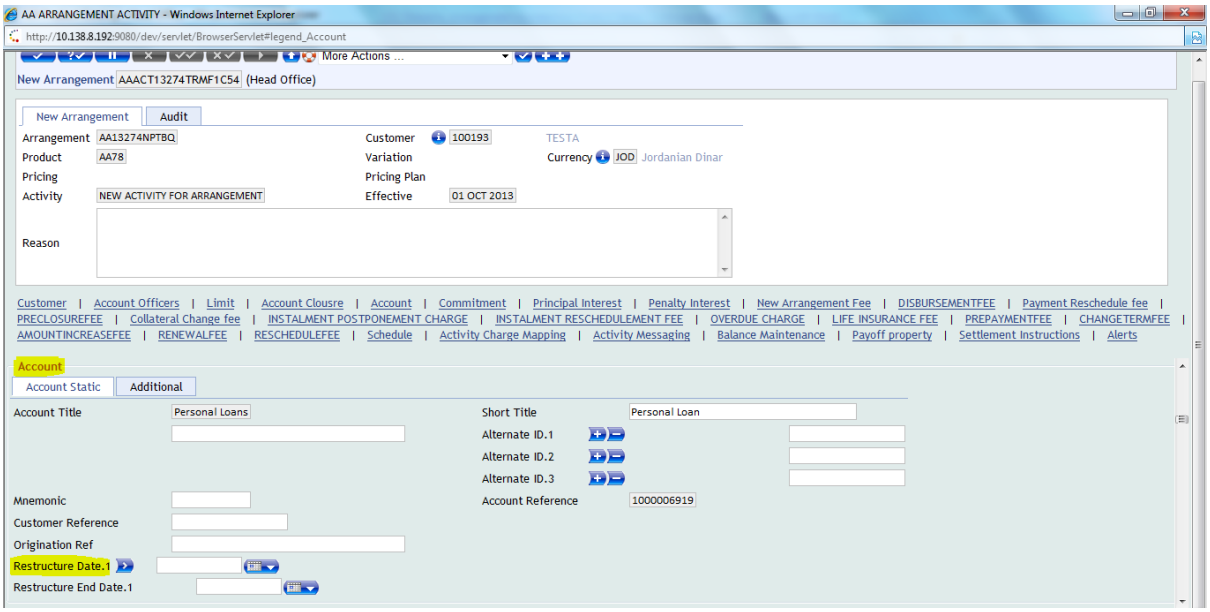
Local Ref table:



The above local field is attached to the version AA.ARR.ACCOUNT,JAB.AA which was attached to the ACTIVITY.PRESENTATION>PERSONAL.LOAN



The above setup will lead to display the LRF in the Account property tab. The screenshot is given below:



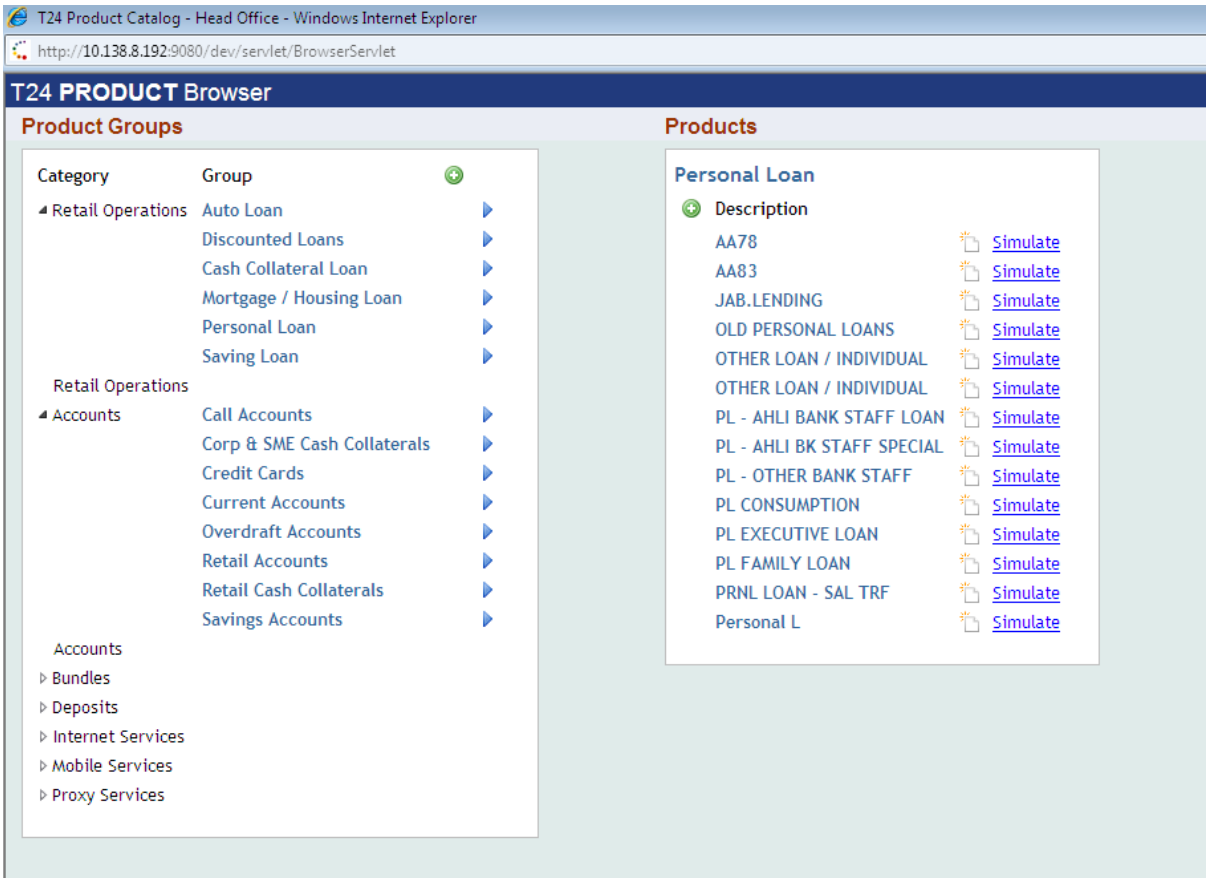
Till now, we have seen the technical functionality of AA. Now, we are going to learn about the business flow in AA.

6. New Arrangement Architecture

While creating a new Arrangement Architecture, it displays multiple Product groups and Product.

Every Product Groups contains different Products.

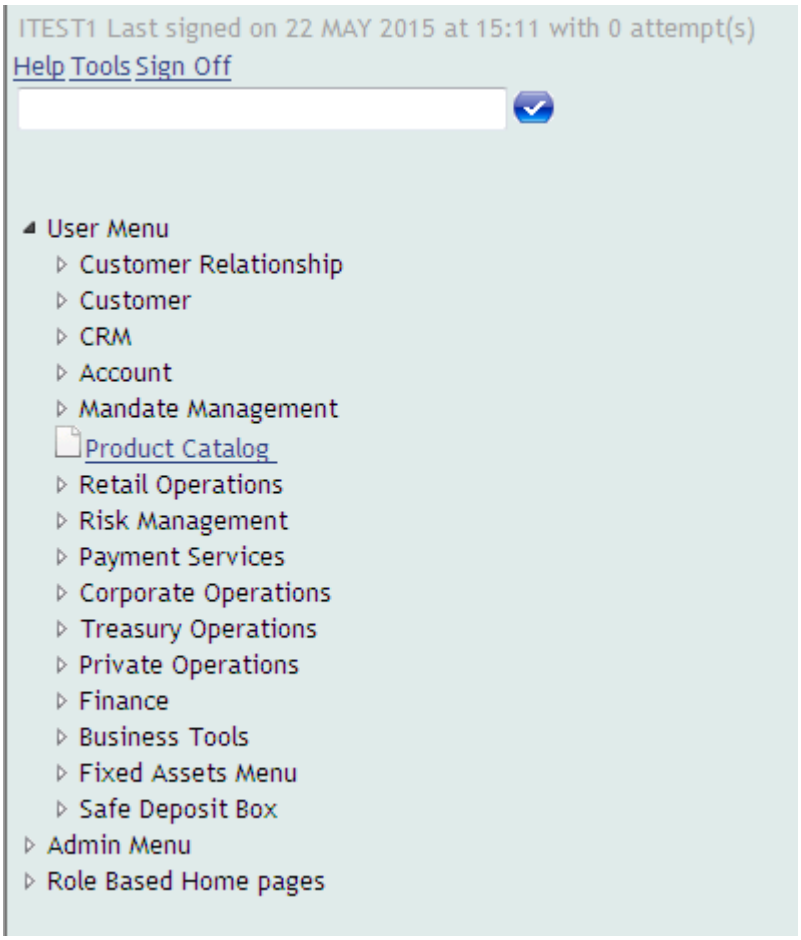
Please find below the product groups and products in the product catalog which is defined in the products and product conditions.



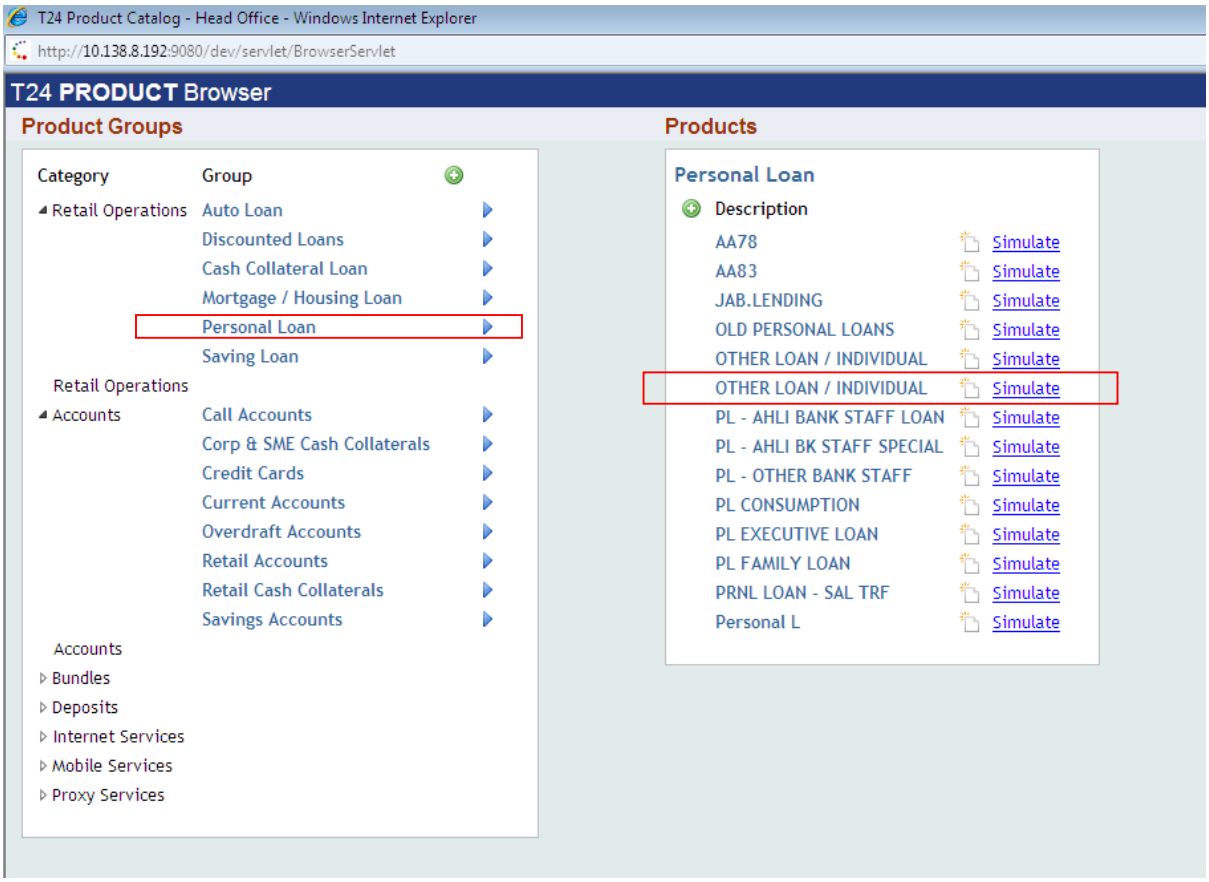
6.1 Create New Arrangement Architecture

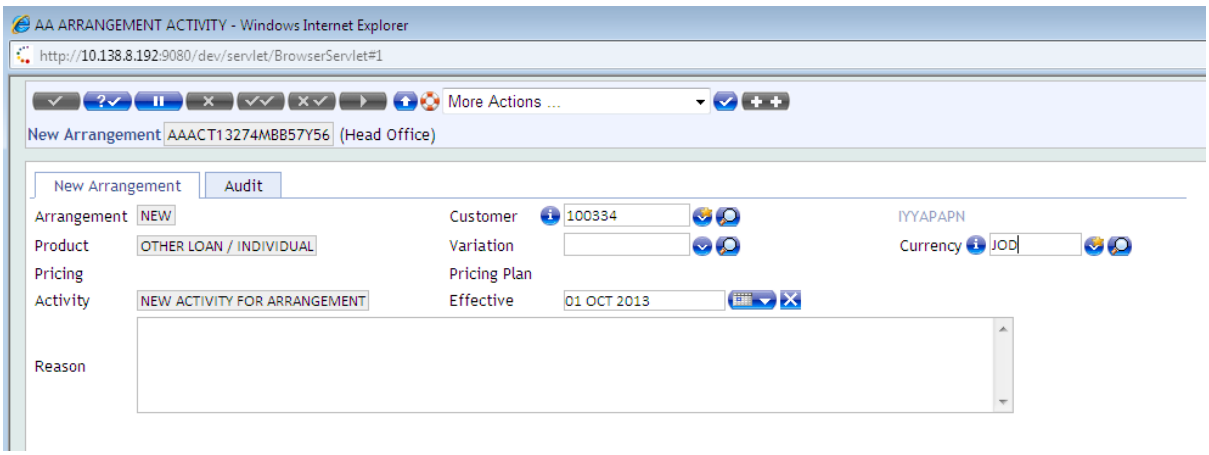
To create New Arrangement Architecture via T24 browser, follow the below steps

Click on User Menu -> Product Catalog

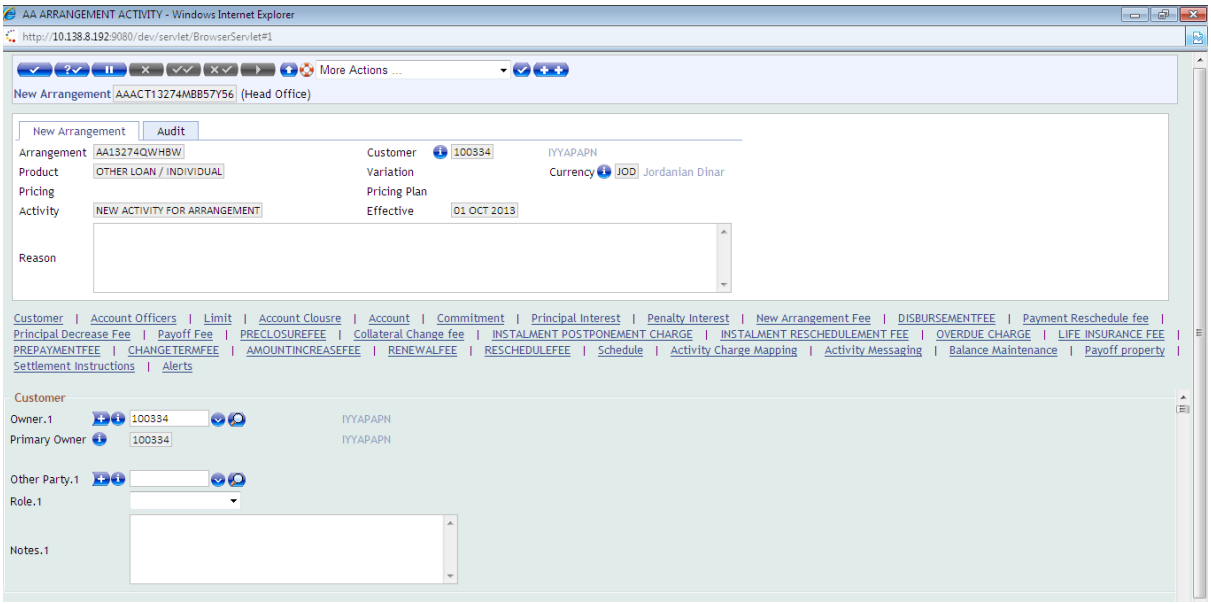


Pop up window will be opened, Choose Product Group as Personal Loan & Product as Other Loan

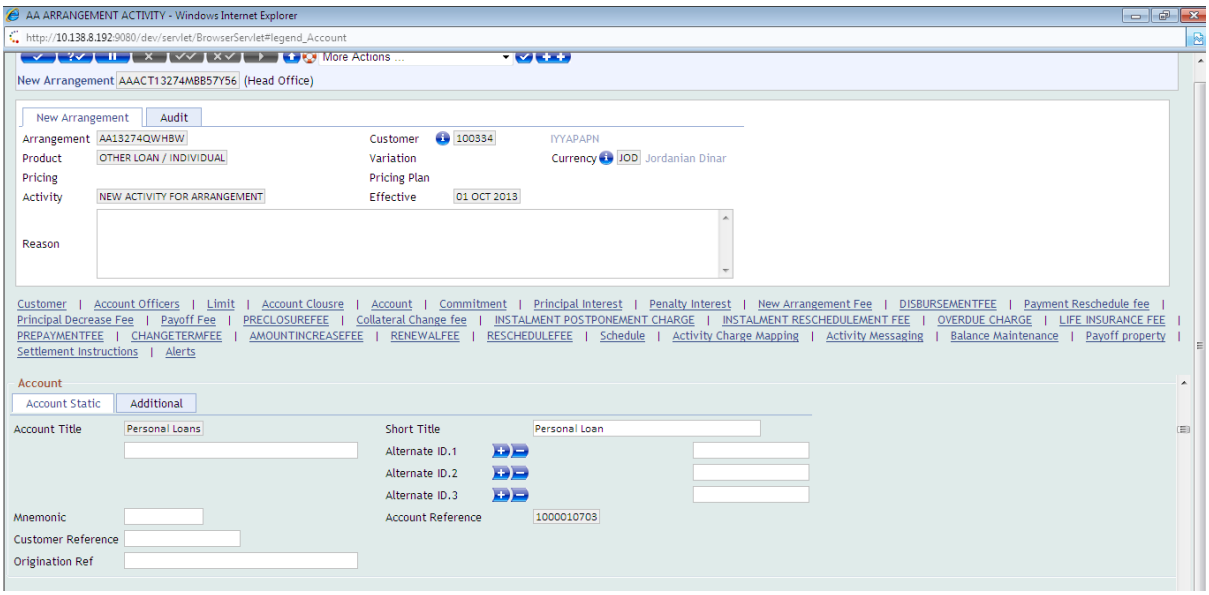




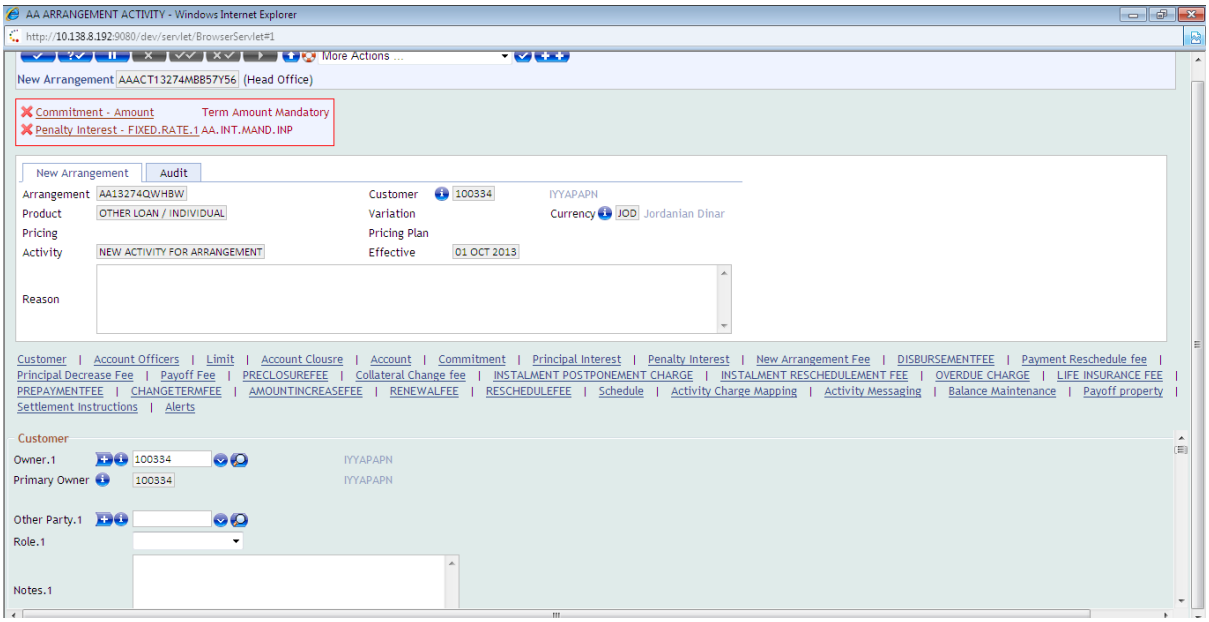
Enter your Customer id and Currency details in the above screen and validate the record, it will open up the arrangement screen to define values.



Account Tab:

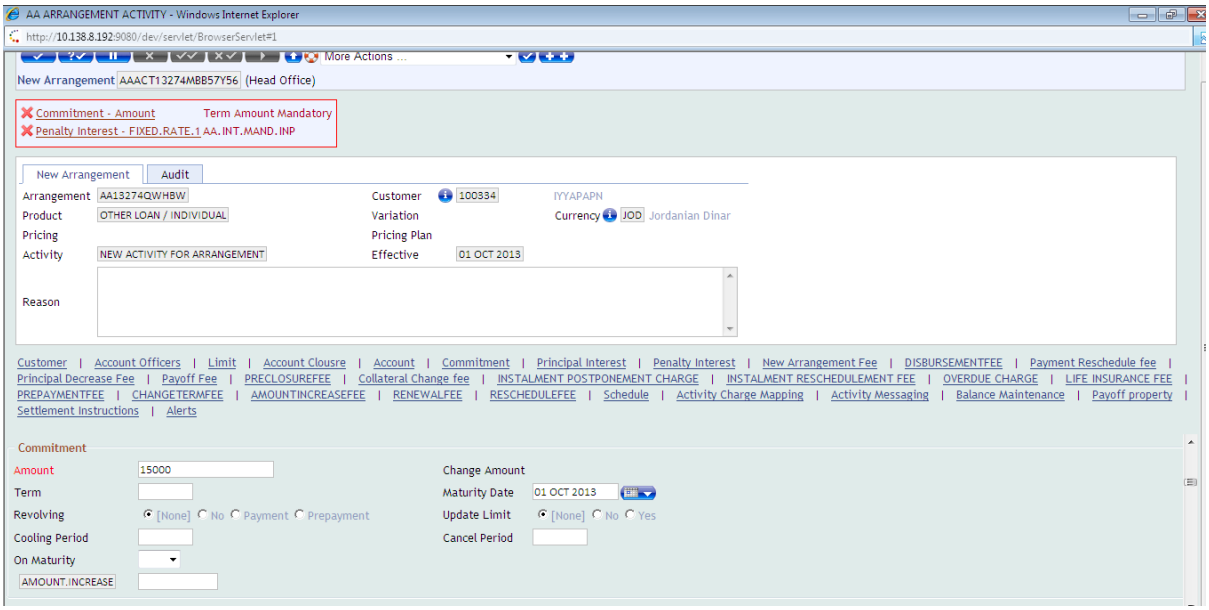


Mandatory input fields are -Term Amount and Penalty interest (Fixed rate)

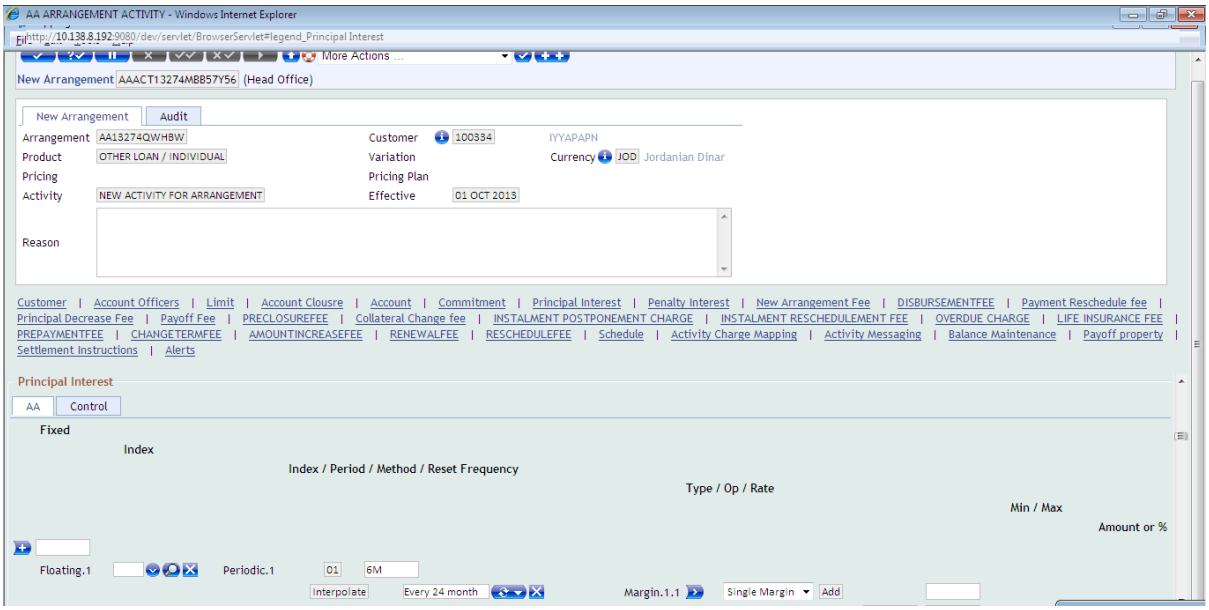


Commitment tab:

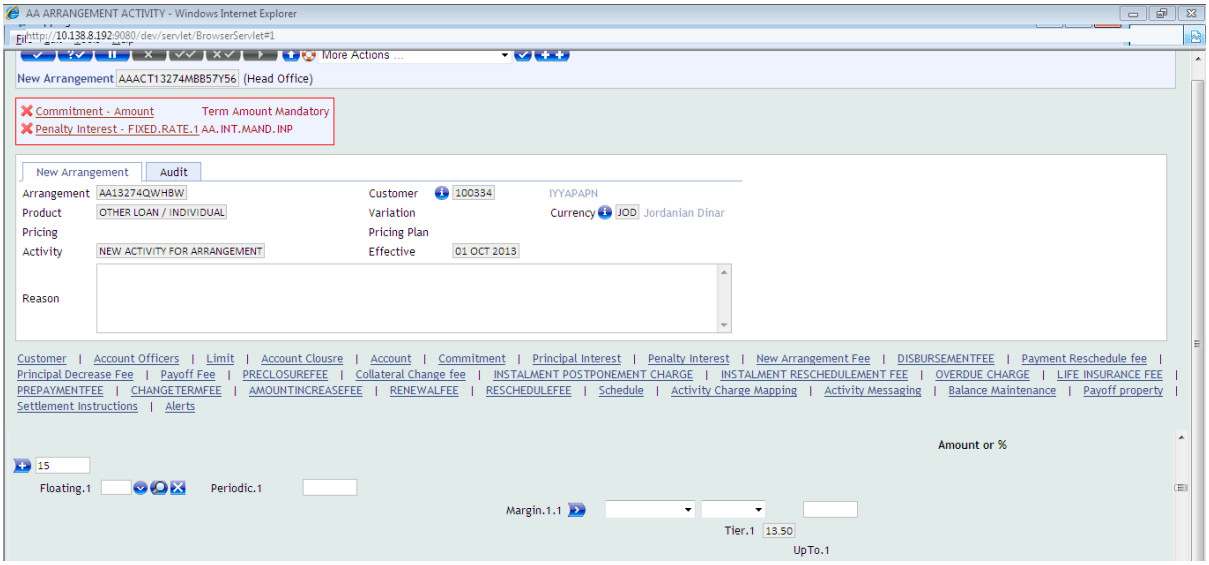
Input the term amount of the loan,



Principal Interest tab:

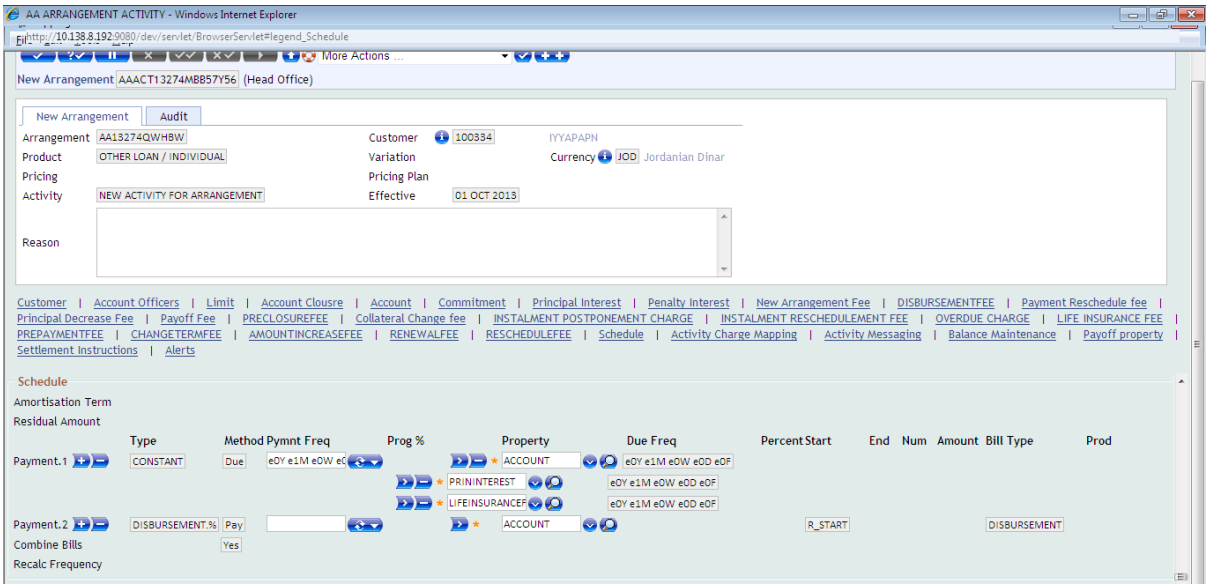


Penalty Interest tab:



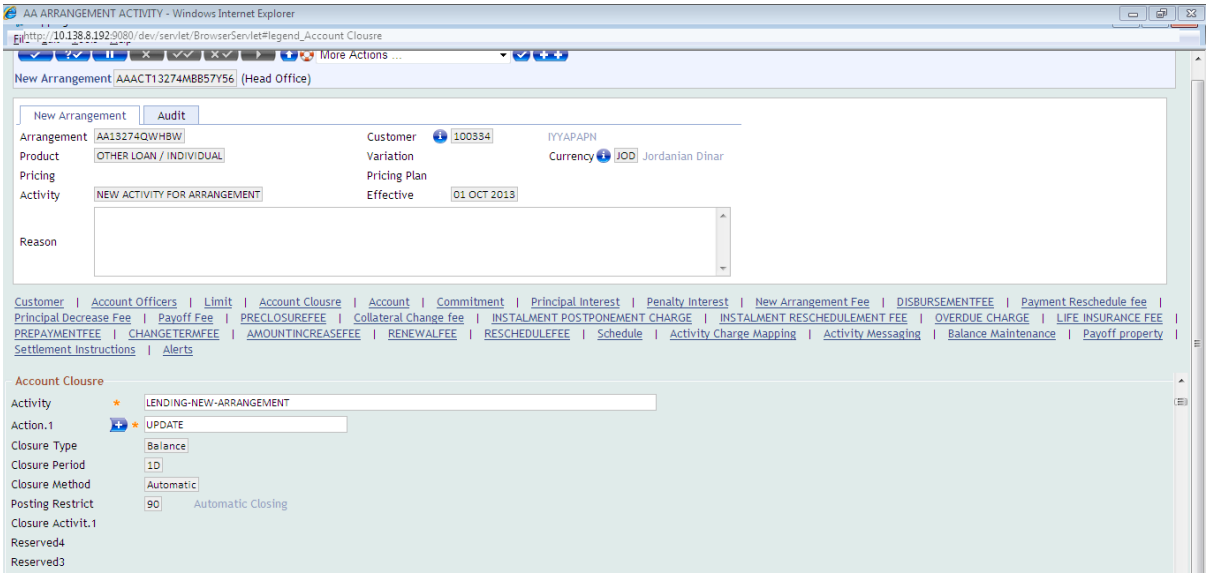
Schedule for the respective Arrangement architecture:

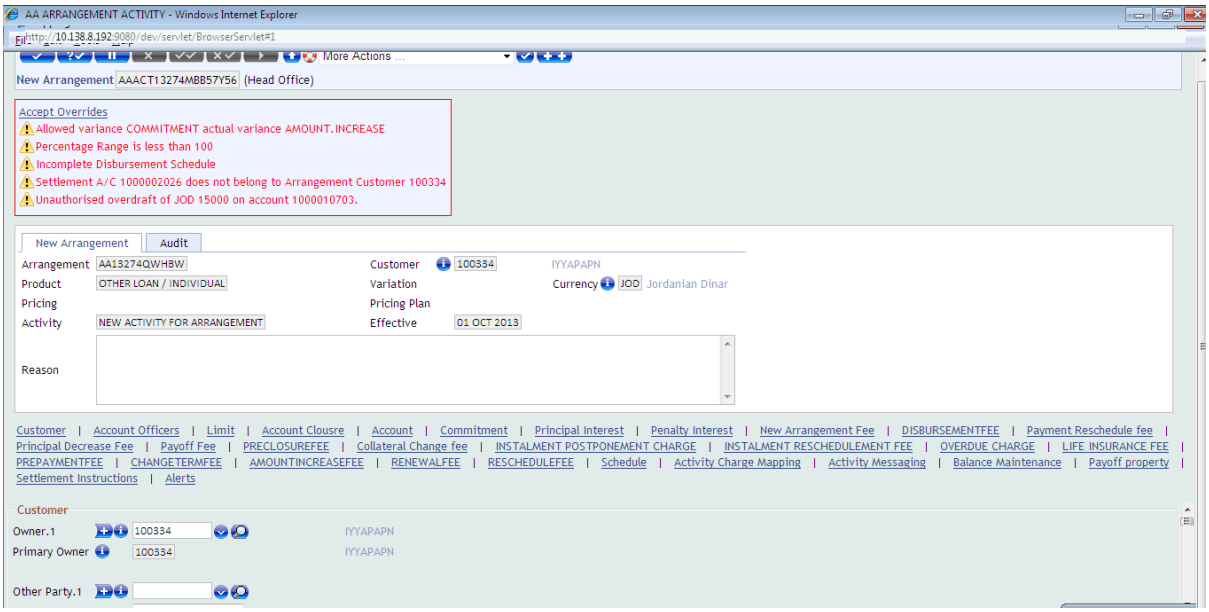
Here defined the Payment type, Method, Property & Frequency.



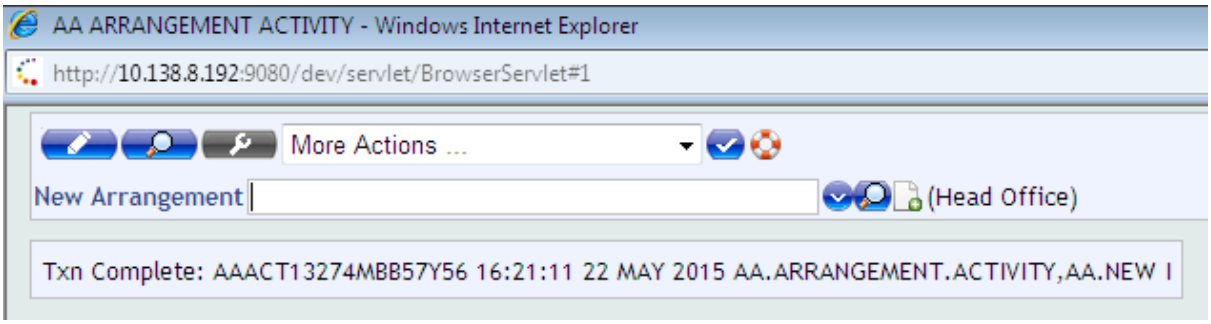
Account Closure:

In account closure Activity and action fields are defaulted based on the Product & Product condition s Setup.





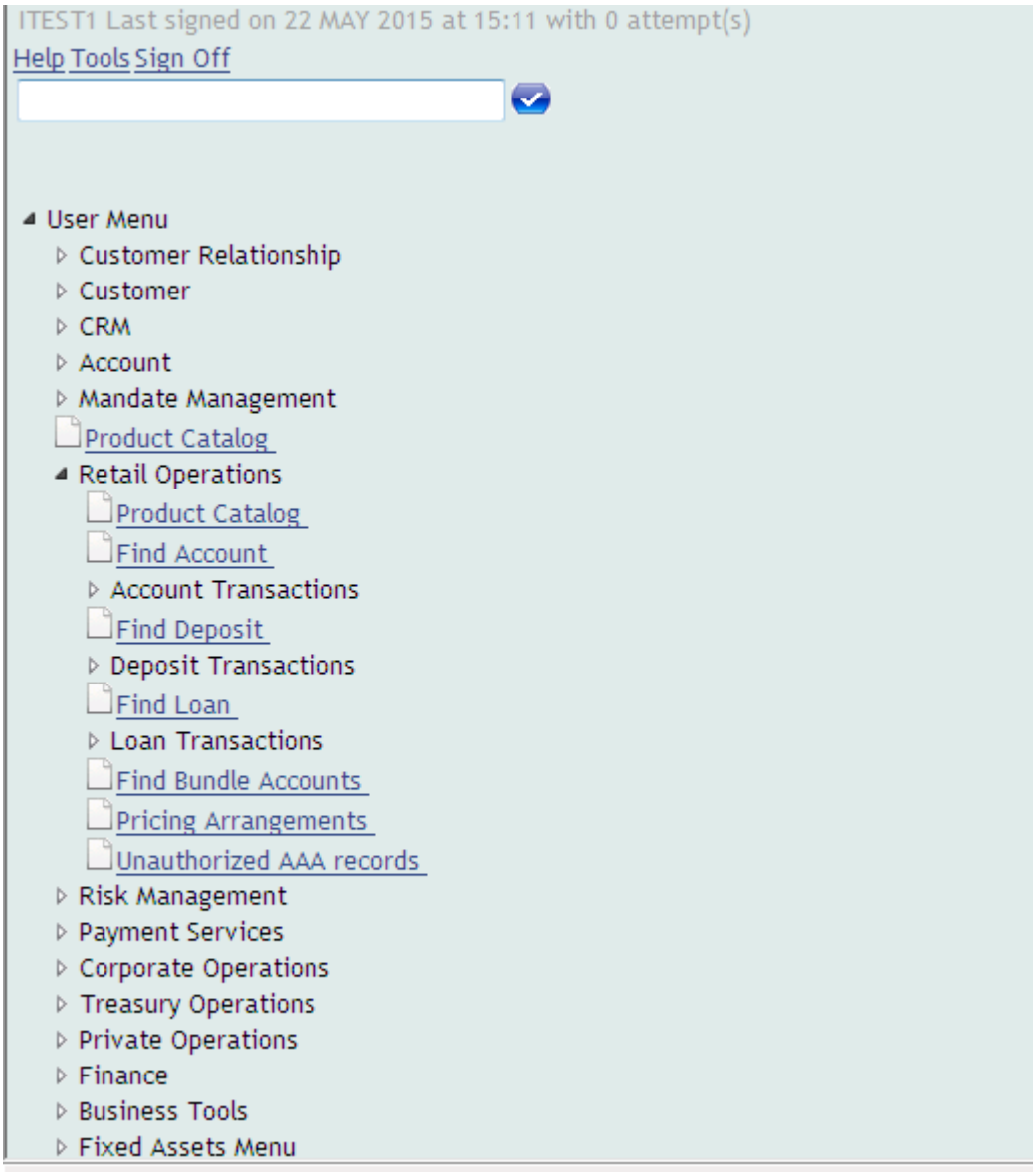
After committing the record,



6.2 Find AA Loan / Authorize the AA Loan

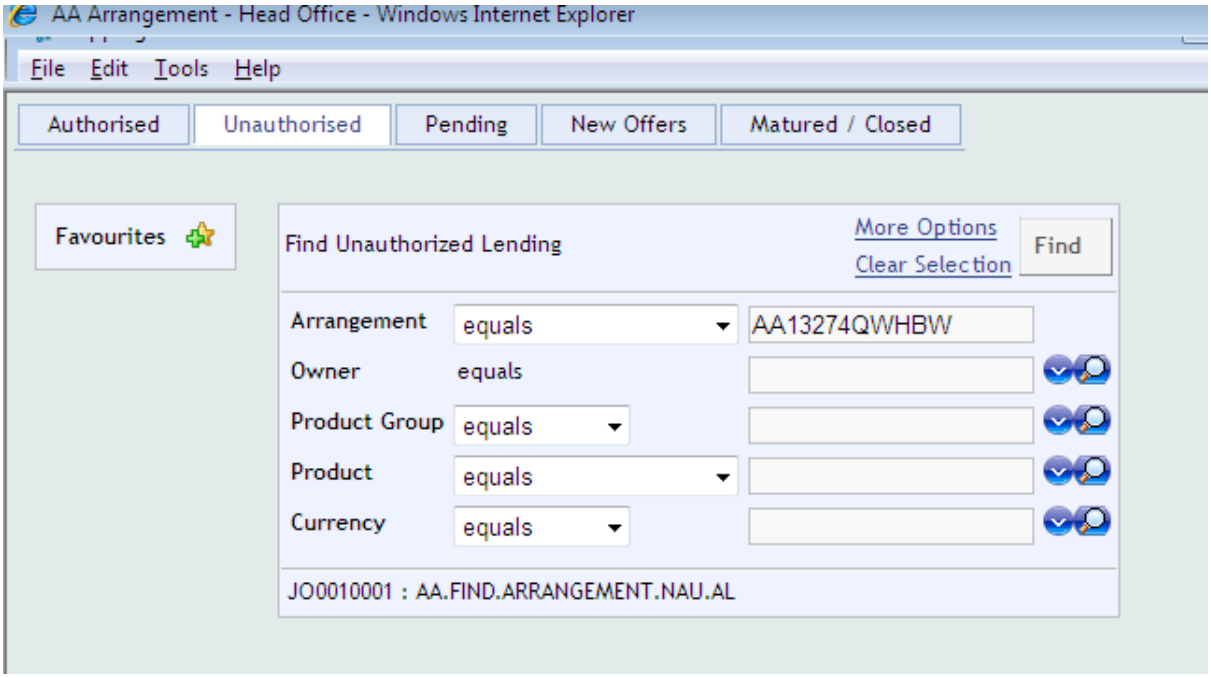
Once Commit the AA record and find the loan/ authorize the record.

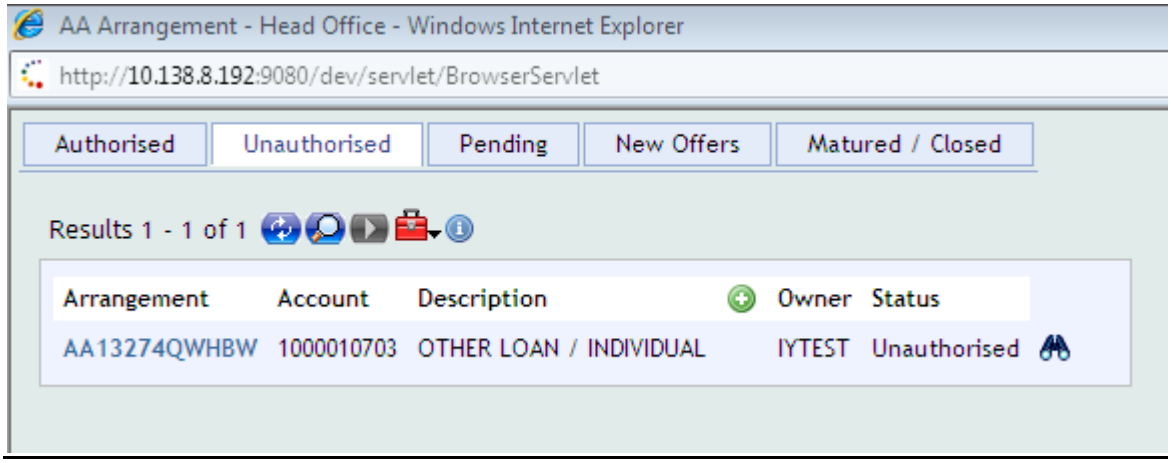
Click user Menu - > Find Loan



Select Unauthorized AAA tab,

Input the Arrangement id - AA13274QWHBW





7. Commonly used Insert file and common variables in AA

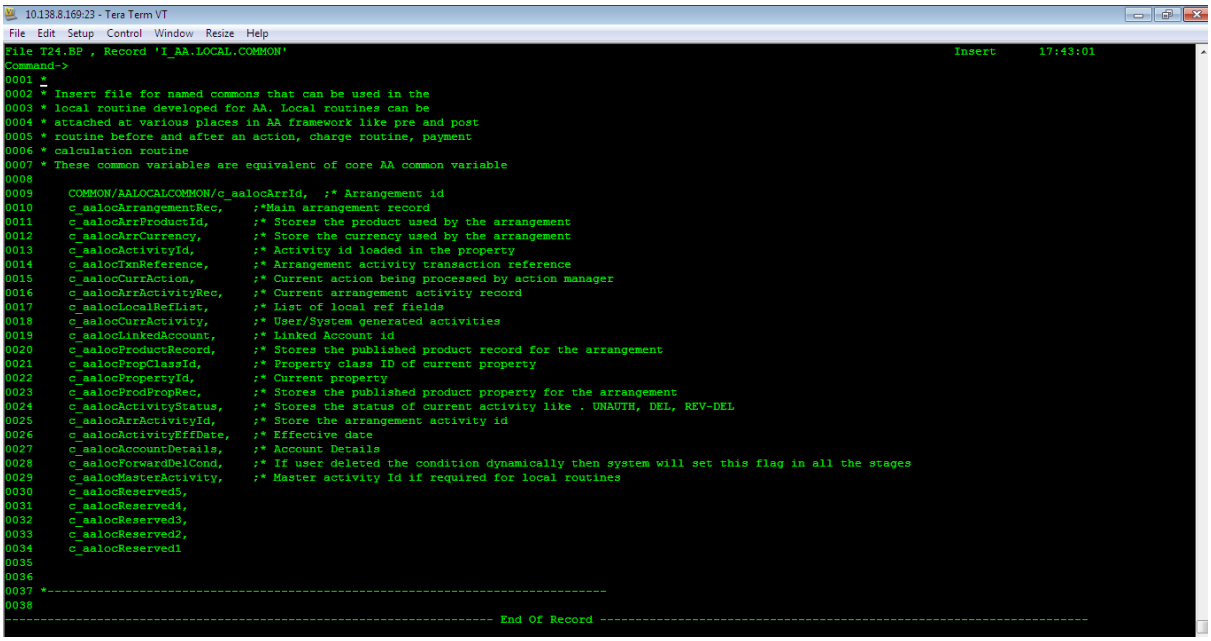
These insert files and common variable are used to find the AA record details.

7.1 Insert Files

$INSERT I_F.AA.INTEREST
$INSERT I_F.AA.TERM.AMOUNT
$INSERT I_F.AA.ACTIVITY.HISTORY
$INSERT I_F.AA.ARRANGEMENT.ACTIVITY
$INSERT I_AA.LOCAL.COMMON
$INSERT I_AA.APP.COMMON
$INSERT I_F.AA.PRODUCT
$INSERT I_AA.ACTION.CONTEXT
$INSERT I_F.AA.TERM.AMOUNT
$INSERT I_F.AA.LIMIT

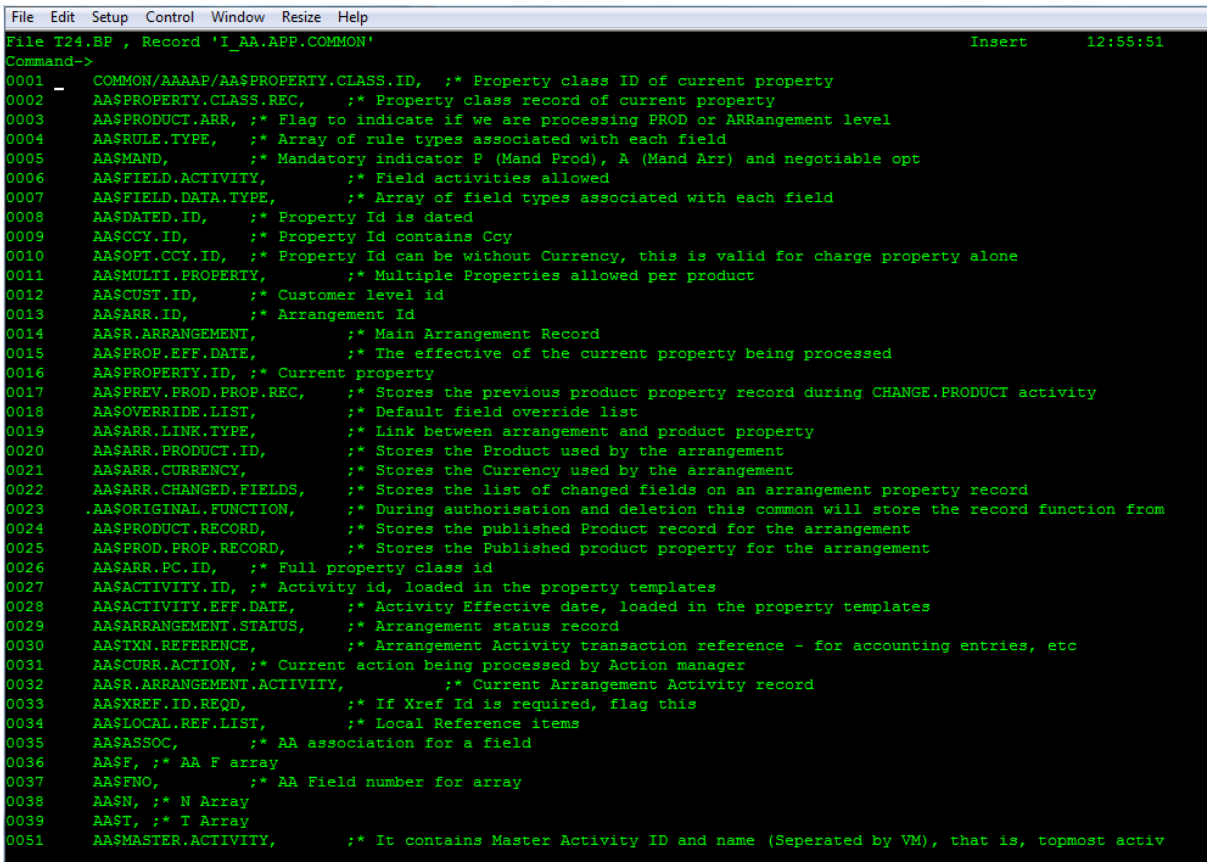
7.2 Common Variables

Find below the common variables of **I_F.AA.LOCAL.COMMON**,

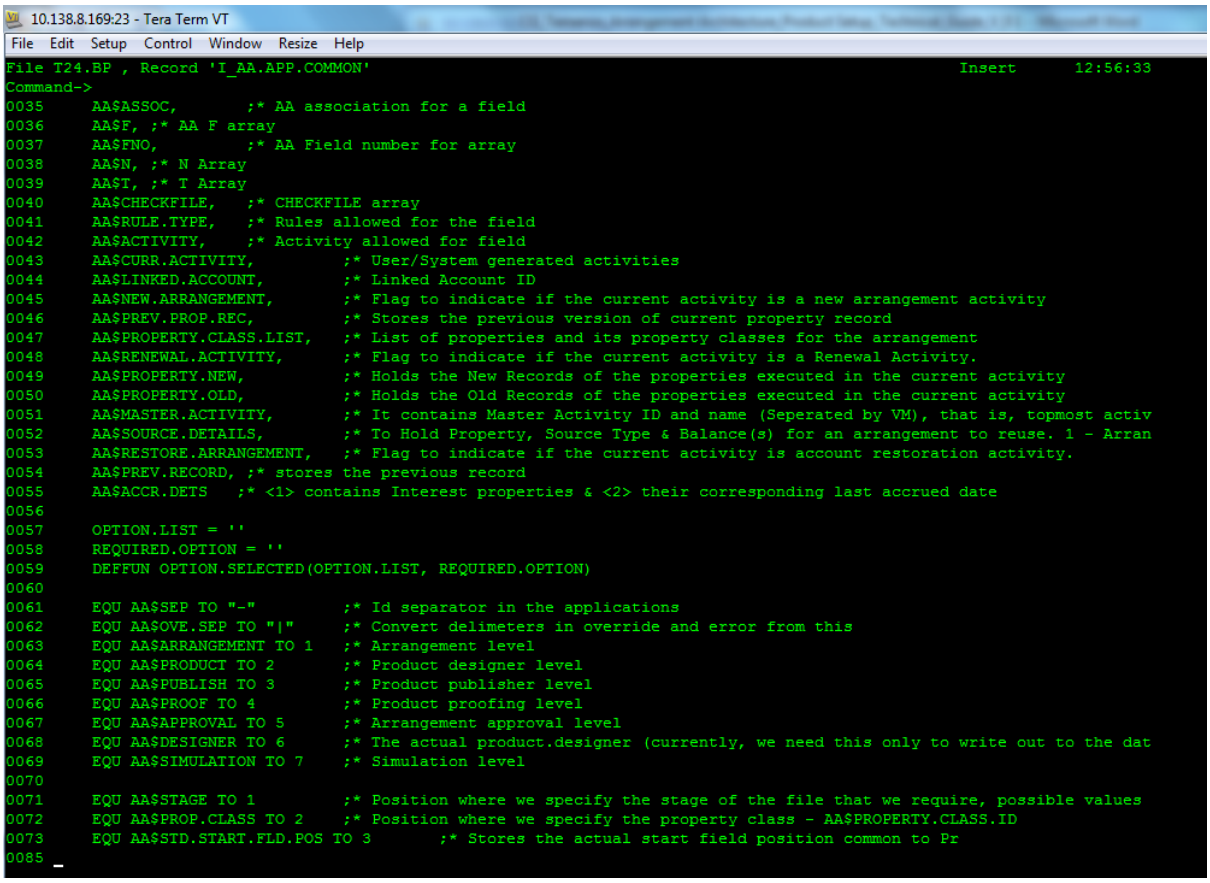


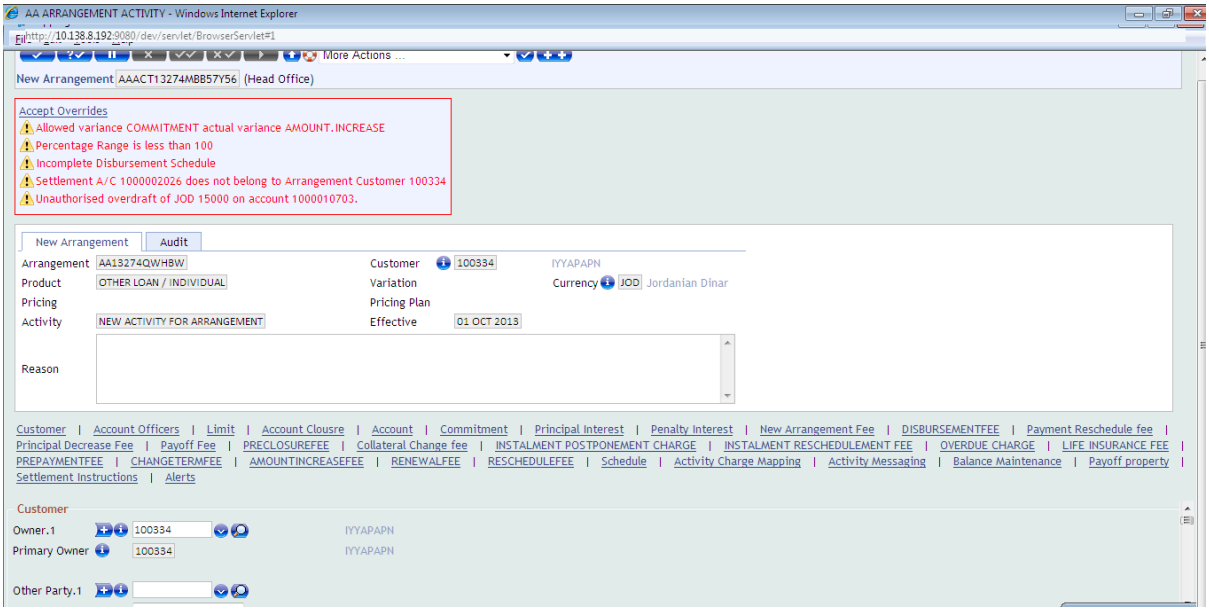
```
10.138.8.169:23 - Tera Term VT
File  Edit  Setup  Control  Window  Resize  Help
File T24.BP , Record 'I_AA.LOCAL.COMMON'                                                    Insert      17:43:01
Command->
0001 *
0002 * Insert file for named commons that can be used in the
0003 * local routine developed for AA. Local routines can be
0004 * attached at various places in AA framework like pre and post
0005 * routine before and after an action, charge routine, payment
0006 * calculation routine
0007 * These common variables are equivalent of core AA common variable
0008
0009     COMMON/AALOCALCOMMON/c_aalocArrId,  ;* Arrangement id
0010     c_aalocArrangementRec,     ;*Main arrangement record
0011     c_aalocArrProductId,       ;* Stores the product used by the arrangement
0012     c_aalocArrCurrency,        ;* Store the currency used by the arrangement
0013     c_aalocActivityId,         ;* Activity id loaded in the property
0014     c_aalocTxnReference,       ;* Arrangement activity transaction reference
0015     c_aalocCurrAction,         ;* Current action being processed by action manager
0016     c_aalocArrActivityRec,     ;* Current arrangement activity record
0017     c_aalocLocalRefList,       ;* List of local ref fields
0018     c_aalocCurrActivity,       ;* User/System generated activities
0019     c_aalocLinkedAccount,      ;* Linked Account id
0020     c_aalocProductRecord,      ;* Stores the published product record for the arrangement
0021     c_aalocPropClassId,        ;* Property class ID of current property
0022     c_aalocPropertyId,         ;* Current property
0023     c_aalocProdPropRec,        ;* Stores the published product property for the arrangement
0024     c_aalocActivityStatus,     ;* Stores the status of current activity like . UNAUTH, DEL, REV-DEL
0025     c_aalocArrActivityId,      ;* Store the arrangement activity id
0026     c_aalocActivityEffDate,    ;* Effective date
0027     c_aalocAccountDetails,     ;* Account Details
0028     c_aalocForwardDelCond,     ;* If user deleted the condition dynamically then system will set this flag in all the stages
0029     c_aalocMasterActivity,     ;* Master activity Id if required for local routines
0030     c_aalocReserved5,
0031     c_aalocReserved4,
0032     c_aalocReserved3,
0033     c_aalocReserved2,
0034     c_aalocReserved1
0035
0036
0037 *-----------------------------------------------------------------------------
0038
------------------------------------------------------------------------ End Of Record ----------------------------------------------------------------------
```

Find below the common variables of **I_F.AA.APP.COMMON**



```
File  Edit  Setup  Control  Window  Resize  Help
File T24.BP , Record 'I_AA.APP.COMMON'                                                      Insert      12:55:51
Command->
0001     COMMON/AAAAP/AA$PROPERTY.CLASS.ID,  ;* Property class ID of current property
0002     AA$PROPERTY.CLASS.REC,     ;* Property class record of current property
0003     AA$PRODUCT.ARR, ;* Flag to indicate if we are processing PROD or ARRangement level
0004     AA$RULE.TYPE,    ;* Array of rule types associated with each field
0005     AA$MAND,         ;* Mandatory indicator P (Mand Prod), A (Mand Arr) and negotiable opt
0006     AA$FIELD.ACTIVITY,         ;* Field activities allowed
0007     AA$FIELD.DATA.TYPE,        ;* Array of field types associated with each field
0008     AA$DATED.ID,     ;* Property Id is dated
0009     AA$CCY.ID,       ;* Property Id contains Ccy
0010     AA$OPT.CCY.ID,   ;* Property Id can be without Currency, this is valid for charge property alone
0011     AA$MULTI.PROPERTY,         ;* Multiple Properties allowed per product
0012     AA$CUST.ID,      ;* Customer level id
0013     AA$ARR.ID,       ;* Arrangement Id
0014     AA$R.ARRANGEMENT,          ;* Main Arrangement Record
0015     AA$PROP.EFF.DATE,          ;* The effective of the current property being processed
0016     AA$PROPERTY.ID, ;* Current property
0017     AA$PREV.PROD.PROP.REC,     ;* Stores the previous product property record during CHANGE.PRODUCT activity
0018     AA$OVERRIDE.LIST,          ;* Default field override list
0019     AA$ARR.LINK.TYPE,          ;* Link between arrangement and product property
0020     AA$ARR.PRODUCT.ID,         ;* Stores the Product used by the arrangement
0021     AA$ARR.CURRENCY,           ;* Stores the Currency used by the arrangement
0022     AA$ARR.CHANGED.FIELDS,     ;* Stores the list of changed fields on an arrangement property record
0023    .AA$ORIGINAL.FUNCTION,      ;* During authorisation and deletion this common will store the record function from
0024     AA$PRODUCT.RECORD,         ;* Stores the published Product record for the arrangement
0025     AA$PROD.PROP.RECORD,       ;* Stores the Published product property for the arrangement
0026     AA$ARR.PC.ID,    ;* Full property class id
0027     AA$ACTIVITY.ID, ;* Activity id, loaded in the property templates
0028     AA$ACTIVITY.EFF.DATE,      ;* Activity Effective date, loaded in the property templates
0029     AA$ARRANGEMENT.STATUS,     ;* Arrangement status record
0030     AA$TXN.REFERENCE,          ;* Arrangement Activity transaction reference - for accounting entries, etc
0031     AA$CURR.ACTION, ;* Current action being processed by Action manager
0032     AA$R.ARRANGEMENT.ACTIVITY,          ;* Current Arrangement Activity record
0033     AA$XREF.ID.REQD,           ;* If Xref Id is required, flag this
0034     AA$LOCAL.REF.LIST,         ;* Local Reference items
0035     AA$ASSOC,        ;* AA association for a field
0036     AA$F, ;* AA F array
0037     AA$FNO,          ;* AA Field number for array
0038     AA$N, ;* N Array
0039     AA$T, ;* T Array
0051     AA$MASTER.ACTIVITY,        ;* It contains Master Activity ID and name (Seperated by VM), that is, topmost activ
```





c_aalocArrProductId - The above contract shows product id is “OTHER LOAN/INDIVIDUAL”
c_aalocArrCurrency - Currency “JOD”
c_aalocActivityId – Activity ID “AAACT13274MBB57Y56”
c_aalocLinkedAccount – Linked acc no – “1000010703”
c_aalocArrId - Arrangement ID “AA13274QWHBW”
c_aalocActivityStatus – Like “UNAUTH” , “REVERSE” ,”DELETE”, “AUTH”
c_aalocPropertyId – Like “PRININTEREST”
c_aalocPropClassId – Like “TERM.AMOUNT”

Likewise, above mentioned AA Insert files are having many common variables.

7.3 Property Classes

For reading the AA record is different from the normal read command (CALL F.READ()), Initially we need to get the Arrangement ID.

Mostly used property classes are,

- CLOSURE
- CUSTOMER
- INTEREST
- LIMIT
- TERM.AMOUNT
- OFFICERS
- PERIODIC.CHARGES
- SETTLEMENT
- BALANCE.MAINTENANCE
- CHARGE.OVERRIDE
- CHARGE

8 AA at routine level

Initially get the Arrangement Id to process the Property class find below details of the routine,

8.1 Core APIs

The following APIs may be utilized when certain information's are required in a local routine:

- **AA.GET.ARRANGEMENT.CONDITION**
  - This routine can be used by any application to retrieve the property conditions that apply to an arrangement for a particular effective date (default value is TODAY).
  - The specific property or property class required must be supplied
  - The records and ids for the property conditions that apply are returned
  - No. of arguments passed – 7

Parameter type	Name	Description
IN (Mandatory)	idArrangementComp	The Id of the arrangement for which property condition is requested.
IN (Optional)	idPropertyClass	Property class for which property details are sought. If the class contains multiple properties in this arrangement, system returns the records for all properties
IN (Optional)	idProperty	Specific property for which details are sought. If class is not supplied, property must be stated.
IN (Optional)	effectiveDate	The date for which property details are sought. If the property record does not have a record on that date, the past record nearest to that date

		is returned. If a date is not supplied, TODAY would be assumed.
OUT	returnIds	List of property Ids for this class separated by FMs
OUT	returnConditions	List of property records – lowered to be in sync with Ids
OUT	returnError	Errors if any during the processing

- **AA.GEN.ARRANGEMENT.ACTIVITY.FIELDS**
  - This method maps the arrangement properties, its field name and its value to new arrangement activity record
  - Values can be supplied to arrangement property conditions from AA.ARRANGEMENT.ACTIVITY record while triggering an activity
  - No. of arguments passed – 4

Parameter Type	Name	Description
IN(Mandatory)	Property list	List of property names of the arrangement whose values are supplied. Properties are separated by value markers
IN(Mandatory)	Field name list	List of field names for each of the properties whose values are supplied. For a property the field names are separated by sub value markers
IN(Mandatory)	Field value list	Actual values for the list of fields for each of the arrangement properties. For a property the field values are separated by sub value markers
OUT	ARR ACT Fields record	The formatted arrangement activity record

- **AA.GEN.NEW.ARRANGEMENT.ACTIVITY**
  - This method appends the new arrangement activity as the secondary activity to the parent activity
  - No. of arguments passed – 7

Parameter Type	Name	Description
IN(Mandatory)	Arrangement Id	The arrangement id reference
IN(Mandatory)	Secondary activity	New secondary activity name to be triggered.
IN(Mandatory)	Effective date	Effective date of the activity
IN(Mandatory)	Txn Details	The arrangement product that is effective on the date passed for this arrangement
IN(Mandatory)	Parent activity id	The list of properties that are eligible for this product(separated by VMs)
IN(Mandatory)	Arrangement activity rec	New arrangement activity record
OUT	Return error	Error message if any

- **AA.GET.ECB.BALANCE.AMOUNT**
  - This method returns the total balance amount for a given balance type on a given date for an account arrangement
  - No. of arguments passed – 5

Parameter Type	Name	Description
IN(Mandatory)	Account Id	The account number of the arrangement
IN(Mandatory)	Balance type	The balance type for which the balance is sought.
IN(Mandatory)	Request date	Date for which balance is sought
OUT	Balance amount	The balance amount as in EB.CONTRACT.BALANCES as on the requested date. If the balance type is virtual then sum of all the virtual balance is returned
OUT	Return error	Error if any

8.2 Reading the property class

Initially get the arrangement using common variable – "c_allocArrId" and read the AA.ACTIVITY.HISTORY table to get the Arrangement Activity id.

Used common variables are – "c_allocArrId", "c_allocActivityEffDate"

The below routine used the Core API – AA.GET.ARRANGEMENT.CONDITIONS to read and write the AA record.

Total Arguments - 7
Passed Arguments are – Arrangement ID, Prop class, Property, R.Conditions and Err message.

8.3 RAISE Conditions

RAISE(R.CONDITION) is used to get the values for the defined property class "TERM.AMOUNT".

The below routine to get the Amount, tenor, and maturity date of the Term amount property class values.

```
ARRANGEMENT.ID = c_aalocArrId
CALL F.READ(FN.AA.ACTIVITY.HISTORY,ARRANGEMENT.ID,R.AA.ACTIVITY.HISTORY,F.AA.ACTIVITY.HISTORY,AA.ACTIVITY.HISTORY.ERR)

LOCATE "LENDING-NEW-ARRANGEMENT" IN R.AA.ACTIVITY.HISTORY<AA.AH.ACTIVITY,1,1> SETTING ACT.POS THEN
    Y.AA.ACT.ID = R.AA.ACTIVITY.HISTORY<AA.AH.ACTIVITY.REF,1,ACT.POS>
    CALL F.READ(FN.AA.ARRANGEMENT.ACTIVITY,Y.AA.ACT.ID,R.AA.ARRANGEMENT.ACTIVITY,F.AA.ARRANGEMENT.ACTIVITY,ARR.ACT.ERR)
    Y.EFF.DATE = R.AA.ARRANGEMENT.ACTIVITY<AA.ARR.ACT.EFFECTIVE.DATE>

    Y.TERM.AMOUNT.ID = ARRANGEMENT.ID:"-COMMITMENT-":Y.EFF.DATE:".1"
    CALL F.READ(FN.AA.TERM,Y.TERM.AMOUNT.ID,REC.TERM.AMT,F.AA.TERM,ERR.AA.TERM)
    Y.AMOUNT = REC.TERM.AMT<AA.AMT.AMOUNT>
    Y.TENOR = REC.TERM.AMT<AA.AMT.TERM>
    Y.MAT.DATE = REC.TERM.AMT<AA.AMT.MATURITY.DATE>
END ELSE

    Y.EFF.DATE = c_aalocActivityEffDate
    PROP.CLASS = "TERM.AMOUNT"
    PROPERTY = ''
    R.CONDITION = ''
    ERR.MSG = ''

    CALL AA.GET.ARRANGEMENT.CONDITIONS(ARRANGEMENT.ID,PROP.CLASS,PROPERTY,'','',R.CONDITION,ERR.MSG)
    R.CONDITION = RAISE(R.CONDITION)
    Y.AMOUNT = R.CONDITION<AA.AMT.AMOUNT>
    Y.TENOR = R.CONDITION<AA.AMT.TERM>
    Y.MAT.DATE = R.CONDITION<AA.AMT.MATURITY.DATE>
END
```

Using this we can get the records for the tables or any property class

8.4 Common variables

Below Common variables are used to check the conditions,

```
IF V$FUNCTION EQ 'I' AND c_aalocPropertyId EQ "PRININTEREST" THEN
    *1975 - S
    IF CALC.ARATE THEN
        GOSUB APR.OVERRDE
    END
    *1975 - E
END

IF c_aalocActivityStatus EQ 'AUTH' AND c_aalocPropertyId EQ "PRININTEREST" THEN
    GOSUB CREATE.CONCAT
END
```

8.5 Select Command in the AA record

The below example is used to select the AA Bill record by using the Core API – AA.GET.BILL.ID
Total Passed Arguments - 8

```
SEL.BILL:
*********

    CALL AA.GET.BILL.ID(ARRANGEMENT.ID, 'NEW', PAYMENT.DATE, BILL.DATE, 'DUE', PAYMENT.METHOD, BILL.IDS, RETURN.ERROR)
    LOOP
        REMOVE Y.AA.BILL.ID FROM BILL.IDS SETTING BILL.ID.POS
    WHILE Y.AA.BILL.ID:BILL.IDS.POS
        GOSUB READ.BILL.DETAILS
    REPEAT

    RETURN
```

9 Updating fields in AA

9.1 Authorizing the record

For updating fields, use below logic, instead of using F.WRITE api,

R.AA.ACC.RECORD<AA.AC.LOCAL.REF,Y.RESTRUCT.DATE.POS,Y.CNT+1> = TODAY
R.AA.ACC.RECORD<AA.AC.LOCAL.REF,Y.END.DATE.POS,Y.CNT+1> = Y.DATE

The below condition is used to check the status of the record ,

REQD.PROCESS = c_arrActivityStatus["-",1,1]

REQD.PROCESS EQ 'UNAUTH' then

Below API “AA.ACTION.LIST.MANAGER” is used to “APPEND” the fields like OFS process.

CALL AA.ACTION.LIST.MANAGER("APPEND", "", "", "")

```
MSG.SYNTAX = "OFS":FM:"FIRST":FM:"BUILD"
RECORD.ERRORS = ""
RECORD.OVERRIDES = ""
OFS.PROCESS = 'PROCESS'
UPDATE.FUNCTION = "I"

CALL AA.ACTION.LIST.MANAGER("INIT", "", "", "")
CALL AA.PROCESS.CONTRACT(OFS.PROCESS, MSG.SYNTAX, "AA.ARR.ACCOUNT", Y.AA.ACC.ID, UPDATE.FUNCTION, R.AA.ACC.RECORD, RECORD.ERRORS, RECORD.OVERRIDES, RET
CALL AA.ACTION.LIST.MANAGER("APPEND", "", "", "")
```

Note:

Use AA.ACTION.LIST.MANAGER api only for updating the fields in AA and for other applications, use F.WRITE.

9.2 Updating existing record

The below logics are used to updating the existing records,

REQD.PROCESS = c_arrActivityStatus["-",1,1]

REQD.PROCESS EQ 'AUTH' then

CALL AA.ACTION.LIST.MANAGER("INSERT", "", "", "")

Below API “AA.ACTION.LIST.MANAGER” is used to “INSERT” the fields like OFS process.

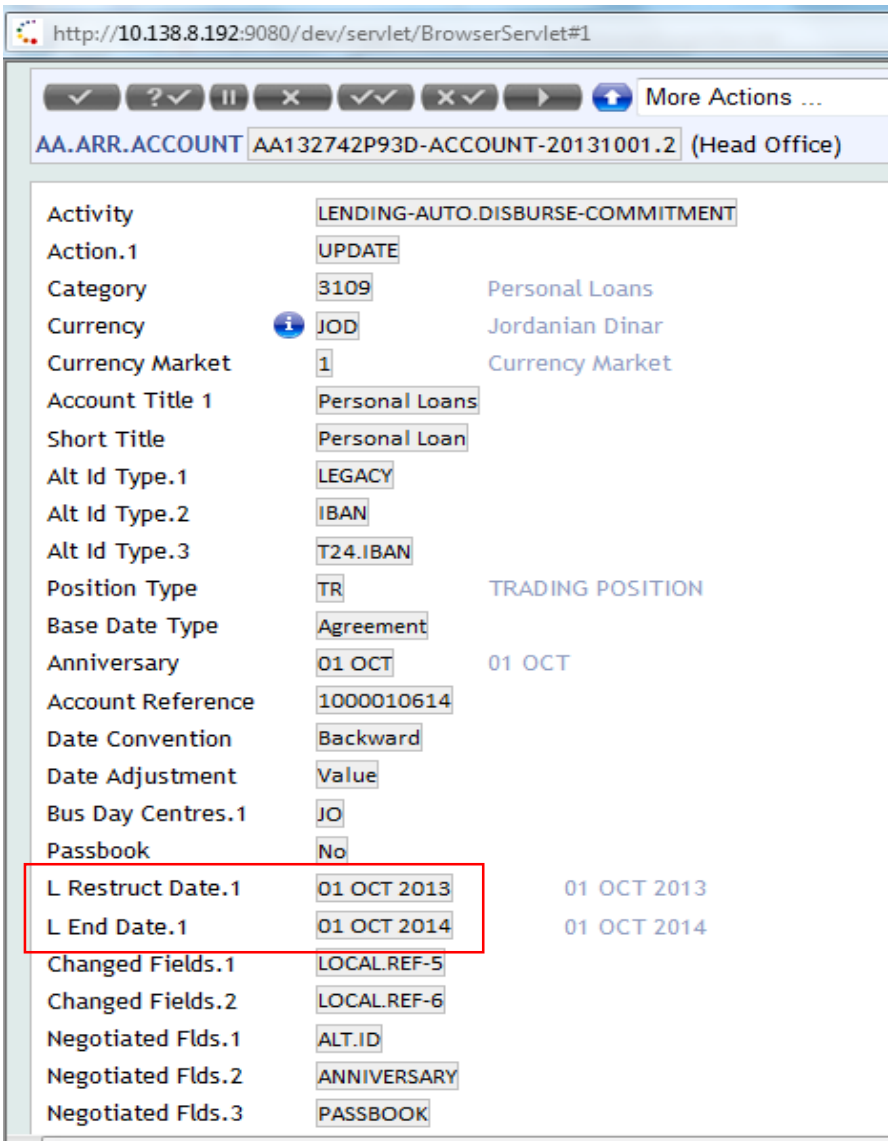
```
IF REQD.PROCESS EQ 'AUTH' THEN

    PROP.CLASS = 'ACCOUNT'
    PROP.NAME  = ''
    returnConditions = ''
    RET.ERR = ''
    CALL F.READ(FN.AA.ARRANGEMENT,ARRANGEMENT.ID,R.AA.ARRANGEMENT,F.AA.ARRANGEMENT,ERR.AA.ARRANGEMENT)
    Y.EFFEC.DATE = R.AA.ARRANGEMENT<AA.ARR.START.DATE>
    CALL AA.GET.ARRANGEMENT.CONDITIONS(ARRANGEMENT.ID,PROP.CLASS,PROP.NAME,Y.EFFEC.DATE,returnIds,returnConditions,E)
    R.AA.ACC.RECORD = RAISE(returnConditions)
    Y.AA.ACC.ID = R.AA.ACC.RECORD<AA.AC.ID.COMP.1>:'-':R.AA.ACC.RECORD<AA.AC.ID.COMP.2>:'-':R.AA.ACC.RECORD<AA.AC.ID.COMP.3>
1-Overview -S
    MSG.SYNTAX = "OFS":FM:"FIRST":FM:"BUILD"
    RECORD.ERRORS = ""
    RECORD.OVERRIDES = ""
    OFS.PROCESS = 'PROCESS'
    UPDATE.FUNCTION = "A"

    CALL AA.ACTION.LIST.MANAGER("INIT", "", "", "")
CALL AA.PROCESS.CONTRACT(OFS.PROCESS, MSG.SYNTAX, "AA.ARR.ACCOUNT", Y.AA.ACC.ID, UPDATE.FUNCTION, R.AA.ACC.RECORD, RECORD.ERRORS, RECORD.OVERRIDES, RET.ERR
    CALL AA.ACTION.LIST.MANAGER("INSERT", "", "", "")

END
```

9.3 Updated Restructure date & End date below



10 T24 Browser Level debug in TAFJ for AA routine

10.1 Browser Listener

Initially start the browser. Listener in the Eclipse then input the contract in the T24 Browser.

If the routine is attached, then during the respective activity, routine is triggered which is attached (like POST, PRE and Validation routine) in the products setup.

10.2 Variables Tab

Below the AA Routine triggered for the respective AA activity.
Already mentioned the AA common variable in the section- 7.2

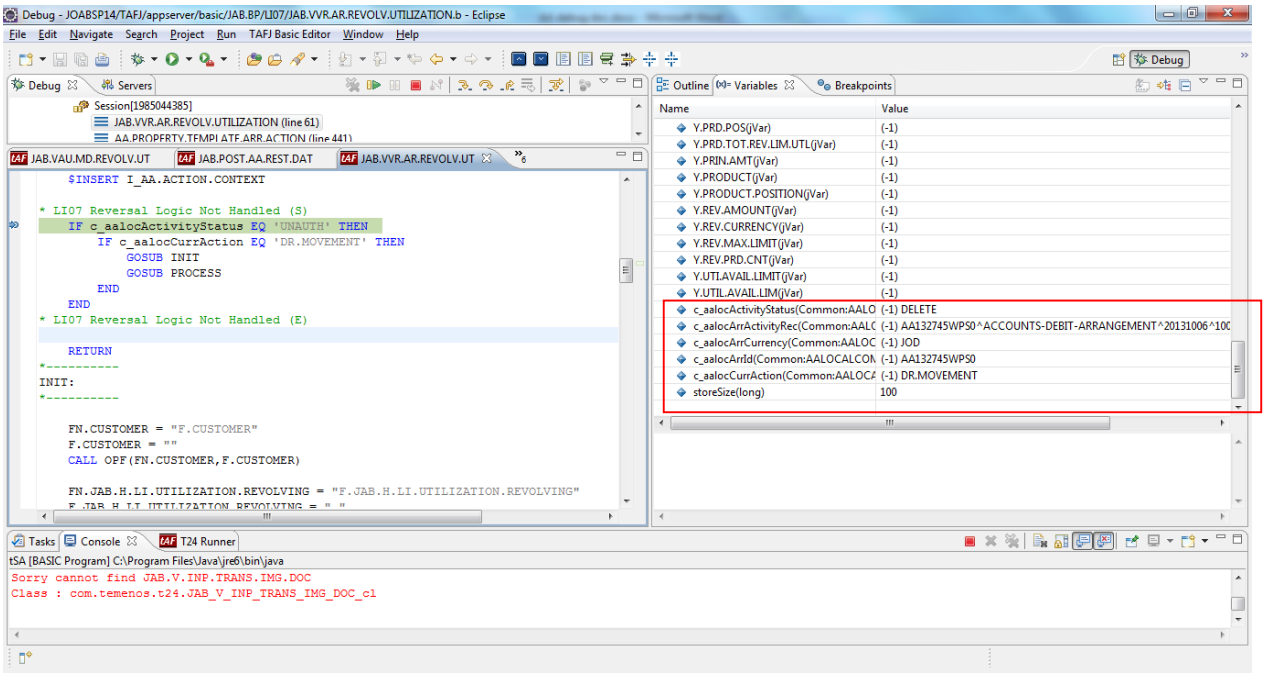
AA Variables and Values:

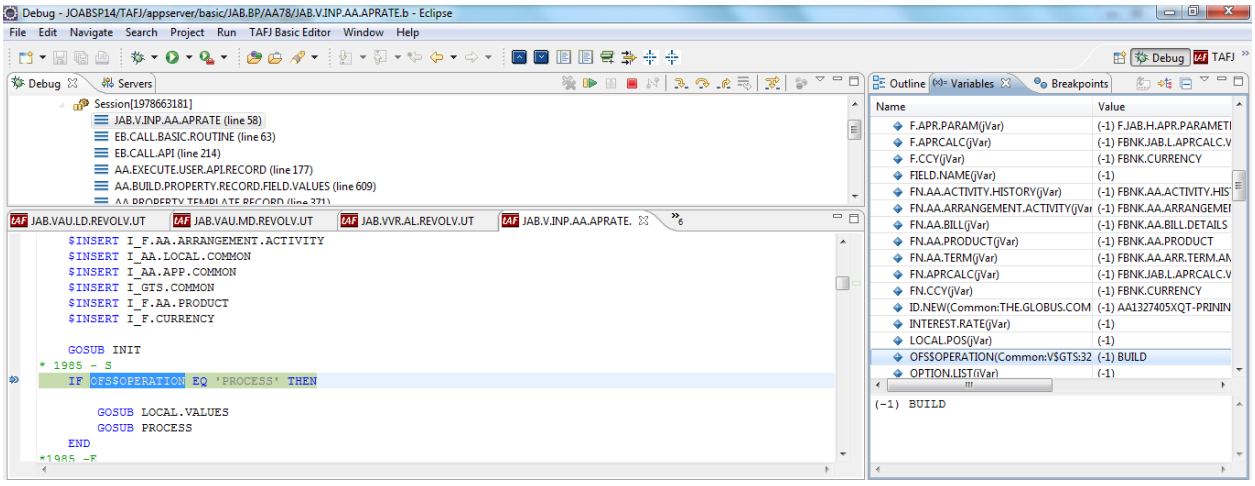
In the variables tab, find the values of the AA common variables,

For Ex:

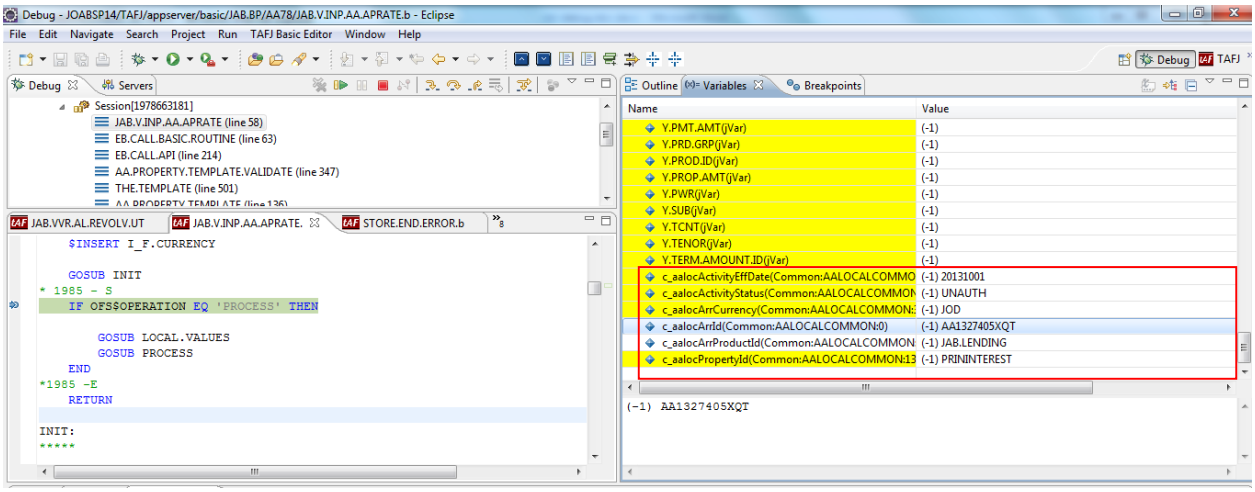
c_aalocActivityStatus - DELETE
c_aalocCurrAction - DR.MOVEMENT.
c_aalocArrCurrency - JOD
c_aalocPropertyId - PRININTEREST
c_aalocArrProductId - JAB.LENDING

While processing batch in COB/ONLINE, Activity Status is like UNAUTH or DELETE.





Below screenshot shows the common variable values of Arrangement ID, Product ID, Property ID, Currency and Status of AA contract.


Debug - JOABSP14/TAFJ/appserver/basic/JAB.BP/AA78/JAB.V.INP.AA.APRATE.b - Eclipse
File Edit Navigate Search Project Run TAFJ Basic Editor Window Help
Debug Servers
Session[1978663181]
JAB.V.INP.AA.APRATE (line 58)
EB.CALL.BASIC.ROUTINE (line 63)
EB.CALL.API (line 214)
AA.PROPERTY.TEMPLATE.VALIDATE (line 347)
THE.TEMPLATE (line 501)
JAB.VVR.AL.REVOLV.UT
JAB.V.INP.AA.APRATE.
STORE.END.ERROR.b
$INSERT I_F.CURRENCY
GOSUB INIT
* 1985 - S
IF OFS$OPERATION EQ 'PROCESS' THEN
GOSUB LOCAL.VALUES
GOSUB PROCESS
END
*1985 -E
RETURN
INIT:
*****
Outline Variables Breakpoints
Name
Value
Y.PMT.AMT(jVar) (-1)
Y.PRD.GRP(jVar) (-1)
Y.PROD.ID(jVar) (-1)
Y.PROP.AMT(jVar) (-1)
Y.PWR(jVar) (-1)
Y.SUB(jVar) (-1)
Y.TCNT(jVar) (-1)
Y.TENOR(jVar) (-1)
Y.TERM.AMOUNT.ID(jVar) (-1)
c_aalocActivityEffDate(Common:AALOCALCOMMO (-1) 20131001
c_aalocActivityStatus(Common:AALOCALCOMMON (-1) UNAUTH
c_aalocArrCurrency(Common:AALOCALCOMMON: (-1) JOD
c_aalocArrId(Common:AALOCALCOMMON:0) (-1) AA1327405XQT
c_aalocArrProductId(Common:AALOCALCOMMON (-1) JAB.LENDING
c_aalocPropertyId(Common:AALOCALCOMMON:13 (-1) PRININTEREST
(-1) AA1327405XQT