Template Programming

TEMENOS EDUCATION CENTRE

Agenda

- What is a Template program?
- Why Template programming?
- Major components in a Template
- Types of Template
- Creation of a new application using the Template

What is a Template Program?

APPLICATION PROGRAM

- Get data
- Stores data

Why T24 uses Template Programming?

Any user who wishes to create a new application on T24 has to make use of an existing program in T24 called a TEMPLATE in order to

- Maintain code consistency and code standard
- Rapid code development
- Eliminate mundane tasks

Major components

When creating a new application, there are various factors that need to be kept in mind, like the

- Functions that can be allowed for the application
- Fields that comprise the application
- Field level validations
- Input level validations
- Authorization level validations
- Accounting entries that need to be generated for the transaction of the application
- Limit checks
- Generation of delivery advices
- End of day processing

Types of Template

Following are the different types of template programs available

- TEMPLATE
- TEMPLATE.L
- TEMPLATE.T
- TEMPLATE.W

INSURANCE application an example

Using the INSURANCE application, a customer can deposit an Insurance policy with the bank. There are different types of policies that a customer can deposit. The INSURANCE application consists of 2 main files.

- The INSURANCE file – Allows a customer to deposit a policy and also allows the bank to charge a commission/charge
- The POLICY.TYPE file that holds details of the various policies that a customer can deposit with the bank.

Fields that comprise the main INSURANCE file

No	Field Name	Type	Description	Mandatory
1	POLICY.NO	16,A	Standard T24 ID – Prefix IN	Y
2	POLICY.TYPE	35,A	Should be a valid Policy Type in the POLICY.TYPE file.(See Below)	Y
3	CUSTOMER.NO	10,CUS	Must be a valid CUSTOMER	Y
4	CURRENCY	3,CCY	Must be a valid CURRENCY	Y
5	COVER.AMOUNT	16,AMT		Y
6	PREMIUM.AMOUNT	16,AMT		Y
7	START.DATE	11,D	Defaults to TODAY	
8	PAYMENT.FRQ	16,FQO	Must be after START.DATE Must be after TODAY	
9	XX<BENEFICIARY	10,CUS or 35,A	Free Text	
10	XX>ADDRESS	35,A	Free Text	
11	AGREEMENT.TYPE	35,A	Only the following options are valid OPTION1 OPTION2 OPTION3 OPTION4 Either the AGREEMENT.TYPE or CONDITIONS must be set	
12	CONDITIONS	35,A	Free Text	
13	XX.LOCAL.REF			

Fields that comprise the POLICY.TYPE application

No	Field Name	Type	Description	Mandatory
1	POLICY.TYPE.ID	7,A		Y
2	XX.LL.DESCRIPTION	35,A	Description of the policy type	Y
3	CATEGORY		Must be a valid CATEGORY Must be in the product range 20000 - 49999	Y
4	RESERVED12-1	NOINPUT		

Steps to create the INSURANCE Template

Step 1:

Make a copy of the TEMPLATE program that is available in the GLOBUS.BP directory on to your local directory (Example TRG.BP). Ensure that you save the TEMPLATE program with a different name (Insurance – Name of the application that we are to create).

```
jsh…>COPY FROM GLOBUS.BP TO DEV.BP TEMPLATE

jsh…>sh

jsh…>cd DEV.BP

jsh…>mv TEMPLATE INSURANCE

jsh…>cd

jsh…>COPY FROM GLOBUS.BP TO DEV.BP TEMPLATE

jsh…>sh

jsh…>cd DEV.BP

jsh…>mv TEMPLATE POLICY.TYPE
```

Steps to create the new INSURANCE Template

Step 2:
Change the name of the template program to your application name (Insurance).

```
* SUBROUTINE TEMPLATE
 ***********************************************************************
 * Enter Program Description Here
 *-----------------------------------------------------------------------
    $INSERT I_COMMON
    $INSERT I_EQUATE
```

Change TEMPLATE to your Application name (Insurance)

Steps to create the new INSURANCE Template

Step 3:

Basic template program is ready

Add relevant code in the appropriate paragraphs.

Add the various fields that the Insurance application needs to contain

Steps to create the new INSURANCE Template

DEFINE.PARAMETERS:

To define the various fields that comprise an application.

Edit and add the necessary fields for our Insurance application.

```
**********************************************************************

DEFINE.PARAMETERS:    * SEE 'I_RULES' FOR DESCRIPTIONS *

REM > CALL XX.FIELD.DEFINITIONS

RETURN

**********************************************************************
```

Remove the REM statement from the line CALL XX.FIELD.DEFINITIONS and replace the XX to INSURANCE (application name)

```
CALL INSURANCE.FIELD.DEFINITIONS
```

XX.FIELD.DEFINITIONS

```
SUBROUTINE XX.FIELD.DEFINITIONS

$INSERT I_COMMON

$INSERT I_EQUATE

$INSERT I_F.ACCOUNT          ; * Other Inserts required for Check files, etc.

$INSERT I_F.CUSTOMER

*-----------------------------------------------------------------------

    GOSUB INITIALISE

    GOSUB DEFINE.FIELDS

    RETURN

*-----------------------------------------------------------------------
```

Paragraph to initialize variables

Paragraph to define the fields of the application

XX.FIELD.DEFINITIONS

```
DEFINE.FIELDS:


ID.F = "ID .NO" ; ID.N = "16" ; ID.T = "A"
    Z=0
REM > Z+=1 ; F(Z) = "XX.XX.FIELD.NAME" ; N(Z) = "35.2" ; T(Z) = "A"
REM > Z+=1 ; F(Z) = "XX.XX.FIELD.NAME" ; N(Z) = "35.2" ; T(Z) = "A"
V = Z + 9
RETURN
*-----------------------------------------------------------------------
INITIALISE:

    MAT F = "" ; MAT N = "" ; MAT T = ""
    MAT CHECKFILE = "" ; MAT CONCATFILE = ""
    ID.CHECKFILE = "" ; ID.CONCATFILE = ""

* Define often used check file variables

    CHK.ACCOUNT = "ACCOUNT":FM:AC.SHORT.TITLE:FM:"L"
    CHK.CUSTOMER = "CUSTOMER":FM:EB.CUS.SHORT.NAME:FM:'.A'
    RETURN
-----------------------------------------------------------------------
END
```

Actual field definition

V:Total number of fields in an application
Z:Number of fields defined
V=Z+9 (To add 9 audit fields)

Steps to create the new Insurance Template

Step 4:

Copy the skeleton program XX.FIELD.DEFINITIONS from GLOBUS.BP to BP with the name INSURANCE.FIELD.DEFINITIONS

```
jsh…>COPY FROM GLOBUS.BP TO DEV.BP XX.FIELD.DEFINITIONS

jsh…>sh

jsh…>cd DEV.BP

jsh…>mv XX.FIELD.DEFINITIONS INSURANCE.FIELD.DEFINITIONS

jsh…>cd
```

```
jsh…>COPY FROM GLOBUS.BP TO DEV.BP XX.FIELD.DEFINITIONS

jsh…>sh

jsh…>cd DEV.BP

jsh…>mv XX.FIELD.DEFINITIONS POLICY.TYPE.FIELD.DEFINITIONS
```

Steps to create the new Insurance Template

Step 5:

INSURANCE.FIELD.DEFINITIONS is ready

Edit and add the necessary code

Specify the various attributes of a field

Steps to create the new Insurance Template

Field Attributes:

- The maximum number of characters the field can hold.
- The minimum number of characters a field can hold
- The type of data the field can hold (Alphanumeric, Integer, Date etc)
- Pick list for the field (if any)
- Other attributes of a field like whether it is a NOINPUT field or a NOCHANGE, MULTIVALUE field, SUBVALUE field, etc.

Attributes of the field

Attributes of a field have to be specified when a field is created. In order to specify the attributes for a field, T24 gives us 3 arrays namely the

- F array
- N Array
- T Array

F Array:
To define the name of the field and to specify if the field is a single value, multi value or a sub value field. This array also allows us to specify a field as a language specific field.

Attributes of the field

Attributes of a field have to be specified when a field is created. In order to specify the attributes for a field, T24 gives us 3 arrays namely the

- F array
- N Array
- T Array

N Array:
To define the maximum and minimum number of characters a field can hold. This array is also used to set a flag to specify whether any other validations apart from the data type validation have to be performed or not.

Attributes of the field

Attributes of a field have to be specified when a field is created. In order to specify the attributes for a field, T24 gives us 3 arrays namely the

- F array
- N Array
- T Array

T Array:
To define pick list for a field, if any.
Used to specify whether a field is a NOINPUT field or a NOCHANGE field.

Attributes of the fields

Following is the way to define a field using the 3 different arrays

F(1) = “Fieldname” ; N(1) = “35.1.C” ;T(1)<1> = “A”

These refer to the field numbers. For definition of the 2nd field, they would become ‘2’ and so on.

F Array

F Array has been used to define the name of the Field.

Define a single value field in the following way
F(1) = “Fieldname”

Define a multi value field in the following way
F(1) = “XX.Fieldname”

Define a sub value field in the following way
F(1) = “XX.XX.Fieldname”

In order to define an associated multi value field the definitions has to be as follows
F(1) = “XX.XX<Fieldname1”
F(2) = “XX.XX-FieldName2”
F(3) = “XX.XX>FieldName3”

N Array

The N array comprises of 3 parts

- The first part is used to specify the maximum number of characters.
- The second part is used to specify the minimum number of characters.
- A value ‘C’ in the third part of this array is used to signify that extra validation has to be done for this field.

Example : N(1) = “35.1.C”

T Array

The T array consists of 3 elements.

Element 1	Element2	Element3
IN2 Routines	Pick Lists	NOINPUT/NOCHANGE

T Array

Element 1:

The first element of the T array is used to define the type of data the field has to contain.

Example : T(1)<1> = ‘A’

The value input for the 1st element refers to a suffix of an IN2 routine.

IN2 Routines

There are a number of IN2 routines existing in T24. Using these IN2 routines we can restrict the type of data that is entered into a field.

IN2 Routine Name	Description
IN2AAA	Characters from a-z and A-Z only
IN2	Numeric data
IN2D	Will accept input in any of several valid date formats. The date input will be expanded to the full date YYYYMMDD and displayed as DD MMM YYYY.
IN2AMT	This routine accepts input of an amount and can edit the input to have the number of decimal places defined for a specified currency.
IN2CUS	Input must be a valid customer number, i.e. 1 - 10 numeric. Input of a customer mnemonic will also be accepted and converted to the customer number.

T Array

Element 2:

The second element of the T array is used to define pick lists for a field.

Example : T(1)<2> = “Value1_Value2_Value3_Value4_Value5”

Incase the pick list for a field has to contain values of another field in another Application, then a component called ‘CHECKFILE’ can be used.

F(1) = “Fieldname” ; N(1) = “35.1.C” ; T(1)<1> = “A”
CHECKFILE = “Filename” : FM : Fieldname : FM : “L”

Name of the file that has to be read(**NO** F Prefix)
Name of the field whose value has to be used as enrichment
Return a language associated enrichment

T Array

Element 3:

The third element of the T array is used to make a field a NOINPUT or an NOCHANGE field.

Note : (Remember a field cannot be NOINPUT and NOCHANGE)
T(1)<3> = “NOINPUT” (or)
T(1)<3> = “NOCHANGE”

POLICY.TYPE.FIELD.DEFINITIONS

```
SUBROUTINE POLICY.TYPE.FIELD.DEFINITIONS
$INSERT I_COMMON
$INSERT I_EQUATE

GOSUB INITIALISE
GOSUB DEFINE.FIELDS
RETURN
```

POLICY.TYPE.FIELD.DEFINITIONS

```
DEFINE.FIELDS:

ID.F="POLICY.TYPE.ID";ID.N="7";ID.T=""
Z=0
Z+=1; F(Z)="XX.LL.DESCRIPTION";N(Z)="35.1"; T(Z)="A"
Z+=1; F(Z)="CATEGORY";N(Z)="5.1.C";T(Z)=""
Z+=1; F(Z)="RESERVED12";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED11";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED10";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED9";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED8";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED7";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED6";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED5";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED4";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
```

POLICY.TYPE.FIELD.DEFINITIONS

```
Z+=1; F(Z)="RESERVED3";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED2";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
Z+=1; F(Z)="RESERVED1";N(Z)="35..";T(Z)="A";T(Z)<3>='NOINPUT'
V=Z+9
RETURN

INITIALISE:

MAT F="";MAT N="";MAT T=""
MAT CHECKFILE = "";MAT CONCATFILE =""
ID.CHECKFILE =="";ID.CONCATEFILE=""
RETURN
END
```

Compiling POLICY.TYPE application

```
EB.COMPILE  DEV.BP POLICY.TYPE.FIELD.DEFINITIONS
EB.COMPILE  DEV.BP POLICY.TYPE
```

FILE.LAYOUT

Capable of creating insert files.

Run this utility from the ‘jsh’ prompt.

FILE.LAYOUT

```
jsh…> FILE.LAYOUT
Use Select List or Input Individually either :
List of FILE names from which to build insert modules via their dictionarys
List of PROGRAM names from which to build insert modules through calling.
Enter Program/File(s) :POLICY.TYPE
Enter Program/File(s) :

Build Insert from File (D)ictionarys or (P)rograms or (Q)uit
<CR = (P)rograms>  :P
Building Insert From Program(s)
```

```
Program is POLICY.TYPE

Enter output Name - <CR> = I_F.Entryname  :I_F.POLICY.TYPE

Enter PREFIX or <CR> = NONE :POL

Enter SUFFIX or <CR> = NONE :

Processed 23 matrix entries for POLICY.TYPE program
```

I_F.POLICY.TYPE

Find below the insert file for the POLICY.TYPE application.

Name : I_F.POLICY.TYPE

```
* File Layout for POLICY.TYPE Created 10 SEP 04 at 09:35AM by g14007
*      PREFIX[POL.]      SUFFIX[]
 EQU POL.DESCRIPTION TO 1,         POL.CATEGORY TO 2,
      POL.RESERVED12 TO 3,       POL.RESERVED11 TO 4,
      POL.RESERVED10 TO 5,        POL.RESERVED9 TO 6,
       POL.RESERVED8 TO 7,        POL.RESERVED7 TO 8,
       POL.RESERVED6 TO 9,       POL.RESERVED5 TO 10,
      POL.RESERVED4 TO 11,       POL.RESERVED3 TO 12,
      POL.RESERVED2 TO 13,       POL.RESERVED1 TO 14,
  POL.RECORD.STATUS TO 15,         POL.CURR.NO TO 16,
       POL.INPUTTER TO 17,       POL.DATE.TIME TO 18,
     POL.AUTHORISER TO 19,         POL.CO.CODE TO 20,
      POL.DEPT.CODE TO 21,    POL.AUDITOR.CODE TO 22,
POL.AUDIT.DATE.TIME TO 23
```

INSURANCE.FIELD.DEFINITIONS

Now let us define the fields for the INSURANCE application.

```
SUBROUTINE INSURANCE.FIELD.DEFINITIONS
$INSERT I_COMMON
$INSERT I_EQUATE
$INSERT I_F.CUSTOMER
$INSERT I_F.CURRENCY
$INSERT BP I_F.POLICY.TYPE

GOSUB INITIALSE
GOSUB DEFINE.FIELDS
RETURN
```

INSURANCE.FIELD.DEFINITIONS

DEFINE.FIELDS:

```
ID.F = "POLICY.NO" ; ID.N = "16" ; ID.T = "A"
Z=0
Z+=1 ; F(Z) = "POLICY.TYPE" ; N(Z) = "35.1." ; T(Z) = "A"
CHECKFILE(Z) = "POLICY.TYPE":FM:POL.DESCRIPTION:FM:"L"
Z+=1 ; F(Z) = "CUSTOMER.NO" ; N(Z) = "10.1" ; T(Z) = "CUS"
CHECKFILE(Z) = "CUSTOMER":FM:EB.CUS.MNEMONIC:"L"
Z+=1 ; F(Z) = "CURRENCY" ; N(Z) = "3.1" ; T(Z) = "CCY"
CHECKFILE(Z) = "CURRENCY":FM:EB.CUR.CCY.NAME:"L"
Z+=1 ; F(Z) = "COVER.AMOUNT" ; N(Z) = "16.1.C" ; T(Z) = "AMT"
Z+=1 ; F(Z) = "PREMIUM.AMOUNT" ; N(Z) = "16.1.C" ; T(Z) = "AMT"
Z+=1 ; F(Z) = "START.DATE" ; N(Z) = "11..C" ; T(Z) = "D"
Z+=1 ; F(Z) = "END.DATE" ; N(Z) = "11.1.C" ; T(Z) = "D"
```

INSURANCE.FIELD.DEFINITIONS

```
Z+=1 ; F(Z) = "PAYMENT.FRQ" ; N(Z) = "16.." ; T(Z) = "FQO"
Z+=1 ; F(Z) = "XX<BENEFICIARY" ; N(Z) = "10..C" ; T(Z) = "A"
Z+=1 ; F(Z) = "XX>ADDRESS" ; N(Z) = "35..C" ; T(Z) = "A"
Z+=1;F(Z)="AGREEMENT.TYPE";N(Z)="35..";T(Z)<2> =
    "OPTION1_OPTION2_OPTION3_OPTION4“
Z+=1 ; F(Z) = "CONDITIONS" ; N(Z) = "35..C" ; T(Z) = "A"
Z+=1 ; F(Z) = "XX.LOCAL.REF" ; N(Z) = "16.." ; T(Z) = "A" ; T(Z)<3> = 'NOINPUT'
V = Z + 9
RETURN

INITIALISE:
   MAT F = "" ; MAT N = "" ; MAT T = ""
   MAT CHECKFILE = "" ; MAT CONCATFILE = ""
   ID.CHECKFILE = "" ; ID.CONCATFILE = ""
* Define often used check file variables
   RETURN
END
```

Compile INSURANCE.FIELD.DEFINITIONS

```
EB.COMPILE DEV.BP INSURANCE.FIELD.DEFINITONS
EB.COMPILE DEV.BP INSURANCE
```

PGM.FILE

Make entries for INSURANCE and POLICY.TYPE application in PGM.FILE

Type H denotes that it is a table capable of holding data and allows addition, modification and deletion of data

Commit Validate Hold Delete Authorise Reverse Veri

PGM.FILE

US0010001 : INSURANCE

Type	H
GB Screen Title	INSURANCE TEMPLATE
Additional Info	.NOH
Product	EB
Curr No	3
Inputter.1	4_TRAINEE03
Date Time.1	02 NOV 05 14:00
Authoriser	4_TRAINEE03
Co Code	US-001-0001 BNK R05BASE
Dept Code	1

Commit Validate Hold Delete Authorise Reverse

PGM.FILE

US0010001 : POLICY.TYPE

Type	H
GB Screen Title	POLICY TYPE APPLICATION
Product	EB
Curr No	1
Inputter.1	2_TRAINEE03
Date Time.1	05 OCT 05 12:44
Authoriser	2_TRAINEE03
Co Code	US-001-0001 BNK R05BASE
Dept Code	1

FILE.CONTROL

Step 8:

Using the FILE.CONTROL application we can set,

- The type of files our application has to store (Live,Unauthorised,History)
- The type(hashed,non – hashed) and modulo of the files
- The other attributes of a file (INT,CUS,FIN) etc

Creation of FILE.CONTROL record

To create a FILE.CONTROL record, we can make a copy of an existing FILE.CONTROL record of any existing application and make the changes.

FILE.CONTROL – INSURANCE

Use the 'Merge' facility provided by jBASE and create the FILE.CONTROL record. Create the FILE.CONTROL record for the POLICY.TYPE application

Jsh..>JED F.FILE.CONTROL INSURANCE

```
001 CUSTOMER
002 ST EB
003 $HIS]$NAU
004 2
005 31
006 CUS
007 CUSTOMER
008 N
009 Y
010 BST
011
012 CUSTOMER
013
014 2
015 1_DIM
016 0205171442
017
018 IE0010001
019 1
```

Change this to the application Name (INSURANCE). It is a description of the FILE.CONTROL record.

Product to which the application belongs to
The types of files apart from the live files that have to be stored

CUS – Shared file

FILE.CONTROL – POLICY.TYPE

Use the 'Merge' facility provided by jBASE and create the FILE.CONTROL record. Create the FILE.CONTROL record for the POLICY.TYPE application

```
Jsh..>JED F.FILE.CONTROL POLICY.TYPE

001 CUSTOMER
002 ST EB
003 $HIS]$NAU
004 2
005 31
006 CUS
007 CUSTOMER
008 N
009 Y
010 BST
011
012 CUSTOMER
013
014 2
015 1_DIM
016 0205171442
017
018 IE0010001
019 1
```

Change this to the application Name (POLICY.TYPE). It is a description of the FILE.CONTROL record.

Product to which the application belongs to
The types of files apart from the live files that have to be stored

CUS – Shared file

Create the insert file using FILE.LAYOUT program

```
Jsh…>FILE.LAYOUT
Use Select List or Input Individually either :

List of FILE names from which to build insert modules via their dictionarys
List of PROGRAM names from which to build insert modules through calling.

Enter Program/File(s) :INSURANCE
Enter Program/File(s) :

Build Insert from File (D)ictionarys or (P)rograms or (Q)uit
<CR = (P)rograms>  :P
Building Insert From Program(s)
```

```
Program is INSURANCE

Enter output Name - <CR> = I_F.Entryname  :I_F.INSURANCE

Enter PREFIX or <CR> = NONE :INS

Enter SUFFIX or <CR> = NONE :

Processed 22 matrix entries for INSURANCE program
```

I_F.INSURANCE

The insert file created by the FILE.LAYOUT program for the INSURANCE application

```
* File Layout for INSURANCE Created 10 SEP 04 at 09:37AM by g14007
*      PREFIX[INS.]      SUFFIX[]
 EQU INS.POLICY.TYPE TO 1,      INS.CUSTOMER.NO TO 2,
        INS.CURRENCY TO 3,     INS.COVER.AMOUNT TO 4,
  INS.PREMIUM.AMOUNT TO 5,       INS.START.DATE TO 6,
        INS.END.DATE TO 7,      INS.PAYMENT.FRQ TO 8,
     INS.BENEFICIARY TO 9,         INS.ADDRESS TO 10,
 INS.AGREEMENT.TYPE TO 11,      INS.CONDITIONS TO 12,
      INS.LOCAL.REF TO 13,   INS.RECORD.STATUS TO 14,
        INS.CURR.NO TO 15,        INS.INPUTTER TO 16,
      INS.DATE.TIME TO 17,      INS.AUTHORISER TO 18,
        INS.CO.CODE TO 19,       INS.DEPT.CODE TO 20,
   INS.AUDITOR.CODE TO 21, INS.AUDIT.DATE.TIME TO 22
```

CREATE.FILES

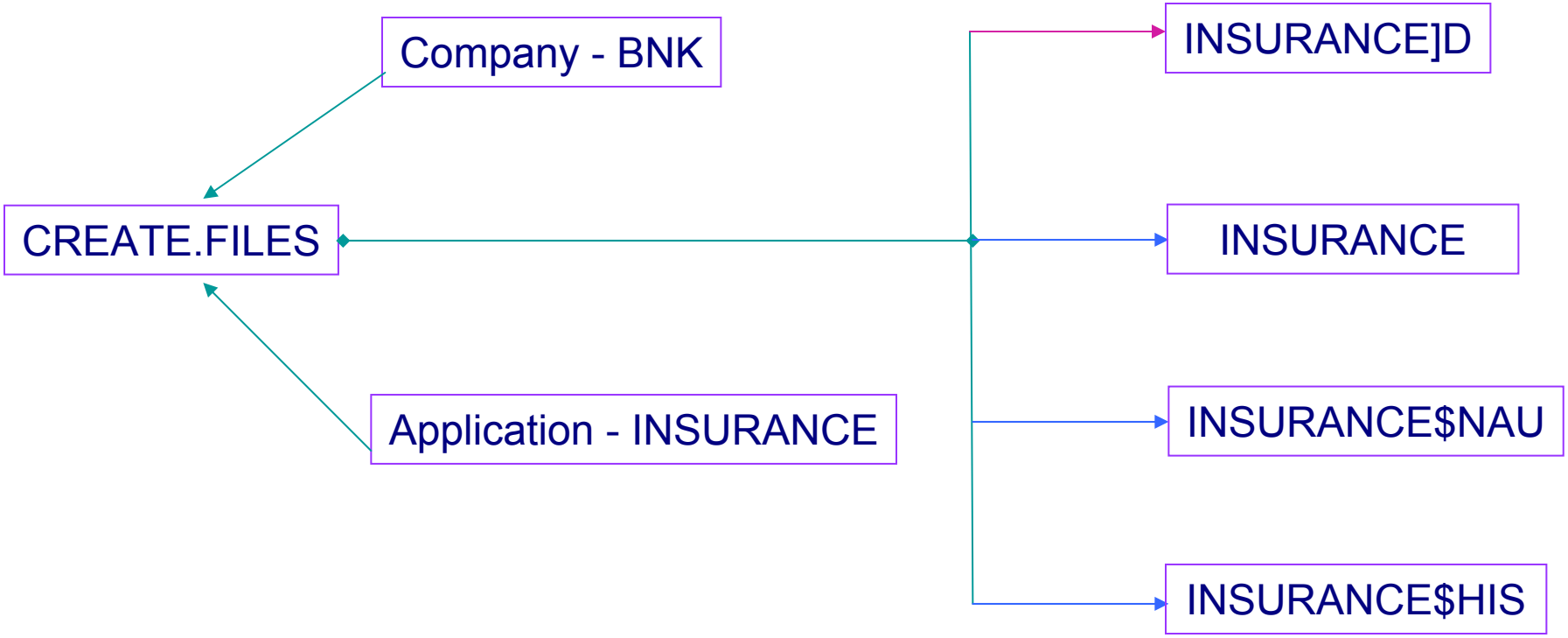
Step 10:

The application CREATE.FILES is the one that looks into the FILE.CONTROL record for the respective application and creates the necessary files.

We need to run this process in classic

CREATE.FILES

This process will create 3 Files



CREATE.FILES

Type CREATE.FILES from within T24

T24 will prompt for the following

```
COMPANY CODE                                          : BNK
FILE NAME                                             : <Application name>*
CREATE FILES FOR Globus Demo Account using FILE NAME(S) INPUT: Y
```

Execute CREATE.FILES for the POLICY.TYPE application. Output below.

```
Creating dictionary for F.POLICY.TYPE
[ 417 ] File ..\mbdemo.dict\F.POLICY.001]D created , type = J4
Creating file FBNK.POLICY.TYPE

 No Records selected

[ 417 ] File ..\mbdemo.data\eb\FBNK.POLI001 created , type = J4
Creating file FBNK.POLICY.TYPE$HIS

 2 Records selected

[ 417 ] File ..\mbdemo.data\eb\FBNK.POLI002 created , type = J4
Creating file FBNK.POLICY.TYPE$NAU

[ 417 ] File ..\mbdemo.data\eb\FBNK.POLI000 created , type = J4
Creating file FBNK.POLICY.TYPE
```

CREATE.FILES

Execute CREATE.FILES for the INSURANCE application.

Output below.

```
Creating dictionary for F.INSURANCE
[ 417 ] File ..\mbdemo.dict\F.INSURANCE]D created , type = J4
Creating file FBNK.INSURANCE

 No Records selected

[ 417 ] File ..\mbdemo.data\eb\FBNK.INSU001 created , type = J4
Creating file FBNK.INSURANCE$HIS

 2 Records selected

[ 417 ] File ..\mbdemo.data\eb\FBNK.INSU002 created , type = J4
Creating file FBNK.INSURANCE$NAU

[ 417 ] File ..\mbdemo.data\eb\FBNK.INSU000 created , type = J4
Creating file FBNK.INSURANCE
```

Rebuild STANDARD.SELECTION

- Files are ready to store data
- Update the field ‘REBUILD.SYS.FIELDS’ to a ‘Y’

Auto id generation

The ID of the INSURANCE application has to be in the following format

Example : IN00000000000001

This number needs to be automatically generated.

In order to automatically generate ids set the applications

- AUTO.ID.START
- COMPANY.

Ready to use

Start inputting records in POLICY.TYPE and INSURANCE applications

Workshops

Wokshop1