Oracle FLEXCUBE Direct Banking

mLEAP Framework(Leap Screen Design)
Developer Guide for Android and iOS
Release 12.0.3.0.0

Part No. E52543-01

April 2014

ORACLE®

mLEAP Framework (Leap Screen Design) Developer Guide for Android and iOS

April 2014

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India
Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax:+91 22 6718 3001
www.oracle.com/financialservices/

Preface

Intended Audience

Any interested party working on the delivery of Oracle FLEXCUBE Direct Banking may read this document. The following profile of users would find this document useful:

- Application Architects
- End to End Designers
- Business Service Detailed Designers and Developers
- Implementation Partners

Specifically, however, this document is targeted at Iimplementation Partners, Customization Development Teams or Vendors providing customization, configuration and implementation services around the Oracle FLEXCUBE Direct Banking product.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to OFSS Support

https://support.us.oracle.com

Contents

Introduction*:*

A general Leap screen is designed in three steps (that incorporates defining three tables). These are:

1) SCREENTEMPLATEMASTER →Defining the number of tables the screen contains.
2) SCREENTABLEDEFINITION →Defining the type and position of tables as per screen.
3) SCREENTEMPLATE →Defining the components in the screen and their behavior.

Defining the Number of Tables the Screen Contains:: SCREENTEMPLATEMASTER Entry.

This is meant for defining the primary details of the screen, if any. Entries other than tablescount and totalstepcount are auto-filled as per the details of the new REQUEST created.
Usually while using the Leap Tool, one even need not worry defining the number of tables explicitly. As we draw a table on the canvas, it is automatically added to the screen (tablescount). Though, one can explicitly add tables too.

The Entries include:

Column Name	Column Type	Description
IDTXN	VARCHAR2(3)	The IdTxn for which the screen is being created
IDREQUEST	VARCHAR2(8)	The IdRequest for which the screen is being created
IDCHANNEL	VARCHAR2(2)	The IdChannel for which the screen is being created
IDENTITY	VARCHAR2(5)	The IdEntity for which the screen is being created
USERTYPE	VARCHAR2(3)	The UserType for which the screen is being created (AndTabs/AndPhone/iPad/iPhone)
IDDESCRIPTION	VARCHAR2(200)	This Column contains the Page Header label to be displayed. Developer can enter XSL template keywords, which will get translated at run time, e.g. 'K_INTERNAL_TRANSFER'

TABLESCOUNT	NUMBER	Number of HTML Tables to be created on the screen
TOTALSTEPCOUNT	NUMBER	Gives the total stages in Baretrail

A) How To Define TablesCount:

First Bottleneck, while designing the screen is to judge the number of tables that the screen needs (or to divide the screen in tables). The Basic idea is to count the tables based on the design of the screen. For this, let's take a screen as example:

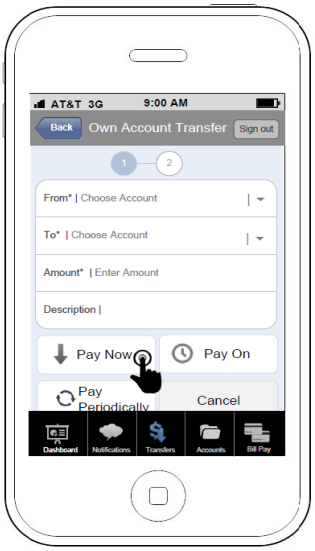
For Tablets:

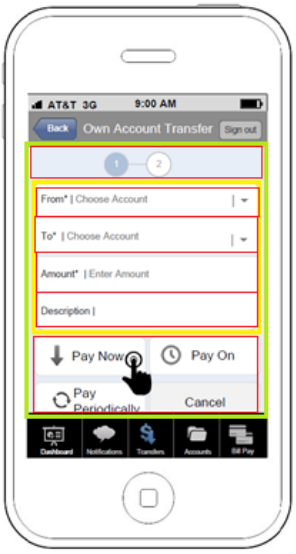
The tables of the screen are marked in red. There is certain difference that needs consideration as in:

1) In case of Android Tablets, the tables are defined as visible on the screen. As in above example the number of tables for the screen is '7'.

2) In case of iPad, the tables remain the same with an addition of two tables at the background over which all the tables are painted. (BackgroundWHA and xmlBG table: height and width as that of screen, used for painting the background of screen). As in above example, the number of tables is 7; the iPad screen will have 7+2 = 9 tables.

For Phones:





Count of tables:
Master table --> 1
Container tables (type W) --> 1
Individual Tables --> 6

Count =8

Individual table that contains one or more component

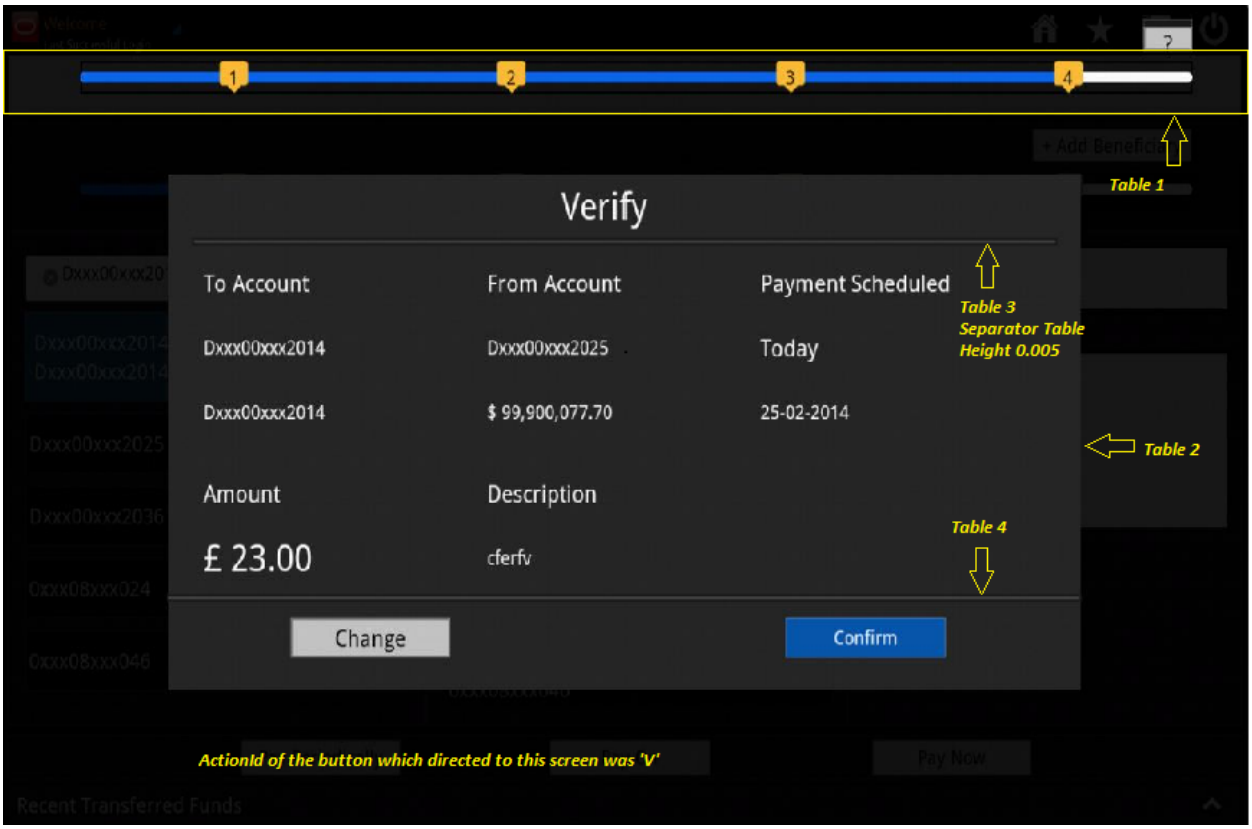
Table of type W which means a table containing one or more tables

Master Table (One that is Mandatory in every screen as the first table)

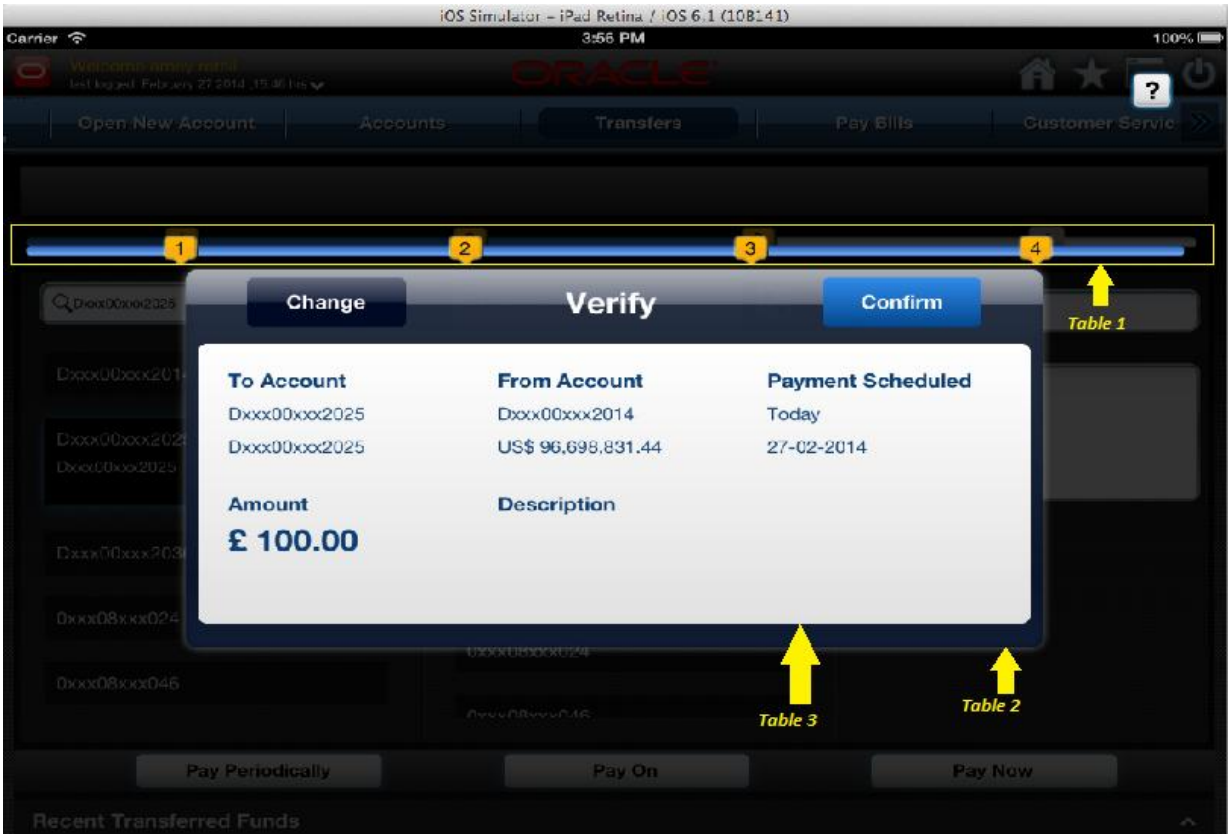
In case of both Android phones and iphones this structure remains the same. Master Table is also a container table, i.e. display type is 'W'.

This is the case of prepare screen. While in **Verify/Confirm screen** we define the tables as:

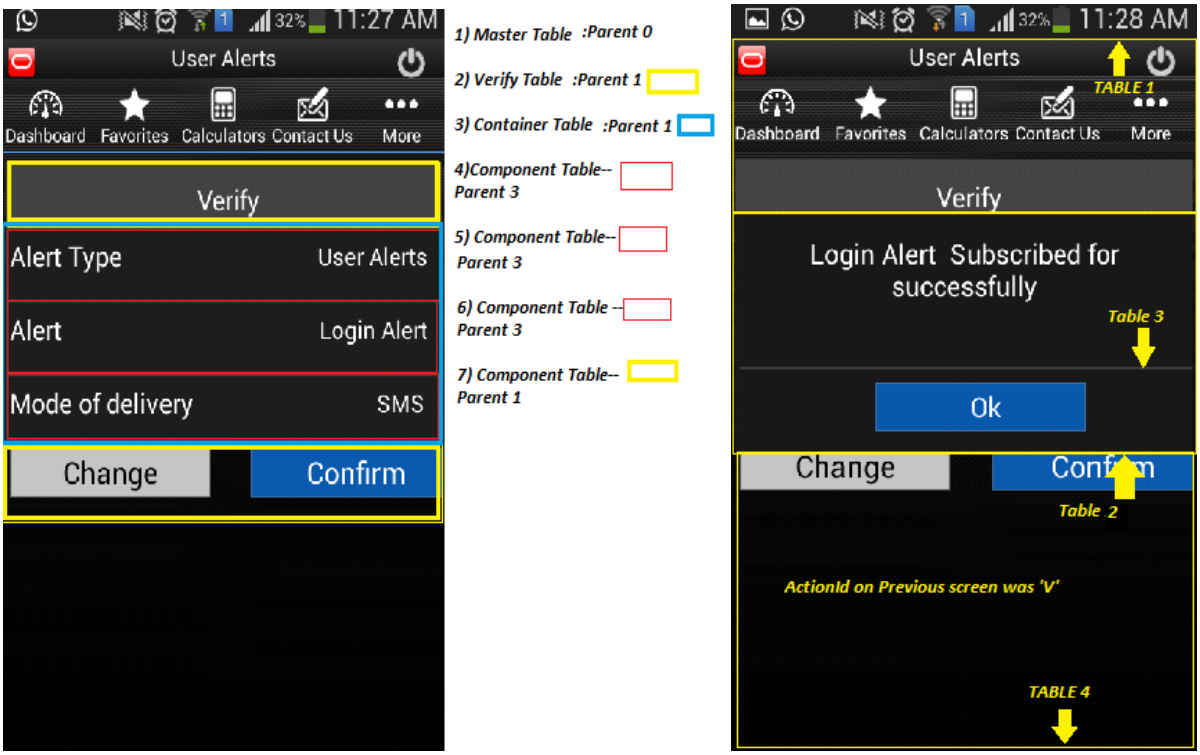
Android Tablets:



iPad:


iOS Simulator – iPad Retina / iOS 6.1 (10B141)
Carrier
3:56 PM
100%
ORACLE
Open New Account
Accounts
Transfers
Pay Bills
Customer Servic
1
2
3
4
Change
Verify
Confirm
Table 1
To Account
Dxxx00xxx2025
Dxxx00xxx2025
From Account
Dxxx00xxx2014
US$ 96,698,831.44
Payment Scheduled
Today
27-02-2014
Amount
£ 100.00
Description
Table 3
Table 2
Pay Periodically
Pay On
Pay Now
Recent Transferred Funds

Phones:



B) How To Define TotalStepCount:

Total step count defines the number of total steps that is shown when a component “BareTrail” is added. As in the above example:

The total number of steps the component shows is “2”.

Now that the number of tables of the screen is defined, we need to define the tables separately.

Defining the type and position of tables as per screen:: SCREENTABLEDEFINITION entry

This is meant for defining the properties of the HTML tables that would be created on the screen.

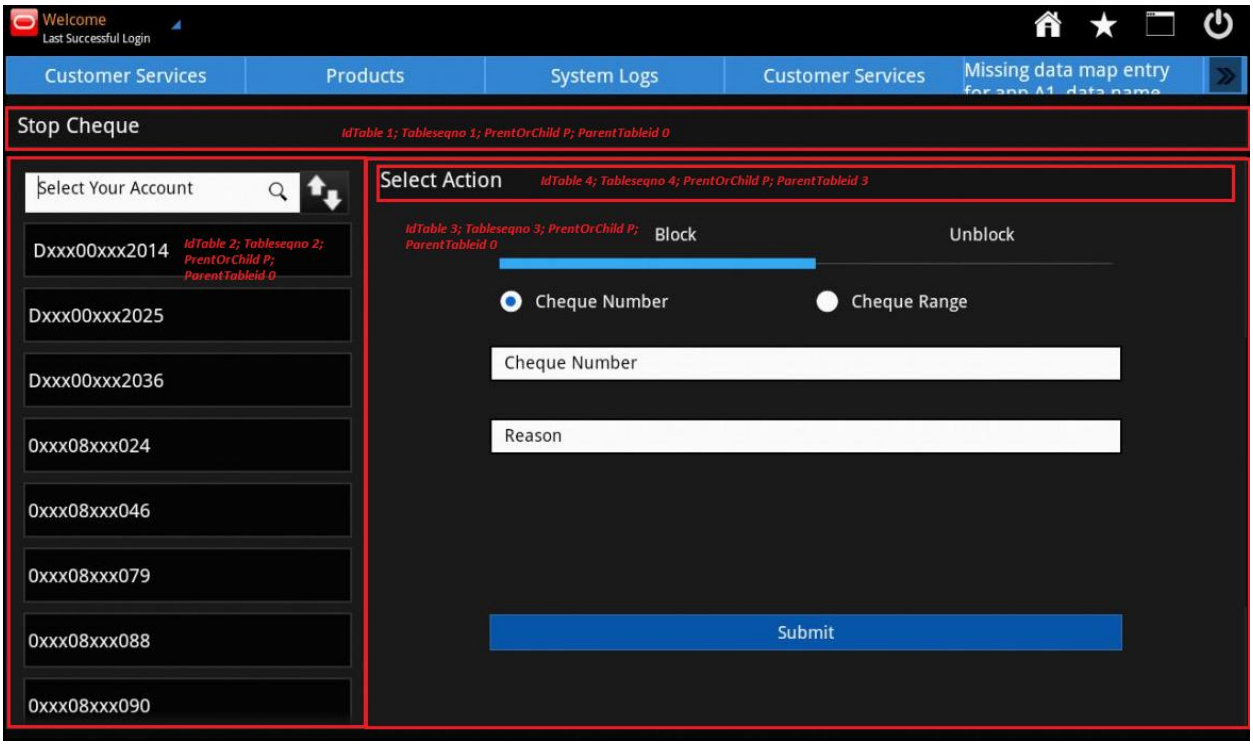
Column Name	Column Type	Description
IDTXN	VARCHAR2(3)	The IdTxn for which the screen is being created
IDREQUEST	VARCHAR2(8)	The IdRequest for which the screen is being created
IDCHANNEL	VARCHAR2(2)	The IdChannel for which the screen is being created
IDENTITY	VARCHAR2(5)	The IdEntity for which the screen is being created
USERTYPE	VARCHAR2(3)	The UserType for which the screen is being created
IDTABLE	NUMBER	The ID of The Table. Unique identification for each table. Continuous sequence maintained though sequence no. can differ.
TABLESEQNO	NUMBER	This column allows a developer to rearrange the table as per the sequence required. A developer needs to give a continuous sequence number for the tables, one should not give any number in between. Tables are painted on screen using this sequence
PARENTORCHILD	VARCHAR2(1)	Defines the Type of table. 'P' is the type if the table is Parent or Child. Though in case of child we define the parenttableid as well.

<table>
<tr><td>PARENTTABLEID</td><td>NUMBER</td><td>For Child table, we have to define the parent tableid which will contain this child table. The Positioning of the child table is done from parent table (not from the entire screen)</td></tr>
<tr><td>DEFAULTDISPLAY</td><td>CHAR(1)</td><td>This column describes the type of display and properties of a particular table.
<table><tr><td>S</td><td>This is type is used for normal tables that are displayed on the screen on load</td></tr><tr><td>H</td><td>This type is used for tables that initially are hidden and are displayed on certain event. This event is mentioned in behavior on screen.</td></tr><tr><td>M</td><td>This type is used to display tables of modal type.</td></tr><tr><td>P</td><td>Pop Up tables are displayed using this type of display</td></tr><tr><td>W</td><td>This is used in Phones only. It defines a table that contains one or more tables as its child.</td></tr></table></td></tr>
</table>

<table>
<tr><td>USERAGENT</td><td>VARCHAR2(1)</td><td>This column allows a user to set different definitions for different clients. (AndTabs, AndPhone, iPad, iPhone)</td></tr>
<tr><td>CONDITION</td><td>VARCHAR2(1000)</td><td>This column contains the display condition of a table. There might be cases in which we have to show a table only if certain condition from response or previous screen is true. These conditions are defined in this column.</td></tr>
<tr><td>RELPOSX</td><td>NUMBER(3,2)</td><td>This column contains the Left Margin of the Table. (From the screen)</td></tr>
<tr><td>RELPOSY</td><td>NUMBER(3,2)</td><td>This column contains the top Margin of the Table. (From the screen)</td></tr>
<tr><td>RELWIDTH</td><td>NUMBER(3,2)</td><td>This column contains the Width of the Table. (On screen in %age composition) (100%) for different useragents:
<table><tr><td>USERAGENT</td><td>TOTAL WIDTH</td></tr><tr><td>AndTabs</td><td>1.000</td></tr><tr><td>AndPhone</td><td>1.000</td></tr><tr><td>iPad</td><td>1.000</td></tr><tr><td>iPhone</td><td>1.000</td></tr></table></td></tr>
<tr><td>RELHEIGHT</td><td>NUMBER(3,2)</td><td>This column contains the Height of the table. . (On screen in %age composition) (100%) for different useragents:</td></tr>
</table>

		USERAGENT	TOTAL HEIGHT
		AndTabs	0.861
		AndPhone	0.800
		iPad	0.861
		iPhone	0.800

A) Evaluating IDTABLE, TABLESEQNO, PARENTORCHILD, PARENTTABLEID:



As we can see in above example, idtable of the tables are in sequence of 1 to 4 and same is with the tableseqno. Difference is that idtable is used to identify table while tableseqno is used to determine the sequence in which the tables are painted on screen, i.e. sequence number of table 4 must always be after table 3 so that it is painted on table 3. Type is to be 'P' for all tables in parentorchild column, while, Parenttableid defines parent for the table.

ParentTableid=0, means the parent is screen on whole, thus x&y is determined on basis of whole screen;
ParentTableid=3, means the parent is tableid 3, thus x&y of the table is determined wrt tableid 3 coords

B) Evaluating DEFAULTDISPLAY:

S:: In the above example all the tables are shown on screen by default. So, DefaultDisplay of all these table is 'S', i.e. show

H::

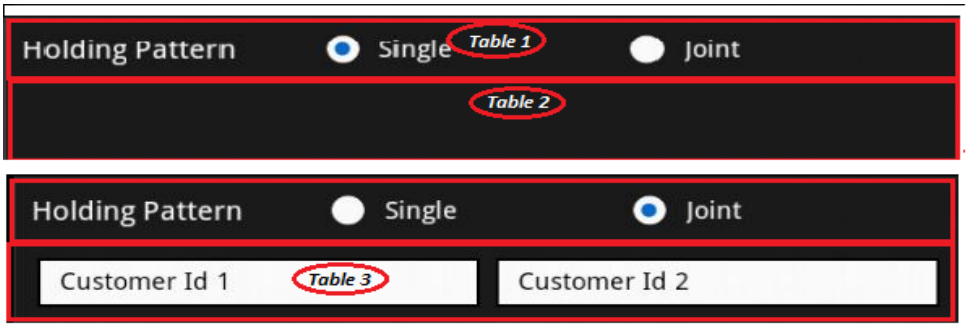
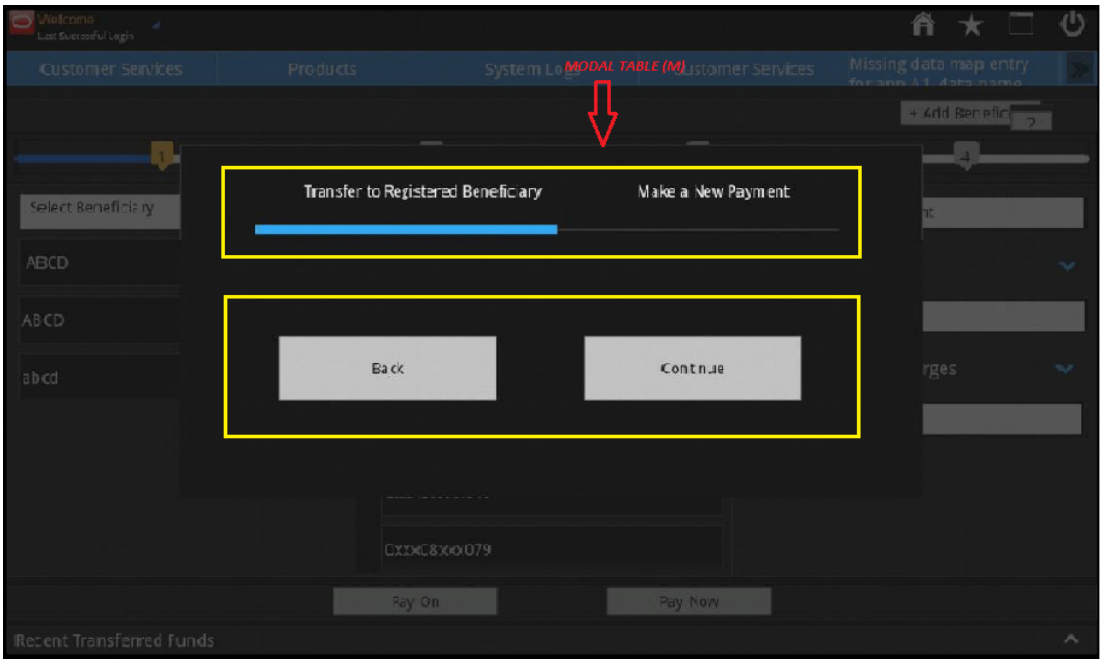


Table 2 is displayed on load of screen while Table 3 is displayed only if joint is selected.

Thus, DefaultDisplay of
Table 2 = S
Table 3 = H

So, Table 3 remains hidden. On click of Joint we put a behavior that makes Table 2 as 'H' and Table 3 as 'S' and vice-versa on Single.

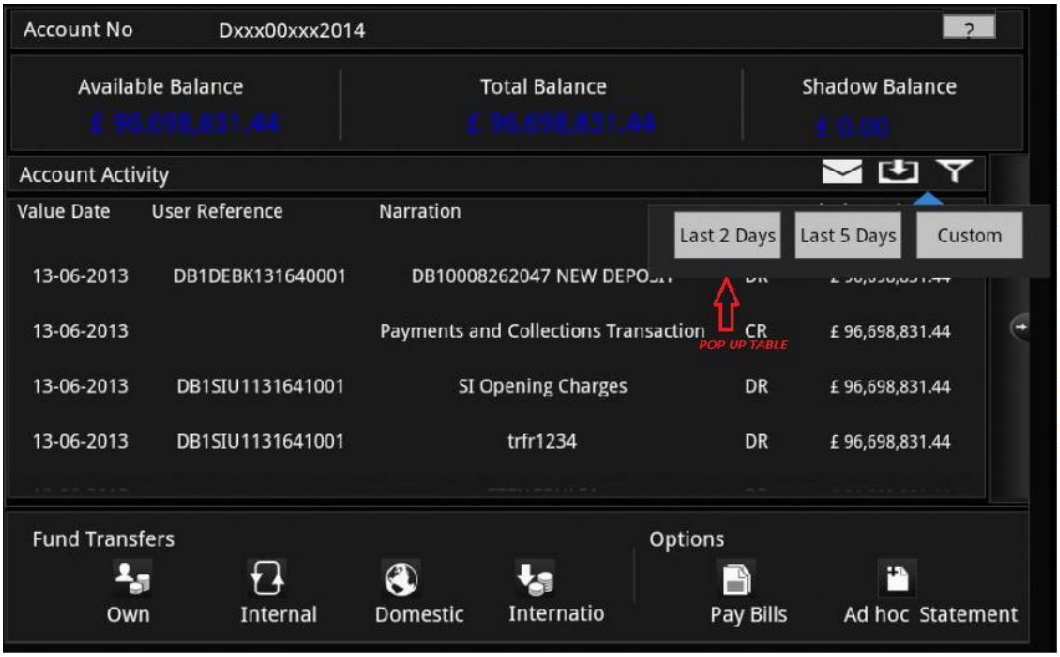
M::



The Table displayed on load of screen (Like the one in example) is called as a modal table.

A Modal table can have one or more Table as its child. As in this case shown in yellow line

P::



As shown in the example the PopUp table is a table that pops up from a button called popup button.

DefaultDisplay of table is given as 'P'. Components are given to the table or one might also add child tables containing components to it.

In case Of mobiles, AndPhones have both this type of (small) popups and full screen popups.
While in case of iPhones Only Full screen Popups are allowed.

W (PHONES ONLY):: Used in Phones Only, this type of table is used to contain one or more tables as its child. This table can contain only tables and is not used for painting any component. Since in case of phones we do not define the X and y coordinate, all the tables are painted according to their sequence number inside the container table (basically master but can use other container table inside master too).

One can see that the screen has 2 Container tables.
1) Master Table : every table on screen is atleast a child to this table (if not any other table) Default Display = W
2) Container Table in ▭ *That contains 4 tables as its child. This table's parent is master table, though, child of this parent has its id as parentid*

▭ *Individual table that contains one or more component*

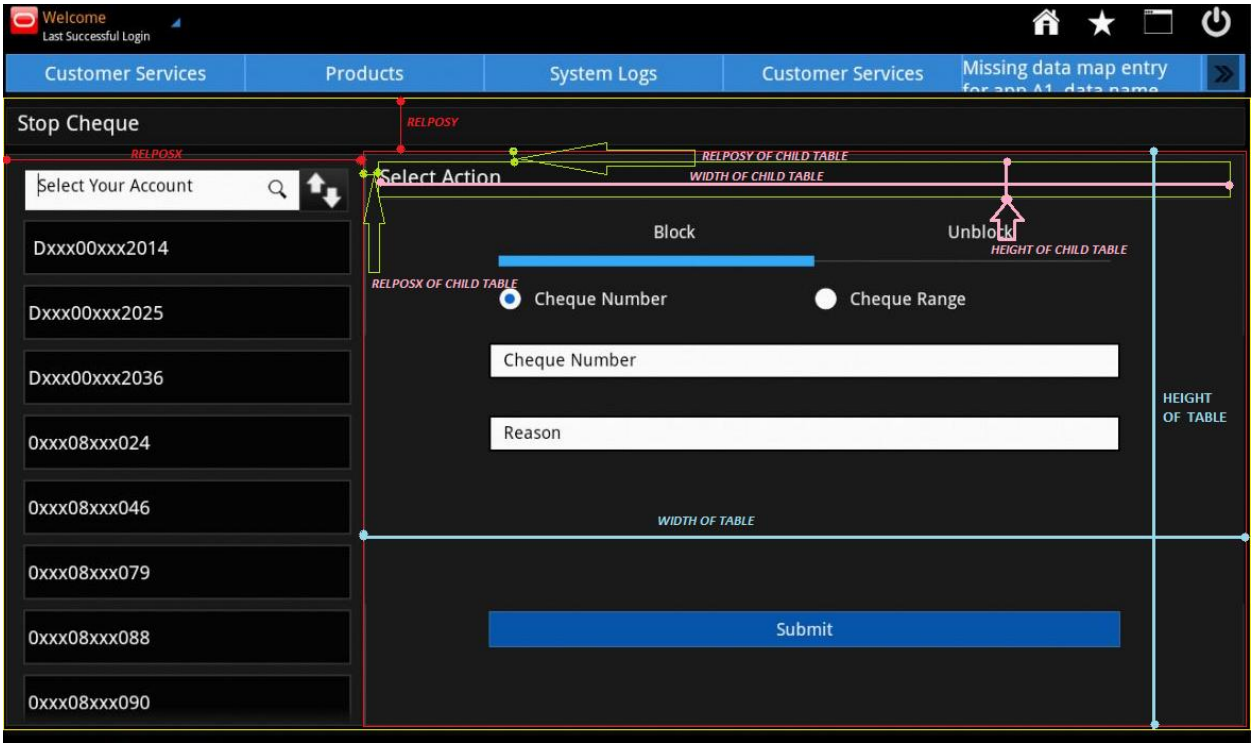
▭ *Table of type W which means a table containing one or more tables*

▭ *Master Table (One that is Mandatory in every screen as the first table)*

C) Evaluating WIDTH, HEIGHT, RELPOSX, RELPOSY:

RELPOSX and RELPOSY of a table is calculated as per its parent. And the width and height is always

relative to the screen. So, RELPOSX and RELPOSY of a table that has no parent id are calculated on basis of screen (as the whole screen becomes its parent). While, RELPOSX and RELPOSY of a table that has parent id is calculated relative to parent table. In case of iPad, we include two tables of height and width of the screen to paint the background.



In case of Phones, we do not define the RELPOSX & RELPOSY of the table. Tables are painted in accordance to their sequence number and are always inside at least one container table. This is the reason why the first table (table with sequence number 1) is always the master table. Using the concept of child we define other tables.

Table seqno 1 Mastertable — Parent:0 Tableid:1

Table seqno 2 Component table — Parent:1 Tableid:2

Table seqno 3 Container table — Parent:1 Tableid:3

Table seqno 4 Component table — Parent:3 Tableid:4

& so on...

Defining the components in the screen and their behavior:: SCREENTEMPLATE entry.

This table is the main table which contains all the details which would be required to create the HTML components on the screen. The details of the table are as follows:

Column Name	Column Type	Description
IDTXN	VARCHAR2(3)	The IdTxn for which the screen is being created
IDREQUEST	VARCHAR2(8)	The IdRequest for which the screen is being created
IDCHANNEL	VARCHAR2(2)	The IdChannel for which the screen is being created
IDENTITY	VARCHAR2(5)	The IdEntity for which the screen is being created
USERTYPE	VARCHAR2(3)	The UserType for which the screen is being created
LABEL	VARCHAR2(200)	Label of the field to be displayed.
NAME	VARCHAR2(50)	Name of the HTML field.
ID	VARCHAR2(50)	ID of the HTML field.
TYPE	VARCHAR2(2)	HTML Data Type of the field defined in MSTHTMLDATATYPES.
NODEVALUE	VARCHAR2(1000)	The Xpath from where the value is to be populated in the field.
IDROW	NUMBER	The Row Id of the table in which the field is to be displayed.
COLUMNID	NUMBER	The Column Id of the Row in which the field is to be displayed.
TABLENO	NUMBER	The HTML Table id. The properties of this HTML Table have to be defined in SCREENTABLEDEFINITION.
FUNCTIONARGS	VARCHAR2(100)	Arguments of the Function.
MANDATORYICON	VARCHAR2(100)	The mandatory Icon, for e.g. '**'
ISMANDATORY	VARCHAR2(1)	The field is Mandatory or Not.

DEFAULTVALUE	VARCHAR2(1000)	This field is used for different datatypes with different purposes. For Radio Button it used to decide which radio button has to be kept default selected on page load, for check box it is used to decide whether to keep the check box checked on page load. Explained in details with the html datatypes description later in this document.
ROWARRAYNODE	VARCHAR2(1000)	This column contains the Xpath on which one can iterate to create multiple rows dynamically.
CONDITION	VARCHAR2(1000)	This column contains the display condition of a field.
RELPOSX	NUMBER(3,2)	This column contains the Left Margin of the Table.
RELPOSY	NUMBER(3,2)	This column contains the top Margin of the Table.
RELWIDTH	NUMBER(3,2)	This column contains the Width of the Table.
RELHEIGHT	NUMBER(3,2)	This column contains the Height of the table.
TOKEN1	VARCHAR2(50)	This field is used for different datatypes with different purposes.
TOKEN2	VARCHAR2(50)	This field is used for different datatypes with different purposes.
TOKEN3	VARCHAR2(50)	This field is used for different datatypes with different purposes.
TOKEN4	VARCHAR2(50)	This field is used for different datatypes with different purposes.
TOKEN5	VARCHAR2(50)	This field is used for different datatypes with different purposes.
STEPNUMBER	NUMBER	This field gives the step number for each component, used for baretrail/swimlanes. If suppose the number is 2 then the field will stay disabled till the Baretail reaches 2, i.e. all conditions for 1 is fulfilled.

A) Evaluating NAME, ID, NODEVALUE, DATACLASS:

When defining ***NAME & ID*** to a component we need to understand the following:

1) Name and id for a component can be same or different. Eg: Name- fldmessbull, ID-fldmessbull OR Name-fldmessbull, ID-fldmessbull1.

2) In a screen ID of component is necessarily unique while Name may or may not be unique.

3) We define any Name & ID to a component, keeping in mind the needed (mandatory) values needed as defined in the xsl of next screen. That means if the next screen needs some values defined as fldacctno - for account number, fldcustid – for customer id then. The screen will have to give those values in component with NAME: fldacctno & fldcustid respectively.

4) The ID of a component is used as {requestid}{componentid}. For Eg: For RROAT61 a component ID fldname is actually identified as **RROAT61fldname**. Client side code accepts ID based on this form of id, i.e. requestid & ID. Though while entering through LEAP we mention just fldname, the requestid is attached while creating f-language. So, while entering values for behavior, we must specify the ID of the component in form of **{requestid}{componentid}** rather than just ID.

Defining ***NODEVALUE***:

1) Though defining nodevalue is different for different component and is defined in detail further. One thing that is important is, the nodevalue of a component is given via xpath. Xpath is obtained from response (a path to the value to be displayed)

Defining ***DATACLASS:***
Dataclass of a component can be given using the CSS Doc. In case of any specific change or form is needed, it is defined in the component section.

B) Evaluating IDROW, COLUMNID, TABLENO:

TABLENO for a component is given as the tableid of table in which the component is to be displayed. COLUMNID for the component is usually ‘1’ (recommended). ROWID for a component:

1) ROWID is a number usually describing the number of component.

2) ROWID mandatorily has to be in a sequence. Eg: 1,2,3,4,5 is right order similar is 1,5,2,4,3 because it is finally in sequence from 1 to 5. While, 1, 2,6,5,7 is not correct and would result in not displaying the components on screen.

C) Evaluating RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT & Defining CONDITION:

Condition is given to a component so as to define its painting on screen only on a specific condition. For

Eg: If the Value of radio on previous screen was Month then display Textbox, otherwise display calendar. Then, we define both the components on current screen, i.e., a Textbox and a calendar but we give condition to both as needed above. (//faml/request/radio='Month' for textbox and //faml/request/radio!='Month' for calendar; supposing radiobutton name='radio' and value passed on selecting Month is 'Month').

RELPOSX & RELPOSY of component are calculated according to their parent, i.e. the table to which the components belong. RELWIDTH & RELHEIGHT are calculated in respect to the screen only.

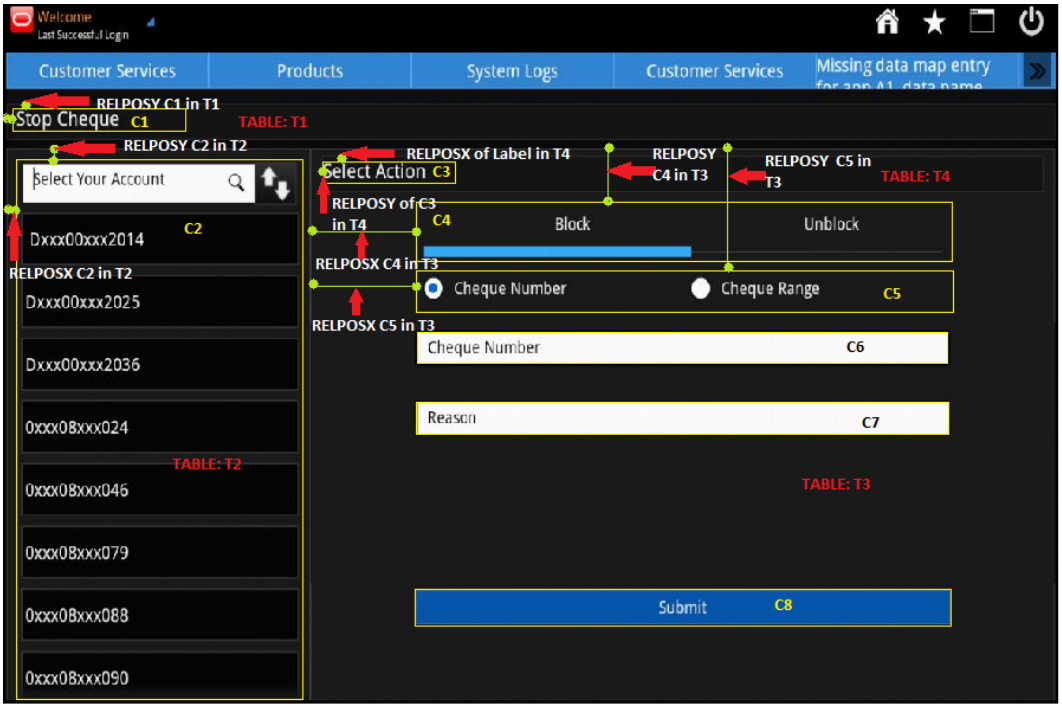


TABLE T1, T2, T3, T4: define table no 1, 2, 3,4.
C1, C2, C3, C4 and so on describe the component 1, 2, 3, 4 and so on.

we can see that RELPOSX and RELPOSY of component is given wrt the table in which the component is held.

D) Defining TYPE (Logical Grouping):

There are many components that are used to present data on screen. We can broadly classify this data into following types:

Components that accept input from user:

TEXTBOX **(T)**	This is a simple input type that can accept data from user. Defining the type of data to be accepted is possible (mentioned later in component description).	
TEXT AREA **(TA)**	Textarea is similar to textbox.	

	Accepts input from user. Has multi-line property.	
DATE FIELD TEXTBOX **(MD)**	A textbox with a calendar (date-month-year). Used to accept date from the user. Date data is only accepted using this component.	
DATE FIELD **(TD)**	This is a scrollable date calendar. Used to accept date.	
LOOK UP TEXTBOX **(TL)**	A textbox with an icon on the side. On click of this icon one can open a new requestid to provide selection option to the user. One can directly enter value in textbox too.	
PASSWORD **(P)**	A textbox that accepts any data from user. But the data is in form of bullets for the purpose of privacy and security.	
DATE FIELD CALENDAR TEXTBOX **(ND)**	A textbox with a calendar (based on number of days). Used to accept date.	
DATE FIELD CALENDAR **(CD)**	A date calendar (based on number of days). Accepts date.	

Components that follow property of button (On click action):

BUTTON (B)	A simple button used to submit data or get a new requestid.	
IMAGEBUTTON (IB)	Similar to button, but has an image instead of label.	
POP-UP BUTTON (PB)	A button with image or label. Used to open a pop-up window on the screen.	

WIDGET BUTTON (WB)	A button that can have image and label both at a time. Used to request based on different values. (Explained later in components)	
BUTTON GROUP (BG)	One or more than one button created based on a particular screen requirement (generated dynamically).	
ACCORDION (AC)	A type similar to button. Usually defined on the corner of screen (bottom, left or right). Drags data (which is defined in its table outside the screen) inside for display on click.	
SLIDER BUTTON (SL)	A button that displays data in form of slider. One can select data on basis of length of slider clicked.	

Components that are used to display values (As in labels):

VERIFY TEXT (V)	A simple label that displays data.	
FORMATTED AMOUNT (FA)	A label that displays data in form of amount (decimal places corrected preceded by currency symbol)	
FORMATTED DATE (FD)	A label that displays data in form of date.	
FORMATTED UNIT (FU)	A label used for Formatting a unit(amount, number)	
TIMEZONE FORMATTED DATE (TZ)	A label used for displaying Formatted Date with time zone.	

LABEL HEADING (L1)	A label used to display sub headings.	
GROUP LABEL (GL)	Used for grouping labels together as a grouped header label.	
VALUE LABEL (VL)	A label that adds certain predefined label to the value obtained from response.	

Components created in form of list to present data:

STATIC DROPDOWN (D1)	A component with or without a label used to show a list of static values.	
DROPDOWN (D)	A component similar to static dropdown just that the values fetched in it are dynamic (based on current response).	
LOOK-UP DROPDOWN (LD)	A component that has values similar to dropdown in addition to a look-up image that requests certain data for selection.	
SEARCH DROPDOWN (SD)	A component that contains dynamic value represented in form of searchable list.	
LIST (L)	A component that shows data in a general list form.	
DROPDOWN UNSORTED (DU)	A component that is used to display data in dropdown in unsorted form.	
ACCOUNTS DROPDOWN (SA)	A component used to display Accounts with option group.	

Components created to present data in forms other than list to present data:

STATIC RADIO (SR)	A component used to support radio based controls.	
DYNAMIC RADIO (DR)	Similar to static radio. But the number of radio options depends on response (dynamic).	
STATIC CHECKBOX (SC)	A component used to display options in checkbox type (multi select).	
DYNAMIC CHECKBOX (DCB)	Similar to static checkbox. But the checkbox options are generated as per response (dynamic).	
SEGMENTED BUTTON (SB)	A component used to draw buttons sequentially.	
DYNAMIC SEGMENTED BUTTON (DSB)	Similar to segmented button. But the number of buttons and their labels are generated dynamically.	

Components that are used to display some form of data:

GRAPH AREA (GR)	A component used to draw data received on a graph.	
SERVER IMAGE (SI)	A component used to draw image obtained from server.	
ATTACH FILE LIST (A)	A component used to draw the attachments obtaining the number of attachments in the response. Eg: in case of mail the number of docs attached.	
PICTURE VIEW (PV)	A component used to paint picture.	

WIDGET VIEW (WV)	A component that can be used to paint a certain response within its area.	
WEBVIEW (W)	A component used to display data in a web view.	

Other Components:

HYPERLINK (HL)	A Link to some other request/URL.	
MAIL LIST (ML)	A type of list used only in case of displaying mails.	
VALUE SELECTOR (VS)		
LOCATION (LOC)		
CAPTCHA (CP)	Used to draw a captcha with image containing words, a button for sound and a button to refresh the captcha.	
CONTACT LIST (CL)		
PASSWORD STRENGTH (PST)		
QR CODE (QR)	Used to draw and accept QR Codes. Also used for NFC Transfer.	
BARETAIL (BT)	Used to define the current stage and maintain the enable and disable of components based on the stages.	
PICTURE UPLOAD (PU)	A component used to provide upload of documents.	
DYNAMIC ADDITION (DA)	Used to display textboxes and add them on request	

	dynamically. Or generate the number dynamically for display.	

Terms to Explain:

1) ***ActionID:*** ActionId is a value used along with the requestid value. Actionid is used to define the type of request that is being hit for the requestid given. For Eg: if a new page is being loaded we give Actionid as 'P'

ACTIONID VALUES	***Meaning***
P	Fire a new prepare screen.
V	Fire a new verify/confirm screen

2) ***TargetID:*** Targetid is a value that is also used along with the requestid value. Targetid is specified in case we want to fire a new request in some table (part) of the current opened screen. In this case we give the targetid as {currentrequestid}{tableid}. For Eg: If we want o open RRTDF62 in some part of RRTDF61, we define the Targetid as RRTDF613; where 3 is the tableid in which RRTDF62 is to be populated.

3) ***RequestID:*** The ID of the request that is to be hit on a particular event is given as requestid.

4) ***ValueIndex:*** Value index is given in case we want some value to stay selected by default in dynamic components. For Eg: In a dropdown we want some value selected, then the value that is being passed (v attribute) for that value in dropdown must be equal to the value we get in valueindex.

5) ***Index:*** Index is somewhat similar to the ValueIndex. The difference is that here we select the value on the basis of its index rather than value. Index contains the number, eg, 1 will select the firstvalue in the list.

Client Side Values Obtained:: F-Language

The XML to be painted on the client end is in form of language called F-Language. The Tables, the components and their information etc. is sent to the client in form of f-lang.
A general format of f-Lang can be given as:

<?xml version="1.0" encoding="UTF-8"?>
<F xmlns:str="http://exslt.org/strings">
<H xmlns:java="http://xml.apache.org/xslt/java" actid="N" t="0"/>
<T s="N" dRX="0.798000" dH="0.861000" dW="0.202000" dY="0.000000" dX="0.000000" g="TableBlack" pid=""
id="RRIMS635" l="" t="S">
<L lw="" lg="" t="" brpos="" bpos="" bd="" dRX="0.053000" dH="0.050000" dW="0.150000" dY="0.002000"
dX="0.070000" g="LabelText" v="" l="Mailbox" id="RRIMS63fldlname" n="fldlname" bt="0"/>
</T>
<T s="N" dRX="0.808000" dH="0.204000" dW="0.182000" dY="0.085000" dX="0.010000" g="Table" pid=""
id="RRIMS631" l="" t="S">
<P r="" fv="" fn="" v="" t="" g="ImageArea" bgi="mail" dX="0.020000" dY="0.062000" dW="0.052000"
dH="0.088000" dRX="0.110000" id="RRIMS63fldpic" bt="0"/>
<IB b="" brpos="" bpos="" bd="" g="ImageButton" bgi="cal" dX="0.115000" dY="0.042000" dW="0.037000"
dH="0.064000" tgtid="" actid="P" dRX="0.030000" r="RRREM61" t="s" id="RRIMS63fldimg1" bt="0" v="RRREM61"
rd="false"/>
<IB b="" brpos="" bpos="" bd="" g="ImageButton" bgi="alert_mail" dX="0.115000" dY="0.116000"
dW="0.037000" dH="0.064000" tgtid="" actid="P" dRX="0.030000" r="RRIMS61" t="s" id="RRIMS63fldimg2"
bt="0" v="RRIMS61" rd="false"/>
</T>
<T s="N" dRX="0.808000" dH="0.145000" dW="0.182000" dY="0.319000" dX="0.010000" g="TableBlack" pid=""
id="RRIMS634" l="" t="S">
<R dRX="0.002000" dW="0.177000" dH="0.135000" dY="0.005000" dX="0.003000" lg="" cg="Radio1" g="Radio"
t="V" n="fldradio" i="" id="RRIMS63fldradio" bt="0" vi="">
<Q r="" tgtid="" actid="" b="set{I@fldFolderId;M@fldMsgType}#fireRequest{RRIMS63,,P}" l="Mails" v="0"
n="fldradio1"/>
<Q r="" tgtid="" actid="" b="set{S@fldFolderId;C@fldMsgType;1@fldradio_ix}#fireRequest{RRIMS65,,P}" l="Sent"
v="1" n="fldradio2"/>
</R>
</T>
<T s="N" dRX="0.000000" dH="0.861000" dW="0.798000" dY="0.000000" dX="0.202000" g="Table" pid=""
id="RRIMS632" l="" t="S">
<B ref="" pi="" tgtid="RRIMS632" actid="" dRX="0.620000" dH="0.052000" dW="0.130000" dY="0.012000"
rd="false" dX="0.012000" g="Button" a="Button" s="def" r="RRIMS67" t="pf" l="+Compose"
id="RRIMS63fldcompose1" n="fldcompose1" pr="" h="RRIMS67,RRIMS632" bt="0" b=""/>
<SB rd="false" dRX="0.219000" dW="0.360000" dH="0.054000" dY="0.012000" dX="0.219000" cg="Radio1"
g="Radio" t="" l="K_LABEL" n="fldmessbull" id="RRIMS63fldmessbull" i="" bt="0" vi="">
<Q r="" tgtid="" actid=""
b="vis{s@RRIMS63fldgrplabel;h@RRIMS63fldgrplabel2;s@RRIMS63fldmailmess;h@RRIMS63fldbulletin}"
l="Messages" v="0" n="fldmessbull1"/>

<Q r="" tgtid="" actid=""
b="vis{s@RRIMS63fldgrplabel2;h@RRIMS63fldgrplabel;s@RRIMS63fldbulletin;h@RRIMS63fldmailmess}"
l="Bulletins" v="1" n="fldmessbull2"/>
</SB>
</T>
<T s="N" dRX="0.202000" dH="0.777000" dW="0.798000" dY="0.080000" dX="0.000000" g="" pid=""
id="RRIMS636" l="" t="S">
<L dRX="0.226000" dH="0.122000" dW="0.422000" dY="0.250000" dX="0.450000" g="VerifyValTextPad" t="s"
l="No Messages in Inbox" id="RRIMS63auto2" n="auto2" bt="0"/>
</T>
<I v="I" vx="" n="fldFolderId" t="h"/>
<I v="M" vx="" n="fldMsgType" t="h"/>
<I v="0" vx="" n="fldradio_ix" t="h"/>
<I v="" vx="" n="fldmessbull_ix" t="h"/>
<I v="" n="fldtxnheading" t="h"/>
</F>

As we can see, the initial is the **F** tag that is used to represent the beginning and end of the f-Lang of a page. The **H** tag is used to pass information regarding the current request received. As, the **t** attribute denotes the success of the response. For eg: t=0 denotes successful result, while t=-1 denotes error in result. Value is picked from the response.
H tag may also contain **M** tag in case the response has some message. This message might be an error message, a warning message or a success message. The **t** attribute denotes the type. Eg t='e' for error message, t='w' for warning message and t='s' for success message where the **l** attribute denotes the message string.

In case of tables, The **T** tag is used to denote individual table. The components of a table are generated inside the T tag (between the opening and end of T tag). The attributes of T tag include:
s= used to denote if the table is scrollable;
dRX= used to denote the relative X position (used in case of RTL languages);
dX= used to denote the X position of table wrt its parent (usually screen).;
dY= used to denote the Y position;
dW & **dH**= used to denote the width and height of the table respectively;
g= used to denote the CSS of the table;
id= used to denote the unique id of the table. Eg RRIMS631 denotes table no 1 of screen RRIMS63;
pid= used to denote the parent id of the table if any. In case the parent is screen, pid is null;
t= type of table, denotes the default display type of the table.

In **case of components**, there are different tags generated to represent different components. For eg: B Tag is used for denoting buttons while IB used for denoting Image button, L is used to denote labels, SB is used to denote segmented button, etc. While the tags are different there are certain attributes in the tag that are common. These can be given as:
r= used to denote request id that is to be fired;

actid = used to denote the action id of the request to be fired;
tgtid= used to denote the target id of the request to be fired;
t= used to denote the type of the component;
g= used to denote the dataclass of the component;
b= used to denote the behavior applied on the component;
dRX= used to denote the relative X position (used in case of RTL languages);
dX= used to denote the X position of component w.r.t. its parent (table).;
dY= used to denote the Y position;
dW & **dH**= used to denote the width and height of the component respectively;
id= used to denote the unique id of the component on the screen. Usually ID of the component or table is used to define behavior (If any) on it;
n= used to denote the name of the component;
bt= used to denote the bare tail number, if this number is less than or equal to the number on BT component, the component is enabled, otherwise disabled;
l= used to define the labels of the component;
v= used to define the value of the component, this value is passed to the next screen and can be accessed via the components' name.

There are many other attributes, which would be defined in the components itself with their purpose.

The **I** tag formed outside any table and of type 'h' (t='h') is used to denote the values of type hidden that are stored on screen and are passed to the next screen where they can be obtained by their name (n attribute). The **v** attribute denotes the value passed while **vx** attribute denotes the value index passed.

Components Explained:

SEGMENTED BUTTON (SB):

'SB' type is used to create a segmented button. In Leap you can select directly 'Data Type' and choose 'Segmented Buttons'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the button.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	SegmentedButton~SegmentedView or Radio~Radio1 (refer CSS doc)	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass
TYPE	**SB**	'SB' defines the component to be segmented button.
NODEVALUE	string('1')~string('2')	'~' separated values (equal to the labels defined in DEFAULTSTATICLABEL). Value of selected button is considered as the value of SB
DEFAULTSTATICLABEL	K_BLOCK~K_UNBLOCK	Defines the total number of segments in the SB. And gives name to each one.
TOKEN1	P~P	Action ID. '~' separated values for each segment in the SB.
TOKEN2		TargetID. '~' separated values for each segment in the SB.
TOKEN3	//faml/request/fldvalue	ValueIndex
TOKEN4	RRVAT62~RRVAT64	REQUESTID. '~' separated values for each segment in the SB.
TOKEN5	Y	Type **Y** — When the behavior and requestid in the button is not auto done on load of page. (Done on Click of

			segment)
		N/'null'	When the behavior or requestid is auto done on page load.

'SB' also has option for index. Such that Value of nameofSB_ix component's value on previous screen is made as the default selected valu on the current screen.

Client side F-Lang::

<SB rd="false" dRX="0.040000" dW="0.960000" dH="0.090000" dY="0.000000" dX="0.000000" cg="Radio1" g="Radio" t="" l=" " n="fldmessbull" id="RRIMS63fldmessbull" i="" bt="0" vi="">

<Q r="" tgtid="" actid="" b="vis{s@RRIMS63fldmailmess;h@RRIMS63fldbulletin;s@RRIMS63fldmailmess;h@RRIMS63fldbulletin}" l="block" v="0" n="fldmessbull1"/>

<Q r="" tgtid="" actid="" b="vis{s@RRIMS63fldbulletin;h@RRIMS63fldmailmess;s@RRIMS63fldbulletin;h@RRIMS63fldmailmess}" l="Unblock" v="1" n="fldmessbull2"/>

</SB>

Where **SB** defines property of segmented button in general & **Q** is for individual segments.
SB: rd= readonly; g= dataclass, cg=cell dataclass, t= type,i= index, vi= valueindex;
All the other attributes are same as defined above.

DYNAMIC SEGMENTED BUTTON (DSB):

'DSB' type is used to create dynamic segmented button. It is similar to 'SB' in presentation and functioning. Though, the number of segments made depends on the values from the response.

Due to this, the entries for 'DSB' are different from 'SB'. The entries for DSB can be given as:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the button.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.

<table>
<tr><td>DATACLASS</td><td>SegmentedButton~SegmentedView or Radio~Radio1 (refer CSS doc)</td><td>Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell’s dataclass</td></tr>
<tr><td>TYPE</td><td>SB</td><td>‘SB’ defines the component to be segmented button.</td></tr>
<tr><td>NODEVALUE</td><td>//faml/response/authorizationstatisticsresponsedto/transactionstatistics/authorizationstatisticsperioddto/statistics/authorizationstatisticsdto~ description~idtxn;status</td><td>X-Path to dto~value to be displayed~value(s) to be passed</td></tr>
<tr><td>TOKEN1</td><td>P</td><td>Action ID</td></tr>
<tr><td>TOKEN2</td><td></td><td>TargetID</td></tr>
<tr><td>TOKEN3</td><td>//faml/request/fldvalue</td><td>ValueIndex</td></tr>
<tr><td>TOKEN4</td><td>RRVAT62</td><td>REQUESTID</td></tr>
<tr><td>TOKEN5</td><td>Y</td><td>Type<table><tr><td>Y</td><td>When the behavior and requestid in the button is not auto done on load of page. (Done on Click of segment)</td></tr><tr><td>N/’null’</td><td>When the behavior or requestid is auto done on page load.</td></tr></table></td></tr>
</table>

Client side F-Lang::

<SB rd="false" dRX="0.040000" dW="0.960000" dH="0.090000" dY="0.000000" dX="0.000000" cg="Radio1" g="Radio" t="" l=" " n="fldmessbull" id="RRIMS63fldmessbull" i="" bt="0" vi="">

<Q r="" tgtid="" actid="" b="vis{s@RRIMS63fldmailmess;h@RRIMS63fldbulletin;s@RRIMS63fldmailmess;h@RRIMS63fldbulletin}" l="Mail" v="0" n="fldmessbull1"/>

<Q r="" tgtid="" actid="" b="vis{s@RRIMS63fldbulletin;h@RRIMS63fldmailmess;s@RRIMS63fldbulletin;h@RRIMS63fldmailmess}" l="Email" v="1" n="fldmessbull2"/>

</SB>

Where **SB** defines property of segmented button in general & **Q** is for individual segments.
SB: rd= readonly; g= dataclass, cg=cell dataclass, t= type,i= index, vi= valueindex;
All the other attributes are same as defined above.

MAIL BOX LIST (ML):

The Mail Box List is created by using type 'ML'. This type of list is used specifically for showing mails for a user.

<table>
<tr><th>COLUMN</th><th>EXAMPLE VALUES</th><th>VALUES DESCRIPTION</th></tr>
<tr><td>LABEL</td><td>K_BLANK</td><td>This column is used to define the Label for the list.</td></tr>
<tr><td>NAME</td><td>fldmailmess</td><td>Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.</td></tr>
<tr><td>ID</td><td>fldmailmess</td><td>Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.</td></tr>
<tr><td>DATACLASS</td><td>List~ListCell (refer CSS doc)</td><td>Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass</td></tr>
<tr><td>TYPE</td><td>ML</td><td>'ML' defines the component to be Mailbox list.</td></tr>
<tr><td>NODEVALUE</td><td>//faml/response/messageandreminderserviceresponsedto/internalmessageresponse/internalmessageserviceresponsedto/internalemessagelist/internalmessagedto[typerecord='M']~sendername;text;received~typerecord;messageid;folderid;hasattachments~hasattachments</td><td>X-Path to dto~values to be displayed~value(s) to be passed~attachment boolean</td></tr>
<tr><td>TOKEN1</td><td>P</td><td>Action ID</td></tr>
<tr><td>TOKEN2</td><td></td><td>TargetID</td></tr>
<tr><td>TOKEN3</td><td>0.078</td><td>Optionheight. Height of an individual cell.</td></tr>
<tr><td>TOKEN4</td><td>RRVAT62</td><td>REQUESTID</td></tr>
<tr><td>TOKEN5</td><td>R~3</td><td>Type<table><tr><td>Value before ~</td><td>Represents the type of list. Refer to List.</td></tr><tr><td>Value after ~</td><td>Number of columns.</td></tr></table></td></tr>
</table>

Client side F-Lang::

<MB bt="1" id="RRIMS63fldmailmess" tgtid="" actid="P" reqid="RRIMS64" n='fldmailmess' tid="" oh="0.078" i='' g="List" og="List" dX="0. 000000" dY="0. 000000" dW="0. 950000" dH="0.760000" dRX="0. 000000" >

<ML hA="Y" v='210987~Loan Department' b="">

<P g="TTransparent" bgi="attachment_icon" dX="0.01000" dY="0.0300" dW="0.05000" dH="0.0550" dRX="0.074000" bt="0"/>

<L g="ListLabel" dX="0.0600" dY="0.01000" dW="0.5600" dH="0.0400" dRX="0.074000" l="Loan Department"/>

<L g="ListLabel" dX="0.0600" dY="0.05500" dW="0.5600" dH="0.0400" dRX="0.074000" l="To close The Loan Account"/>

<L g="ListRGrey" dX="0.6500" dY="0.01000" dW="0.4000" dH="0.1000" dRX="0.074000" l="24-08-2013 12:00:23"/>

</ML>

</MB>

Where **MB** defines property of Mailbox List in general & **ML** is for individual List elements, **P** & **L** etc are individual components for the list element.
MB: oh= option height; g= dataclass, og=cell dataclass, i= index;
ML: hA = has attachment, if hA=Y only then the element contains P tag (Picture View); v=value to be passed to next screen; b=behavior.
P : bgi= image name to be shown when mail has attachments.
All the other attributes are same as defined above.

ATTACH FILE LIST (A):

The Attach File List is created by using type 'A'. It is used to show the attachments, in case user has some or has uploaded some, for eg in a mail.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABELCLASS	VerifyDescriptionLabel	This column is used to define the Label class for the elements of attachment list.
NAME	fldattachment	Gives a name, usually unique in the screen. Value selected on button

<table>
<tr><td></td><td></td><td>can be obtained on next screen using this name.</td></tr>
<tr><td>ID</td><td>fldattachment</td><td>Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.</td></tr>
<tr><td>DATACLASS</td><td>LandingListLabel (refer CSS doc)</td><td>Used to draw the component in a specific design layout</td></tr>
<tr><td>TYPE</td><td>A</td><td>‘A’ defines the component to be Attachment list.</td></tr>
<tr><td>NODEVALUE</td><td>//faml/response/internalmessageserviceresponsedto/internalemessagelist/internalmessagedto/attachments/uploadfiledto~filename~filereference~filesizekb</td><td>X-Path to dto~values to be displayed~value(s) to be passed~size of the file in KB</td></tr>
<tr><td>TOKEN1</td><td>C</td><td>Type of attachment.
<table>
<tr><td>C</td><td>Compose Type. Attachments added.</td></tr>
<tr><td>S</td><td>Received Message Type. One can SEE (download,open ,etc. an attachment)</td></tr>
</table></td></tr>
<tr><td>TOKEN2</td><td>RRIMS31</td><td>REQUESTID</td></tr>
</table>

Client side F-Lang::

<A bt="0" id="RRIMS64fldattachment" r="RRIMS91" l="" eg=" VerifyDescriptionLabel " t="S" n='fldattachment' m='N' i=' ' c=' ' g=" LandingListLabel " dX="0.020000" dY="0.500000" dW="0.800000" dH="0.600000" dRX="0.100000">

<E l="Resume.png" v='001547856 ' s="4.87 KB" />

<E l="File.png" v='001547860 ' s="2.79 KB" />

</A>

Where **A** defines property of Attachment List in general & **E** is for individual List elements.

A: eg= label class CSS; g= dataclass; c=default value, if any; i= index;
E: s = size of the attachment in ‘KB’.
All the other attributes are same as defined above.

LIST (L):

The List is created by using type ‘L’. It is used to show all the possible Elements from the response for a particular value. Entry for a list can be given as:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label for the list.
LABELCLASS	VerifyValueLabel~ListChildHeader	‘~’ separated CSS for each label/component defined in the DISPLAY values of the ‘NODEVALUE’
LABELWIDTH	0.25~0.58#0.05~0.04	This column is used to define the width and height for each label/component defined in the DISPLAY values of the ‘NODEVALUE’. ‘~’ separated width/height value for each label in the LD. Values before # are widths and after # are heights.
NAME	fldbulletin	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldbulletin	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	List~ListCell (refer CSS doc)	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell’s dataclass
TYPE	**L**	‘L’ defines the component to be List.
NODEVALUE	//faml/response/messageandreminderserviceresponsedto/internalmessageresponse/internalmessageserviceresponsedto/internalemessagelist/internalmessagedto[typerecord='B']~sendername;text;received~typerecord;messageid;folderid;hasattachments~messageid	X-Path to dto~values to be displayed (‘;’ separated) ~value(s) to be passed (‘;’separated) ~filter value. Filter value is given in case there is a filter behavior defined for some component based on the selection of list element.
TOKEN1	P	Action ID

<table>
<tr><td>TOKEN2</td><td></td><td>TargetID</td></tr>
<tr><td>TOKEN3</td><td>0.078</td><td>Optionheight. Height of an individual cell.</td></tr>
<tr><td>TOKEN4</td><td>RRVAT62</td><td>REQUESTID</td></tr>
<tr><td>TOKEN5</td><td>R~3#NG</td><td>Type.
Value before #:<table><tr><td>Value before ~</td><td>Represents the type of list.</td></tr><tr><td>Value after ~</td><td>Number of columns.</td></tr></table>Value after #: Defines the type of grouping the list forms. Possible Values are:<table><tr><td>NG</td><td>No Grouping. Default value.</td></tr><tr><td>NTG</td><td>Grouping Present. The List is already in Expanded form, i.e. on click expand disabled and all values displayed by default.</td></tr><tr><td>ETG</td><td>Grouping present. The list is in Contracted form, i.e. on click the list expands to display all values.</td></tr></table></td></tr>
</table>

The Type of List varies from the Functionality to the Display. Values Of ‘TYPE’ can be given as:

In case Of Android Devices:

LIST TYPE	**DESCRIPTION**
R	Used to represent General List structure.

I	Used to represent a List with an image. The First Component of this list is always an Image.
LL	Lazy Loading Type of list is used to upload data in a list from a different request Id. Eg: If a list has to upload image for different elements through request ids, i.e., Individual request id gets image for individual element of the list.
IN	Inner Type list is used in case we want to have a list inside a list. An Inner List is always defined before its parent List. i.e. rowid of child< rowid of parent if tableid is same OR Table Seq. of child< Table Seq. of parent
C	When all the values of the list are to be passed to the next request id irrespective of what is selected, we use this type.

In case of IOS Devices:

LIST TYPE	**DESCRIPTION**
R	Used to represent General List structure.
A	Used to represent list where all the value to be displayed are displayed by default.
G	Used in iPad for lists that need the elements in box form, each box separated from other by some space.
NS	Similar as G, this List has no separator between different element boxes.
E	Used in iPhone for lists that need the elements in box form, each box separated from other by some space.
I	Used to represent a List with an image. The First Component of this list is always an Image.
B	A list type that contains another list or simply a List that is a parent to some other list is of this type.
IN	Inner Type list is used as a child list in case we want to have a list inside a list. An Inner List is always defined before its parent List. i.e. rowid of child< rowid of parent if tableid is same OR Table Seq. of child< Table Seq. of parent
LL	Lazy Loading Type of list is used to upload data in a list from a different request Id. Eg: If a list has to upload image for different elements through request ids, i.e., Individual request id gets image for individual element of the list.

C	When all the values of the list are to be passed to the next request id irrespective of what is selected, we use this type.

Possible Display Values in a List includes a variety of other components. These are defined in the DISPLAY VALUES of the List in the following manner:

COMPONENT TYPE	**EXAMPLE**	**DESCRIPTION**
Formatted Amount (FA)	FA#goalccy#currentgoalamt#//faml/response/sessioninfo/@idlang	'#' separated values where values mean: 1) FA →Formatted amount type 2) Currency used 3) Amount to be formatted 4) ID language
Image (P)	P#L#accttype#c	'#' separated values where values mean: 1) P → Image type where image name is obtained. 2) Inclination of the image, Possible values : a) 'L'- Left inclined b) 'R'- Right inclined 3) Image name evaluated as the value obtained. 4) From where to get the image, Possible values: a) 'c'- value stored at client end b) 's'- value stored at server end
Image (PV)	PV#RRUGD71#LL#fldimglocator^fldRequestType#goalrefno^string('GOAL_CATEG1')	'#' separated values where values mean:

		1) PV → Image type where image is obtained from a different request id. 2) REQUESTID. 3) Type. 'LL' Lazy Loading 4) '^' separated names to be passed to the next screen 5) '^' separated value to be passed corresponding to the names passed.
Formatted Date (FD)	FD#string('BDATEFORMAT')#redeemptiondate#//faml/response/sessioninfo/@idlang	'#' separated values where values mean: 1) FD→ Formatted Date type 2) Date Format to be followed 3) Date to be formatted 4) ID Language
Label with Value (KV)	KV#%%K_REDEMPTION_REF_NO%%:#redeemptionrefno	'#' separated values where values mean: 1) KV→ Label with Value 2) Label between '%%' followed by any sign if needed eg ':' 3) Label value
Label with Formatted Amount (KVA)	KVA#%%K_REDEMPTION_AMOUNT%%:#//faml/request/fldcurr#redeemptionamount#//faml/response/sessioninfo/@idlang	'#' separated values where values mean: 1) KVA→ Label with Formatted Amount 2) Label between '%%' followed by any sign if needed. 3) Currency used

		4) Amount to be formatted 5) ID Language
Label with Formatted Date (KVD)	KVD#%%K_REDEMPTION_DATES%%:#string('BDATEFORMAT')#redeemptiondate#//faml/response/sessioninfo/@idlang	'#' separated values where values mean: 1) KVD→ Label with Formatted Date 2) Label between '%%' followed by any sign if needed. 3) Date Format to be followed 4) Date to be formatted 5) ID Language
Widget Button (WB)	WB#U#main#RRIMS64#U	'#' separated values where values mean: 1) WB→ Widget Button 2) Content Id of the WB 3) Widget Id of the WB 4) Request id 5) Type as for WB
Button (B)	B#string('%%K_RECENTTXN%%')#RRIFA70	'#' separated values where values mean: 1) B→ Button 2) Label to be displayed on Button. 3) Request id
Textbox (T)	T#categoryamount#categAmtI#res#00.00#categorycod	'#' separated values where values mean: 1) T→ Textbox

		2) Nodevalue of the textbox 3) Name of the textbox 4) Behavior 'Function' of Textbox 5) 'Functionargs' of Textbox 6) 'DynamicFunctionArgs' of Textbox
Slider button (SL)	SL#Y#string('100')#string('0')#categorypercentage#0.5^0.75^1	'#' separated values where values mean: 1) SL→ Slider button 2) Is Read Only of 'SL' 3) End Range Of 'SL' 4) Start Range of 'SL' 5) REQUEST ID 6) Range Limit of 'SL'
List [Inner List which is child to the present list] (LC)	LC#RRVBT61fldcateg_ch#string('%%K_VIEW_SUB_CATEGORIES%%')#Y#categorycode **NOTE:** In case of inner list, 1) Inner List sequence should be such that it is painted first. 2) ID of the inner list should be IDofParentList_Anything.	'#' separated values where values mean: 1) LC→ List Type 2) ID of the child list. 3) Label on the button on which list appears (if clickable) OR after which inner list is made (if not clickable) 4) Is Enabled. Possible Values: a) 'Y' : Click is enabled b) 'N' : Click is disabled 5) Value on which filter is applied, such that when 'v' of inner list is equal to the element's filter value, it is

		considered as its child.

Client side F-Lang::

<LO gr="" dRX="0.040000" dH="0.660000" dW="0.960000" dY="0.000000" dX="0.000000" og="ListCell" g="List" i="" oh="0.100" vi="" r="RRIMS64" tgtid="" actid="" nc="3" t="R" n="fldbulletin" rd="" id="RRIMS63fldbulletin" bt="0">

<O b="set{I@fldFolderId;B@fldMsgType}" v="B~1902~I~Bank admin~CUSTOM~N" f="">

<L dH="0.067" dW="0.7" g="ListLabel" v="Bank admin"/>

<L dH="0.067" dW="0.23" g="ListRGrey" v="30-12-2013 00:00:00"/>

<L dH="0.04" dW="0.7" g="ListLabel" v="Test bulletin"/>

</O>

<O b="set{I@fldFolderId;B@fldMsgType}" v="B~22~I~Bank admin~CUSTOM~N" f="">

<L dH="0.067" dW="0.7" g="ListLabel" v="Bank admin"/>

<L dH="0.067" dW="0.23" g="ListRGrey" v="25-07-2013 00:00:00"/>

<L dH="0.04" dW="0.7" g="ListLabel" v="TEST"/>

</O>

</LO>

Where **LO** defines property of List in general & **O** is for individual List elements.

LO: gr= type of grouping; g= dataclass; og=element dataclass; vi=valueindex; i= index; nc= number of columns
O: b = behavior; f= filter value.
Different Components in the O tag depends on the components defined in the Display value of list.
All the other attributes are same as defined above.

Label Heading (L1):

'L1' type is used to create a Label Heading. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION

LABEL	K_BLANK	This column is used to define the Label to be displayed.
NAME	fldfieldname	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldfieldname	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ValueLabel,FBold-AndPhone VerifyLblTextPad - AndTabs VerifyLblTextPad - iPad LabelTextPhone - iPhone (refer CSS doc)	Used to draw the component in a specific design layout.
TYPE	**L1**	'L1' defines the component to be " Label Heading ".
DEFAULTSTATICLABEL		

Client side F-Lang::

<L dRX="0.600000" dH="0.070000" dW="0.400000" dY="0.001000" dX="0.000000" g="ValueLabel,FLeft" t="s" l="Customer Id 1" id="RRATO61fldjointcust1lbl" n="fldjointcust1lbl" bt="0"/>

TEXT BOX (T):

'T' type is used to create a Text Box. In Leap you can select directly 'Data Type' and choose 'Text Box'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label textbox.
NAME	fldfieldname	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldfieldname	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	IPadding,Input -AndPhone Input - AndTabs Input2,BevelEffect - iPad	Used to draw the component in a specific design layout.

	InputText - iPhone (refer CSS doc)	
TYPE	**T**	'T' defines the component to be Text Box.
NODEVALUE	Input	The input is considered as the value of Text Box
DEFAULTSTATICLABEL	K_BLOCK~K_UNBLOCK	Defines the total number of segments in the SB. And gives name to each one.
TOKEN1	N	Text type. N – For numbers E – For email ids S – For any string value D – For decimal value
TOKEN2		deptype
TOKEN3	//faml/request/fldvalue	ValueIndex
TOKEN4	RRQRC66qrcode	Fill only if the type is "LL" in token 5
TOKEN5	i	Type i a QR LL

Client side F-Lang::

<I b="" t="" dRX="0.004000" dH="0.054000" dW="0.306000" dY="0.160000" dX="0.010000" p="S" g="Input" rd="false" k="Input" a="Description" s="35" v="" m="N" n="fldpurposedesc" l="Description of Remittance Purpose" id="RRITG61fldpurposedesc" bt="3" dv="" dt=""/>

TEXT AREA (TA):

'TA' type is used to create a Text Area. In Leap you can select directly 'Data Type' and choose 'Text Area'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the textarea.

NAME	fldfieldname	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldfieldname	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	IPadding,Input -AndPhone Input - AndTabs Input - iPad InputText - iPhone (refer CSS doc)	Used to draw the component in a specific design layout.
TYPE	**TA**	'TA' defines the component to be Text Area".
NODEVALUE	//faml/request/fldplaceholde	
DEFAULTSTATICLABEL		

Client side F-Lang::

<I dRX="0.010000" dH="0.052000" dW="0.313000" dY="0.442000" dX="0.010000" g="Input" rd="false" v="" m="N" t="b" s="35" n="fldnarrative" l="Description" id="RRITR62fldnarrativenat" bt="3"/>

LOOKUP TEXT BOX (TL):

'TL' type is used to create a Look up Text Box. In Leap you can select directly 'Data Type' and choose 'Text Box with Look up'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the textbox.
NAME	fldfieldname	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldfieldname	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	IPadding,Input -AndPhone Input - AndTabs Input2,BevelEffect - iPad	Used to draw the component in a specific design layout.

	InputText - iPhone (refer CSS doc)	
TYPE	**TL**	'TL' defines the component to be " Look up Text Box ".
NODEVALUE	//faml/request/fldplaceholde	
DEFAULTSTATICLABEL		
Token1	RRSRS61	REQUESTID
Token5		

Client side F-Lang::

<I mod="" dRX="0.010000" dH="0.050000" dW="0.480000" dY="0.230000" dX="0.010000" g="Input" r="RRATO64" f="" s="" n="fldbankdetail" m="Y" l="Swift\Bank Code" t="l" id="RRATO61fldbankdetail" v="" rd="true" bt="2"/>

CAPTCHA (CP):

'CP' type is used to create a Captcha Component.

'CP' provides features for 'Captcha Image', 'Captcha Audio', and 'Refresh Button'.

Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL		This column is used to define the Labels.
NAME	fldcaptchaname	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldcaptchaid	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	CSS1~CSS2~CSS3	CSS1 : Entire component CSS CSS2: CSS for Audio Button CSS3: CSS for Refresh Button.
TYPE	**CP**	'CP' defines the component to be Captcha.
NODEVALUE	Not Required	Not Required.

IMAGE	I1~I2	I1: Image name for audio button I2: image for Refresh button
TOKEN2	Request Id	Request Id that will fetch Captcha data (Audio and Image data) when Referesh Button is clicked.

Request Id that is fired should contain 3 fields:

1. fldIdTuring : Hidden Field (H) value= 'Turing Id' of current data.
2. fldaudiostream : Text field (T) with type(token5)= 'a'. value= 'Audio bytestream'
3. fldimagestream : Text field (T) with type(token5)= 'i'. value= 'Image bytestream'

The corresponding TextField used for entering 'Captcha Text' will be validated against the 'Turing Id'.

Client side f-lang format ::

<CP g="**DATACLASS**" r="**TOKEN2**" dX="0.010000" dY="0.180000" dW="0.300000" dH="0.100000" dRX="0.690000" id="RequestId+**ID** " n="**NAME**">

<GR …/>

<IB …/>

<IB … />

</CP>

Sample client side F-Lang::

Screen Containing CP:

<CP g="transparentButton" r="RRTIA65" dX="0.010000" dY="0.180000" dW="0.300000" dH="0.100000" dRX="0.690000" id="RRORG67fldcapimage" n="fldcapimage">

<GR v="" dX="0.003" dY="0.004" dW="0.244" dH="0.092" dRX="0" id="RRORG67fldcapimage1" n="fldcapimagegraph"/>

<IB t="c" dRX="0.0" tgtid="" dH="0.04" dW="0.04" dY="0.004" dX="0.257" id="RRORG67fldcapimage2" n="fldcapimageaudio" bgi="captcha_sound" g="ImageButton"/>

<IB t="r" dRX="0.0" tgtid="" dH="0.04" dW="0.04" dY="0.060" dX="0.257" id="RRORG67fldcapimage3" n="fldcapimagerefresh" bgi="captcha_refresh" g="ImageButton"/>

</CP>

Screen Containing bytestreams:

<F xmlns:str="http://exslt.org/strings">

<H xmlns:java="http://xml.apache.org/xslt/java" actid="N" t="0"/>

<T s="N" dRX="1.000000" dH="0.000000" dW="0.000000" dY="0.000000" dX="0.000000" g="" pid="" id=" " l="" t="S">

***<I b="" t="i"** dRX="0.000000" dH="0.000000" dW="0.000000" dY="0.000000" dX="0.000000" p="" g="Input2,BevelEffect" rd="false" k="Input2,BevelEffect" a="" s="" **v="Image Bytestream"** m="N" n="fldimagestream" l="" id="RequestID+fldimagestream" bt="0" dv="" dt=""/>*

***<I b="" t="a"** dRX="0.000000" dH="0.000000" dW="0.000000" dY="0.000000" dX="0.000000" p="" g="Input2,BevelEffect" rd="false" k="Input2,BevelEffect" a="" s="" **v=" Audio Bytestream"** m="N" n="fldaudiostream" l="" id="RequestId+fldaudiostream" bt="0" dv="" dt=""/>*

</T>

***<I v="LWVcKjMqcOjX55hkOZO8kYcelYV4x1dXAtCm5QN22Mlwj9c1"** vx="" **n="fldIdTuring" t="h"**/>*

<I v="K_NON_LOGIN_PSC5" n="fldtxnheading" t="h"/>

</F>

Add UDF to Request (AU)

'AU' type is used to re-create udf fields from Request.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	No Label Required	
NAME	fldaddudf	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldaddudf	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	Not Required	
TYPE	**AU**	'AU' defines the component to add udf fields from request to next

		request.
NODEVALUE	Not Required	
DEFAULTSTATICLABEL	Not Required	Not Required
TOKEN1	Not Required	Not Required
TOKEN2	Not Required	Not Required
TOKEN3	Not Required	Not Required
TOKEN4	Not Required	Not Required
TOKEN5	Not Required	Not Required

In case, if there are UDF fields in Request, 'AU' component creates them as 'Hidden UDF' fields.

Client side F-Lang::

Screen 1 : (with udf fields)

<T ……>
<S n="fldtypeowner" id="RRORG171fldtypeowner" ….>
<O />
<O />
</S>
<U n="fldtypeowner"/>

<I n="fldother" id="RRORG171fldother" />
<U n="fldother"/>

<S n="fldbusinesstype" id="RRORG171fldbusinesstype" >
<O />
<O />
</S>
<U n="fldbusinesstype"/>

<U n="fldbussinessname"/>
<I n="fldbussinessname" id="RRORG171fldbussinessname" />

<U n="fldother1"/>
<I n="fldother1" id="RRORG171fldother1" />

<U n="fldturnover"/>
<I n="fldturnover" id="RRORG171fldturnover" />
…
…

Screen 2: (with AU component)

<T …>
<I v="Partnership Firm~Partnership Firm" vx="" n="fldtypeowner" t="h"/>
<U n="fldtypeowner"/>
<I v="" vx="" n="fldother" t="h"/>
<U n="fldother"/>
<I v="Retail~Retail" vx="" n="fldbusinesstype" t="h"/>
<U n="fldbusinesstype"/>
<I v="ffggt" vx="" n="fldbussinessname" t="h"/>
<U n="fldbussinessname"/>
<I v="" vx="" n="fldother1" t="h"/>
<U n="fldother1"/>
<I v="343" vx="" n="fldturnover" t="h"/>
<U n="fldturnover"/>
<I v="USD~US Dollar" vx="" n="fldturnoverccy" t="h"/>
<U n="fldturnoverccy"/>
</T>

Sample Response:

Screen 2: (Screen with AU)

<fldbusinesstype>
<![CDATA[Retail~Retail]]>
</fldbusinesstype>
<fldudf>
<![CDATA[fldbusinesstype]]>
</fldudf>
<fldturnover>
<![CDATA[343]]>
</fldturnover>
<fldudf>
<![CDATA[fldturnover]]>
</fldudf>

<fldtypeowner>
<![CDATA[Partnership Firm~Partnership Firm]]>
</fldtypeowner>
<fldudf>
<![CDATA[fldtypeowner]]>
</fldudf>
< fldother>
<![CDATA[]]>
</ fldother>
<fldudf>
<![CDATA[fldother]]>
</fldudf>
< fldother1>
<![CDATA[]]>
</ fldother1>
<fldudf>
<![CDATA[fldother1]]>
</fldudf>
<fldturnoverccy>
<![CDATA[USD~US Dollar]]>
</fldturnoverccy>
<fldudf>
<![CDATA[fldturnoverccy]]>
</fldudf>
<fldbussinessname>
<![CDATA[ffggt]]>
</fldbussinessname>
<fldudf>
<![CDATA[fldbussinessname]]>
</fldudf>

If you add AU component in Screen 3, udf fields generated in screen2 will be auto-generated as 'Hidden UDF' in f-xml.

BARETRAIL (BT):

'BT' type is used to create Baretrail. In Leap you can select directly 'Data Type' and choose 'Baretrail'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldbaretrailname	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldbaretrailid	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	CSS	CSS for Baretrail (BT) component.
TYPE	**BT**	'BT' defines the component to be segmented button.
NODEVALUE	Not Required	Not Required.
SCREEN LEVEL : TOTALSTEPCOUNT	N	N : Number of digits to show on the baretrail.(BT)
TOKEN1	Enabled Stages	Number upto which the components will be enabled.
TOKEN2	Initial Stages	Number upto which the baretrail component will be highlighted.

For every Component in Leap, we have an entry for 'Step Number'. If the components 'Step Number' is less than or equal to 'Enabled Stages', the component is accessible for input. Otherwise the component is disabled.

Whenever components with 'Step Number' less than or equal to 'Enabled Stages' are filled or updated, 'Enabled Stages' is incremented by one, BT is updated to show the next digit, and the components with 'Step Number' equal to the incremented value are made accessible for input.

Client side F-Lang format::

<BT dRX="RTL-X" dH="RELHEIGHT " dW="RELWIDTH " dY="RELPOSY " dX="RELPOSX " e="TOKEN1" g="DATACLASS" c="TOKEN2" ts=" TOTALSTEPCOUNT " id="RequestId+ID "/>

</T>

Sample client side F-Lang::

<BT dRX="0.010000" dH="0.060000" dW="0.980000" dY="0.010000" dX="0.010000" e="2" g="StepTable" c="2" ts="5" id="RRORG271baretrail"/>

</T>

BUTTON GROUP (BG):

'BG' type is used to create a segmented button.

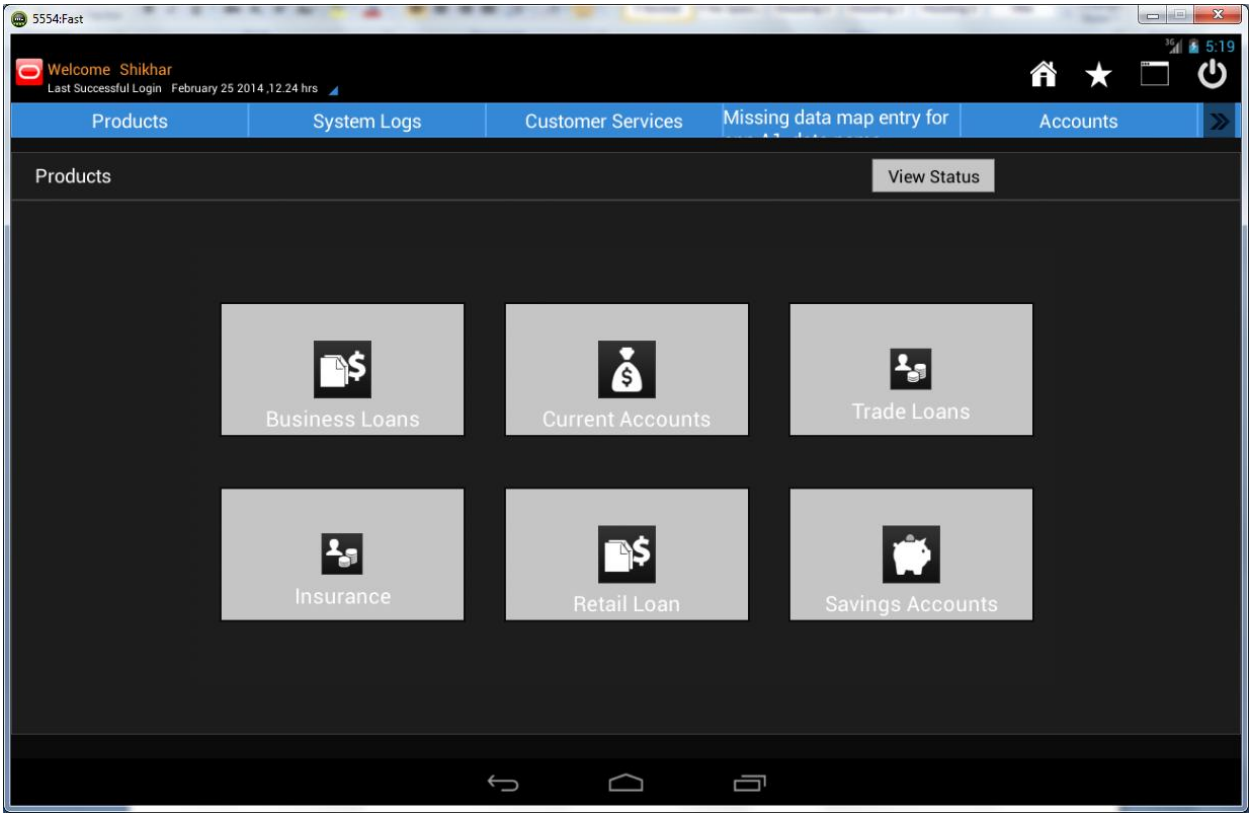
'BG' is a dynamic component only. Static version of this component is not available. To have the feel of the static Version, it is advised to add individual buttons with proper positions and dimensions.

Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	Not Required
NAME	fldcomponentname	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldcomponentid	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
LABELWIDTH	BW~BH	Button Width ~ Button Height
LABELCLASS	CSS-T	CSS for Text in Buttons.
DATACLASS	CSS1~CSS2	CSS for entire BG component ~ CSS for Buttons in 'BG'
TYPE	**BG**	'SB' defines the component to be segmented button.
NODEVALUE	xPath~Desc~Code	xPath of DTO from response ~ relative path of labels to display on the buttons ~ relative path of the values to be submitted when a button is clicked
IMAGE	xPath	Relative path of the node from response that contains image names to display on the buttons in BG
RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT		Positions and Dimensions of the entire 'BG' component irrespective of the number of Buttons
TOKEN1	Action Id	Action Id for the next screen
TOKEN2	Target Id	Target Id for the next screen
TOKEN3	RH	Height of an Individual Row in the BG
TOKEN4	Request Id	Request Id to fire when a button is clicked from the BG. All Buttons will fire the same Request Id.

TOKEN5	NC	Number of Columns in the BG.

Component ::



F-lang Format :

<BG bt="{Baretrail Step Number}" id="{ RequestId+ **ID** }" rH="{**TOKEN3** }" lg="{ **LABELCLASS** }" n="{ **NAME**}" nc="{ **TOKEN5** }" bW="{ **LABELWIDTH (BW)**}" bH="{ **LABELWIDTH(BH)** }" g="{ **DATACLASS(CSS1)** }" bg="{ **DATACLASS(CSS2)** }" dX="{ **RELPOSX**}" dY="{ **RELPOSY** }" dH="{ **RELHEIGHT** }" dW="{ **RELWIDTH**}" dRX="{$posrx}" actid="{**TOKEN1**}" tgtid="{**TOKEN2**}" r="{**TOKEN4**}" >

<B bgi=”**IMAGE**” l=”**NODEVALUE(DESC)”** v=”**NODEVALUE(CODE)**” **/>**

</BG>

Sample client side F-Lang::

<BG r="RRPSC62" tgtid="" actid="" dRX="0.150000" dW="0.700000" dH="0.647000" dY="0.070000" dX="0.150000" bg="ConfirmButton" g="Radio1" bH="0.2" bW="0.2" nc="3" n="fldprodcateg" lg="VerifyItemLabel" rH="0.25" id="RRPSC61fldprodcateg" bt="0">

<B bgi="loans" l="Business Loans" v="B"/>

<B bgi="casa" l="Current Accounts" v="C"/>

<B bgi="ownicon" l="Trade Loans" v="D"/>

<B bgi="ownicon" l="Insurance" v="I"/>

<B bgi="loans" l="Retail Loan" v="L"/>

<B bgi="deposits" l="Savings Accounts" v="S"/>

</BG>

PICTURE UPLOAD (PU):

'PU' type is used to create a picture upload component. In Leap you can select directly 'Data Type' and choose 'Picture Upload'. Entries needed for this type are:

COLUMN	**EXAMPLE VALUES**	**VALUES DESCRIPTION**
NAME	fldmessbull	Gives a name, usually unique in the screen.
ID	fldmessbull	Gives a unique id to the component in the screen.
DATACLASS	Button (refer CSS doc)	Used to draw the component in a specific design layout.
TYPE	**PU**	'PU' defines the component to be picture

		upload. This type enables the user to upload picture from client.
NODEVALUE	//faml/response/imagepreviewresponsedto/uploadfiledetails/uploadfiledto/uploadedfilecontent	DTOpath of the image in bytestream format.
TOKEN1	RRIMS93	Request ID.
TOKEN5	c	Type

Client side F-Lang::

<PU t="c"
v="R0lGODlhAwHCAPcAAAICAgQFCQUIBQYJCQoDAwoGCQkJBAoLCwwNEQwQDQ4RERMJCRUNERQTDBITExUVGRUYFhYZGRoTFRsVGRkaFBobGw8VIxwdIR4hDhwgHR0iIhslMSQEACUKEyYYBiIZHDgZAjETHSUdISIdMTUcJCUkDiQkGy4wHjcpFzMyHSYnJystMCwxLi0xMjMoKzUtMTQzKzIzMzU1OTU4NTY5OTozNTo1OTo6NTo7Oz09QTxAPD5BQkQZBEYsA0IlHUYxGlgoB1QyF0IjLUUpMkk1I0I6O1Y5JFM+ ==" n="fldcurntimgrefnoid" r="RRIMS93" g="Button" dX="0.000000" dY="0.010000" dW="0.200000" dH="0.120000" dRX="0.000000" id="RRUGD66fldcurntimgrefnoid"/>

Where **PU** defines Picture Upload in general.
PU: t=type; g= dataclass, r=request id,v=value;
All the other attributes are same as defined above.

WIDGET VIEW (WV):

'WV' type is used to create a widget view. In Leap you can select directly 'Data Type' and choose 'Widget View'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldwidget	Gives a name, usually unique in the screen.
ID	fldwidget	Gives a unique id to the component in the screen.
DATACLASS	contentWidgetStyle (refer CSS doc)	Used to draw the component in a specific design layout.

<table>
<tr><td>TYPE</td><td>WV</td><td>‘WV’ defines the component to be widget view.</td></tr>
<tr><td>TOKEN1</td><td>U</td><td>Content ID.
<table>
<tr><th>Content Id Values</th><th>Description</th></tr>
<tr><td>U</td><td>When the Widget view is on Dashboard.</td></tr>
<tr><td>IP</td><td>When the Widget View is inside any menu transaction.</td></tr>
</table></td></tr>
<tr><td>TOKEN2</td><td>RRUGD71</td><td>Request ID.</td></tr>
<tr><td>TOKEN3</td><td>U</td><td>Type
<table>
<tr><th>Type</th><th>Description</th></tr>
<tr><td>R</td><td>To pick XML from client-side</td></tr>
<tr><td>X</td><td>View Presentation is on Client-side but the XML is on server side.</td></tr>
<tr><td>U</td><td>Server side XML.</td></tr>
<tr><td>N</td><td>Notification View.</td></tr>
</table></td></tr>
<tr><td>TOKEN4</td><td>fldgoalName^fldgoalOwner</td><td>Field Name. ‘^’ separated values</td></tr>
<tr><td>TOKEN5</td><td>Sedan^amey retail</td><td>Field Value. ‘^’ separated values</td></tr>
</table>

Client side F-Lang::

<WV fv=" Sedan~amey retail " fn="fldgoalName~fldgoalOwner" dRX="0.515000" dH="0.320000" dW="0.485000" dY="0.010000" dX="0.000000" r="RRUGD71" g="contentWidgetStyle" cid="R" t="RH" wid="RRUGD63fldimage"/>

WV: cid=contentid; g= dataclass, r=request id, t= type,fn= fieldnames, fv= fieldvalues, wid= widgetid; All the other attributes are same as defined above.

LOOKUP DROPDOWN (LD):

'LD' type is used to create a lookup dropdown. In Leap you can select directly 'Data Type' and choose 'Lookup Dropdown'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the textbox appearing in the lookup dropdown.
LABELCLASS	VerifyLblTextPad~VerifyLblTextPad or lblBlack2~lblBlack2(refer CSS doc)	This column is used to define the dataclasses for each label. . '~' separated values for each label in the LD
LABELWIDTH	0.30~0.15#0.04~0.04	This column is used to define the width and height for each label. . '~' separated width/height value for each label in the LD. Values before # are widths and after # are heights.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value selected on list can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	FilterBox~Select (refer CSS doc)	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass
TYPE	**LD**	'LD' defines the component to be lookup dropdown.
NODEVALUE	//faml/response/ bankdetailsdto~ nambank; swiftcode~ swiftcode; city	Value before first ~ is the path of the dto.Value after first ~(or before second ~) is the value which needs to be shown in the list(; separated values for multiple labels to be shown).Value after second ~ is the value which needs to be passed further(; separated multiple values)
TOKEN1	P	Action ID.
TOKEN2	RRVAT614	TargetID.

TOKEN3	0.08	Option Height.(Height of each row of the list.)
TOKEN4	RRVAT62	REQUESTID.
TOKEN5	A~2	Type Value before ~ defines the type of list and Value after ~ defines the number of columns in the list. For the possible values of Type : Refer List DataType.

Client side F-Lang::

<LD dRX="0.000000" dH="0.900000" dW="1.000000" dY="0.060000" dX="0.000000" g="List" m="Y" id="RRCGS65fldcode" bt="0">

<I dRX="0.0" dH="0.06" dW="1.000000" dY="0.01" dX="0.005" g="FilterInput" l="Please Select Bank Name:" n="fldcode" v="" s="" a="" k="" r="false"/>

<LO dRX="0.0" dH="Match" dW="1.000000" dY="0.08" dX="0.005" og="ListCell" g="List" i="" oh="0.08" vi="" r="" tgtid="" actid="" nc="2" t="R" n="fldcode" rd="" id="RRCGS65fldcode" bt="">

<O b="set{BANKCL1000@RRCGS67fldrecvamntdcnbnkcdtxt;BANK FOR CL1 BRANCH@RRCGS67fldrecvamntdcnbknametxt;CL Bank@RRCGS67fldrecvamntdcnbnkadrstxt;LONDON@RRCGS67fldrecvamntdcnbnkcitytxt}" v="BANKCL1000~BANK FOR CL1 BRANCH~LONDON~~CL Bank~2nd Street~London" f="">

<L dH="0.08" dW="0.8" g="ValueText,FLeft" v="BANK FOR CL1 BRANCH CL Bank 2nd Street London LONDON "/>

<L dH="0.06" dW="0.7" g="ValueText,FLeft" v=" BANKCL1000"/>

</O>

</LO>

</LD>

Where **LD** defines lookupdropdown in general ,**I** is for input(textfield) ,**LO** is for list ,**O** is for individual row & **L** is for each label within a row.
LD: g= dataclass;

I: g=dataclass,l=label;

LO: g=dataclass,og=dataclass for each row,oh=option height,t=type,r=request id,nc=number of columns;

O: b=behavior;

L: g=dataclass for each label,v=value;
All the other attributes are same as defined above.

CUSTOM TEMPLATE (CT):

'CT' type is used to create a specific field. In Leap you can select directly 'Data Type' and choose 'Custom Template'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldct	Gives a name, usually unique in the screen.
ID	fldct	Gives a unique id to the component in the screen.
TYPE	**CT**	'CT' defines the component to be custom type.

Name and ID are required just to make LEAP entries .

CT is used for creating custom f-lang respective to clients. For each RequestId containing CT component, there needs to be custom f-lang response in usabilitycustomtemplate.xsl

Client side F-Lang::

(In usabilitycustomtemplate.xsl)

```
<xsl:if test="//faml/request/fldRequestId='RRSUC61'">
        <R n="fldchqnumrangeradio" g="Radio" dX="0.01" dY="0.11" dW="0.62" dH="0.05" dRX="">
                <Q v="1" n="fldchqnum1"  l="Cheque Number" e="fldchqno"
d="fldchqrangestart,fldchqrangeend"/>
                <Q v="2" n="fldchqrange1"  l="Cheque Range" e="fldchqrangestart,fldchqrangeend"
d="fldchqno"/>
        </R>
</xsl:if>
```

GRAPH AREA (GR):

'GR' type is used to create a graph area. In Leap you can select directly 'Data Type' and choose 'Graph Area'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the graph.
NAME	fldmessbull	Gives a name, usually unique in the screen.
ID	fldmessbull	Gives a unique id to the component in the screen.
NODEVALUE	/faml/response/savingscalculatorresponsedto/graphimage	DTO Path of the bytestream for image.

Client side F-Lang::

<GR n=' fldgraph' id="RRSCL62fldgraph" dRX="0.00" dH="0.50" dW="1.0" dY="0.01" dX="0.0" v='
R0lGODlhAwHCAPcAAAICAgQFCQUIBQYJCQoDAwoGCQkJBAoLCwwNEQwQDQ4RERMJCRUNERQTDBITE
xUVGRUYFhYZGRoTFRsVGRkaFBobGw8VIxwdIR4hDhwgHR0iIhslMSQEACUKEyYYBiIZHDgZAjETHSUdISId
MTUcJCUkDiQkGy4wHjcpFzMyHSYnJystMCwxLi0xMjMoKzUtMTQzKzIzMzU1OTU4NTY5OTozNTo1OTo6
NTo7Oz09QTxAPD5BQkQZBEYsA0llHUYxGlgoB1QyF0IjLUUpMkk1I0I6O1Y5JFM+NGUqA2E1Eng7BHc8FU
Q9QkI/UERBK0REO1tJJlVHOFRSPGtHGGVHJWlTOndPLHJVKnpZMnxkPkVFRkxMUU1RTU1RUlJMSlRNUVZ
WS1NTU1VVWVVYVVZZWllUVVpWWFpaVVtbW1tcYV9hTFxiW15hYmRPVmFeSmJcXWJdYmRiS2VkWHZn
SHJqVnd1XmdnZ2xtcWtxam5xc3JsbHJtcnZ3Z3d3d3x8gX+EZnyBe32BgoI8AIpKDZhYKIZtOJRoN5BzL5t4PK
BPDqdXGKlnLKJkMYRlSIFmWIt2RptuRZ96RZ17WIR/ZoF6faR+V8h1J8Z/PdB/PYN9hJWAQZqDVYaFd42ReZ
6Ea5OOeJOUeqOIW66RQ6aQXqSGbK2Qbr6rZt6LPeiNPseCR9uMUd6VUNabY8WiUdm9XMKob+aOROGT
SfmdSv+tWOSuf/mycubRfv/FboeHh4yNkY2QjY6RkZCLipOOkZaWh5eXl52doJ2gnJ6hoaCdiqObm6Ke'/>

Where **GR** defines Graph Area in general.
GR: v=value;
All the other attributes are same as defined above.

DYNAMIC ADDITION (DA):

'DA' type is used to create a component with dynamic addition property. In Leap you can select directly 'Data Type' and choose 'Dynamic Addition'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABELCLASS	ImageArea~Input2,BevelEffect	This column is used to define the DataClass of the component.Value before ~ defines the css of ImageView

		and value after ~ defines the css of input component.
LABELWIDTH	0.08~0.40#0.08~0.07	This column is used to define the width and height of component . '~' separated width/height value for image and input component respectively. Values before # are widths and after # are heights.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value entered in input field can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen.
DATACLASS	StepTable~StepTable (refer CSS doc)	Used to draw the component in a specific design layout. Value before ~ is parent's dataclass and after ~ is child's dataclass
TYPE	**DA**	'DA' defines the component to be dynamic addition.
NODEVALUE	//faml/response/goaldetailsresponsedto/goal/goaldetailsdto/participantlist/participantlistdto~participant~participant	DTO Path of the Value(This is required only when DA is of R type)
IMAGE	add-button-retina	Name of the Image to be shown.
TOKEN1	A or R	Type. **A** Addition **R** Removal.
TOKEN2	5	Maxlimit.(Max. number of rows incase of type A)
TOKEN3	0.08	ChildHeight
TOKEN4	E	TextType. **E** Email **S** String. **D** Decimal.
TOKEN5	RRANP63~V	Request Id. Value before ~ is request Id and Value after ~ is action id.

Client side F-Lang::

For type A:

<DA imgbgi="add-button-retina" imgG="StepTable" imgH="0.08" imgW="0.20" t="A" cc="5" cH="0.10" cg="StepTable" g="StepTable" dH="0.500000" dW="0.980000" dY="0.070000" dX="0.010000" id="fldnpemailid" n="fldnpemailid">

<DC v="">

<I b="" t="" dRX="" dH="0.08" dW="0.75" dY="" dX="" p="E" g="Input1" rd="" k="Input1" a="" s="" v="" m="N" n="fldnpemailid1" l="" id="" bt="" dv="" dt=""/>

</DC>

</DA>

For type R:

<DA imgbgi="" imgG="StepTable" imgH="0.08" imgW="0.20" t="R" cc="5" cH="0.10" cg="StepTable" g="StepTable" dH="0.500000" dW="0.980000" dY="0.070000" dX="0.010000" id="fldRemoveParticipant" n="fldRemoveParticipant">

<DC actid="V" r="RRANP63" v="asdrfgh">

<I b="" t="" dRX="" dH="0.08" dW="0.75" dY="" dX="" p="" g="Input1" rd="" k="Input1" a="" s="" v="asdrfgh" m="N" n="fldRemoveParticipant1" l="" id="" bt="" dv="" dt=""/>

</DC>

<DC actid="V" r="RRANP63" v="qawsed">

<I b="" t="" dRX="" dH="0.08" dW="0.75" dY="" dX="" p="" g="Input1" rd="" k="Input1" a="" s="" v="qawsed" m="N" n="fldRemoveParticipant1" l="" id="" bt="" dv="" dt=""/>

</DC>

</DA>

Where **DA** defines dynamic addition in general , **DC** is for individual row & **I** is for input field.

DA: imgbgi= imgname; imgG= dataclass of Image, t= type,cc= maxcount,cH=child height, cg= child dataclass,g=parent dataclass;

DC:actid=acrtion id,r=request id,v=value to be passed.

I:g=dataclass,v=value to be passed
All the other attributes are same as defined above.

CONDITIONAL FIELD (CF):

'CF' type is used to create a conditional field. In Leap you can select directly 'Data Type' and choose 'Conditional Field'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of data type defined in the Conditional Field node value.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS		
TYPE	**CF**	'CF' defines the component to be Conditional field.
NODEVALUE	I+not(//faml/response/registeredbillerlistresponsedto/ registeredbillerdetails/registeredbillerdetailsdto)||L1|| %%K_BILLER_EXIST%%||	Value (it should be '+' and '||' separated.)
FUNCTION ARGS		Argument for data type ,

		defined in node value.

Client side F-Lang::

There is no specific client code for the “Conditional Field”. You will get the particular datatype client code as defined in node value of “CF”.

FOREX RATE (FX):

‘FX’ type is used to show the FOREX Rates on the screen. In Leap you can select directly ‘Data Type’ and choose ‘FOREX Rate ’. There are no other entries needed for ‘FX’ type.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	Not Required.	
NAME	fldfxrate	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldfxrate	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS		
TYPE	FX	‘FX’ defined the component to render the Forex Rates on the screen.

Client side F-Lang::

<D>

<L v="GREAT BRITAIN POUND(GBP)" l="From Currency:"/>

<L v="EURO(EUR)" l="To Currency:"/>

<L v="1.05" l="Cash Buy:"/>

<L v="1.20" l="Cash Sell:"/>

<L v="1.06" l="TT Buy:"/>

<L v="1.19" l="TT Sell:"/>

</D>

ACCORDION(AC):

'AC' type is used to create a ACCORDION. An effect of animation for a table can be obtained using this type. In Leap you can select directly 'Data Type' and choose 'Accordion'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the button on accordion.
NAME	fldaccordion	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldaccordion	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	SecondaryButtonPad, HorizontalSliderButton	Used to draw the component in a specific design layout.
TYPE	**AC**	'AC' defines the component to be ACCORDIAN type.
NODEVALUE	Not required	
TOKEN1	RROAT617	Table ID which needs to be effected.
TOKEN2	RROAT6110	TargetID.
TOKEN3		
TOKEN4	RRVAT70	REQUESTID.
TOKEN5	B	Type **B** Bottom Transition **T** Top Transition **R** Right Transition

		L	Left Transition

Client side F-Lang::

<AC tgtid="RROAT6110" dRX="0.000000" tid="RROAT617" dH="0.050000" dW="1.000000" dY="0.000000" dX="0.000000" g="SecondaryButtonPad" t="B" l="Recent Transferred Funds" r="RRVAT70" id="RROAT61fldaccordion" f="" bt="0" bgi="" rd="false"/>

PICTURE VIEW (PV):

'PV' type is used to display image. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label.
NAME	fldimage	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldimage	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ImageArea, darkLine, lightLine, headerLine	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass
TYPE	**PV**	'PV' defines the component to be picture view.
NODEVALUE	//faml/response/imagename	Dynamic loading of images, the path for image name is given here. And type='di' is a must
Image	Image name	Name of the image to be pasted, the image file(.png) must be present in the client directory of images. Avoid writing file extension, name is enough.
TOKEN1	Fldimglocator ^ fldRequestType	Field Names to be passed. '^' separated values define the fieldnames.
TOKEN2	//faml/response/goaldetailsresponsedto/goalrefno ^ string('GOAL_CATEG1')	Field values to be passed. '^' separated values define the field values corresponding to the field names defined in Token 1

<table>
<tr><td>TOKEN3</td><td>RRUGD71</td><td>Request Id</td></tr>
<tr><td>TOKEN4</td><td>//faml/request/fldcurntimgrefnoid_loc</td><td>Used when image has to be picked up from the path specified, type='c'</td></tr>
<tr><td>TOKEN5</td><td>LL</td><td>Type
<table>
<tr><td>LL</td><td>Lazy Loading, when the image has to be fetched from a certain request id on the basis of the fields and corresponding values in Token 1 and Token 2.</td></tr>
<tr><td>C</td><td>Give the path in Token4 from where image has to be picked.</td></tr>
<tr><td>Di</td><td>Dynamic loading of images</td></tr>
<tr><td>No type</td><td>Specify the name of the image in Image column.</td></tr>
</table></td></tr>
</table>

Client side F-Lang::

<P r="RRUGD71" fv="2192~GOAL_CATEG1" fn="fldimglocator~fldRequestType" v="" t="LL" g="contentWidgetStyle" bgi="" dX="0.025000" dY="0.085000" dW="0.200000" dH="0.200000" dRX="0.432000" id="RRUGD62fldimage" bt="0"/>

fn: the fieldnames separated by '~',fv: the fieldvalues separated by '~'

FORMATED DATE (FA):

'FA' type is used to create a Formatted Date. In Leap you can select directly 'Data Type' and choose 'Formatted Date'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the field.
NAME	fldamount	Gives a name, usually unique in the screen.
ID	fldamount	Gives a unique id to the component in the screen. ID is used to define

		behavior to the particular component.
DATACLASS	ValueLabel,FRight	Used to draw the component in a specific design layout.
TYPE	**FA**	'FA' defines the component of Formatted Date.
NODEVALUE	//faml/request/fldCurrency#//faml/request/fldAmount#//faml/response/sessioninfo/@idlang	Needed 3 arguments separated by '#' 1.Currency 2.Amount 3.Language ID
TOKEN1	**KV** Or **KVA** Or **null**	1.If NODEVALUE (Currency)='RATE' then TOKEN1='KV' or null depending if the label is to be shown along with value or not, respectively. 2.If NODEVALUE (Currency)!='RATE' then TOKEN1='KVA' or null depending upon whether the label is shown along with the value or not, respectively.
TOKEN2	0.260000	List Width

Client side F-Lang::

<L dRX="0.024000" dH="0.050000" dW="0.460000" dY="0.200000" dX="0.476000" g="ValueLabel,FRight" v="200.00~GBP" t="a" id="RRPPP62fldamount" bt="0"/>

Where **L** defines label for formatted Amount.

L: g= dataclass,v=value, t= type,id=ID+RequestID,bt=baretail step;

All the other attributes are same as defined above.

Formatted Date (FD):

'FD' type is used to create a Formatted Date. In Leap you can select directly 'Data Type' and choose 'Formatted Date'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the field.
NAME	fldprefdatevrfy	Gives a name, usually unique in the screen.
ID	fldprefdatevrfy	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ValueLabel,FRight	Used to draw the component in a specific design layout.
TYPE	**FD**	'FD' defines the component to be Formatted Date.
NODEVALUE	string('BDATEFORMAT')#//faml/response/leadoriginationresponsedto/applicationdetails/origapplicationdetailsdto/applicantdetails/origapplicantdetailsdto/preferredcontactdate#//faml/response/sessioninfo/@idlang	Needed 3 arguments separated by '#' 1.Date Formate 2.Processing Date 3.Language ID
TOKEN1	**null** Or **KVD**	1.If Label is not shown with the value, type = null 2.If Label is shown with the value, type = KVD
TOKEN2	0.260000	List Width

Client side F-Lang::

<L dRX="0.024000" dH="0.050000" dW="0.460000" dY="0.200000" dX="0.476000" lg="ValueLabel,FRight" v="12-12-2013" id="RRLAP71 flddobvrfy" bt="0"/>

Where **L** defines label for formatted Date.
L: lg= dataclass,v=value,id=RequestID+ID,bt=baretail step;
All the other attributes are same as defined above.

GROUP LABELS (GL):

'GL type is used to create a widget button. In Leap you can select directly 'Data Type' and choose 'Group Label'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_SENDER~K_SUBJECT~K_DATE	This column is used to define the Labels to show ~ separated.
NAME	auto2	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	auto2	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	HeaderLabel~HeaderLabel~HeaderLabel	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass
TYPE	**GL**	'GL' defines the component to be group label.
NODEVALUE	string('1')~string('2')~string('3')	
DEFAULTSTATICLABEL		
TOKEN1	0.30~0.30~0.30	Labels Width (it should be '~') separated.
TOKEN5	Y	

Client side F-Lang::

```
<GL dRX="0.056000" dH="0.062000" dW="0.730000" dY="0.000000" dX="0.012000" n="auto2" g="HeaderLabel~HeaderLabel~HeaderLabel" id="RRIMS61auto2" bt="0">

<L g="HeaderLabel" dW="0.30" l="Sender"/>

<L g="HeaderLabel" dW="0.30" l="Subject"/>

<L g="HeaderLabel" dW="0.30" l="Date"/>

</GL>
```

Where **GL** defines property of labels group & **L** is for individual label.
GL: rd= readonly; g= dataclass, cg=cell dataclass;

All the other attributes are same as defined above.

HYPERLINK (HL):

'HL' type is used to create a hyper links. In Leap you can select directly 'Data Type' and choose 'hyperlinks '. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the hyperlink.
NAME	fldlink	Gives a name, usually unique in the screen.
ID	fldlink	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	Hyperlink	Used to draw the component in a specific design layout.
TYPE	**HL**	'HL' defines the component to be a hyperlink.
NODEVALUE		
DEFAULTSTATICLABEL		
TOKEN1	P	Action ID.
TOKEN2	RRXXX619	Target ID
TOKEN3	RRXXX62	Request ID
TOKEN4	p~RRXXX616	Token4 should be <type>~<tableID>.
TOKEN5		

Client side F-Lang::

<HL b="" tid="RRXXX619" t="P" g="Hyperlink" r="RRXXX62" tgtid="RRXXX619" actid="P" u="" dX="0.020000" dY="0.005000" dW="0.400000" dH="0.040000" dRX="0.580000" id=" RRXXX61fldlink" l="K_LINK" n="fldlink"/>

Where **HL** defines property of hyperlink.
All the other attributes are same as defined above.

WEBVIEW (W):

'W' type is used to get a webview component which can be manipulated using jsp's. In Leap you can select directly 'Data Type' and choose 'Webview'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label.
NAME	fldwebview	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldwebview	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	WebView	Used to draw the component in a specific design layout.
TYPE	**W**	'W' defines the component to be webview component.
NODEVALUE	**Case1:** string('https://10.180.58.246:7003/T001/static/RRPTF62.jsp') **Case2:** concat(string('https://10.180.58.246:7003/T001/static/RRSPA61temp.jsp?IDSESSION='),//faml/request/fldSessionId),string('&JSESSIONID='),//faml/request/fldjsessionid)	Enclose the path of the jsp to be called as shown. In case some fields are to be passed then follow the case 2 format to append the values.
DEFAULTSTATICLABEL	Not Required	

TOKEN1	n	**n**	Load request	
		l	Relative path	
		h	Html string	
		modal	Modal type	
		x	Jsp call	
		s	Appended with session data	

Client side F-Lang::

<W t="n" v="https://10.180.58.246:7003/T001/static/RRPTF62.jsp" dRX="-0.010000" dH="0.800000" dW="0.700000" dY="0.010000" dX="0.010000" n="fldgraph" g="WebView" id="RRSPA61fldgraph" bt="0"/>

Date picker (TD):

'TD' type is used to create a Date picker component. In Leap you can select directly 'Data Type' and choose 'Date picker'. In this date component, calendar comes on page load .Entries needed for this type is:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the date textbox.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value entered in textbox can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ListLabel	Used to draw the component in a specific design layout
TYPE	**TD**	'TD' defines the component to be Date picker.
NODEVALUE		NA

Client side F-Lang::

<I dRX="0.000000" dH="0.380000" dW="0.700000" dY="0.050000" dX="0.100000" g="ListLabel" s="" n="fldpaylaerdate" v="" l="" t="d" id="RRDTF61fldpaylaerdate" bt="0"/>

Date Textbox (MD):

'MD' type is used to create a date textbox. In Leap you can select directly 'Data Type' and choose 'Date Field'. In this component a textbox is shown along with calendar image and when user click on this image the calendar pops up . Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the date textbox.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value entered in textbox can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	Input,FBlack,FS18,TPadding	Used to draw the component in a specific design layout
TYPE	**MD**	'MD' defines the component to be date textbox.
NODEVALUE		NA

Client side F-Lang::

<I dRX="0.056000" dH="0.050000" dW="0.820000" dY="0.350000" dX="0.030000" g="Input,FBlack,FS18,TPadding" s="" m="N" n="fldstartdate" l="Start Date" t="c" id="RRREM62fldstartdate" bt="0" v=""/>

Password (P):

'P' type is used to create textbox for entering password. In Leap you can select directly 'Data Type' and choose ''Password''. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the Password textbox.

NAME	fldmessbull	Gives a name, usually unique in the screen. Value entered in textbox can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	Input,IPadding	Used to draw the component in a specific design layout
TYPE	**P**	Textbox for entering password.
NODEVALUE		Give the xpath for which you want to show value.
TOKEN1		Type.

Client side F-Lang::

<I dRX="0.010000" dH="0.080000" dW="0.990000" dY="0.100000" dX="0.000000" g="Input,IPadding" s="" n="fldnewpwd" m="N" l="New Password" t="p" id="RRCPW61fldnewpwd" bt="0"/>

Value Label (VL):

'VL' type is used to create a Label component. In Leap you can select directly 'Data Type' and choose ''Value label''. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label.
NAME	fldmessbull	Gives a name, usually unique in the screen.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ValueLabel,FShaded,FRight	Used to draw the component in a specific design layout
TYPE	**VL**	Create the label component.
NODEVALUE	//faml/detailsresponsedto/details/detaildto/days##//faml/detailsresponsedto/details/detaildto/months##//faml/detailsresponsedto/details/detaildto/year	Give the xpath for which you want to show value. '##' separated values

DEFAULTSTATICLABELS	K_DAYS~K_MONTHS~K_YEAR	'~' separated label for each value attached at the end of value.

Client side F-Lang::

<L lw="" lg="" t="" brpos="" bpos="" bd="" dRX="0.100000" dH="0.060000" dW="0.350000" dY="0.005000" dX="0.550000" g="ValueLabel,FShaded,FRight" **v="04 days 01 months 11 years"** id="RRTDF62flddepdate1" n="flddepdate1" bt="0"/>

TIMEZONEFORMATTED DATE (TZ):

'TZ' type is used to create a Time zone Formatted Date. In Leap you can select directly 'Data Type' and choose 'Time Zone'. Entries needed for this type are:

COLUMN	**EXAMPLE VALUES**	**VALUES DESCRIPTION**
LABEL	K_BLANK	This column is used to define the Label.
NAME	fldmessbull	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldmessbull	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ValueText (refer CSS doc)	Used to draw the component in a specific design layout.
TYPE	**TZ**	'TZ' defines the component to be Time zone Formatted Date.
NODEV	concat(//faml/response/sessioninfo/@identity, '.DATETIMEFORMAT')#//faml/response/authviewresponsedto/currents	Needed 4 arguments

ALUE	tatics/authorizationstatisticsdto/datecreated#//faml/response/*/extendedresponse/extendedresponsedto/entitytimezone#//faml/response/sessioninfo/@idlang	separated by '#' 1.Date Formate 2.Processing Date 3. Zone 4.Language ID

Client side F-Lang::

<L bt="0" id="RRVAT62fldcomp" l="Formatted Time Zone Date" v='12-12-2014 00:15:36' g="ValueText" dX="0.000000" dY="0.000000" dW="0.200000" dH="0.050000" dRX="0.000000" />

FORMATTED UNIT (FU):

'FU' type is used to create a formatted unit. In Leap you can select directly 'Data Type' and choose 'Format Units'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_BLANK	This column is used to define the Label of the field.
NAME	fldfxrate	Gives a name, usually unique in the screen.
ID	fldfxrate	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	**FU**	'FU' defines the component to be formatted unit .
NODEVALUE	string('USD')#//faml/response/exchangerateinquiryresponsedto/exchangerates/exchangeratedto/cashsellrate#//faml/response/sessioninfo/@idlang	Given node value needs following arguments: 1. Name Format 2. Value 3. Id Lang

Client side F-Lang::

<L bt="0" id="RRVAT62fldcomp" l="Formatted Amount" v='45.00~GBP' g="ValueText" dX="0.000000" dY="0.000000" dW="0.200000" dH="0.050000" dRX="0.000000" />

Drop Down (D):

'D' type is used to create a segmented button. In Leap you can select directly 'Data Type' and choose 'Drop Down'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_FREQUENCY	This column is used to define the Label of the Drop Down. Label will appear at the first position of the Drop Down in the form of Hint text. If LABEL is not provided, then first value (first index value) coming in NODEVALUE will be displayed.
NAME	fldfreq	Gives a name, usually unique in the screen. Value selected on Drop Down can be obtained on next screen using this name, also the value displayed after selection is passed automatically on the next screen by a unique name which appears as DropdownName_txt ,i.e., in this case it will be fldfreq_txt.
ID	fldfreq	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component. The Component Name and ID can be same.
NODEVALUE	//faml/response/ datadto~desc~code	Value after first ~ is the values which will appear once the Drop Down is clicked. Values after second ~ are the values which will be passed on the next screen, if more than one value needs to be passed then they need to be separated by comma. E.g. value1, value2 and so on. If the dropdown options need to be filtered for some specific value, then

<table>
<tr><td></td><td></td><td>that value should be given after third '~'.
Thus the node value can be described as:
" XPath~ display value~ passing value ~filter value "</td></tr>
<tr><td>TOKEN1</td><td>V</td><td>Action ID.</td></tr>
<tr><td>TOKEN2</td><td>RRALR615</td><td>Target ID.</td></tr>
<tr><td>TOKEN3</td><td>RRALR66</td><td>Request ID.</td></tr>
<tr><td>TOKEN4</td><td>Y/M/D or any dynamic value.</td><td>TYPE. Defines the type of data need to be shown in Drop Down.The possible values for Type are:
<table>
<tr><td>Type</td><td>Description</td></tr>
<tr><td>Y</td><td>Displays data of Year format.</td></tr>
<tr><td>D</td><td>Displays data of Year format.</td></tr>
<tr><td>M</td><td>Displays data of Month format.</td></tr>
<tr><td>fr</td><td>It populates a new request once the dropdown is loaded, it needs either some Behavior to fire the Request or the request Id must be specified in Token3.</td></tr>
<tr><td>Currency</td><td>Display data with Currency.</td></tr>
<tr><td>NFC</td><td>For NFC related screens, the</td></tr>
</table></td></tr>
</table>

			type must be specified as NFC. It specifies the maximum limit time to start the scan.	
TOKEN5	str:split(//faml/request/fldalerts,"~")[10]	VALUE Index.		

Client side F-Lang::

<S dRX="0.640000" dH="0.050000" dW="0.135000" dY="0.264000" dX="0.015000" g="DropDown" c="" i="" m="N" n="fldaddparam" l="Frequency" r="RRALR66" tgtid="RRALR615" actid="" id="RRALR65fldfreq" bt="0" rd="false" d="dp" vi="W" t="">

<O b="" og=" DropDownCellLabel" l="Frequency"/>

<O b="" v="D" l="Daily" f=""/>

<O b="" v="W" l="Weekly" f=""/>

<O b="" v="M" l="Monthly" f=""/>

<O b="" v="Y" l="Yearly" f=""/>

<O b="" v="JF" l="callForexAlertWindow" f=""/>

</S>

Where **S** defines property of Drop Down in general & **O** is for individual options which will appear in the Drop Down.
D: g= Dataclass, t= type= index, vi= VALUE Index, tgtid=Target ID, actid=Action ID, r=Request ID;

O: b=behavior, v=passing value, l=label displayed=filter value;

All the other attributes are same as defined above.

Static Drop Down (D1):

'D1' type is used to create a Static Drop Down. In Leap you can select directly 'Data Type' and choose 'Static Drop Down'. In case of Static Drop Down the values to be populated in options of Drop Down are pre-known. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_FREQUENCY	This column is used to define the Label of the Drop Down. Label will appear at the first position of the Drop Down in the form of Hint text. If LABEL is not provided, then first value (first index value) coming in NODEVALUE will be displayed.
NAME	fldfreq	Gives a name, usually unique in the screen. Value selected on Drop Down can be obtained on next screen using this name, also the value displayed after selection is passed automatically on the next screen by a unique name which appears as DropdownName_txt ,i.e., in this case it will be fldfreq_txt.
ID	fldfreq	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component. The Component Name and ID can be same.
NODEVALUE	string('RTD')~string('TDF')	'~' separated values (equal to the labels defined in DEFAULTSTATICLABEL). If label is given then, it is the default display of static dropdown else the first value id default display of Static labels.
DEFAULTSTATICLABEL	K_REDEEMTD~K_VIEWALL	Defines the total number of options in the D1. And gives name to each one.

TOKEN1	V	Action ID.
TOKEN2	RRALR615	Target ID
TOKEN4	RRALR66	Request ID.
TOKEN5	String('RTD')	VALUE Index.

Client side F-Lang:

<S dRX="0.580000" dH="0.043000" dW="0.220000" dY="0.003000" dX="0.200000" g="Dropdown" c="" m="N" n="fldddheader_ato" vi="" l="Open Term Deposit" id="RRATO61fldddheader_ato" i="" bt="0" rd="false" d="dp">

<O b="" og="" l="Open Term Deposit" f=""/>

<O r="" tgtid="" actid="" v="RTD" l="Redeem Term Deposits" f=""/>

<O r="" tgtid="" actid="" v="TDF" l="View All" f=""/>

</S>

Where **S** defines property of Drop Down in general & **O** is for individual options which will appear in the Drop Down.

S: g= dataclass, t= type,i= index, vi= valueindex, tgtid=TargetID, actid=ActionID, r=RequestID;

O: b=behavior, v=passing value, l=label displayed=filter value.

All the other attributes are same as defined above.

Drop Down Complex (SD):

'SD' type is used as Search Drop Down. The DropDown Complex consists of Input Box with list of options. User can either enter the option or can select the option from the list. In case of Dropdown Complex, more than one value can be displayed at a time. In Leap you can select directly 'Data Type' and choose 'Search Dropdown'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION

LABEL	K_FREQUENCY	This column is used to define the Label of the Drop Down. Label will appear at the first position of the Drop Down in the form of Hint text. If LABEL is not provided, then first value (first index value) coming in NODEVALUE will be displayed.
NAME	fldfreq	Gives a name, usually unique in the screen. Value selected on Drop Down can be obtained on next screen using this name, also the value displayed after selection is passed automatically on the next screen by a unique name which appears as DropdownName_txt ,i.e., in this case it will be fldfreq_txt.
ID	fldfreq	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component. The Component Name and ID can be same.
LABELCLASS	VerifyLblTextPad~VerifyLblTextPad/ lbl1~lbl2 (refer CSS doc)	Define the styles for each value that are displayed when SD is loaded or clicked. The class for each label is provided individually by separating with '~', if more than one label is to be displayed.
LABELWIDTH	0.28~0.28#0.04~0.04	Label width provides height as well as the width for each label displayed in SD. Values before '#' defines the width section and values after '#' defines the Height section. i.e. 0.28~0.28 → width section and 0.04~0.04 → height section '~' separated value defines the width of each label individually and similar is the case of height.

TOKEN1	V	Action ID.
TOKEN2	RROAT612	Target ID.
TOKEN 3	0.08	Option Height, it is the optional height that specifies the height of the list containing label in SD, when it is loaded initially.
TOKEN4	RROAT62	Request ID.
TOKEN5	R~1	Token 5 is used differently in diff cases. **Case 1**: When SD needs to be used normally, then TOKEN 5 is used as the TYPE for the list which comes in SD. It defines how the data in list need to be displayed. The value given is : R~1, Where R-describes the type and value after '~' describes the number of columns to be displayed when SD is loaded initially. The Types available are: **R –** General Type in which initially first row is shown in list and after the click rest are shown and selected. **A –** This type displays all the rows once the screen is loaded. On the click of the list only selection is done. **Case 2**: By Default the component SD contains a search Input box and a sort button beside it. But if the button need to fire some request

		then, token 5 is used in following manner: Firstly the type mentioned in case 1 is given, followed by '^' symbol and following is the RequestID that need to be fired on the click of sort Button. E.g. A~1^RRBTG61

In case of Search Dropdown, the options and labels are treated individually. Options will contain the values which need to be passed to next screen and labels with contain the value which will be displayed. This will be clearer with client side F-lang.

Client side F-Lang::

<D dRX="0.010000" dH="0.573000" dW="0.320000" dY="0.020000" dX="0.010000" g="FilterBox" m="Y" t="R" id="RROAT61flddestacctno" bt="1" n="flddestacctno">

<I dRX="0.0" dH="0.06" dW="0.275" dY="0.01" dX="0.005" g="FilterInput" l="Select To Account" n="flddestacctnoinput" v="" s="" a="" k="" r="false"/>

<IB dRX="0.0" tgtid="" dH="0.06" dW="0.04" dY="0.01" dX="0.278" bgi="sort" g="ImageButton"/>

<LO dRX="0.0" dH="Match" dW="0.320000" dY="0.08" dX="0.005" og="ListItemBand" g="FilterBox" i="" oh="" vi="" r="" tgtid="" actid="" nc="1" t="R" n="flddestacctno" rd="" id="RROAT61flddestacctno" bt="">

<O v="000003171225~0.000000~GBP~000003171~000~GBP~C~0xxx03xxx225~0xxx03xxx225" f="">

<L dH="" dW="" g="VerifyLblTextPad" v="0xxx03xxx225"/>

<L dH="" dW="" g="VerifyLblTextPad" v="0xxx03xxx225"/>

</O>

</LO>

</D>

Where **D** defines property of Drop Down in general , **I** is for Search Textbox which will appear at the top of Complex dropdown, **IB** is for the Image button used for sorting present beside the Textbox, **LO** in general is for List option, **O** describes the option of the List, **L** is for the Label displayed .

D: g= dataclass before '~';m=is mandatory;bt=step number

I: l=label displayed;

LO: t=type, nc=no. of columns, og=dataclass after '~', g=dataclass before'~';

L: g=label class,v=value displayed;

All the other attributes are same as defined above.

POPUP BUTTON (PB):

'PB' type is used to create a popup region. In Leap you can select directly 'Data Type' and choose 'Popup Button'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_PAY_ON	This column is used to define the Label of the button.
NAME	fldsubmitlater	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldsubmitlater	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	"Button","PayOn" "BSubmit,FWhite,FS20,FBold" (refer CSS doc)	Used to draw the component in a specific design layout. The button will be shown using the given CSS
TYPE	**PB**	'PB' defines the component to be Pop-up button.
TOKEN1	RRITG6121	It describes the tableid which is displayed inside the pop-up. There is a special Case in this when used for iPhones: 1) Requestid~act : Eg: RRXXX61~act where act is passed as type that recognizes the pop-up action stylesheet for button. 2) Requestid~action: Eg: RRXXX61~action where action is passed as type that recognizes the pop-up

		action stylesheet for image button.
TOKEN2	paylater	Name of the Image to be displayed in background of the button
TOKEN3	0.9	Height of the popup
TOKEN4	1.000	Width of the popup
TOKEN5	0.00~0.20	Represents the XY position of the popup being displayed. Xposition~Yposition

Client side F-Lang::

<PB t="" pW="1.000" pH="0.90" pi="paylater" tid="RRITG6121" dRX="0.080000" dH="0.100000" dW="0.430000" dY="0.010000" dX="0.490000" g="BSubmit,FWhite,FS20,FBold" n="fldsubmitlater" s="def" l="Pay On" id="RRITG61fldsubmitlater" pY="0.20" pX="0.00" bt="4" bgi="" rd=""/>

Where **PB** defines property of button in general & **Q** is for individual segments.
SB: rd= readonly; g= dataclass, cg=cell dataclass, t= type,i= index, vi= valueindex;
All the other attributes are same as defined above.

WIDGET BUTTON (WB):

'WB' type is used to create a widget button. In Leap you can select directly 'Data Type' and choose 'Widget Buttons'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_LABEL	This column is used to define the Label of the Widget Button.
NAME	fldwidgetbtn	Gives a name, usually unique in the screen.
ID	fldwidgetbtn	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	WidgetButtonVertical	Used to draw the component in a specific design layout.
TYPE	**WB**	'WB' defines the component to be widget button.
IMAGE	analysis_icon	Image name.
TOKEN1	U	Content-ID.

TOKEN2	RRATO61	REQUESTID.
TOKEN3	P	Type (It can be '~' separated).
TOKEN4	widget1	Widget ID.(Widget contents to be render on this widget.)
TOKEN5	1# 0.22# 0.12	Badge (It should be '#' separated). Badgevalue#badgeposition#(badgeposition in case of RTL).

Client side F-Lang::

<WB brpos="0.22" bpos="0.12" bd="1" dRX="0.580000" dH="0.040000" dW="0.400000" dY="0.005000" dX="0.020000" r=" RRATO61" g=" WidgetButtonVertical " cid="U" bgi=" analysis_icon " wid="widget1" l="K_BUTTON_LABEL" p="" t="P" id="RRQKT61fldwidgetbtn"/>

Where **WB** defines property of button in general.
WB: cid=Content Id; g= dataclass; wid='Widget ID; bd=Badge Value; bpos=Badge position
All the other attributes are same as defined above.

IMAGE BUTTON (IB):

'IB' type is used to create an Image Button. In Leap you can select directly 'Data Type' and choose 'Image Button'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_ERECEIPT	This column is used to define the Label of the button.
NAME	fldereciepts	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldereciepts	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.

DATACLASS	“ImageButton”,”transparentButton” “ConfirmButton,SaveButton” (refer CSS doc)	Used to draw the component in a specific design layout. The button will be shown using the given CSS
TYPE	**IB**	‘IB’ defines the component to be Image button.
NODEVALUE	p~RRPDF01	Type~REQUESTID. ‘~’ separated values for type of image button and the request id to be fired.
TOKEN1	P	Action ID.
TOKEN2		TargetID.
TOKEN3		Value if any to be passed to with the component.
TOKEN5	1# 0.22# 0.12	Badge (It should be ‘#’ separated). Badgevalue#badgeposition#(badgeposition in case of RTL).
IMAGESRC	viewall	Image that has to be displayed on the screen which was already save from the client end.

Client side F-Lang::

<IB b="" brpos="" bpos="" bd="" g="transparentButton" bgi="viewall" dX="0.596000" dY="0.000000" dW="0.037000" dH="0.064000" tgtid="" actid="" dRX="0.067000" r="RRSPA67" t="s" id="RRSPA61fldviwtrnscicon" bt="0" v="RRSPA67" rd="false"/>

Where **IB** defines property of button in general & **Q** is for individual segments.
IB: rd= readonly; g= dataclass, cg=cell dataclass, t= type,i= index, vi= valueindex;
All the other attributes are same as defined above.

CONTACT LIST (CL):

‘CL’ type is used to create a Contact List from Mobile/Tablet/iPad Device. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION

LABEL	K_RECEIVERNAME_OR_NUMBER	This column is used to define the Label for Contact List.
NAME	fldcontactdtlvalue	Gives a name, usually unique in the screen.
ID	fldcontactdtlvalue	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	FilterBox~ListItemBand (refer CSS doc)	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass
TYPE	**CL**	'CL' defines the component to be Contact List.
NODEVALUE	C#name~email~number#number	'#' separated values indicates values to be displayed[first set of values;'~' separated] & values passed while submitting request["number" only in this example]
DEFAULTSTATICLABEL	K_RECEIVERNAME_OR_NUMBER	Default static label.
TOKEN1	Y~0.06~0.06~L	Use to show image before every contact in list. The ~ separated values are [from left to right] 1. Has Picture –Y/N 2. Width 3. Height 4. Orientation- L/R
TOKEN2	C	This "C" parameter need to provide to fetch Contact List from Mobile Device.
TOKEN5	3	Number Of Fields to be displayed on Screen.

Client side F-Lang::

<CL dRX="0.017000" dH="0.540000" dW="0.966000" dY="0.022000" dX="0.010000" g="FilterBox" n="fldcontactdtlvalue" t="C" m="N" id="RRPPP66flddecontactdetail" bt="0">

<I dRX="0.0" dH="0.06" dW="0.805" dY="0.0" dX="0.0" g="FilterInput" l="Receiver Name Or Number:" n="fldcontactdtlvalueinput" v="" s="" a="" k="" r="false"/>

<IB dRX="0.0" tgtid="" dH="0.06" dW="0.151" dY="0.0" dX="0.8150000000000001" bgi="sort" g="ImageButton"/>

<CLO b="set{NUMBER@RRPPP61fldcontactdtlvalue;NAME@RRPPP61fldHName}" pO="L" pH="0.06" pW="0.06" hP="Y" v="number" seq="name~email~number" id="RRPPP66flddecontactdetailvalue" n="fldcontactdtlvalue" t="C" nc="3" oh="" og="ListItemBand" g="FilterBox" dRX="0.017000" dH="0.48510000000000003" dW="0.966000" dY="0.054900000000000004" dX="0.000">

<O>

<L dH="0.04" dW="0.50" g="ValueText,FLeft"/>

<L dH="0.04" dW="0.50" g="ValueText,FLeft"/>

<L dH="0.04" dW="0.50" g="ValueText,FLeft"/>

</O>

</CLO>

</CL>

VERIFY FIELD (V):

'V' type is used to create a label. It is generally used in case of Verify & Confirm Screen. In Leap you can select directly 'Data Type' and choose 'Verification Data'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_EXAMPLE	Label to be given. Is displayed only if there is no value in the Nodevalue
NAME	fldname	Gives a name, usually unique in the screen.
ID	fldname	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ValueLabel,FLeft	Used to draw the component in a specific design layout.
TYPE	**V**	'V' defines the component to be verification field.
NODEVALUE	1. //faml/request/fldName **OR** 2. String('%%K_NAME%%')	1. XPath 2. Hardcoded String value

Client side F-Lang::

<L lw="" lg="" t="" brpos="" bpos="" bd="" dRX="0.232000" dH="0.050000" dW="0.719000" dY="0.025000" dX="0.009000" g="ValueLabel,FLeft" v="" id="RRPPP62fldname" n="fldname" bt="0"/>

Here **V** data type is changed to **L** tag while generating f language.

PASSWORD STRENGTH (PST):

'PST' type is used to find the strength of the password. This data type need to be coupled with Input Box/Text Box to identify the strength of the password entered in Input field. To couple this data type with Input Field/Text box, you need to mention "pst" in token5 of the Input Field/Text Box. The possible results are – weak, better, medium, strong, very strong. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_PASSWORD	This column is used to define the Label.
NAME	fldstrength	Gives a name, usually unique in the screen.
ID	fldstrength	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	PasswordStrengthTable	Used to draw the component in a specific design layout.
TYPE	**PST**	'PST' defines the component to be segmented button.
NODEVALUE	N/A	N/A
DEFAULTSTATICLABEL	K_PASSWORD	Default Static Label

Client side F-Lang::

<I b="" t="pst" dRX="0.113000" dH="0.050000" dW="0.217000" dY="0.226000" dX="0.010000" p="" g="Input" rd="false" k="Input" a="Login Password" s="" v="" m="Y" n="fldppassword" l="Login Password" id="RRSBC62fldppassword" bt="2" dv="" dt=""/>

<PST dRX="0.018000" dH="0.200000" dW="0.200000" dY="0.190000" dX="0.229000" lg="VerifyLblTextPad" g="PasswordStrengthTable" n="fldstrength" l="Password" id="RRSBC62fldstrength"/>

VALUE SELECTOR (VS):

'VS' type is used for creating Value Selector. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_GROSS_INCOME	This column is used to define the Label.
LABELCLASS	ValueLabel~Input~ValueLabel~LoginSeekBar	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's label class
NAME	fldgrossincome	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldgrossincome	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	Table	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass
TYPE	**VS**	'VS' defines the component to be value selector
LABELCLASS	ValueLabel~Input~ValueLabel~LoginSeekBar	To define the CSS for various Components inside VS. Lbael~Inputbox~Label~SliderButton
DEFAULTVALUE	java:com.iflex.fcat.infra.JFFormatter.formatCurrency('INPCUR',//faml/response/prepareloancalculatorresponsedto/minmonthlyexpense,/faml/request/fldLangId)	To define the default value of the value selector.
TOKEN1	java:com.iflex.fcat.infra.JFFormatter.formatCurrency(/faml/response/prepareloancalculatorresponsedto/currency/currencydto/codcurrency,//faml/response/prepareloancalculatorresponsedto/minmonthlyexpense,/faml/request/fldLangId)	Number of Decimal Places allowed is defined using this token.
TOKEN2	//faml/response/prepareloancalculatorresponsedto/minmonthlyexpense~//faml/response/prepareloancalculatorresponsedto/maxmonthly	Range is defined with '~' separated values. StartRange~EndRange

	expense	
TOKEN3	/faml/response/prepareloancalculatorresponsedto/currency/currencydto/codcurrency	Defines the Default Currency of the VS
TOKEN4	N	Defines the Text type of the input. Refer Textbox.
TOKEN5	R	Type. Its values can be R – For Integer Values C – For Decimal Values

Client side F-Lang::

<**VS** d="0.00" g="Table" dX="0.010000" dY="0.010000" dW="0.950000" dH="0.200000" dRX="0.040000" id="RRLEC61fldexpense" n="fldexpense">

<L v="Expenses" l="Expenses" dRX="0.0" dH="
			0.12
			" dW="0.57" dY="0.01" dX="0.01" g="ValueLabel"/>

<I s="15" p="D" n="fldexpense" v="0.00" dRX="0.0" dH="0.06" dW="0.285" dY="0.01" dX="0.62" g="Input"/>

<L v="GBP" t="a" dRX="0.0" dH="0.06" dW="
			0.04
			" dY="0.01" dX="0.58" g="ValueLabel"/>

<SL dr="1000000.000000" sr="0.000000" dRX="0.0" dH="0.08" dW="0.9309999999999999" dY="
			0.12
			" dX="0.01" g="LoginSeekBar"/>

</**VS**>

Where **VS** defines property of Value Selector in general .
All the other attributes are same as defined above.

CONFIRM BUTTON (B):

'B' type is used to create a submit button. In Leap you can select directly 'Data Type' and choose 'Button'. Entries needed for this type are:

COLUMN	**EXAMPLE VALUES**	**VALUES DESCRIPTION**

LABEL	K_SUBMIT	This column is used to define the Label of the button.
NAME	fldsubmit	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldsubmit	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	ConfirmButton	Used to draw the component in a specific design layout. Value before ~ is dataclass and after ~ is individual cell's dataclass
TYPE	**B**	'B' defines the component to be Submit button.
NODEVALUE	s~RRADT02	Type~REQUESTID. '~' separated values for type of image button and the request id to be fired.
DEFAULTSTATICLABEL	K_SUBMIT	Label for button.
TOKEN1	P	ACTIONID.
TOKEN2		TARGETID.
TOKEN3	uparrow	IMAGE to be displayed on the button along with Text, if needed.
TOKEN4	fldpaymode=L--	Name and value to be passed. Name=Value--Name2=Value2--type.
TOKEN5	flddoctype	REFERENCE NUMBER to be passed If any.

Client side F-Lang::

<B ref="" pi="" tgtid="" actid="" dRX="0.070000" dH="0.070000" dW="0.200000" dY="0.130000" rd="false" dX="0.330000" g="ConfirmButton" a="ConfirmButton" s="def" r="RRDTF62" t="s" l="Deposit to Account" id="RRDTF61flddepacc" n="flddepacc" pr="" h="" bt="0" b="set{1@flddeliverymode_ix}"/>

Where **B** defines property of button in general .

B: rd= readonly; g= dataclass, cg=cell dataclass, t= type,i= index, vi= valueindex;

All the other attributes are same as defined above.

STATIC RADIO (SR):

'SR' type is used to create an Static Radio Button. In Leap you can select directly 'Data Type' and choose 'Static Radio'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldnatclearingcodetype	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldbenenatnlclrcodtype11	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	"MRadio~Radio1"," Radio~Radio1" "RadioContentStyle~RadioVertical," (refer CSS doc)	Used to draw the component in a specific design layout. '~' separated value for Radio button and its individual segments
TYPE	**SR**	'SR' defines the component to be Static button.
NODEVALUE	string('T')~string('P')	'~' separated values (equal to the labels defined in DEFAULTSTATICLABEL). If label is given then, it is the default display of static radio else the first value is default display of Static labels.
TOKEN1	V	Action ID.
TOKEN2		Target ID.
TOKEN3		VALUE Index.
TOKEN4	RROAT61	Request ID.

TOKEN5	H	Type. H – Horizontal V – Vertical
DEFAULTSTATICLABELS	K_TENURE~K_MATURITYDATE	The name of the Radio Button to be display on the screen.

Client side F-Lang::

<R dRX="0.043000" dW="0.945000" dH="0.060000" dY="0.010000" dX="0.012000" lg="" cg="Radio1" g="MRadio" t="" n="fldpattern" i="" id="RRATO61fldpattern" bt="1" vi="">

<Q r="" tgtid="" actid="" b="vis{h@RRATO616;h@RRATO616}" l="Single" v="0" n="fldpattern1"/>

<Q r="" tgtid="" actid="" b="vis{s@RRATO616;s@RRATO616}" l="Joint" v="1" n="fldpattern2"/>

</R>

Where **R** defines property of static Radio Button in general & **Q** is for individual segments.
SR: g= dataclass, cg=cell dataclass, t= type,i= index, vi= valueindex, tgtid=TargetID, actid=ActionID,b=behaviour;
All the other attributes are same as defined above.

QR CODE (QR):

'QR' type is used to create an QR Code. In Leap you can select directly 'Data Type' and choose 'QR Code'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	qrcode	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	qrcode	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.

DATACLASS	“TTransparent~ValueLabel~ValueLabel”, “TransparentTable~LabelTextWhiteBold~ LabelTextWhiteBold” (refer CSS doc)	Used to draw the component in a specific design layout.
TYPE	**QR**	‘QR’ defines the component to be QR Code.
TOKEN1	Y	Is Encoded. Possible Values are Y/N.
TOKEN2	QR	Type. It defines the type here that must be required to paint the QR Code QR – For QR code generation NFC – For NFC type transfer
TOKEN3	100	Time given for the device to establish connection

Client side F-Lang::

<QR b="" e="" t="QR" dRX="0.000000" dH="0.480000" dW="1.000000" dY="0.020000" dX="0.000000" g="TTransparent~ValueLabel~ValueLabel" n="qrcode" id="RRQRC66qrcode"/>

Where **QR** defines property of QR Code in general.
QR: e=encode,g= dataclass,t= type;
All the other attributes are same as defined above.

DYNAMIC RADIO (DR):

‘DR’ type is used to create an Dynamic Radio Button. In Leap you can select directly ‘Data Type’ and choose ‘Dynamic Radio Button Group’. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
LABEL	K_FREQUENCY	This column is used to define the Label.

<table>
<tr><td>NAME</td><td>fldsiexecutionfreq</td><td>Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.</td></tr>
<tr><td>ID</td><td>fldsiexecutionfreq</td><td>Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.</td></tr>
<tr><td>DATACLASS</td><td>“Dropdown”,” Radio~Radio1”
“RadioTableVertical~Radio” (refer CSS doc)</td><td>Used to draw the component in a specific design layout. ‘~’ separated value for Radio button and its individual segments</td></tr>
<tr><td>TYPE</td><td>DR</td><td>‘DR’ defines the component to be Dynamic Radio button.</td></tr>
<tr><td>NODEVALUE</td><td>//faml/response/genericpaymentresponsedto/responsedata/fundtransferresponsedto/frequency/datamapdto~description~datavalue</td><td>The values after first ‘~’ are the The values after the second ‘~’ are the values that need to be passed on next screen. If more than one value need to be passed then it has to be separated by ‘,’
Thus the node value can be described as:
“ XPath~ display value~ passing value “</td></tr>
<tr><td>TOKEN1</td><td>V</td><td>Action ID.</td></tr>
<tr><td>TOKEN2</td><td></td><td>Target ID.</td></tr>
<tr><td>TOKEN3</td><td></td><td>VALUEINDEX</td></tr>
<tr><td>TOKEN4</td><td>RROAT62</td><td>Request ID.</td></tr>
<tr><td>TOKEN5</td><td>H</td><td>TYPE.<table><tr><td>H</td><td>Horizontal alignment</td></tr><tr><td>V</td><td>Vertical alignment</td></tr></table></td></tr>
</table>

Client side F-Lang::

<R dRX="0.020000" dH="0.065000" dW="0.370000" dY="0.070000" dX="0.110000" lg="" cg="Radio1" g="Radio" i="" m="N" n="fldaccttransinstruction" l="" id="RRATO61fldaccttransinstruction" bt="0" tgtid="" actid="" r="" t="" vi="">

<Q b="checkVisibiltyCallVis{RRATO61fldaccttransinstruction^s@RRATO6111;h@RRATO6112;h@RRATO6113}" v="1~Own Account" l="Own Account"/>

<Q b="checkVisibiltyCallVis{RRATO61fldaccttransinstruction^s@RRATO6112;h@RRATO6111;h@RRATO6113}" v="2~Internal Bank Account" l="Internal Bank Account"/>

<Q b="checkVisibiltyCallVis{RRATO61fldaccttransinstruction^s@RRATO6113;h@RRATO6111;h@RRATO6112}" v="3~Use Domestic Network" l="Use Domestic Network"/>

</R>

Where **R** defines property of Radio Group in general & **Q** is for individual Radio Buttons.
DR: g= dataclass, cg=cell dataclass, t= type, vi= valueindex;
All the other attributes are same as defined above.

STATIC CHECKBOX (SC):

'SC' type is used to create an Static CheckBox. In Leap you can select directly 'Data Type' and choose 'Check Box'. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldchecktnc	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldchecktnc1	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.

<table>
<tr><td>DATACLASS</td><td>“Radio~Radio1”,”
VerifyTable~Table”
“CheckBoxstyle~CheckBoxVertical”
(refer CSS doc)</td><td>Used to draw the component in a specific design layout. ‘~’ separated value for Checkbox and its individual segments</td></tr>
<tr><td>TYPE</td><td>SC</td><td>‘SC’ defines the component to be Static CheckBox.</td></tr>
<tr><td>NODEVALUE</td><td></td><td>‘~’ separated values (equal to the labels defined in DEFAULTSTATICLABEL). If label is given then, it is the default display of static checkbox else the first value is default display of Static labels.</td></tr>
<tr><td>TOKEN1</td><td>2</td><td>It defines the number of columns.</td></tr>
<tr><td>TOKEN2</td><td>0.080</td><td>It defines the row height.</td></tr>
<tr><td>TOKEN3</td><td></td><td>VALUE Index. If the requirement needs to display some specific value once the dropdown is painted, then that value is given as VALUE Index.</td></tr>
<tr><td>TOKEN5</td><td>v</td><td>Type.<table><tr><td>H</td><td>Horizontal Alignment</td></tr><tr><td>V</td><td>Vertical Alignment</td></tr><tr><td>AF</td><td>Add Favourite</td></tr><tr><td>LF</td><td>List</td></tr><tr><td>EF</td><td>Edit</td></tr></table></td></tr>
</table>

Client side F-Lang::

<CG pv="" cH="0.06" dRX="0.010000" dW="0.980000" dH="0.070000" dY="0.010000" dX="0.010000" lg="ValueLabel" cg="Radio1" g="Radio" t="h" n="fldchecktnc" i="" id="RRORG63fldchecktnc" bt="0" nc="1">

<CB b="" l="I accept Terms and Conditions" v="1"/>

</CG>

Where **CG** defines property of button in general & **CB** is for individual segments.
CG: lg=label, g= dataclass, cg=cell dataclass, t= type,i= index,b=behaviour;
All the other attributes are same as defined above.

DYNAMIC CHECKBOX (DCB):

'DCB' type is used to create an Dynamic Check Box. In Leap you can select directly 'Data Type' and choose 'Check Box. Entries needed for this type are:

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldcontactmodes	Gives a name, usually unique in the screen. Value selected on button can be obtained on next screen using this name.
ID	fldcontactmodesP	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
DATACLASS	"Radio~Radio1"," Table~Table" "CheckBoxstyle~NonSeparatorTable" (refer CSS doc)	Used to draw the component in a specific design layout. '~' separated value for Checkbox and its individual segments
TYPE	**DCB**	'DCB' defines the component to be Dynamic Check Box.
NODEVALUE	//faml/response/preparemanageprofileresponsedto/contactmodes/datadto~description~code	The values after first '~' are the values that are to be displayed..The values after the second '~' are the values that need to be passed on next screen. If more than one value need to be passed then it has to be separated by ',' Thus the node value can be described as: " XPath~ display value~ passing value "
LABELCLASS	CheckBoxLabel1	It is used to paint the label of Dynamic CheckBox in different style, so that the options appear in one form and label appear as hint text.

<table>
<tr><td>TOKEN1</td><td>2</td><td>It defines the number of column.</td></tr>
<tr><td>TOKEN2</td><td>0.080</td><td>It defines the row height.</td></tr>
<tr><td>TOKEN3</td><td>/faml/response/preparemanageprofileresponsedto/extendedresponse/extendedresponsedto/user/userdto/prefcontactmode/datadto~code</td><td>VALUE Index.</td></tr>
<tr><td>TOKEN5</td><td>VD</td><td>Type.<table><tr><td>VD</td><td>On long press of the Dynamic CheckBox will fire the RequestId.</td></tr><tr><td>g</td><td>Used in case of IOS devices.</td></tr></table></td></tr>
</table>

Client side F-Lang::

<CG pv="" cH="0.08" lw="" dRX="0.030000" dH="0.080000" dW="0.940000" dY="0.010000" dX="0.030000" lg="ValueLabel" cg="Radio1" g="Radio" i="" m="N" n="fldcontactmodes" l="" id="RRMPR61fldcontactmodesP" bt="0" nc="2" t="">

<CB b="" v="E" l="Email"/>

<CB b="check(vis{s@RRMPR6126})uncheck(vis{h@RRMPR6126})" v="M" l="Mobile"/>

</CG>

Where **CG**defines property of button in general & **CB** is for individual segments.
DCB: g= dataclass, cg=cell dataclass, t= type,i= index, b= behavior,l='Label';
All the other attributes are same as defined above.

LOCATION (LOC):

'LOC' type is used to show the current Location of the device.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.

ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	LOC	'LOC' is used to get the current latitude and longitude position of the device being used.
TOKEN1	/faml/response/devicepositionresponsedto/extendedresponse/extendedresponsedto/latitude	Latitude position of the device
TOKEN2	/faml/response/devicepositionresponsedto/extendedresponse/extendedresponsedto/longitude	Longitude position of the device

Client side F-Lang::

<loc lat="0" long="0" />

MENU TEMPLATE (CM):

'CM' type is used to show the Menu Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	CM	'CM' defined the component to render the Menu template on the dashboard screen.

Client side F-Lang::

<N xmlns:java="http://xml.apache.org/xslt/java" r="RRCPW61" l="Change Password" t1="N" t2="000" l2="Dashboard" l1="Customer Services"/>

<N xmlns:java="http://xml.apache.org/xslt/java" l1="Dashboard" l2="Dashboard" t2="000" t1="000" l="Products" r="RRPSC61"/>

<N xmlns:java="http://xml.apache.org/xslt/java" l1="Customer Services" l2="Dashboard" t2="000" t1="680" l="Loan Top Up Request" r="RRTOP61"/>

<Z xmlns:java="http://xml.apache.org/xslt/java" l="Current and Savings" r="RRADT61"/>

<Z xmlns:java="http://xml.apache.org/xslt/java" l="Open Term Deposit" r="RRATO61"/>

<Z xmlns:java="http://xml.apache.org/xslt/java" l="Deposit Redemption" r="RRRTD61"/>

<X xmlns:java="http://xml.apache.org/xslt/java" n="fldfav"/>

<X xmlns:java="http://xml.apache.org/xslt/java" n="fldnotify"/>

POLICY TEMPLATE (PP):

'PP' type is used to show the Policy Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	PP	'PP' defined the component to

		render the Policy template on the screen.

Client side F-Lang::

<PP lH="" lW="1.000000" dRX="0.000000" dH="0.350000" dW="1.000000" dY="0.060000" dX="0.000000" gl="ValueLabel" g="PasswordPolicy" n="loginpolicy" id="RRCPW61loginpolicy">

<L v="-Should be minimum 8 characters."/>

<L v="-Should be maximum 20 characters."/>

<L v="-Can contain lowercase alphabets."/>

<L v="-Can contain uppercase alphabets."/>

<L v="-Can contain numeric characters."/>

<L v="-Must contain one of the following as first character:"/>

<L v=" --Lowercase alphabets"/>

<L v=" --Uppercase alphabets"/>

<L v=" --Numeric characters"/>

<L v="-Must contain one of the following as last character:"/>

<L v=" --Lowercase alphabets"/>

<L v=" --Uppercase alphabets"/>

<L v=" --Numeric characters"/>

<L v="-Allowed special characters"/>

<L v="."/>

<L v="_"/>

<L v="-Can contain 5 successive characters."/>

<L v="-Can contain 5 repetitions."/>

<L v="-Password can not be same as last 10 password."/>

</PP>

SECURITY QUESTION TEMPLATE (SQ):

'SQ' type is used to show the Security Question Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	SQ	'SQ' defined the component to render the Security Question template on the screen.

Client side F-Lang::

<I v="3" n="fldtotalnodes" t="h"/>

<S og="" vi="" r="" dH="0.07" dW="1.000000" dY="0.01" dX="0.000000" g="Dropdown,SPadding" c="Security Question 1" i="" a="" m="Y" n="fldquestionset1" l="Security Question 1" id="RRUSQ61fldquestionset1" d="dp">

<O b="" l="Security Question 1"/>

<O b="" v="1~SECURITY_QUESTION2~What is your Mother's name?" l="What is your Mother's name?"/>

<O b="" v="1~SECURITY_QUESTION3~How many Brother's do you have?" l="How many Brother's do you have?"/>

<O b="" v="1~SECURITY_QUESTION4~How many Sister do you have?" l="How many Sister do you have?"/>

<O b="" v="1~SECURITY_QUESTION5~What is your date of birth?" l="What is your date of birth?"/>

<O b="" v="1~SECURITY_QUESTION7~How many pets you have?" l="How many pets you have?"/>

</S>

<I dRX="0" dH="0.06" dW="1.000000" dY="0.09" dX="0.000000" p="s" g="Input,IPadding" r="" k="Input,IPadding" a="" s="40" v="" m="Y" n="fldAns1" l="Enter Answer" id="RRUSQ61fldAns1"/>

SECURITY QUESTION VERIFY TEMPLATE (SV):

'SV' type is used to show the Security Question Verify Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	SV	'SV' defined the component to render the Security Question Verify template on the screen.

Client side F-Lang::

<L dH="0.07" dW="0.920" g="ValueText,FCenter" dY="0.020000000000000004" dX="0.000000" v="Please check security question and its answers properly. " id="RRUSQ62fldverifytext1" n="fldverifytext1"/>

<L dH="0.07" dW="0.920" g="ValueLabel,FLeft,FS16" dY="0.080000" dX="0.05" v="1. What is your Mother's name?" id="RRUSQ62fldquestion1" n="fldquestion1"/>

<L dRX="0.07999999999999996" dH="0.05" dW="0.920" g="ValueLabel,FLeft,FS16" dY="0.15000000000000002" dX="0.05" v="mom" id="RRUSQ62fldanswer1" n="fldanswer1"/>

<L dH="0.07" dW="0.920" g="ValueLabel,FLeft,FS16" dY="0.21000000000000002" dX="0.05" v="2. What is your father's name?" id="RRUSQ62fldquestion1" n="fldquestion2"/>

<L dRX="0.07999999999999996" dH="0.05" dW="0.920" g="ValueLabel,FLeft,FS16" dY="0.28" dX="0.05" v="dad" id="RRUSQ62fldanswer1" n="fldanswer2"/>

<L dH="0.07" dW="0.920" g="ValueLabel,FLeft,FS16" dY="0.34" dX="0.05" v="3. Which city you were born?" id="RRUSQ62fldquestion1" n="fldquestion3"/>

<L dRX="0.07999999999999996" dH="0.05" dW="0.920" g="ValueLabel,FLeft,FS16" dY="0.41000000000000003" dX="0.05" v="hospital" id="RRUSQ62fldanswer1" n="fldanswer3"/>

TRANSACTION DETAILS (AUTHORISATION) TEMPLATE (VAT):

'VAT' type is used to show the Authorization Template of transaction details. It is a template written that on being called fetches a specific value for specific user agent that is used at client end for transaction details screen.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	VAT	'VAT' defined the component to render the Transaction Details Authorization template on the screen.

Client side F-Lang::

It Checks if the idtxn is 'ACC or CHL' and gives f-lang accordingly.

1) If it is **'ACC'** i.e. Account closure then, it provides screen for data that is used for closure of account.

2) If it is **'CHL'** i.e. Credit Card Hotlisting then, it provides screen for data that will be used for hotlisting the credit card declared.

FCP POLICY TEMPLATE (FP):

'FP' type is used to show the FCP POLICY Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end for Force Change Password.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	FP	'FP' defined the component to render the FCP POLICY template on the screen.

Client side F-Lang::

<TV l=" Policy to be followed for password " t='c'/>

<L t='s' l="Policy to be followed for password"/>

<L v="Should be minimum 8 characters."/>

<L v="Should be maximum 20 characters."/>

<L v="-Can contain lowercase alphabets."/>

<L v="-Can contain uppercase alphabets."/>

<L v="-Can contain numeric characters."/>

<L v="-Must contain one of the following as first character:"/>

<L v=" --Lowercase alphabets"/>

<L v=" --Uppercase alphabets"/>

<L v=" --Numeric characters"/>

<L v="-Must contain one of the following as last character:"/>

<L v=" --Lowercase alphabets"/>

<L v=" --Uppercase alphabets"/>

<L v=" --Numeric characters"/>

<L v="-Allowed special characters"/>

<L v="."/>

<L v="_"/>

<L v="-Can contain 5 successive characters."/>

<L v="-Can contain 5 repetitions."/>

<L v="-Password cannot be same as last 10 password."/>

ACCOUNT TEMPLATE (AD):

'AD' type is used to show the Account Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end for User Accounts.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define

		behavior to the particular component.
TYPE	AD	'AD' defined the component to render the Account template on the screen.

Client side F-Lang::

It fetches data by the type of account needed. Example: if IDTXN='ASC' it fetches CASA accounts, if IDTXN='AST' it fetches Term Deposit accounts, & if IDTXN='ASL' it fetches Loan accounts.

F-lang obtained:
<S n='fldacctno' l='Select Account*:'>

<O l='Dxxx00xxx2014' v='DB10008262014~00008262~C~DB1'/>

<O l='Dxxx00xxx2025' v='DB10008262025~00008262~C~DB1'/>

<O l='Dxxx00xxx2036' v='DB10008262036~00008262~C~DB1'/>

</S>

ACCOUNT TEMPLATE (AD):

'AD' type is used to show the Account Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end for User Accounts.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	AD	'AD' defined the component to render the Account template on

		the screen.

Client side F-Lang::

It fetches data by the type of account needed. Example: if IDTXN=’ASC’ it fetches CASA accounts, if IDTXN=’AST’ it fetches Term Deposit accounts, & if IDTXN=’ASL’ it fetches Loan accounts.

F-lang obtained:
<S n='fldacctno' l='Select Account*:'>

<O l='Dxxx00xxx2014' v='DB10008262014~00008262~C~DB1'/>

<O l='Dxxx00xxx2025' v='DB10008262025~00008262~C~DB1'/>

<O l='Dxxx00xxx2036' v='DB10008262036~00008262~C~DB1'/>

</S>

USER POLICY TEMPLATE (UP):

‘UP’ type is used to show the USER POLICY Template. It is a template written that on being called fetches a specific value for specific user agent that is used at client end for User Policies.

COLUMN	EXAMPLE VALUES	VALUES DESCRIPTION
NAME	fldx	Gives a name, usually unique in the screen. Value selected on Conditional Field can be obtained on next screen using this name.
ID	fldx	Gives a unique id to the component in the screen. ID is used to define behavior to the particular component.
TYPE	UP	‘UP’ defined the component to render the User Policy template on the screen.

Client side F-Lang::

<PP lH="" lW="1.000000" dRX="0.000000" dH="0.350000" dW="1.000000" dY="0.060000" dX="0.000000" gl="ValueLabel" g="PasswordPolicy" n="loginpolicy" id="RRCPW61loginpolicy">

<L v="-User id should be minimum 8 characters."/>

<L v="-User id should be maximum 20 characters."/>

<L v="-User id can contain lowercase alphabets."/>

<L v="-User id can contain uppercase alphabets."/>

<L v="-User id can contain numeric characters."/>

<L v="-User id can contain special characters "/>

<L v="-User id should contain at least 1 Uppercase alphabets"/>

<L v="-User id should contain at least 1 Lowercase alphabets"/>

<L v="-User id should contain at least 1 Numeric characters"/>

<L v="-User id should contain at least 1 Special characters "/>

<L v="-User id must contain one of the following as first char:"/>

<L v=" --Uppercase alphabets"/>

<L v=" --Lowercase alphabets"/>

<L v=" --Numeric characters"/>

<L v="-Special characters"/>

<L v="-Must contain one of the following as last character:"/>

<L v=" --Lowercase alphabets"/>

<L v=" --Uppercase alphabets"/>

<L v=" --Numeric characters"/>

<L v="-Special characters"/>

<L v="-Allowed special characters"/>

<L v="."/>

<L v="_"/>

<L v="-User Id Can contain 5 successive characters."/>

<L v="-User Id Can contain 5 repetitions."/>

<L v="-Password can not be same as last 10 password."/>

</PP>