Oracle FLEXCUBE Direct Banking

mLEAP Framework Developer Guide for BB and J2ME

Release 12.0.3.0.0

Part No. E52543-01

April 2014

mLEAP Framework Developer Guide for BB and J2ME

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/

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

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

Structure

This document provides the structure of LEAP related tables and their purpose. It then lists all possible data types for mobile banking along with various configurations applicable for each of the data type.

Related Information Sources

Introduction

LEAP (Lightweight Extensions and Application Programming) framework is a framework which allows a developer to design its screen without writing the XSL's. A developer needs to define the screen layout in a set of tables as per the guidelines given in this document. Once the screen definition is done in the tables, they need to generate an xml using a tool which has been discussed later. Once the xml is generated this framework will paint the screen as per the screen definition.

The LEAP framework has been extended to support mobile clients and mobile browsers. A set of new data types and new functionalities have been added in the LEAP framework for this purpose. This enhanced framework is called as mLEAP (Mobile Lightweight Extensions and Application Programming). mLEAP framework is capable of generating the content for application based mobile clients as well as mobile browsers. In case of application based channel (43), the output is in the form of XML. In case of mobile browser channel (42), the output is in the form of HTML. This is described in detail in subsequent sections.

All new screens for mobile application and browsers are supposed to be developed through this framework, until and unless a developer faces any major limitation in this framework.

Table Design for XML Output

The LEAP framework has been designed using following five tables. This section describes the structure of these tables. New columns added for mLEAP have been highlighted in blue.

- MSTHTMLDATATYPES
- SCREENTEMPLATEMASTER
- SCREENTABLEDEFINITION
- SCREENTEMPLATE

MSTHTMLDATATYPES

This table is meant for defining a Data Type. The details of the table are as follows:

Column Name	Column Type	Description
IDDATA	VARCHAR2(2)	This column is used to give an ID for a data type, e.g. 'D' for Dropdowns.
DESCRIPTION	VARCHAR2(250)	A Description is mandatory to be given for the data type being defined, so that a developer can understand the purpose of the data type.

SCREENTEMPLATEMASTER

This Table is meant for defining the primary details of the screen like the page heading, number of HTML tables and disclaimer, if any. 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
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

SCREENTABLEDEFINITION

This Table is meant for defining the properties of the HTML tables that would be created 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
IDTABLE	NUMBER	The ID of The Table
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.
PARENTORCHILD	VARCHAR2(1)	‘P’ for Parent Table and ‘C’ if it is a Child Table (inner table within an html <td>). Default Value is ‘P’.
PARENTTABLEID	NUMBER	For C value in ParentorChild column the developer has to mention the parent tableid which will contain this child table.
DEFAULTDISPLAY	CHAR(1)	This column allows a developer to mark a table as hidden when the page loads for the first time. ‘S’ for showing and ‘H’ for hiding the table on page load. Default value is always ‘S’.
USERAGENT	VARCHAR2(1)	This column allows a user to set different definitions for different clients
CONDITION	VARCHAR2(1000)	This column contains the display condition of a table.
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.

SCREENTEMPLATE

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.

App Based Mobile – XSL Design

The LEAP framework has been designed using following XSL's.

XSL Name	Package
genericscreentemplate.xsl	com\iflex\fcat\datafiles\gui\ENU\43\template
mleapscreenlayout.xsl	com\iflex\fcat\datafiles\gui\CMN
mleapdatatypes.xsl	com\iflex\fcat\datafiles\gui\CMN
conditions.xsl	com\iflex\fcat\datafiles\gui\CMN
uidownload.xsl	com\iflex\fcat\datafiles\gui\ENU\43\template

The logic for rendering the screen on the basis of screen definition xml has been written in the above set of XSL's.

App Based Mobile – Supported mLEAP Data Types

IDDATA	DESCRIPTION
T	This Data Type is for creating Text Boxes.
MD	This Data Type is for creating date input field.
P	This Data Type id for Password.
D	Drop Down
L1	This Data Type is for sub headings
V	This Data Type is for Verification Fields
B	This Data Type is for Buttons.
FA	This Data Type is for Formatted Amount.
FD	This Data Type is for Formatted Date.
TZ	This Data Type is for Formatted Date with time zone.
FU	This Data Type is for Formatted unit(amount, number)
H	This Data Type is for Hidden Fields.
D1	Drop Down with static values
PD	This Id is for pagination data
AD	This Id is for custom template of account dropdown for account activity
FX	This Id is for custom template of exchange rate
CM	This Id is for custom template of menu

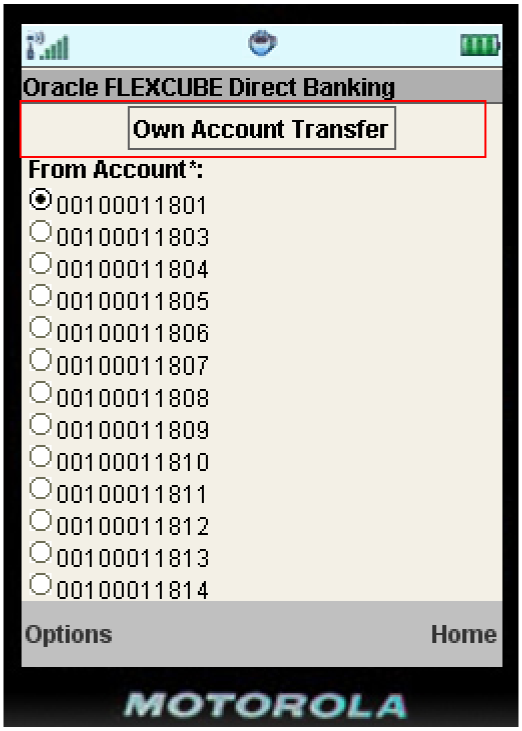
PP	This Id is for custom template of password policy
FP	This Id is for custom template of password policy in case of force change password
TA	This Data Type is for Text Area
SA	This Data Type is for Accounts with optgroup
PB	Data type created to support buttons which results in a ‘popup’ event
SR	This data type is used to support radio based controls.
GL	This data type is used for grouping labels together as a grouped header label.
SB	This Data Type Creates a Segmented Button.
L	This Data Type is used to create List.
AC	This Data Type supports Accordion.
W	This Data Type shows Webview on the client side.
DSB	This Data Type Creates a dynamic Segmented Button.
BG	This Data Type Creates grouped buttons
GR	This Data Type Creates graph area

App Based Mobile – Sample Screen Components

Following are the samples of creating different screen components using this framework.

Adding a Page Heading

In the screen shot below page header has been highlighted, a developer can configure this parameter by adding the entries in *screentemplatemaster*. The value entered in iddescription column gets reflected as page header.

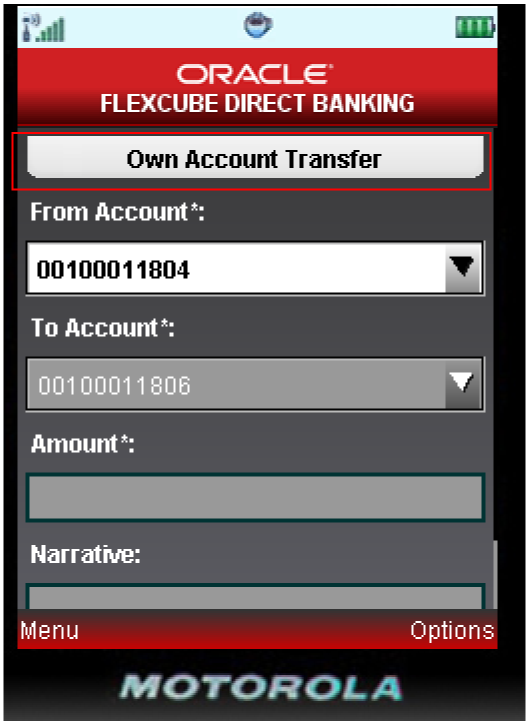


Oracle FLEXCUBE Direct Banking

Own Account Transfer

From Account*:

● 00100011801
○ 00100011803
○ 00100011804
○ 00100011805
○ 00100011806
○ 00100011807
○ 00100011808
○ 00100011809
○ 00100011810
○ 00100011811
○ 00100011812
○ 00100011813
○ 00100011814



Own Account Transfer

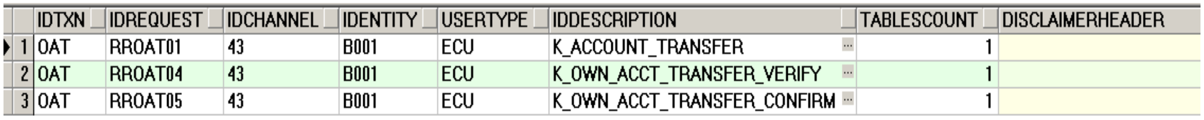
From Account*: 00100011804

To Account*: 00100011806

Amount*:

Narrative:

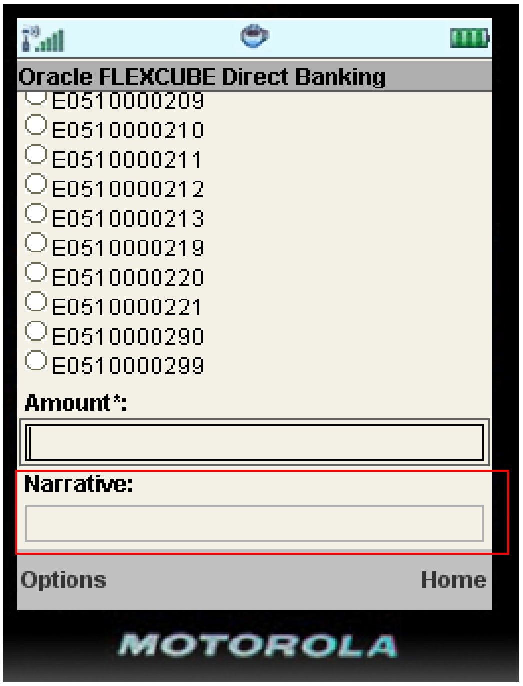
The entries for the idtxn, idrequest, identity, idchannel and usertype has to be done



	IDTXN	IDREQUEST	IDCHANNEL	IDENTITY	USERTYPE	IDDESCRIPTION	TABLESCOUNT	DISCLAIMERHEADER
1	OAT	RROAT01	43	B001	ECU	K_ACCOUNT_TRANSFER	1	
2	OAT	RROAT04	43	B001	ECU	K_OWN_ACCT_TRANSFER_VERIFY	1	
3	OAT	RROAT05	43	B001	ECU	K_OWN_ACCT_TRANSFER_CONFIRM	1	

Adding a Text Box

The following screen shot displays a text box created using the definition in *screentemplate*.

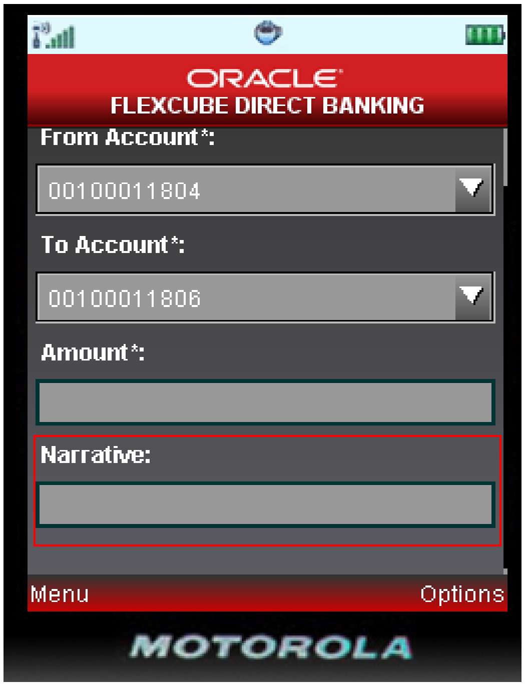


Oracle FLEXCUBE Direct Banking

○ E0510000209
○ E0510000210
○ E0510000211
○ E0510000212
○ E0510000213
○ E0510000219
○ E0510000220
○ E0510000221
○ E0510000290
○ E0510000299

Amount*:

Narrative:



From Account*: 00100011804

To Account*: 00100011806

Amount*:

Narrative:

Now as per the screen shot there is a label “User Reference” for the field and a data field where the value is to be entered.

The entries to be done in *screentemplate* for this field are as follows

LABEL	K_NARRATIVE

NAME	fldnarrative
ID	fldnarrative
TYPE	T
IDROW	1
COLUMNID	1
TABLENO	1
ALT	K_NARRATIVE
INPUTMAXLENGTH	20
TOKEN1	<Text type>

Provide text type in token1

Possible values of text type::

- N- for giving numbers
- E- for giving emails
- S- for giving any string value
- D- for giving any decimal value

Adding a Date Field

The date field can be defined similar to text box except the type is to be declared as “MD” in the type column.

Adding a Date Picker

The date field can be defined as a date picker and the type is to be declared as “TD” in the type column.

Adding a Password Field

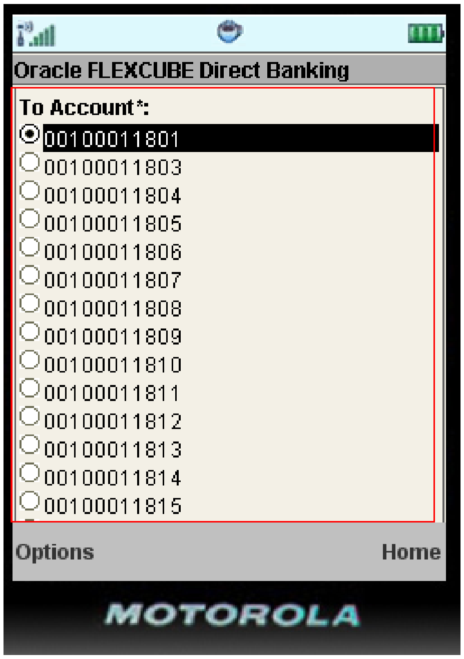
The password field can be defined similar to text box except the type is to be declared as “P” in the type column.

Adding Text Area

The text area can be defined similar to text box except the type is to be declared as “TA” in the type column.

Adding a Drop Down

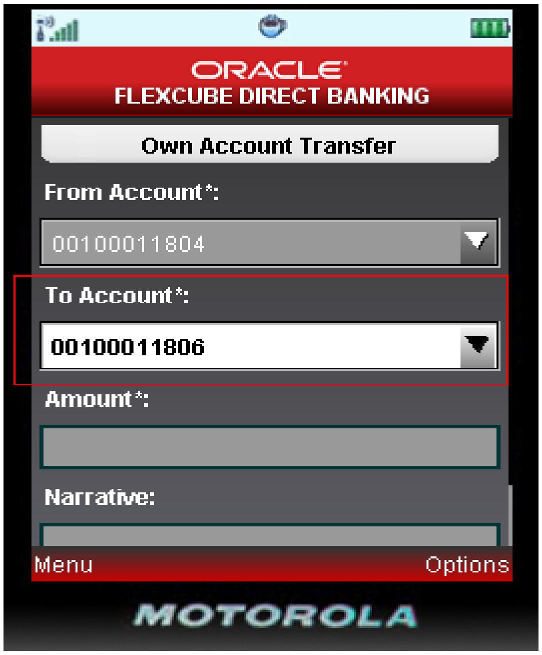
The following Screen Shot displays a Drop Down created using the definition in *screentemplate*.



Oracle FLEXCUBE Direct Banking

To Account*:

● 00100011801
○ 00100011803
○ 00100011804
○ 00100011805
○ 00100011806
○ 00100011807
○ 00100011808
○ 00100011809
○ 00100011810
○ 00100011811
○ 00100011812
○ 00100011813
○ 00100011814
○ 00100011815



ORACLE
FLEXCUBE DIRECT BANKING

Own Account Transfer

From Account*: 00100011804

To Account*: 00100011806

Amount*:

Narrative:

Now as per the screen shot there is a label “Destination Account” for the field and a Drop Down from where the value is to be selected.

The entries to be done in *screentemplate* for this field are as follows

LABEL	K_TO_ACCNO
NAME	flddestacctno
ID	flddestacctno
TYPE	D
NODE VALUE	//faml/response/genericpaymentresponsedto/responsedata/fundtransferresponsedto/srccustomeraccdto/customeraccountdto/accounts/accountnodto~accountdisplayname ~nbraccount,availablebalance,codcurrency,idcustomer,nbrbranch,ccydesc,accttype~codcurrency
IDROW	3
COLUMNID	1
TABLE NO	1
Token1	<Action id>
Token2	<Target id>
Token3	<request id>
Token4	type
Token5	<valueindex>
ISUDF	Y
ALT	K_TO_ACCNO

Now as per the definition this field should appear at 3 position on the screen.

Node value is the XPATH from where the value would be picked to create the drop down. In the example attached the developer needs to give the XPATH till the parent element and use '~' separator to define the name and value for the drop down. If a developer wants to assign '~' separated multiple values to a drop down which is generally required for account dropdowns where the value of drop down contains the name of the branch, currency of the account, it can be done using giving the element name using comma separator as shown below. If required we can filter the options on some value, on which the list has to be filtered

//faml/response/genericpaymentresponsedto/responsedata/fundtransferresponsedto/srccustomeraccdto/customeraccountdto/accounts/accountnodto~accountdisplayname

~nbraccount,availablebalance,codcurrency,idcustomer,nbrbranch,ccydesc,accttype~codcurrency

Label gives the default label that needs to be given for the dropdown

Token1 (actionid) Contains the Name of "Template Screen" which is invoked on Client Side to present the data. For example 'V' denotes a Verify Template Screen and 'M' denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token3 (reqid) contains Request Id.

Here RRATO62 indicates the requestId which will become visible on selection of particular option in Static Dropdown.

Token4(type)

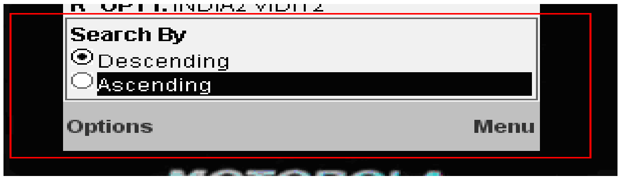
This indicates the type given for dropdown. It can be 'F' when filter is required, i.e when the list needs to be filtered for distinct values.

Token5 (type) indicates the value index for the dropdown. This value index will indicate, the default value to be set for the dropdown.

Adding a Static Drop Down

This Data Type is used for creating static drop down.

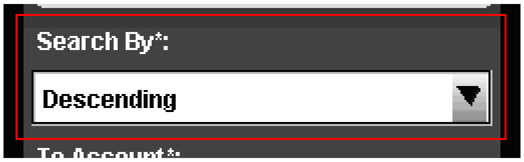
The following Screen Shot displays a static dropdown created using the definition in *screentemplate*.



Search By

● Descending

○ Ascending



Search By*:

Descending ▼

The entries to be done in *screentemplate* for this data type are as follows:

NAME	fldpattern
ID	fldpattern
TYPE	D1
DATACLASS	
NODEVALUE	string('0')~string('1')
DEFAULTSTATICLABEL	K_SINGLE~K_TDJOINT
RELPOSX	0.11
RELPOSY	0.13
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	

TOKEN2	RRATO612
TOKEN4	RRATO62
TOKEN5	<Valueindex>

Label gives the default label that needs to be given for the dropdown

NODEVALUE contains the value which needs to be passed on selection of specific Option under Dropdown.

Here string ('0') ~string ('1') indicates that on selection of first child '0' is passed and on selection of second child '1'is passed. If there had been more Childs for particular Static Dropdown then they would have been again separated by ~.

DEFAULTSTATICLABEL contains the labels which will appear near the respective options under Static Dropdown.

Here K_SINGLE~K_TDJOINT indicates that first option contains the label of "Single" and second option contains the label of "Joint". If there had been more Childs for particular Static Dropdown then they would have been again separated by ~.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of "Template Screen" which is invoked on Client Side to present the data. For example 'V' denotes a Verify Template Screen and 'M' denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token3(default option) contains default option

The default option for a dropdown is the default label that appears as the first option of the dropdown. For instance "Select Account".

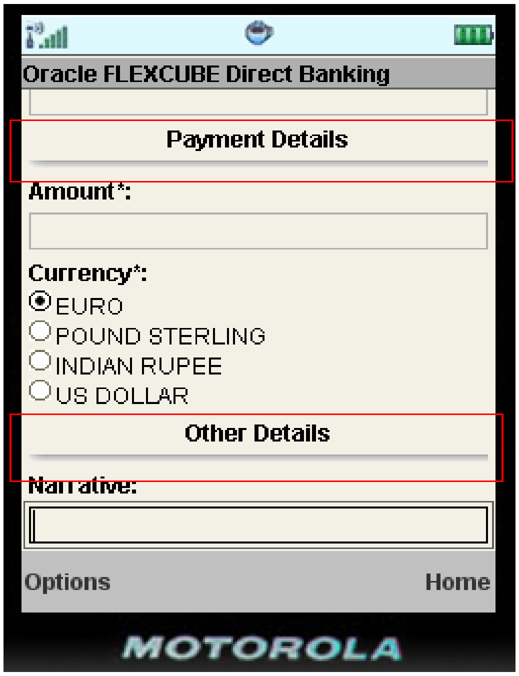
Token4 (reqid) contains Request Id.

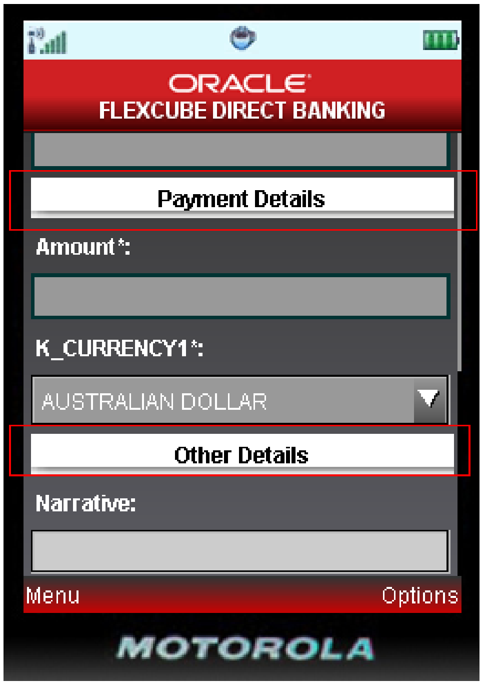
Here RRATO62 indicates the requestId which will become visible on selection of particular option in Static Dropdown.

Token5 (type) indicates the value index for the dropdown. This value index will indicate, the default value to be set for the dropdown.

Adding Subheading Data Field

The following Screen Shot displays a subheading field created using the definition in *screentemplate*. This data type is used to show subheadings on the screen.





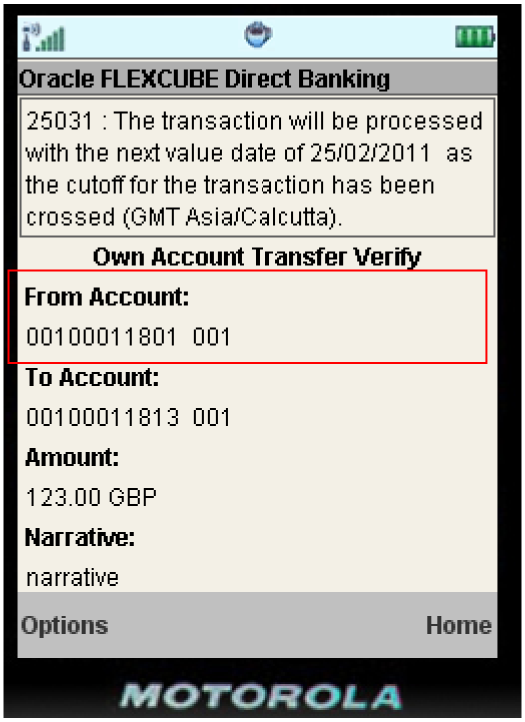
The entries to be done in *screentemplate* for this field are as follows

LABEL	K_PAYMENT_DETAILS
NAME	fldfield1

ID	fldfield1
TYPE	L1
NODEVALUE	
IDROW	1
COLUMNID	1
TABLENO	1

Adding Verification Data Field

The following Screen Shot displays a verification field created using the definition in *screentemplate*. This data type is meant for those data's where fields and there values are non-editable, commonly used in verification screen's.



Oracle FLEXCUBE Direct Banking

25031 : The transaction will be processed with the next value date of 25/02/2011 as the cutoff for the transaction has been crossed (GMT Asia/Calcutta).

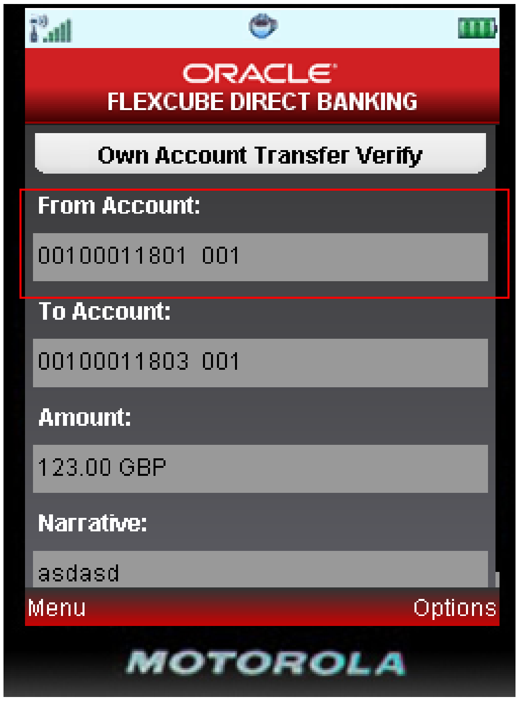
Own Account Transfer Verify

From Account:
00100011801 001

To Account:
00100011813 001

Amount:
123.00 GBP

Narrative:
narrative



Own Account Transfer Verify

From Account:
00100011801 001

To Account:
00100011803 001

Amount:
123.00 GBP

Narrative:
asdasd

Now as per the screen shot there is a label "Source Account" for the field and data which is non-editable.

2 entries to be done in *screentemplate* for this field are as follows

1st entry:

LABEL	K_TO_ACCOUNT
NAME	fldfield1
ID	fldfield1
TYPE	V
NODEVALUE	
IDROW	1
COLUMNID	1
TABLENO	1

2nd entry:

LABEL	
NAME	fldfield2
ID	fldfield2
TYPE	V
NODEVALUE	//faml/request/fldsrcacctno_txt
IDROW	1
COLUMNID	1
TABLENO	1

Now as per the definition this field, label should appear at first position and value should appear at second position.

Adding Static Radio

This data type is used to support radio based controls.

The entries to be done in *screentemplate* for this data type are as follows:

NAME	fldpattern
ID	fldpattern
TYPE	SR
DATACLASS	
NODEVALUE	string(‘1')~string('2')
DEFAULTSTATICLABEL	K_CHEQUENUMBER~K_CHEQUERANGE
RELPOSX	0.11
RELPOSY	0.13
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	
TOKEN2	RROAT611
TOKEN4	RROAT62
TOKEN5	H

DATACLASS contains the name of CSS class for the component

In case of Static radio it will contain two data classes separated by “~”. First Data Class will denote the CSS class name for the Parent Radio i.e., Radio Group and second Data Class will denote the CSS class name for the Childs (radios) present in Radio Group.

NODEVALUE contains the value which needs to be passed on selection of specific radio under Radio Group.

Here string ('1') ~string ('2') indicates that on selection of first child ‘1’ is passed and on selection of second child ‘2’is passed. If there had been more Childs for particular Radio Group then they would have been again separated by ~.

DEFAULTSTATICLABEL contains the labels which will appear near the respective Radio Buttons under Radio Groups.

Here K_CHEQUENUMBER~K_CHEQUERANGE indicates that first radio contains the label of “Cheque Number” and second radio contains the label of “Cheque Range”. If there had been more Childs for particular Radio Group then they would have been again separated by ~.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Here RROAT611 indicates that RROAT61 is the screen on which the data will become visible and 1 appended after RROAT61 indicates the table number in which the data will be populated.

Token4 (reqid) contains Request Id.

Here RROAT62 indicates the requestId which will become visible on selection of particular radio in Static Radio.

Token5 (type) Indicates orientation of Radio Group. The orientation can be either horizontal (H) or vertical (V).

Here ‘H’ indicates that the static radio will be horizontally aligned.

Adding Static CheckBox

This data type is used to support checkbox based controls.

The entries to be done in *screentemplate* for this data type are as follows:

NAME	fldpattern
ID	fldpattern
TYPE	SC
DATACLASS	
NODEVALUE	string('1')~string('2')
DEFAULTSTATICLABEL	K_CHEQUENUMBER~K_CHEQUERANGE
RELPOSX	0.11
RELPOSY	0.13
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	2
TOKEN5	H

DATACLASS contains the name of CSS class for the component

In case of Static checkbox it will contain two data classes separated by "~". First Data Class will denote the CSS class name for the Parent checkbox i.e., checkbox Group and second Data Class will denote the CSS class name for the Childs (checkbox) present in checkbox Group.

NODEVALUE contains the value which needs to be passed on selection of specific checkbox under checkbox Group.

Here string ('1') ~string ('2') indicates that on selection of first child '1' is passed and on selection of second child '2'is passed. If there had been more Childs for particular checkbox Group then they would have been again separated by ~.

DEFAULTSTATICLABEL contains the labels which will appear near the respective checkbox Buttons under checkbox Groups.

Here K_CHEQUENUMBER~K_CHEQUERANGE indicates that first checkbox contains the label of "Cheque Number" and second checkbox contains the label of "Cheque Range". If there had been more Childs for particular checkbox Group then they would have been again separated by ~.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 denotes the number of columns to be given for checkboxes.

Token5 (type) Indicates orientation of Radio Group. The orientation can be either horizontal (H) or vertical (V).

Here ‘H’ indicates that the static radio will be horizontally aligned.

Adding Grouped Labels

It is used for grouping labels together as a grouped header label.

The entries to be done in *screentemplate* for this data type are as follows:

NAME	fldpattern
ID	fldpattern
TYPE	GL
LABEL	K_DESCRIPTION~K_COUNT~K_STATUS
LABELCLASS	AccountDetails~AccountDetails~AccountDetails
DATACLASS	
NODEVALUE	string('1')~string('2')~string('3')
RELPOSX	0.11
RELPOSY	0.13
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	0.25~0.1~0.1

LABEL contains the labels which will appear near the respective Labels under Grouped labels.

Here K_DESCRIPTION~K_COUNT~K_STATUS indicates that label group (parent label) contains the label of “Description” and child labels contains the label of “Count” and “Status” respectively. If there had been more Childs for particular Grouped Label then they would have been again separated by ~.

LABELCLASS contains the respective CSS class for label which will appear on the Grouped Labels.

For example over here the name for parent label CSS class is AccountDetails followed by child labels of same name separated by ‘~’.If there had been more Childs for particular Grouped Label then they would have been again separated by ~.

NODEVALUE contains the value which needs to be passed on selection of specific Label under Grouped Label.

Here string('1')~string('2')~string('3') indicates that on selection of first child ‘1’ is passed, on selection of second child ‘2’is passed and on selection of third child ‘3’ will be passed. If there had been more Childs for particular Radio Group then they would have been again separated by ~.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

TOKEN1 contains the Label Width. Each label width are denoted separated by “~”.

Here 0.25 indicates the label width of parent label, and 0.1 for child labels. .If there had been more Childs for particular Grouped Label then they would have been again separated by ~.

Adding Pop up Button

This Data Type is used to support buttons which results in a ‘popup’ event.

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL
NAME	fldsubmitlater

ID	fldsubmitlater
DATACLASS	Button
TYPE	PB
IDROW	2
COLUMNID	1
TABLENO	6
NODEVALUE	s~RROAT06
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	1
TOKEN2	0.6
TOKEN3	0.4

NODE VALUE defines the type of button and request id to be fired on that button. Here s~RROAT06 indicated that on click of Popup Button RequestId RRAOT06 will be passed.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

TOKEN1 contains the tableid which will appear as a pop-up on the screen. On the click of the Popup Button the table whose id is mentioned in the tableid will appear on the screen as a pop-up.

TOKEN2 and TOKEN3 contains the pop-up height and pop-up width respectively and their value ranges from 0.0 – (any float value).

Adding Image Button

This Data Type is used to support buttons which results in a 'popup' event.

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL
NAME	fldsubmitlater
ID	fldsubmitlater
DATACLASS	ImageButton
TYPE	IB
IDROW	2
COLUMNID	1
TABLENO	6
NODEVALUE	s~RROAT06
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	V
TOKEN2	RROAT061
TOKEN5	<Badge details>
IMAGESRC	<Image src>

NODE VALUE defines the type of button and request id to be fired on that button. Here s~RROAT06 indicated that on click of Popup Button RequestId RRAOT06 will be passed.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of "Template Screen" which is invoked on Client Side to present the data. For example 'V' denotes a Verify Template Screen and 'M' denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token5 contains the badge details (# seperated).Badgevalue#badgepos#badgrertlpos

ImageSrc contains the image name.

Adding Segmented Button

This Data Type Creates a Segmented Button.

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL
NAME	fldstopunstopchq
ID	fldstopunstopchq
DATACLASS	SegmentedButton
TYPE	SB
IDROW	1
COLUMNID	1
TABLENO	5
NODEVALUE	string('1')~string('2')
DEFAULTSTATICLABEL	K_BLOCK~K_UNBLOCK
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20

RELHEIGHT	0.06
TOKEN1	V
TOKEN2	
TOKEN4	RRVAT62
TOKEN5	H

NODEVALUE contains the value which needs to be passed on selection of specific button under Segmented Button.

Here string ('1') ~string ('2') indicates that on selection of first child ‘1’ is passed and on selection of second child ‘2’is passed. If there had been more Childs for particular Segmented Button then they would have been again separated by ~.

DEFAULTSTATICLABEL contains the labels which will appear near the respective Buttons under Segmented Button.

Here K_BLOCK~K_UNBLOCK indicates that first buttoncontains the label of “ Block ” and second button contains the label of “unblock”. If there had been more Childs for particular Radio Group then they would have been again separated by ~.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token4 (reqid) contains Request Id.

Here RRVAT62 indicates the requestId which will become visible on selection of particular Button in Segmented Button.

Token5 (type) indicates orientation of Segmented Button. The orientation can be either horizontal (H) or vertical (V).

Here ‘H’ indicates that the Segmented Button will be horizontally aligned.

Adding List/Searchdropdown

It is used to create List.

The entries to be done in *screentemplate* for this data type are as follows:

LABELCLASS	AcctActDetails~AcctActDetails~AcctActDetails
LABELWIDTH	0.26~0.09~0.08#0.08~0.08~0.06
NAME	fldTxns
ID	fldTxns
DATACLASS	Select
TYPE	L for list and SD for searchdropdown
IDROW	1
COLUMNID	1
TABLENO	5
NODEVALUE	//faml/response/authorizationstatisticsresponsedto/transactionstatistics/authorizationstatisticsperioddto/statistics/authorizationstatisticsdto~txndescription;count;status;description~idtxn;status~status
ALT	K_TRANSACTIONS
DEFAULTSTATICLABEL	K_TRANSACTION_LIST
RELPOSX	0.40
RELPOSY	0.02

RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	
TOKEN2	RRVAT641
TOKEN3	0.08
TOKEN4	RRVAT68
TOKEN5	R~3

LABELCLASS contains the respective CSS class for components which will appear on the List.

For Example here “AcctActDetails~AcctActDetails~AcctActDetails” three label class name are present. First class is for entire list structure, second is for list and third is for options in list.

Label Width indicates the width corresponding to the label class separated by ~ # labelheight, for each label, separated by ~.

Example 0.26~0.09~0.08#0.08~0.08~0.06

Here 0.26 is the width for 1st label and 0.097 is the width for second label and 0.08 for third. Similarly 0.08 is the height for 1st label and 0.08 is for 2nd and so on.

NODEVALUE contains the labels that need to be displayed and values which need to be passed on selection of specific option under list. The nodevalue is given as, dto~displayname(each separated by ‘;’)~values(each separated by ‘;’)~value on which the list has to be filtered, only if required. In case display name consists of amount or date. Please enter the displayname as FA#codcurrency#balance#idlang, in case of amount and FD#dateformat #date#idlang, in case of date.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token2(tableid) specifies the table id in which the list will get populated. actionid is used either with targeted or tableid. It is not used with both at same time. For example if transaction requires list to become visible in table then tableid is used and if transaction needs to populate list on template screen then tragetid is used.

Token3(option height) contains the height of the options available for the list.

Here 0.08 specifies that each option within the list will be of height 0.08.

Token4 (reqid) contains Request Id.

Here RRVAT68 indicates the requestId which will become visible on selection of particular List.

Token5 (type) indicates type of List along with number of columns separated by ~.

It is mentioned as: Type~Number of Columns in List.

The List type can be as:

1) **P**: describes list which will result in Pop-over Type.
2) **R**: describes the list which will not result in pop-over (otherwise).

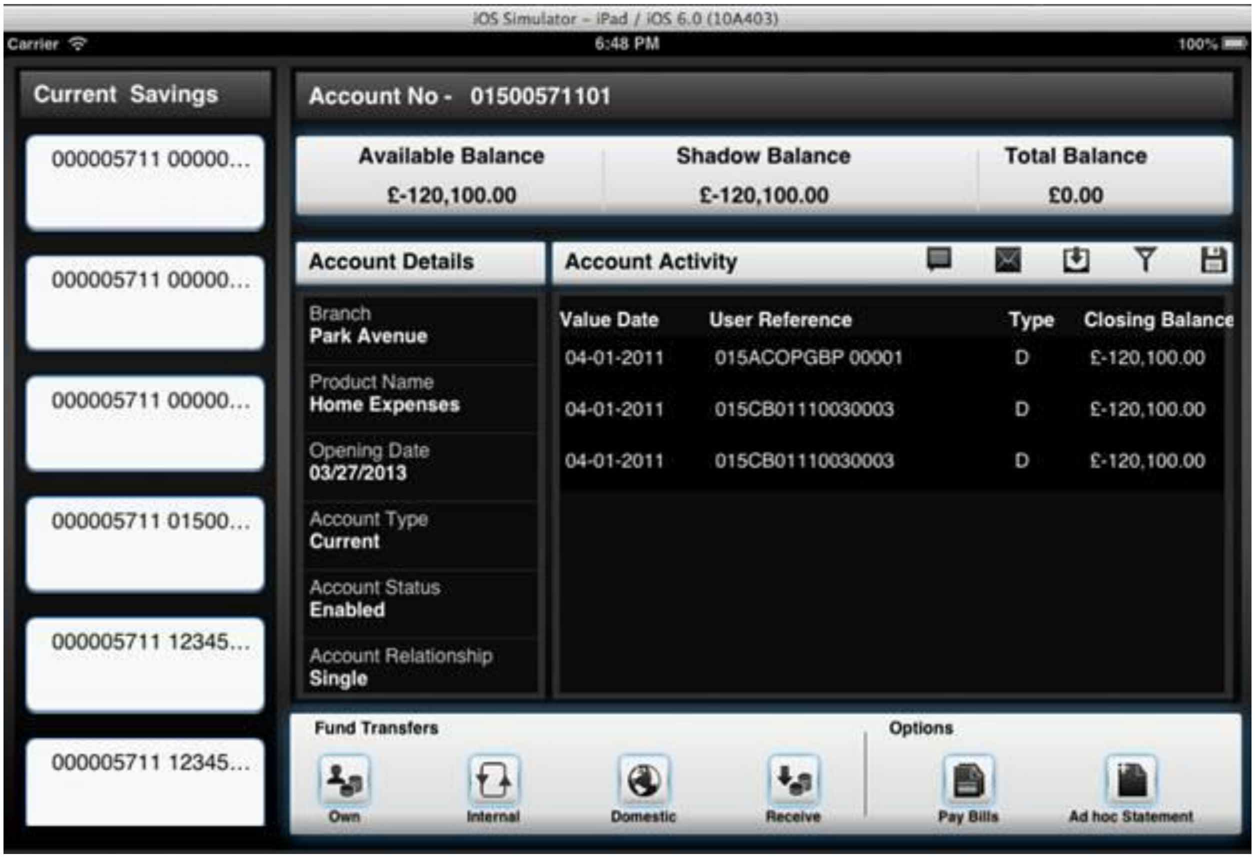
Here in TOKEN5 'R~3' R indicates the type of list which is not pop-over and 3 indicates the number of columns in the list.

Note:- In case of a grouped list, you need to add '~G' in token5 , after type~no. of columns.

Example:: P~2~G. In case of plain list no need to provide any style , example:: P~2

Please see the below image, it depicts the style of list::

Grouped list **Plain List**



Account No - 01500571101

Available Balance	Shadow Balance	Total Balance
£-120,100.00	£-120,100.00	£0.00

Branch
Park Avenue

Product Name
Home Expenses

Opening Date
03/27/2013

Account Type
Current

Account Status
Enabled

Account Relationship
Single

Value Date	User Reference	Type	Closing Balance
04-01-2011	015ACOPGBP 00001	D	£-120,100.00
04-01-2011	015CB01110030003	D	£-120,100.00
04-01-2011	015CB01110030003	D	£-120,100.00

Adding Accordion

This tag supports Accordion.

The entries to be done in *screentemplate* for this data type are as follows:

NAME	fldweb
DATACLASS	Accordion
TYPE	AC
IDROW	1
COLUMNID	1
TABLENO	5
DATACLASS	Select

NODEVALUE	
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	1
TOKEN2	0.24
TOKEN3	0.56
TOKEN4	RRMST61
TOKEN5	L

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

TOKEN1 (table id) Contains the id for the table which will appear on slide of Accordion.

Here ‘1’ denotes the table id which will appear on the slide of the accordion.

TOKEN2 (table width) contains Width of the table which appears on slide.

Here 0.24 is the table width which will appear once the slide action is performed on the Accordion.

TOKEN3 contains the Height of the table which appears on slide.

Here 0.56 is the table height which will appear once the slide action is performed on the Accordion.

TOKEN4 (reqid) contains Request Id.

Here RRMST61indicates the requestId which will become visible on slide of Accordion.

TOKEN5 (type) specifies the positioning of the Accordion which appears on the slide. The position is of three types:

B: Bottom L: Left R: Right

Here the Accordion will be on the ‘L’ Left side of the screen.

Adding Webview

It shows Webview on the client side.

The entries to be done in *screentemplate* for this data type are as follows:

NAME	fldweb
DATACLASS	WebView
TYPE	W
NODEVALUE	
IDROW	1
COLUMNID	1
TABLENO	5
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	l

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

NODEVALUE contains the URL to be given for the Webview.

Token1 indicates the type of Webview on client side. The value of Token1 can be either l or n, for Login required and Login not required Webview respectively.

Here ‘l’ indicates that the Webview on client side requires login.

Adding Value-Label

VL' datatype is used as 'Value-Label' pair

Eg.	1 -	5 years 8 days
	2 -	6 ft 3 in
	3 -	10 lakhs 50 thousand

NAME	Fld1
DATACLASS	Fld1
TYPE	VL
NODEVALUE	//faml/response/tdpayoutinstrresponsedto/tdpayoutinstrdtls/tdpayoutinstrdetailsdto/termyears##//faml/response/tdpayoutinstrresponsedto/tdpayoutinstrdtls/tdpayoutinstrdetailsdto/termmonths##//faml/response/tdpayoutinstrresponsedto/tdpayoutinstrdtls/tdpayoutinstrdetailsdto/termdays
IDROW	1
COLUMNID	1
TABLENO	5
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
DEFAULTSTATICLABELS	K_YEARS~K_MONTHS~K_DAYS

Nodevalue : XPath for 'Value', ## separated,

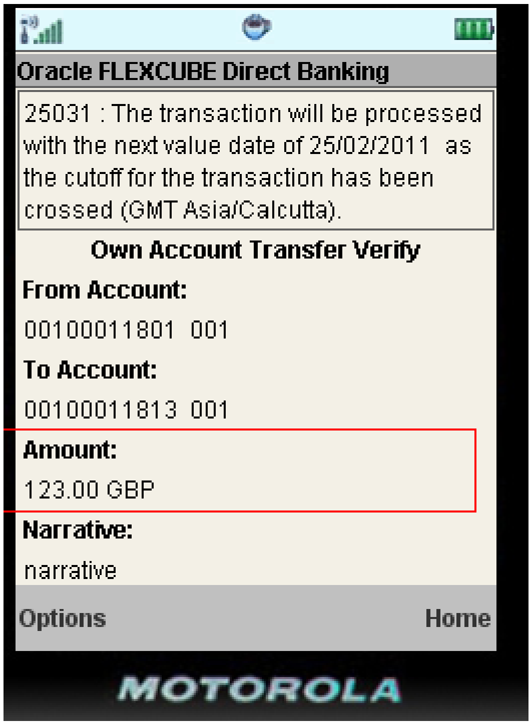
Eg: //faml/response/fldyear ##//faml/response/month##//faml/response/day

DefaultStaticLabels : Labels , ~ separated,

Eg: K_YEARS~K_MONTHS~K_DAYS

Adding Formatted Amount

Formatted Amount is a type of verification field, but the data displayed in this case is formatted using formatCurrency method of JFFormatter. The following screen shot displays a formatted amount value using the definition in *screentemplate*.



Oracle FLEXCUBE Direct Banking

25031 : The transaction will be processed with the next value date of 25/02/2011 as the cutoff for the transaction has been crossed (GMT Asia/Calcutta).

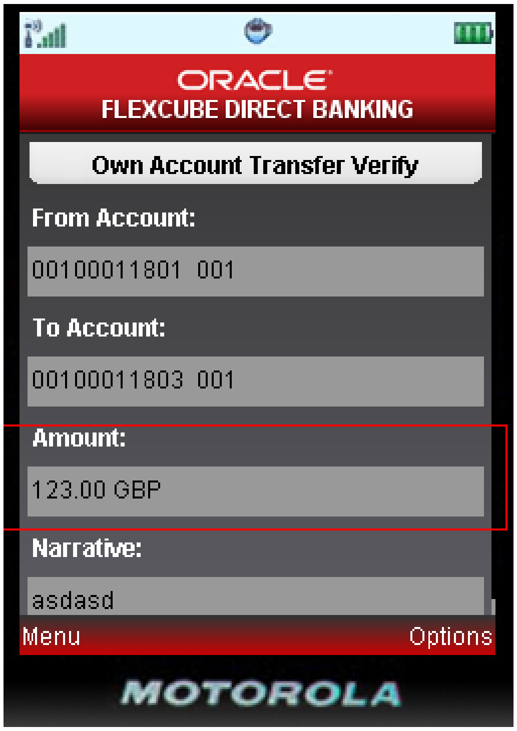
Own Account Transfer Verify

From Account:
00100011801 001

To Account:
00100011813 001

Amount:
123.00 GBP

Narrative:
narrative



Own Account Transfer Verify

From Account:
00100011801 001

To Account:
00100011803 001

Amount:
123.00 GBP

Narrative:
asdasd

Now as per the screen shot there is a label “Transfer Amount” for the field and data which is formatted and non-editable.

2 entries to be done in *screentemplate* for this field are as follows

1st entry:

LABEL	K_XFER_AMOUNT
NAME	fldfield5
ID	fldfield5
TYPE	V
NODEVALUE	
IDROW	3
COLUMNID	1
TABLENO	1

2nd entry:

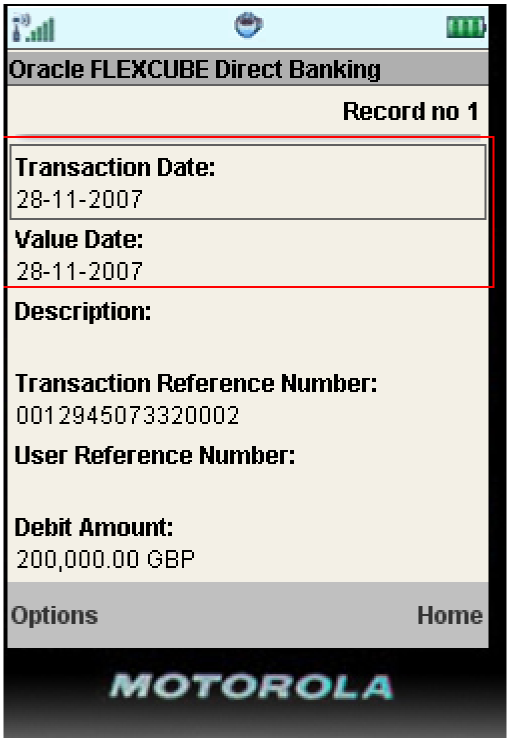
LABEL	
NAME	fldfield6
ID	fldfield6
TYPE	FA
NODEVALUE	str:split(//faml/request/flddestacctno,"~")[6]#//faml/request/fldtxnamount#//faml/request/fldlangid
IDROW	4
COLUMNID	1
TABLENO	1

Now as per the definition this field should appear in the second table’s fifth row and first column.

Node value should contain ‘#’ separated values of the currency to be applied, the value and the langId.

Adding Formatted Date

Formatted Date is a type of verification field, but the data displayed in this case is formatted using formatDate method of JFFormatter. The following screen shot displays a formatted date value using the definition in *screentemplate*.



Oracle FLEXCUBE Direct Banking

Record no 1

Transaction Date:
28-11-2007

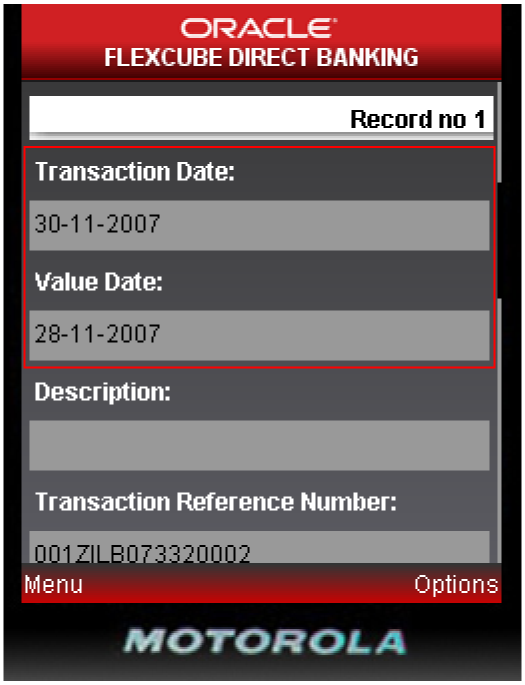
Value Date:
28-11-2007

Description:

Transaction Reference Number:
0012945073320002

User Reference Number:

Debit Amount:
200,000.00 GBP



ORACLE
FLEXCUBE DIRECT BANKING

Record no 1

Transaction Date:
30-11-2007

Value Date:
28-11-2007

Description:

Transaction Reference Number:
001ZILB073320002

Now as per the screen shot there is a label "Transaction Date" for the field and data which is formatted and non-editable.

2 entries to be done in *screentemplate* for this field are as follows

1st entry:

LABEL	K_TXN_DATE
NAME	flddata21
ID	flddata21

TYPE	V
NODEVALUE	
IDROW	21
COLUMNID	1
TABLENO	1

2nd entry:

LABEL	
NAME	flddata22
ID	flddata22
TYPE	FD
NODEVALUE	string('BDATEFORMAT')#//faml/response/viewcreditcardinforesponsedto/carddetails/creditcarddetailsdto[position()=$rowiteration]/paymentduedate#//faml/response/sessioninfo/@idlang
IDROW	22
COLUMNID	1
TABLENO	1

Node value should contain '#' separated values of the format to be applied, the value and the langId.

Adding Formatted Date with time zone

Formatted Date with time zone is a type of verification field, but the data displayed in this case is formatted using formatDate method of JFFormatter.

2 entries to be done in *screentemplate* for this field are as follows

1st entry:

LABEL	K_CREATEDON
NAME	flddata21
ID	flddata21
TYPE	V
NODEVALUE	
IDROW	21
COLUMNID	1
TABLENO	1

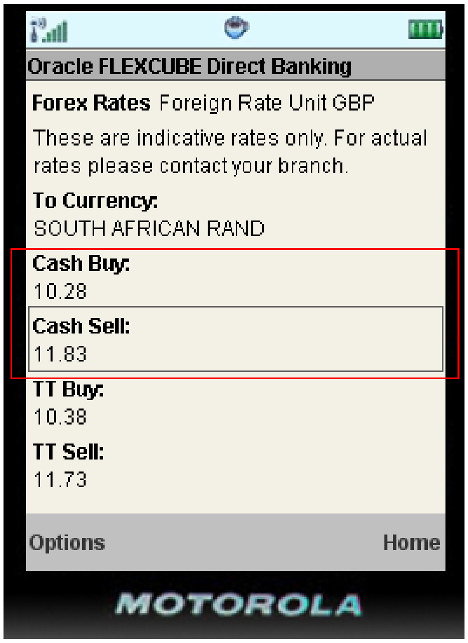
2nd entry:

LABEL	
NAME	flddata22
ID	flddata22
TYPE	TZ
NODEVALUE	concat(//faml/response/sessioninfo/@identity,'.DATETIMEFORMAT')#//faml/response/viewcreditcardinforesponsedto/carddetails/creditcarddetailsdto[position()=$rowiteration]/paymentduedate#//faml/response/*/extendedresponse/extendedresponsedto/entitytimezone#//faml/response/sessioninfo/@idlang
IDROW	22
COLUMNID	1
TABLENO	1

Node value should contain '#' separated values of the format to be applied, the value ,the timezone and the langId.

Adding Formatted Unit

Formatted unit is a type of verification field, but the data displayed in this case is formatted using different format methods of JFFormatter. The following screen shot displays a formatted unit value using the definition in *screentemplate*.



Oracle FLEXCUBE Direct Banking

Forex Rates Foreign Rate Unit GBP

These are indicative rates only. For actual rates please contact your branch.

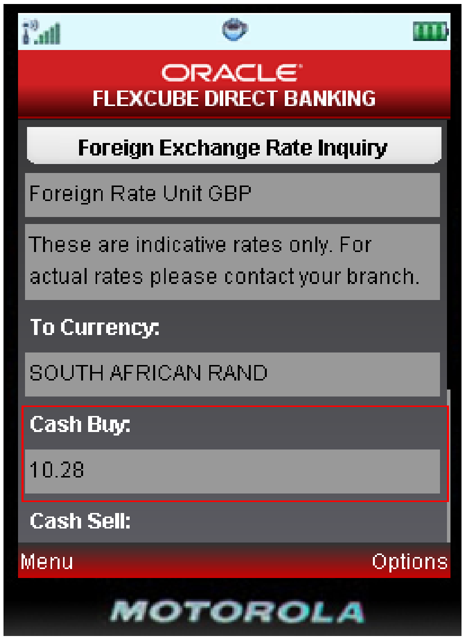
To Currency:
SOUTH AFRICAN RAND

Cash Buy:
10.28

Cash Sell:
11.83

TT Buy:
10.38

TT Sell:
11.73



ORACLE
FLEXCUBE DIRECT BANKING

Foreign Exchange Rate Inquiry

Foreign Rate Unit GBP

These are indicative rates only. For actual rates please contact your branch.

To Currency:
SOUTH AFRICAN RAND

Cash Buy:
10.28

Cash Sell:

2 entries to be done in *screentemplate* for this field are as follows

1st entry:

LABEL	K_CASHBUY
NAME	flddata21
ID	flddata21
TYPE	V
NODEVALUE	
IDROW	2
COLUMNID	1
TABLENO	1

2nd entry:

LABEL	
NAME	flddata22
ID	flddata22
TYPE	FU
NODEVALUE	string('USD')#//faml/response/exchangerateinquiryresponsedto/exchangerates/exchangeratedto/cashbuyrate#//faml/response/sessioninfo/@idlang
IDROW	3
COLUMNID	1
TABLENO	1

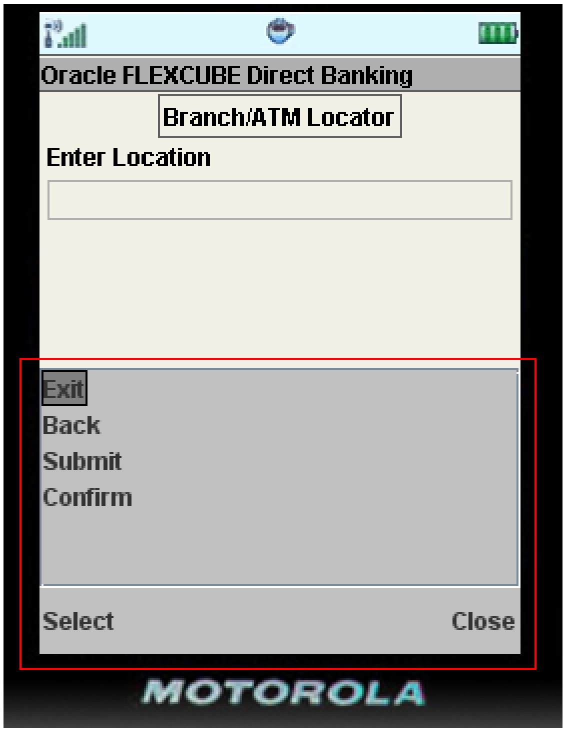
Node value should contain '#' separated values of the format to be applied, the value and the langId.

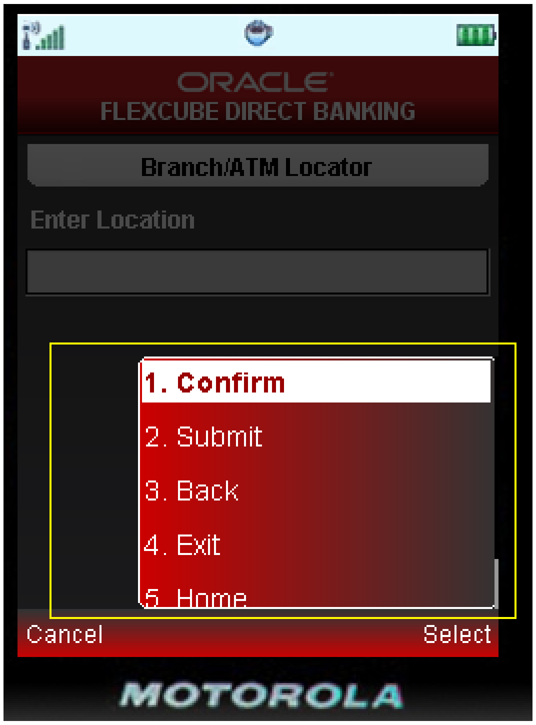
Adding Buttons

The following Screen Shot displays buttons created using the definition in *screentemplate*.

1: Basic Buttons

- Back
- Submit
- Confirm





The entries to be done in *screentemplate* for this field are as follows

LABEL	
NAME	fldback
ID	fldback
TYPE	B
IDROW	3
COLUMNID	1

TABLENO	1
NODEVALUE	b~RROAT01
TOKEN1	<Actionid>
TOKEN2	<targeted>

Here type defines the data type and node value defines the type of button and request id to be fired on that button. Node value should be ~ separated first value defines the type of button (i.e. b= Back, c= Confirm, s=Submit) and second value defines the request id of that button.

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

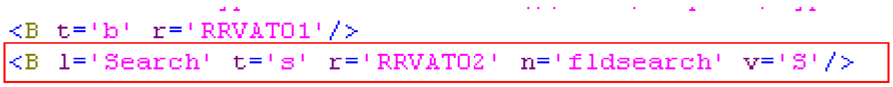
2: Custom buttons

If user wants to add a custom button

The entries to be done in *screentemplate* for this field are as follows

LABEL	K_SEARCH
NAME	fldsearch
ID	fldsearch
TYPE	B
IDROW	3
COLUMNID	1

TABLENO	1
NODEVALUE	s~RRVAT02
FUNCTIONARGS	S
TOKEN1	<Actionid>
TOKEN2	<targeted>



```
<B t='b' r='RRVAT01'/>
<B l='Search' t='s' r='RRVAT02' n='fldsearch' v='S'/>
```

Here type defines the data type and node value defines the type of button and request id to be fired on that button. Node value should be ~ separated first value defines the type of button (for custom buttons this value is always set to be's') and second value defines the request id of that button and FUNCTIONARGS defines the value to be passed in the request with the name defines in NAME (in above case fldsearch=S)

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated

Adding Hidden Data

The entries to be done in *screentemplate* for this field are as follows

NAME	fldRequestId
ID	fldRequestId
TYPE	H

NODEVALUE	string('TFT02') or /faml/request/fldRequestId
TABLE	1

If a developer wants to assign a String value instead of XPATH, they can declare the node value as string (‘<value>’).

Behavior, for all static/dynamic datatypes

Form of behavior string::

set{value@key;value@key}#res{value@key}#vis{showView/hideView@key; hideView/showView@key}

here, various functions(set,res,vis) etc are # separated.

For static datatypes(SegmentedButton,Staticradio,StaticDropdown)::

Function column of screen template:

Comma separated list of static values, for which the behavior has to be applied.

Sample::

Block,UnBlock

FunctionArgs column of screen template(In case of static parameters, i.e the values to be set in the behavior string):

<First static value>:List of static parameters ‘,’comma separated, to be applied for first static value ^ <second static value>: List of static parameters ‘,’comma separated, to be applied for second static value and so on.

Sample::

Block:val1,val2^UnBlock:val3

DynamicFunctionArgs column of screen template(In case of dynamic parameters, i.e the values to be set in the behavior string)::

<First static value>: XPaths for fetching the dynamic parameters ’,’ comma separated ^ <second static value>: XPaths for fetching the dynamic parameters ’,’ comma separated, to be applied for second static value and so on.

Sample::

Block:Xpath1,Xpath2^UnBlock:Xpath3

<u>For dynamic datatypes(SearchDropDown,List)::</u>

Function column of screen template:

There can be two types of functions for dynamic datatypes::

- Type where the behavior string is common for all options of a list, only the values to be set are different. In this case you need to give * in this column.
- Type where the behavior string is different for all options of a list. In this case you need to mention the XPath for fetching dynamic values. **Please note that you need not mention the full XPath, just the value to be fetched from the XPath.**

Sample::
//faml/response/getaccountsresponsedto/custaccounts/customeraccountdto/accounts/accountnodto/nbraccount

Here, you just need to mention nbraccount, instead of the whole XPath.

FunctionArgs column of screen template(In case of static parameters):

<First value>:List of static parameters ‘,’comma separated, to be applied for first value ^<second value>: List of static parameters ‘,’comma separated, to be applied for second value and so on.

Sample::

nbraccount1:val1,val2 ^ nbraccount2:val3 ^ nbraccount3:val4,val5

Here you must have a pre-knowledge of all the possible dynamic values. For instance, here you must be knowing all the possible values of nbraccount.

DynamicFunctionArgs column of screen template(In case of dynamic parameters)::

List of XPaths for fetching these dynamic parameters ’,’ comma separated. Here also you need not mention the complete XPath, just the value to be fetched from the XPath.

Sample::

accountdisplayname,nbraccount

<u>Entry in ApplData table for both static/dynamic datatypes::</u>

DataName - Mobile_BH_<REQUEST ID>_<Field ID>

DataValue - <Static/dynamic Value for which behavior has to be applied>

ValueString - <FUNCTION_NAME>{PARAMETER_Place_HOLDERS>@<FIELD_ID_for which value has to be applied>}#<FUNCTION_NAME>{PARAMETER_Place_HOLDERS>@<FIELD_ID_for which value has to be applied>}

PARAMETER_Place_HOLDERS will contain tokens such as $S1, $S2 etc for static parameters and $D1,$D2 etc for dynamic parameters, which will be replaced dynamically.

Sample::

Static datatype::

DATANAME	DATAVALUE	VALUESTRING
Mobile_BH_RRSUC61fldstopunstopchq	Block	set{$S1@fld1;$D1@fld2;$S2@fld3}#upd{$D2@fld4}
Mobile_BH_RRSUC61fldstopunstopchq	UnBlock	set{$S1@fld1;$D1@fld2}

Dynamic datatype::

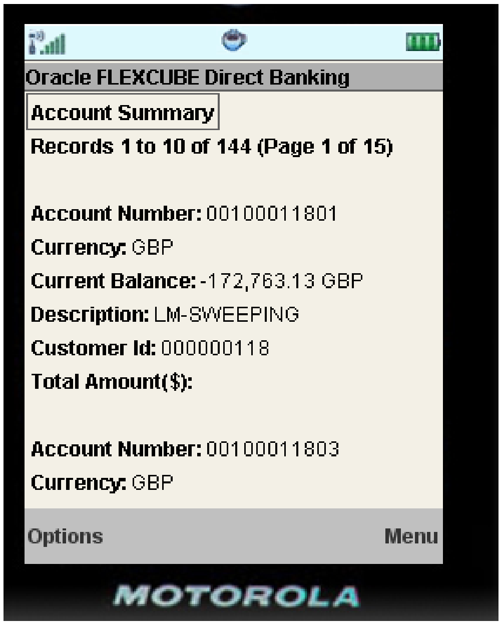
DATANAME	DATAVALUE	VALUESTRING
Mobile_BH_RRSUC61fldtxns	*	set{$D1@fld1;$D2@fld2}
Mobile_BH_RRVCD61fldacc	Nbraccoun1	set{$D1@fld1;$D2@fld2;$S1@fld3;$S2@fld4}
Mobile_BH_RRVCD61fldacc	Nbraccoun2	set{$D1@fld1;$D2@fld2;$S1@fld3}
Mobile_BH_RRVCD61fldacc	Nbraccoun3	set{$D1@fld1;$D2@fld2;$S1@fld3;$S2@fld4}

*, Is used, where the behavior string is common for all options of a list, else for each option you will be making a new entry in appldata

Adding UI Download (pagination data)

The following screenshot displays a UI download data (pagination) incorporated using the screen definition. To incorporate UI download there is **one** entry required in *screentemplate.*

.



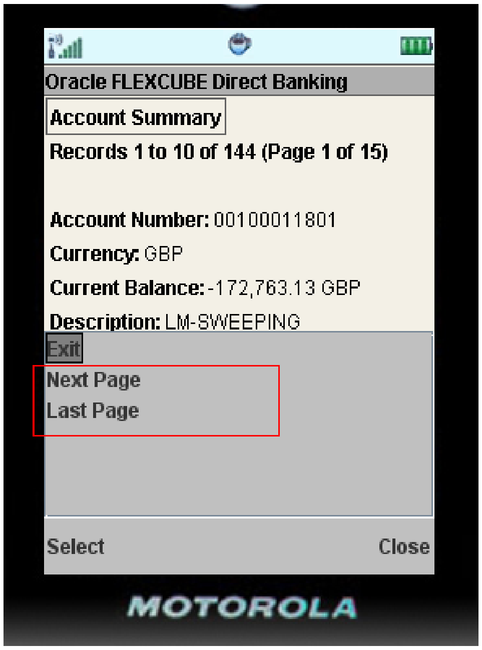
Oracle FLEXCUBE Direct Banking

Account Summary

Records 1 to 10 of 144 (Page 1 of 15)

Account Number: 00100011801
Currency: GBP
Current Balance: -172,763.13 GBP
Description: LM-SWEEPING
Customer Id: 000000118
Total Amount($):

Account Number: 00100011803
Currency: GBP

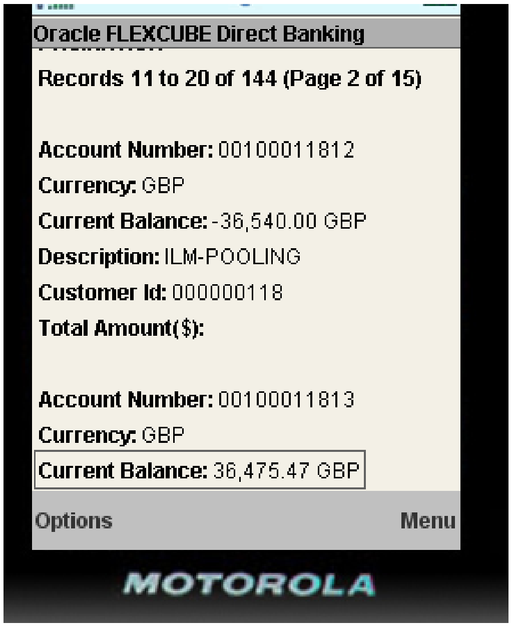


Oracle FLEXCUBE Direct Banking

Account Summary

Records 1 to 10 of 144 (Page 1 of 15)

Account Number: 00100011801
Currency: GBP
Current Balance: -172,763.13 GBP
Description: LM-SWEEPING

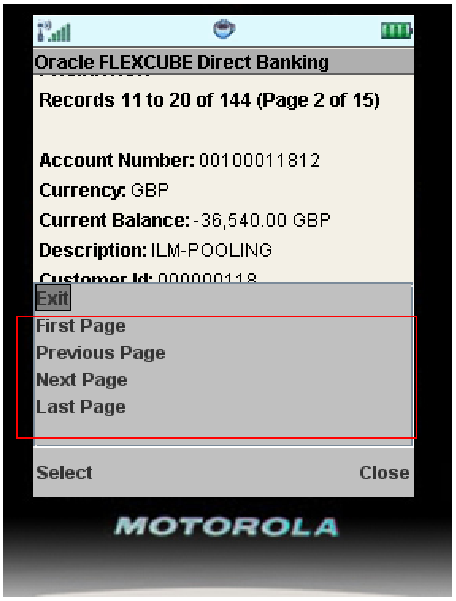


Oracle FLEXCUBE Direct Banking

Records 11 to 20 of 144 (Page 2 of 15)

Account Number: 00100011812
Currency: GBP
Current Balance: -36,540.00 GBP
Description: ILM-POOLING
Customer Id: 000000118
Total Amount($):

Account Number: 00100011813
Currency: GBP
Current Balance: 36,475.47 GBP



Oracle FLEXCUBE Direct Banking

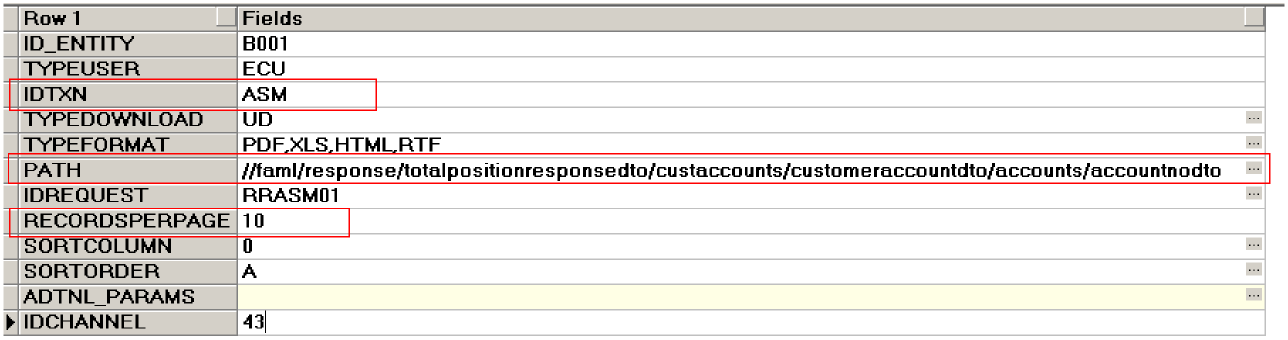
Records 11 to 20 of 144 (Page 2 of 15)

Account Number: 00100011812
Currency: GBP
Current Balance: -36,540.00 GBP
Description: ILM-POOLING
Customer Id: 000000118

IDTABLE	1
TYPE	PD

And rest screen definition entries should be done in **mstuidownloadparams** and **mstuidownload**

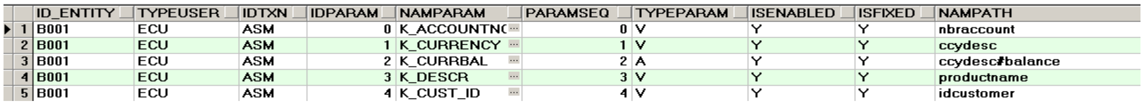
Entries of mstuidownload:



Row 1	Fields
ID_ENTITY	B001
TYPEUSER	ECU
IDTXN	ASM
TYPEDOWNLOAD	UD
TYPEFORMAT	PDF,XLS,HTML,RTF
PATH	//faml/response/totalpositionresponsedto/custaccounts/customeraccountdto/accounts/accountnodto
IDREQUEST	RRASM01
RECORDSPERPAGE	10
SORTCOLUMN	0
SORTORDER	A
ADTNL_PARAMS	
IDCHANNEL	43

PATH should be the XPATH where the actual data is coming.

Entries of mstuidownloadparams:



	ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
1	B001	ECU	ASM	0	K_ACCOUNTNC	0	V	Y	Y	nbraccount
2	B001	ECU	ASM	1	K_CURRENCY	1	V	Y	Y	ccydesc
3	B001	ECU	ASM	2	K_CURRBAL	2	A	Y	Y	ccydesc#balance
4	B001	ECU	ASM	3	K_DESCR	3	V	Y	Y	productname
5	B001	ECU	ASM	4	K_CUST_ID	4	V	Y	Y	idcustomer

Entries of screen components should be done in mstuidownloadparams in the sequence as on screen.

Currently supported data types in pagination:

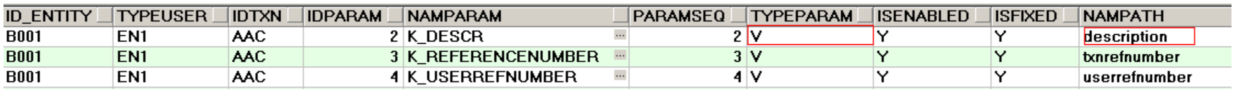
Currently only verify field data is supported in pagination

TYPEPARAM

- V (label and value pare as in screentemplate)
- A (label and value pare for amount field in formatted form)
- D (label and value pare for date field in formatted form)
- B (button)

For V data-type value of NAMPARAM would be the label and evaluated value of NAMPATH

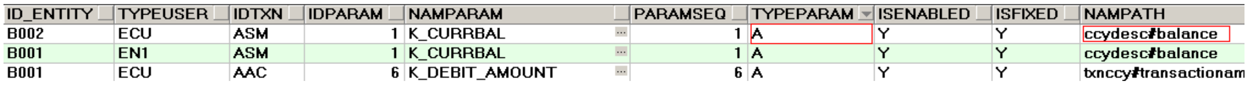
e.g.



ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B001	EN1	AAC	2	K_DESCR	2	V	Y	Y	description
B001	EN1	AAC	3	K_REFERENCENUMBER	3	V	Y	Y	txnrefnumber
B001	EN1	AAC	4	K_USERREFNUMBER	4	V	Y	Y	userrefnumber

For A data-type value of NAMPARAM would be the label and NAMPATH value should be # separated value of currency and amount ie (currency#amount) and internally this field would formatted by using JFFormatter .

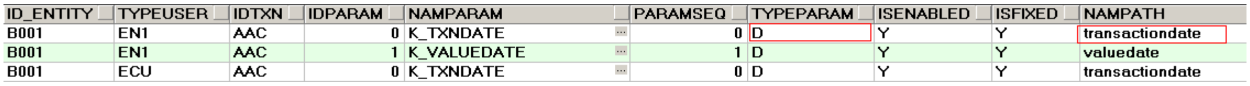
e.g.



ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B002	ECU	ASM	1	K_CURRBAL	1	A	Y	Y	ccydesc#balance
B001	EN1	ASM	1	K_CURRBAL	1	A	Y	Y	ccydesc#balance
B001	ECU	AAC	6	K_DEBIT_AMOUNT	6	A	Y	Y	txnccy#transactionam

For D data-type value of NAMPARAM would be the label and NAMPATH value should value of date and internally this field would formatted by using JFFormatter.

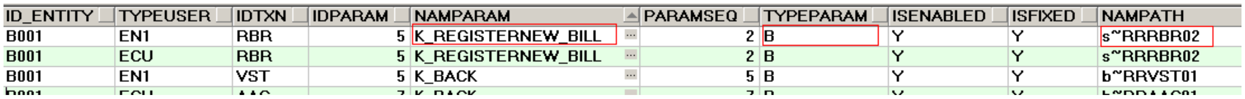
e.g.



ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B001	EN1	AAC	0	K_TXNDATE	0	D	Y	Y	transactiondate
B001	EN1	AAC	1	K_VALUEDATE	1	D	Y	Y	valuedate
B001	ECU	AAC	0	K_TXNDATE	0	D	Y	Y	transactiondate

For B data-type value of NAMPARAM would be the label of button and NAMPATH value should be ~separated value of button type and idrequestid

e.g.

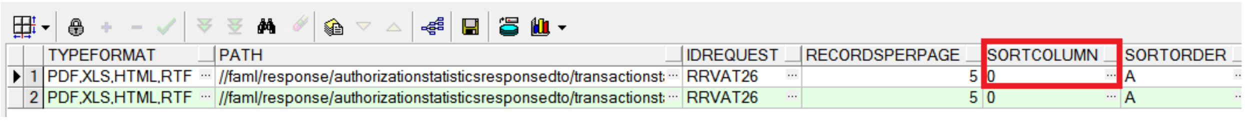


ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B001	EN1	RBR	5	K_REGISTERNEW_BILL	2	B	Y	Y	s~RRRBR02
B001	ECU	RBR	5	K_REGISTERNEW_BILL	2	B	Y	Y	s~RRRBR02
B001	EN1	VST	5	K_BACK	5	B	Y	Y	b~RRVST01
B001	ECU	AAC	7	K_BACK	7	B	Y	Y	b~RRAAC01

Sorting records is also possible. This can be achieved as following:

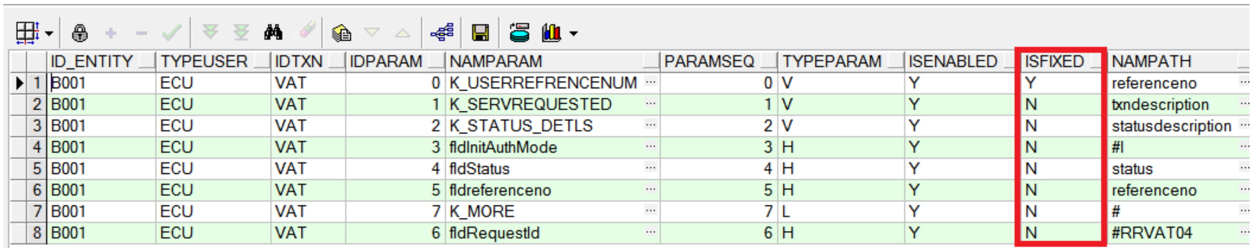
1.> Mstuidownload: column "SORTCOLUMN" will mention the idparam

(entry in mstuidownloadparams column "IDPARAM") which would form the basis for sorting the records.



	TYPEFORMAT	PATH	IDREQUEST	RECORDSPERPAGE	SORTCOLUMN	SORTORDER
1	PDF,XLS,HTML,RTF	//faml/response/authorizationstatisticsresponsedto/transactionst	RRVAT26	5	0	A
2	PDF,XLS,HTML,RTF	//faml/response/authorizationstatisticsresponsedto/transactionst	RRVAT26	5	0	A

2.> Also one of the fields in table mstuidownloadparams can be set as Header. This can be achieved by configuring a 'Y' in column "ISFIXED". Note only one column should be set as 'Y'. If multiple are set as 'Y' the one having lesser IDPARAM value will be picked by the framework as Header.

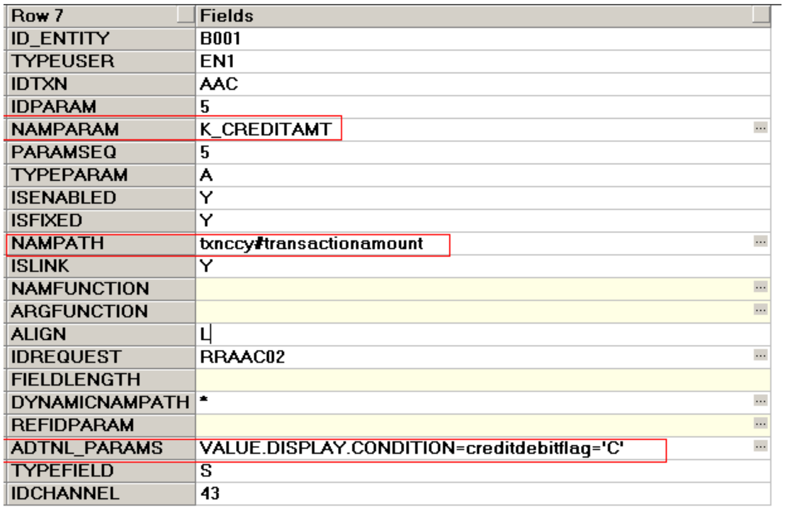


	ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
1	B001	ECU	VAT	0	K_USERREFRENCENUM	0	V	Y	Y	referenceno
2	B001	ECU	VAT	1	K_SERVREQUESTED	1	V	Y	N	txndescription
3	B001	ECU	VAT	2	K_STATUS_DETLS	2	V	Y	N	statusdescription
4	B001	ECU	VAT	3	fldInitAuthMode	3	H	Y	N	#I
5	B001	ECU	VAT	4	fldStatus	4	H	Y	N	status
6	B001	ECU	VAT	5	fldreferenceno	5	H	Y	N	referenceno
7	B001	ECU	VAT	7	K_MORE	7	L	Y	N	#
8	B001	ECU	VAT	6	fldRequestId	6	H	Y	N	#RRVAT04

Condition to show and hide a field can be added in pagination by doing an entry in column (ADTNL_PARAMS)

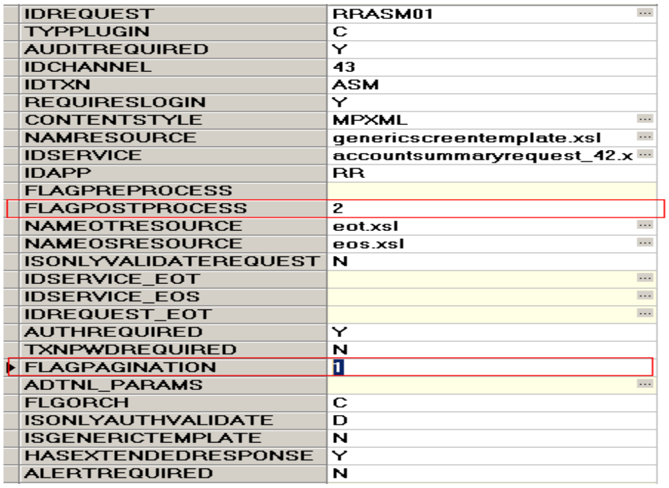
Entry should be like VALUE.DISPLAY.CONDITION= <*condition to show or hide*>

The entries to be done in mstuidownloadparams for this field are as follows



Row 7	Fields
ID_ENTITY	B001
TYPEUSER	EN1
IDTXN	AAC
IDPARAM	5
NAMPARAM	K_CREDITAMT
PARAMSEQ	5
TYPEPARAM	A
ISENABLED	Y
ISFIXED	Y
NAMPATH	txnccy#transactionamount
ISLINK	Y
NAMFUNCTION	
ARGFUNCTION	
ALIGN	L
IDREQUEST	RRAAC02
FIELDLENGTH	
DYNAMICNAMPATH	*
REFIDPARAM	
ADTNL_PARAMS	VALUE.DISPLAY.CONDITION=creditdebitflag='C'
TYPEFIELD	S
IDCHANNEL	43

MSTCHANNELATS configuration for Pagination:



IDREQUEST	RRASM01
TYPPLUGIN	C
AUDITREQUIRED	Y
IDCHANNEL	43
IDTXN	ASM
REQUIRESLOGIN	Y
CONTENTSTYLE	MPXML
NAMRESOURCE	genericscreentemplate.xsl
IDSERVICE	accountsummaryrequest_42.x
IDAPP	RR
FLAGPREPROCESS	
FLAGPOSTPROCESS	2
NAMEOTRESOURCE	eot.xsl
NAMEOSRESOURCE	eos.xsl
ISONLYVALIDATEREQUEST	N
IDSERVICE_EOT	
IDSERVICE_EOS	
IDREQUEST_EOT	
AUTHREQUIRED	Y
TXNPWDREQUIRED	N
FLAGPAGINATION	1
ADTNL_PARAMS	
FLGORCH	C
ISONLYAUTHVALIDATE	D
ISGENERICTEMPLATE	N
HASEXTENDEDRESPONSE	Y
ALERTREQUIRED	N

For pagination support use has to define FLAGPOSTPROCESS as 2 and FLAGPAGINATION as 1

Adding grouped buttons

It is used where buttons are required to be grouped together. Very similar to the 'list' type, here buttons are generated dynamically instead of labels.

The entries to be done in *screentemplate* for this data type are as follows:

LABELCLASS	AcctActDetails
LABELWIDTH	0.26~0.09
NAME	fldTxns
ID	fldTxns
DATACLASS	Select
TYPE	BG
IDROW	1
COLUMNID	1
TABLENO	5
NODEVALUE	//faml/response/authorizationstatisticsresponsedto/transactionstatistics/authorizationstatisticsperioddto/statistics/authorizationstatisticsdto~txndescription ~idtxn;status~status

ALT	K_TRANSACTIONS
DEFAULTSTATICLABEL	K_TRANSACTION_LIST
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
IMAGESRC	image
TOKEN1	V
TOKEN2	RRVAT641
TOKEN3	0.08
TOKEN4	RRVAT68
TOKEN5	3

LABELCLASS contains the respective CSS class for button label

Label Width indicates the width of the button ~ button height

NODEVALUE contains the labels that need to be displayed and values which need to be passed on selection of specific option under list. The nodevalue is given as, dto~button label~values(each separated by ‘;’

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token3(row height) contains the height of the row available for the list.

Token4 (reqid) contains Request Id.

Here RRVAT68 indicates the requestId which will become visible on selection of particular List.

Token5 (type) indicates Number of Columns in the group.

Adding graph area

It is used for creating a graph area

The entries to be done in *screentemplate* for this data type are as follows:

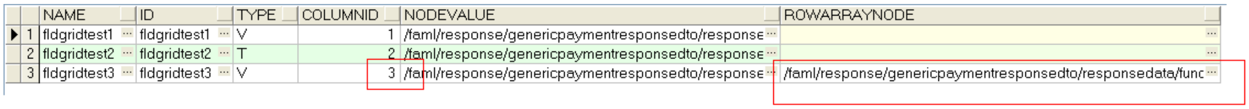
NAME	fldTxns
ID	fldTxns
TYPE	GR
IDROW	1
COLUMNID	1
TABLENO	5
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06

Adding data in for each loop

Data which is reapting on the basic of for each loop , for those perticular components

User has to define a separate table in screentabledefination. All entries in screentemplate should be done against that TABLENO and user has to give the xpath for for:loop in *rowarraynode* against the last COLUMNID's row

The entries to be done in *screentemplate* are as follows



	NAME	ID	TYPE	COLUMNID	NODEVALUE	ROWARRAYNODE
1	fldgridtest1	fldgridtest1	V	1	/faml/response/genericpaymentresponsedto/response	
2	fldgridtest2	fldgridtest2	T	2	/faml/response/genericpaymentresponsedto/response	
3	fldgridtest3	fldgridtest3	V	3	/faml/response/genericpaymentresponsedto/response	/faml/response/genericpaymentresponsedto/responsedata/func

The entries done in *screentemplate* for column nodevalue and rowarraynode is as follows

NAME	fldgridtest1
ID	fldgridtest1
TYPE	V
COLUMNID	1
NODEVALUE	((//srccustomeraccdto/customeraccountdto/accounts/accountnodto/nbraccount)[position()= $rowiteration]/preceding::idcustomer[1][. !=((//srccustomeraccdto/customeraccountdto/accounts/accountnodto/nbraccount)[position()= ($rowiteration)-1]/preceding::idcustomer[1]) or not((//fundtransferresponsedto/srccustomeraccdto/customeraccountdto/accounts/accountnodto/nbraccount)[position()=($rowiteration)-1])])
ROWARRAYNODE	

NAME	fldgridtest2
ID	fldgridtest2

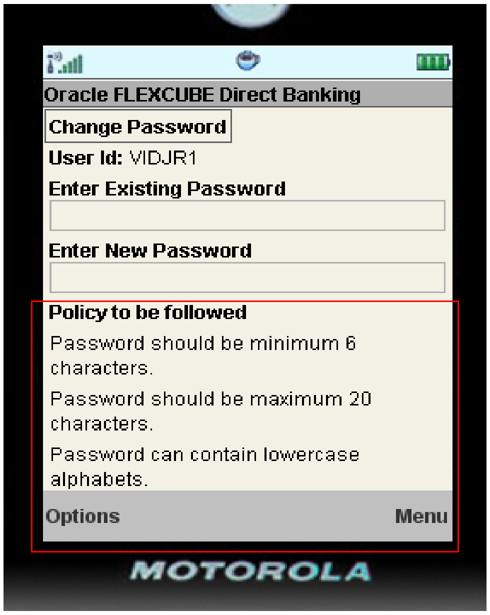
TYPE	T
COLUMNID	2
NODEVALUE	(/faml/response/genericpaymentresponsedto/responsedata/fundtransferresponsedto/srccustomeraccdto/customeraccountdto/accounts/accountnodto/nbraccount)[position()=$rowiteration]
ROWARRAYNODE	

NAME	fldgridtest3
ID	fldgridtest3
TYPE	V
COLUMNID	1
NODEVALUE	(/faml/response/genericpaymentresponsedto/responsedata/fundtransferresponsedto/srccustomeraccdto/customeraccountdto/accounts/accountnodto/availablebalance)[position()=$rowiteration]
ROWARRAYNODE	/faml/response/genericpaymentresponsedto/responsedata/fundtransferresponsedto/srccustomeraccdto/customeraccountdto/accounts/accountnodto/nbraccount

Now as shown in the entries above for nodevalue and rowarraynode, system will count the number of elements in the xpath given in rowarraynode, in the example given above system will count the nbraccount and create that number of rows in the grid layout. Node value contains the xpath that would be evaluated dynamically. To get the nodevalue on the basis of current position of loop an xsl variable rowiteration has been defined in the system, so a developer needs to use the same variable name while doing the entry in nodevalue as shown in the example above.

Adding Custom Templates

In certain scenarios, it may be possible that one may find certain limitations in placing their layout in the screen using the Screen Layout defined in the system, where one can define their customized templates and call them at desired position of the screen. In the sample screen below a developer has defined custom template policy information


Oracle FLEXCUBE Direct Banking

Change Password

User Id: VIDJR1

Enter Existing Password

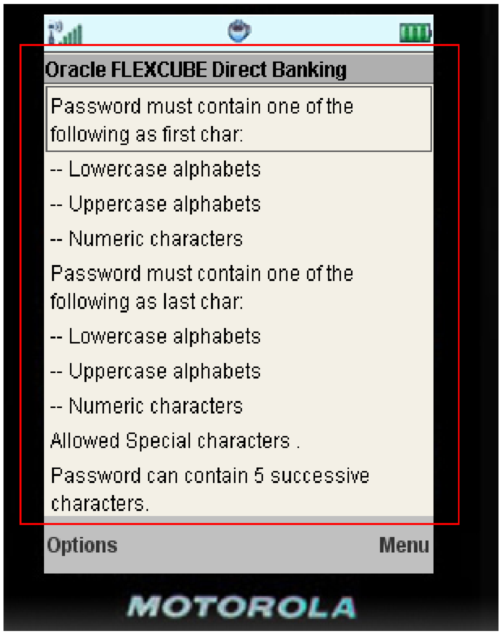
Enter New Password

Policy to be followed

Password should be minimum 6 characters.

Password should be maximum 20 characters.

Password can contain lowercase alphabets.


Oracle FLEXCUBE Direct Banking

Password must contain one of the following as first char:

-- Lowercase alphabets

-- Uppercase alphabets

-- Numeric characters

Password must contain one of the following as last char:

-- Lowercase alphabets

-- Uppercase alphabets

-- Numeric characters

Allowed Special characters .

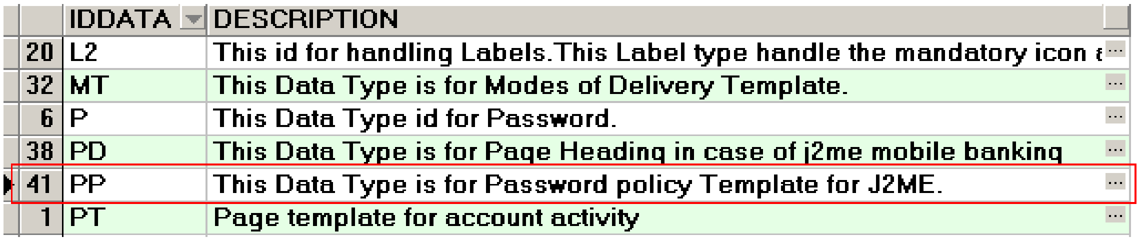
Password can contain 5 successive characters.

A developer needs to follow the following steps to create their custom XSL Templates.

- Declare your ID in msthtmldatatypes table.
- Use your new ID in TYPE column of screentemplate table.
- Write a template and call it from mleapscreenlayout.xsl on the basic of ID created.

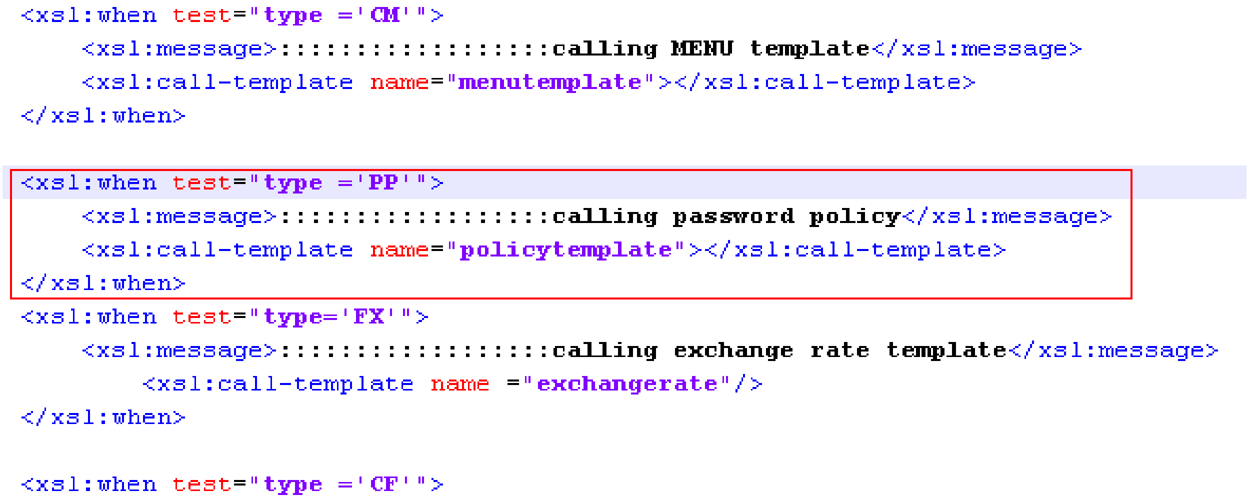
Below is the example of calling a custom template for password policy.

Entry in msthtmldatatypes table



	IDDATA	DESCRIPTION
20	L2	This id for handling Labels.This Label type handle the mandatory icon
32	MT	This Data Type is for Modes of Delivery Template.
6	P	This Data Type id for Password.
38	PD	This Data Type is for Page Heading in case of j2me mobile banking
41	PP	This Data Type is for Password policy Template for J2ME.
1	PT	Page template for account activity

Entry of mleapscreenlayout.xsl

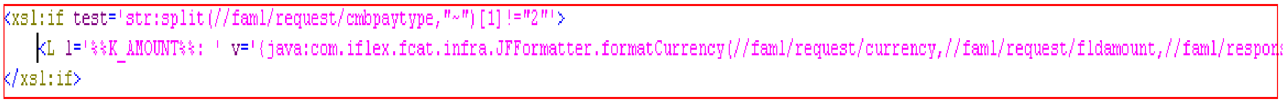


```
<xsl:when test="type ='CM'">
    <xsl:message>::::::::::::::::::calling MENU template</xsl:message>
    <xsl:call-template name="menutemplate"></xsl:call-template>
</xsl:when>

<xsl:when test="type ='PP'">
    <xsl:message>::::::::::::::::::calling password policy</xsl:message>
    <xsl:call-template name="policytemplate"></xsl:call-template>
</xsl:when>
<xsl:when test="type='FX'">
    <xsl:message>::::::::::::::::::calling exchange rate template</xsl:message>
        <xsl:call-template name ="exchangerate"/>
</xsl:when>

<xsl:when test="type ='CF'">
```

Adding Conditional fields

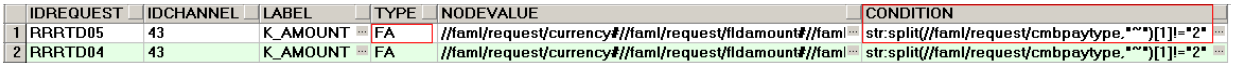
Conditional fields can be added by using a column in screentemplate table i.e. CONDITION

User has to configure the entry in screentemplate



```
<xsl:if test='str:split(//faml/request/cmbpaytype,"~")[1]!="2"'>
    <L l='%%K_AMOUNT%%: ' v='{java:com.iflex.fcat.infra.JFFormatter.formatCurrency(//faml/request/currency,//faml/request/fldamount,//faml/respon
</xsl:if>
```

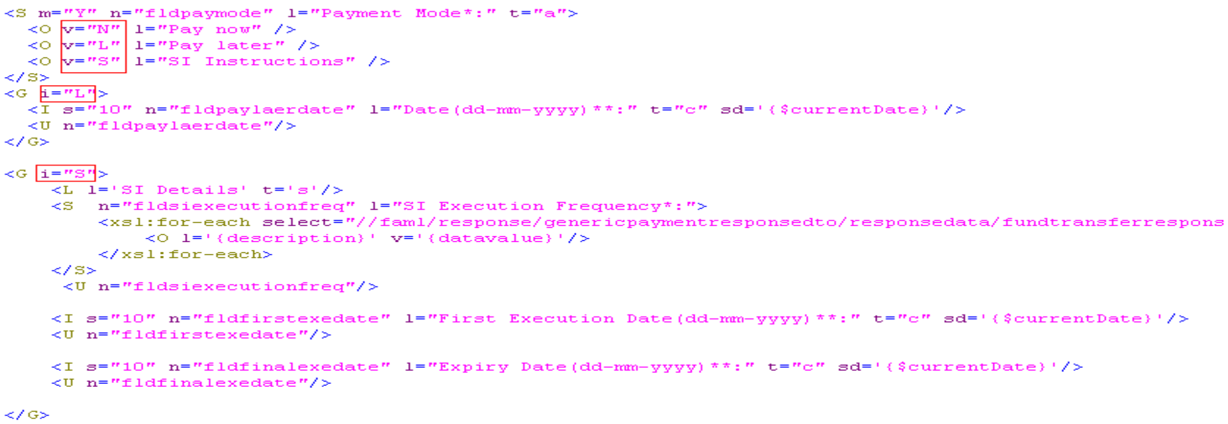
Equivalent entry in screentemplate for above example



	IDREQUEST	IDCHANNEL	LABEL	TYPE	NODEVALUE	CONDITION
1	RRRTD05	43	K_AMOUNT	FA	//faml/request/currency#//faml/request/fldamount#//faml	str:split(//faml/request/cmbpaytype,"~")[1]!="2"
2	RRRTD04	43	K_AMOUNT	FA	//faml/request/currency#//faml/request/fldamount#//faml	str:split(//faml/request/cmbpaytype,"~")[1]!="2"

Adding Dependent Fields

When a fields value is depend upon the change of a dropdown value in case of tablets We use a custom template named “tabcustomtemplate”, below is the snapshot of this template.



```
<S m="Y" n="fldpaymode" l="Payment Mode*:" t="a">
  <O v="N" l="Pay now" />
  <O v="L" l="Pay later" />
  <O v="S" l="SI Instructions" />
</S>
<G i="L">
  <I s="10" n="fldpaylaerdate" l="Date(dd-mm-yyyy)**:" t="c" sd='{$currentDate}'/>
  <U n="fldpaylaerdate"/>
</G>

<G i="S">
    <L l='SI Details' t='s'/>
    <S  n="fldsiexecutionfreq" l="SI Execution Frequency*:">
        <xsl:for-each select="//faml/response/genericpaymentresponsedto/responsedata/fundtransferrespons
            <O l='{description}' v='{datavalue}'/>
        </xsl:for-each>
    </S>
     <U n="fldsiexecutionfreq"/>

    <I s="10" n="fldfirstexedate" l="First Execution Date(dd-mm-yyyy)**:" t="c" sd='{$currentDate}'/>
    <U n="fldfirstexedate"/>

    <I s="10" n="fldfinalexedate" l="Expiry Date(dd-mm-yyyy)**:" t="c" sd='{$currentDate}'/>
    <U n="fldfinalexedate"/>

</G>
```

Here dependent fields are grouped by a tag **"<G>"** , this tag has a attribute **(i)** which is the id of grouped fields which needs to be shown on change of dropdown value . Value of **"i"** should be same as the value of dropdown's option attribute **("v")**

In the above example in case of pay now only one date field should be visible

And for SI Instruction 3 field's frequency, first execution and last execution should be visible. In this case value of option is used as the group id (<G i='value of dropdown option'>

e.g. dropdown value of pay now is "**N**" (<O **v='N'** l='Pay Now') hence the dependent fields are grouped under <G **i='N'**>.

Adding Accounts Dropdown with optgroup

The entries to be done in *screentemplate* for this field are as follows

LABEL	K_SELECTACCOUNT
NAME	fldacct
ID	fldacct

TYPE	SA
NODEVALUE	//faml/response/accountdetailsresponsedto/custaccounts/customeraccountdto/accounts/accountnodto~accountdisplayname~nbraccount,nbrbranch
IDROW	1
COLUMNID	1
TABLENO	1
ISUDF	Y
ALT	K_SELECTACCOUNT

Adding dynamic Segmented Button

This Data Type Creates a dynamic Segmented Button.

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL
NAME	fldstopunstopchq
ID	fldstopunstopchq
DATACLASS	SegmentedButton
TYPE	DSB
IDROW	1
COLUMNID	1
TABLENO	5
NODEVALUE	//faml/response/authorizationstatisticsresponsedto/transactionstatistics/authorizationstatisticsperioddto/statistics/authorizationstatisticsdto~txndescription;count;status;description~idtxn;status

RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	V
TOKEN2	
TOKEN3	
TOKEN4	RRVAT62
TOKEN5	H

NODEVALUE contains the name of the buttons that need to be displayed and values which need to be passed on selection of specific button. The nodevalue is given as, dto~displayname~values(each separated by ‘;’).

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token3 (valueindex) contains value index

Value index is the value of that button which is pre selected, for the dynamic segmented button.

Token4 (reqid) contains Request Id.

Here RRVAT62 indicates the requestId which will become visible on selection of particular Button in Segmented Button.

Token5 (type) indicates orientation of Segmented Button. The orientation can be either horizontal (H) or vertical (V).

Here ‘H’ indicates that the Segmented Button will be horizontally aligned.

Adding dynamic Radio Button

This Data Type Creates a dynamic radio button

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL
NAME	fldstopunstopchq
ID	fldstopunstopchq
DATACLASS	Radio
TYPE	DR
IDROW	1
COLUMNID	1
TABLENO	5
NODEVALUE	//faml/response/authorizationstatisticsresponsedto/transactionstatistics/authorizationstatisticsperioddto/statistics/authorizationstatisticsdto~txndescription;count;status;description~idtxn;status
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	V
TOKEN2	
TOKEN4	RRVAT62
TOKEN5	H

NODEVALUE contains the name of the buttons that need to be displayed and values which need to be passed on selection of specific button. The nodevalue is given as, dto~displayname~values(each separated by ‘;’).

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token4 (reqid) contains Request Id.

Here RRVAT62 indicates the requestId which will become visible on selection of particular Button in Segmented Button.

Token5 (type) indicates orientation of Segmented Button. The orientation can be either horizontal (H) or vertical (V).

Here ‘H’ indicates that the Segmented Button will be horizontally aligned.

Adding dynamic CheckBox Button

This Data Type Creates a dynamic checkbox button

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL

NAME	fldstopunstopchq
ID	fldstopunstopchq
DATACLASS	Checkbox
TYPE	DCB
IDROW	1
COLUMNID	1
TABLENO	5
NODEVALUE	//faml/response/authorizationstatisticsresponsedto/transactionstatistics/authorizationstatisticsperioddto/statistics/authorizationstatisticsdto~txndescription;count;status;description~idtxn;status
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	2
TOKEN5	H

NODEVALUE contains the name of the buttons that need to be displayed and values which need to be passed on selection of specific button. The nodevalue is given as, dto~displayname~values(each separated by ';').

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 denotes the number of columns to be given for checkboxes.

Token5 (type) indicates orientation of Segmented Button. The orientation can be either horizontal (H) or vertical (V).

Here 'H' indicates that the Segmented Button will be horizontally aligned.

Adding Lookup Dropdown

It is used to create a lookup dropdown.

The entries to be done in *screentemplate* for this data type are as follows:

LABELCLASS	AcctActDetails~AcctActDetails~AcctActDetails
LABELWIDTH	0.26~0.09~0.08#0.08~0.08~0.06
NAME	fldTxns
ID	fldTxns
DATACLASS	Select
TYPE	LD
IDROW	1
COLUMNID	1
TABLENO	5
NODEVALUE	//faml/response/authorizationstatisticsresponsedto/transactionstatistics/authorizationstatisticsperioddto/statistics/authorizationstatisticsdto~txndescription;count;status;description~idtxn;status~status
ALT	K_TRANSACTIONS
DEFAULTSTATICLABEL	K_TRANSACTION_LIST
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGH	0.06

T	
TOKEN1	
TOKEN2	RRVAT641
TOKEN3	0.08
TOKEN4	RRVAT68
TOKEN5	R~3

LABELCLASS contains the respective CSS class for components which will appear on the List.

For Example here “AcctActDetails~AcctActDetails~AcctActDetails” three label class name are present. First class is for entire list structure, second is for list and third is for options in list.

Label Width indicates the width corresponding to the label class separated by ~ # labelheight, for each label, separated by ~.

Example 0.26~0.09~0.08#0.08~0.08~0.06

Here 0.26 is the width for 1st label and 0.097 is the width for second label and 0.08 for third. Similarly 0.08 is the height for 1st label and 0.08 is for 2nd and so on.

NODEVALUE contains the labels that need to be displayed and values which need to be passed on selection of specific option under list. The nodevalue is given as, dto~displayname(each separated by ‘;’)~values(each separated by ‘;’))~value on which the list has to be filtered, only if required. In case display name consists of amount or date. Please enter the displayname as FA#codcurrency#balance#idlang, in case of amount and FD#dateformat #date#idlang, in case of date.

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 (actionid) Contains the Name of “Template Screen” which is invoked on Client Side to present the data. For example ‘V’ denotes a Verify Template Screen and ‘M’ denotes the Confirm Template Screen and so on.

Token2 (targetid) Contains the target Id (id of view or panel) in which data is being populated. It conatins the requestid of the screen on which entire content will become visible, appended by the table number in which the content will be populated.

Token2(tableid) specifies the table id in which the list will get populated. actionid is used either with targeted or tableid. It is not used with both at same time. For example if transaction requires list to become visible in table then tableid is used and if transaction needs to populate list on template screen then tragetid is used.

Token3(option height) contains the height of the options available for the list.

Here 0.08 specifies that each option within the list will be of height 0.08.

Token4 (reqid) contains Request Id.

Here RRVAT68 indicates the requestId which will become visible on selection of particular List.

Token5 (type) indicates type of List along with number of columns separated by ~.

It is mentioned as: Type~Number of Columns in List.

Adding Baretrail

It is used to create a baretrail/swimlanes type of a structure

The entries to be done in *screentemplate* for this data type are as follows:

NAME	fldTxns
ID	fldTxns
DATACLASS	<CSS class>
TYPE	BT

IDROW	1
COLUMNID	1
TABLENO	5
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	<Enabled stages>
TOKEN2	<initial completed stages>

Token1 describes the the stage whose Fields are to enabled initially(~ seperated).

Token2 gives the initial(on first Load) completed stages(~ seperated).

Adding Widget Button

This Data Type Creates a widget button .

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL
NAME	fldstopunstopchq
ID	fldstopunstopchq
DATACLASS	widgetbutton
TYPE	WB
IDROW	1
COLUMNID	1

TABLENO	5
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	<contentid>
TOKEN2	<request id>
TOKEN3	<type>
TOKEN4	<widgetid>
IMAGESRC	<imagesrc>

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 represents the contentid

Token2 represents the request id .

Token3 represents the type of widget button.

Token4 represents the widgetid.

ImageSrc gives the name of the image.

Adding Widget View

This Data Type Creates a widget view

The entries to be done in *screentemplate* for this data type are as follows:

LABEL	K_PL
NAME	fldstopunstopchq
ID	fldstopunstopchq
DATACLASS	widgetbutton
TYPE	WV
IDROW	1
COLUMNID	1
TABLENO	5
RELPOSX	0.40
RELPOSY	0.02
RELWIDTH	0.20
RELHEIGHT	0.06
TOKEN1	<contentid>
TOKEN2	<request id>
TOKEN3	<type>

Here RELPOSX, RELPOSY, RELWIDTH, RELHEIGHT indicates Left Margin, Top margin, Width, Height of the component respectively and contains value ranging from 0.0 – (any float value).

Token1 represents the contentid

Token2 represents the request id .

Token3 represents the type of widget button.

Browser Based Mobile – XSL Design

The LEAP framework has been designed using following XSL's.

XSL Name	Package
genericscreentemplate.xsl	com\iflex\fcat\datafiles\gui\ENU\42\template
genericscreenlayout.xsl	com\iflex\fcat\datafiles\gui\ENU\42\template
htmldatatypes.xsl	com\iflex\fcat\datafiles\gui\ENU\42\template
htmldatatypesextensions.xsl	com\iflex\fcat\datafiles\gui\ENU\42\template
uidownload.xsl	com\iflex\fcat\datafiles\gui\ENU\42\template
commonftfunctions.xsl	com\iflex\fcat\datafiles\gui\ENU\42\template

The logic for rendering the screen on the basis of screen definition xml has been written in the above set of XSL's.

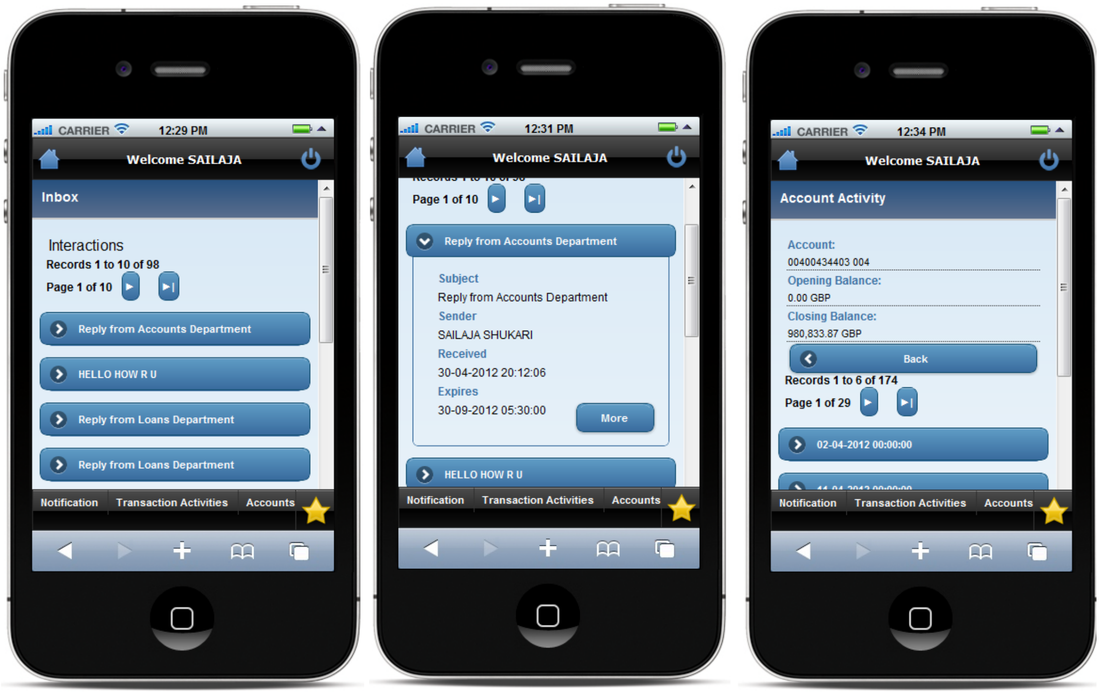
Browser Based Mobile – Supported mLEAP Data Types

IDDATA	DESCRIPTION
T	This Data Type is for creating Text Boxes.
P	This Data Type id for Password.
D	Drop Down
D1	Static Drop Down
V	This Data Type is for Verification Fields
B	This Data Type is for Buttons.
FB	This Data Type is for Different Button on different form
FA	This Data Type is for Formatted Amount.
FD	This Data Type is for Formatted Date.
TZ	This Data Type is for Formatted Date with time zone.
FU	This Data Type is for Formatted unit(amount, number)
H	This Data Type is for Hidden Fields.
D1	Drop Down with static values
PD	This Id is for pagination data
SA	This Id is for custom template of account dropdown
FX	This Id is for custom template of exchange rate
TA	This Data Type id for Text Area

PP	This Id is for custom template of password policy
FP	This Id is for custom template of password policy in case of force change password
RB	This Id is for custom template of Register Biller

Adding UI Download (Pagination Data)

The following screenshot displays a UI download data (pagination) incorporated using the screen definition. To incorporate UI download there is **one** entry required in *screentemplate.*



Records 1 to 10 of 98
Page 1 of 10

Reply from Accounts Department
HELLO HOW R U
Reply from Loans Department
Reply from Loans Department

Page 1 of 10

Reply from Accounts Department

Subject
Reply from Accounts Department
Sender
SAILAJA SHUKARI
Received
30-04-2012 20:12:06
Expires
30-09-2012 05:30:00

HELLO HOW R U

Account:
00400434403 004
Opening Balance:
0.00 GBP
Closing Balance:
980,833.87 GBP

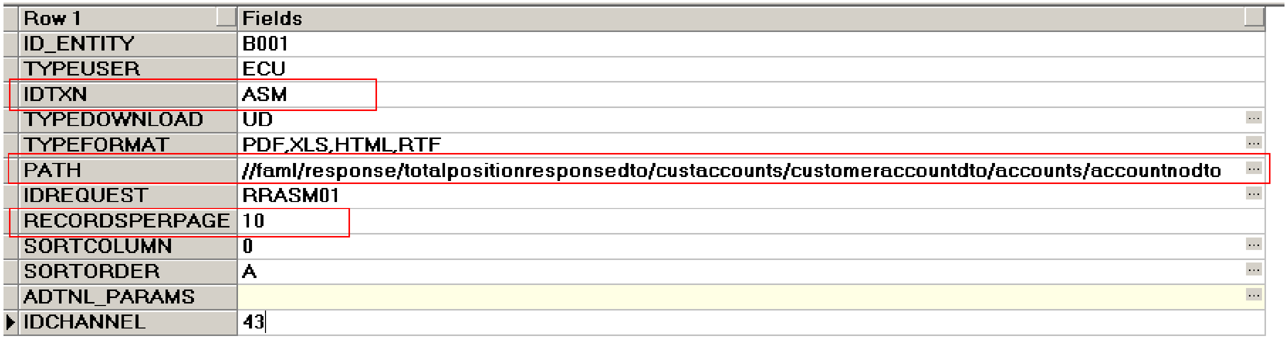
Records 1 to 6 of 174
Page 1 of 29

02-04-2012 00:00:00

IDTABLE	1
TYPE	PD

And rest screen definition entries should be done in **mstuidownloadparams** and **mstuidownload**

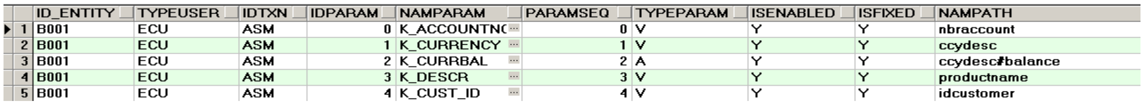
Entries of mstuidownload:



Row 1	Fields
ID_ENTITY	B001
TYPEUSER	ECU
IDTXN	ASM
TYPEDOWNLOAD	UD
TYPEFORMAT	PDF,XLS,HTML,RTF
PATH	//faml/response/totalpositionresponsedto/custaccounts/customeraccountdto/accounts/accountnodto
IDREQUEST	RRASM01
RECORDSPERPAGE	10
SORTCOLUMN	0
SORTORDER	A
ADTNL_PARAMS	
IDCHANNEL	43

PATH should be the XPATH where the actual data is coming.

Entries of mstuidownloadparams:



	ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
1	B001	ECU	ASM	0	K_ACCOUNTNC	0	V	Y	Y	nbraccount
2	B001	ECU	ASM	1	K_CURRENCY	1	V	Y	Y	ccydesc
3	B001	ECU	ASM	2	K_CURRBAL	2	A	Y	Y	ccydesc#balance
4	B001	ECU	ASM	3	K_DESCR	3	V	Y	Y	productname
5	B001	ECU	ASM	4	K_CUST_ID	4	V	Y	Y	idcustomer

Entries of screen components should be done in mstuidownloadparams in the sequence as on screen.

Currently supported data types in pagination:

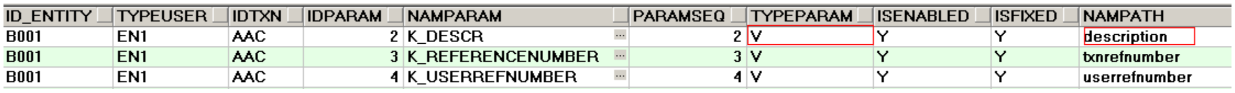
Currently only verify field data is supported in pagination

TYPEPARAM

- V (label and value pare as in screentemplate)
- A (label and value pare for amount field in formatted form)
- D (label and value pare for date field in formatted form)
- B (button)
- L (Link)
- H (Hidden Fields)
- T (Timestamp)

For V data-type value of NAMPARAM would be the label and evaluated value of NAMPATH

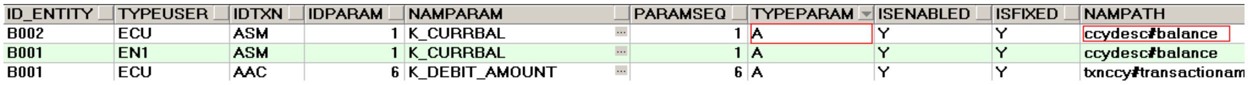
e.g.



ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B001	EN1	AAC	2	K_DESCR	2	V	Y	Y	description
B001	EN1	AAC	3	K_REFERENCENUMBER	3	V	Y	Y	txnrefnumber
B001	EN1	AAC	4	K_USERREFNUMBER	4	V	Y	Y	userrefnumber

For A data-type value of NAMPARAM would be the label and NAMPATH value should be # separated value of currency and amount ie (currency#amount) and internally this field would formatted by using JFFormatter .

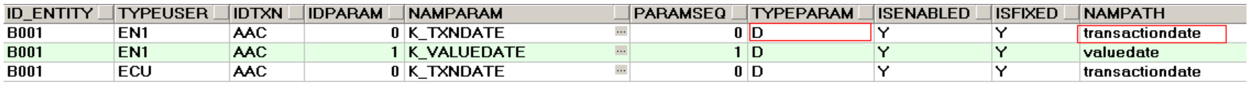
e.g.



ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B002	ECU	ASM	1	K_CURRBAL	1	A	Y	Y	ccydesc#balance
B001	EN1	ASM	1	K_CURRBAL	1	A	Y	Y	ccydesc#balance
B001	ECU	AAC	6	K_DEBIT_AMOUNT	6	A	Y	Y	txnccy#transactionam

For D data-type value of NAMPARAM would be the label and NAMPATH value should value of date and internally this field would formatted by using JFFormatter.

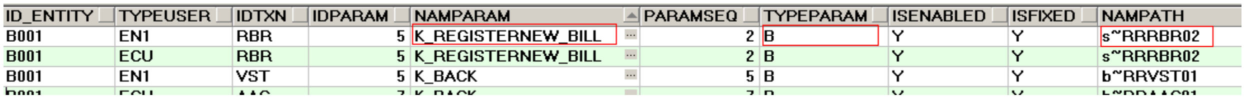
e.g.



ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B001	EN1	AAC	0	K_TXNDATE	0	D	Y	Y	transactiondate
B001	EN1	AAC	1	K_VALUEDATE	1	D	Y	Y	valuedate
B001	ECU	AAC	0	K_TXNDATE	0	D	Y	Y	transactiondate

For B data-type value of NAMPARAM would be the label of button and NAMPATH value should be ~separated value of button type and idrequestid

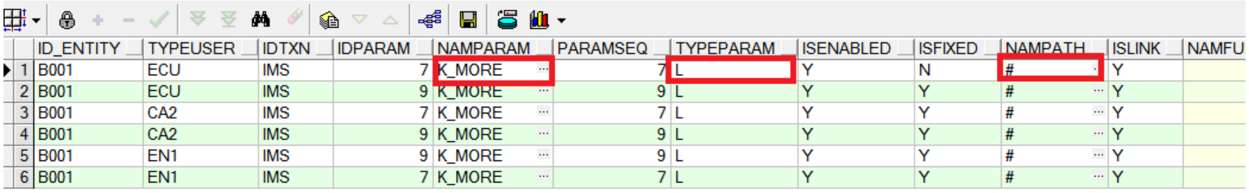
e.g.



ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
B001	EN1	RBR	5	K_REGISTERNEW_BILL	2	B	Y	Y	s~RRRBR02
B001	ECU	RBR	5	K_REGISTERNEW_BILL	2	B	Y	Y	s~RRRBR02
B001	EN1	VST	5	K_BACK	5	B	Y	Y	b~RRVST01

For L data-type value of NAMPARAM would be the label of link button and NAMPATH value should be just: “#”.This would tell the pagination framework to paint it like a Link button. This allows a user to migrate from pagination framework to any of the mapped transaction screen. In other words this enables form submission which can be used to switch to a different transaction screen.

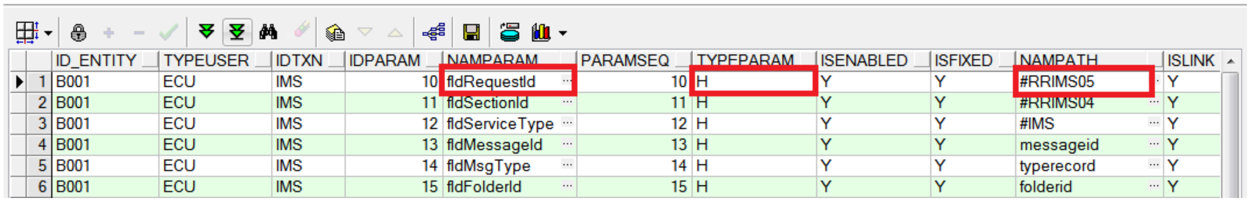
e.g.



	ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH	ISLINK	NAMFU
1	B001	ECU	IMS	7	K_MORE	7	L	Y	N	#	Y	
2	B001	ECU	IMS	9	K_MORE	9	L	Y	Y	#	Y	
3	B001	CA2	IMS	7	K_MORE	7	L	Y	Y	#	Y	
4	B001	CA2	IMS	9	K_MORE	9	L	Y	Y	#	Y	
5	B001	EN1	IMS	9	K_MORE	9	L	Y	Y	#	Y	
6	B001	EN1	IMS	7	K_MORE	7	L	Y	Y	#	Y	

For H data-type value of NAMPARM would be the name of the field and NAMPATH value will be in the format: “ #<value>” . “value” being the value given to the hidden field. This helps in creating the hidden fields that are required by the screen/transaction to which one migrates via a link (typeparm L).

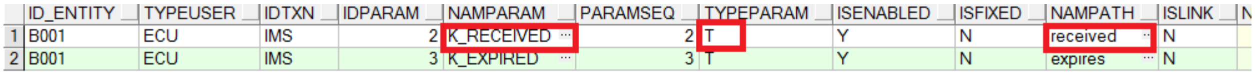
e.g.



	ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH	ISLINK
1	B001	ECU	IMS	10	fldRequestId	10	H	Y	Y	#RRIMS05	Y
2	B001	ECU	IMS	11	fldSectionId	11	H	Y	Y	#RRIMS04	Y
3	B001	ECU	IMS	12	fldServiceType	12	H	Y	Y	#IMS	Y
4	B001	ECU	IMS	13	fldMessageId	13	H	Y	Y	messageid	Y
5	B001	ECU	IMS	14	fldMsgType	14	H	Y	Y	typerecord	Y
6	B001	ECU	IMS	15	fldFolderId	15	H	Y	Y	folderid	Y

For T data-type value of NAMPARAM would be the label and NAMPATH value should value of date and internally this field would formatted by using JFFormatter.

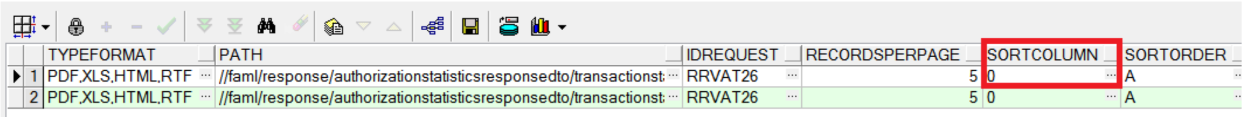
e.g.



	ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH	ISLINK	N
1	B001	ECU	IMS	2	K_RECEIVED	2	T	Y	N	received	N	
2	B001	ECU	IMS	3	K_EXPIRED	3	T	Y	N	expires	N	

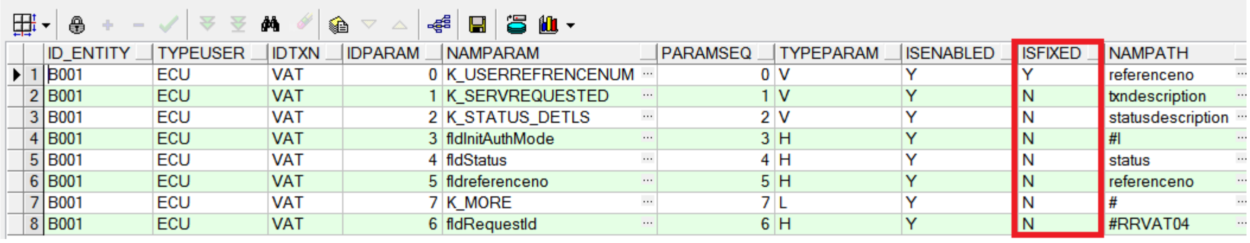
Sorting records is also possible. This can be achieved as following:

3.> Mstuidownload: column “SORTCOLUMN” will mention the idparam
(entry in mstuidownloadparams column “IDPARAM”) which would form the basis for sorting the records.

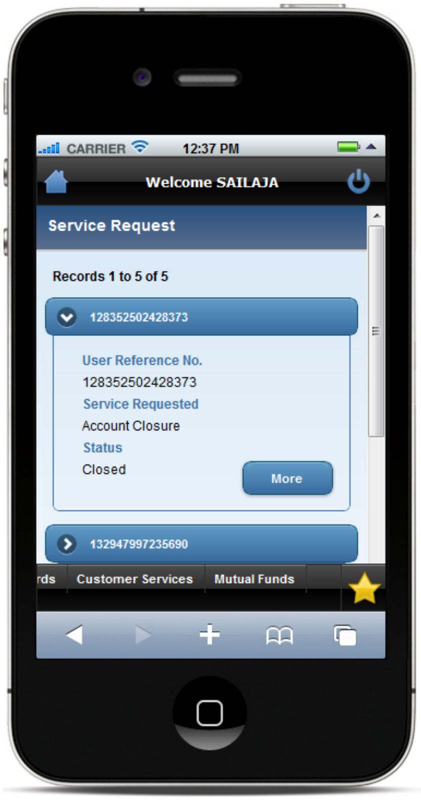


	TYPEFORMAT	PATH	IDREQUEST	RECORDSPERPAGE	SORTCOLUMN	SORTORDER
1	PDF,XLS,HTML,RTF	//faml/response/authorizationstatisticsresponsedto/transactionst	RRVAT26	5	0	A
2	PDF,XLS,HTML,RTF	//faml/response/authorizationstatisticsresponsedto/transactionst	RRVAT26	5	0	A

4.> Also one of the fields in table mstuidownloadparams can be set as Header. This can be achieved by configuring a ‘Y’ in column “ISFIXED”. Note only one column should be set as ‘Y’. If multiple are set as ‘Y’ the one having lesser IDPARAM value will be picked by the framework as Header.



	ID_ENTITY	TYPEUSER	IDTXN	IDPARAM	NAMPARAM	PARAMSEQ	TYPEPARAM	ISENABLED	ISFIXED	NAMPATH
1	B001	ECU	VAT	0	K_USERREFRENCENUM	0	V	Y	Y	referenceno
2	B001	ECU	VAT	1	K_SERVREQUESTED	1	V	Y	N	txndescription
3	B001	ECU	VAT	2	K_STATUS_DETLS	2	V	Y	N	statusdescription
4	B001	ECU	VAT	3	fldInitAuthMode	3	H	Y	N	#I
5	B001	ECU	VAT	4	fldStatus	4	H	Y	N	status
6	B001	ECU	VAT	5	fldreferenceno	5	H	Y	N	referenceno
7	B001	ECU	VAT	7	K_MORE	7	L	Y	N	#
8	B001	ECU	VAT	6	fldRequestId	6	H	Y	N	#RRVAT04



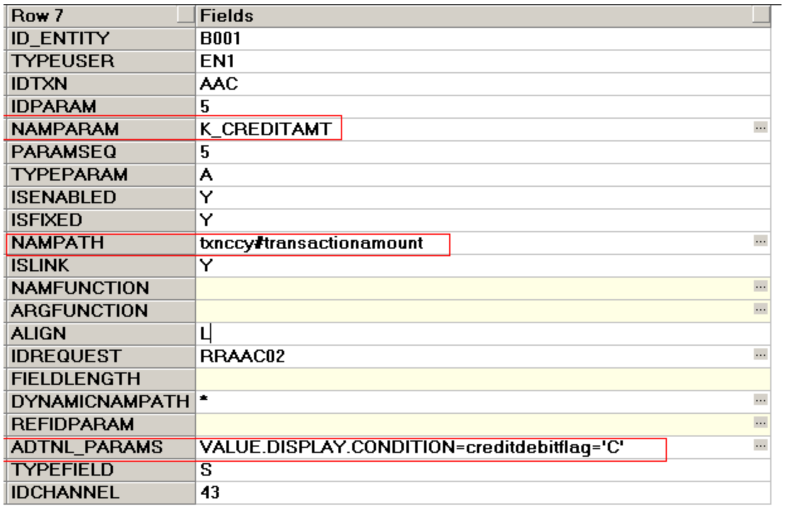
User Reference No.
128352502428373
Service Requested
Account Closure
Status
Closed

This example shows the header being set as per configuration settings mentioned above.

Condition to show and hide a field can be added in pagination by doing an entry in column (ADTNL_PARAMS)

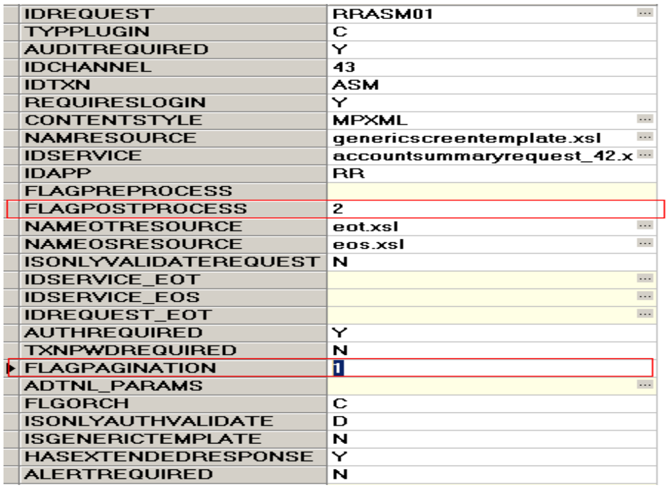
Entry should be like VALUE.DISPLAY.CONDITION= <*condition to show or hide*>

The entries to be done in mstuidownloadparams for this field are as follows



Row 7	Fields
ID_ENTITY	B001
TYPEUSER	EN1
IDTXN	AAC
IDPARAM	5
NAMPARAM	K_CREDITAMT
PARAMSEQ	5
TYPEPARAM	A
ISENABLED	Y
ISFIXED	Y
NAMPATH	txnccy#transactionamount
ISLINK	Y
NAMFUNCTION	
ARGFUNCTION	
ALIGN	L
IDREQUEST	RRAAC02
FIELDLENGTH	
DYNAMICNAMPATH	*
REFIDPARAM	
ADTNL_PARAMS	VALUE.DISPLAY.CONDITION=creditdebitflag='C'
TYPEFIELD	S
IDCHANNEL	43

MSTCHANNELATS configuration for Pagination:

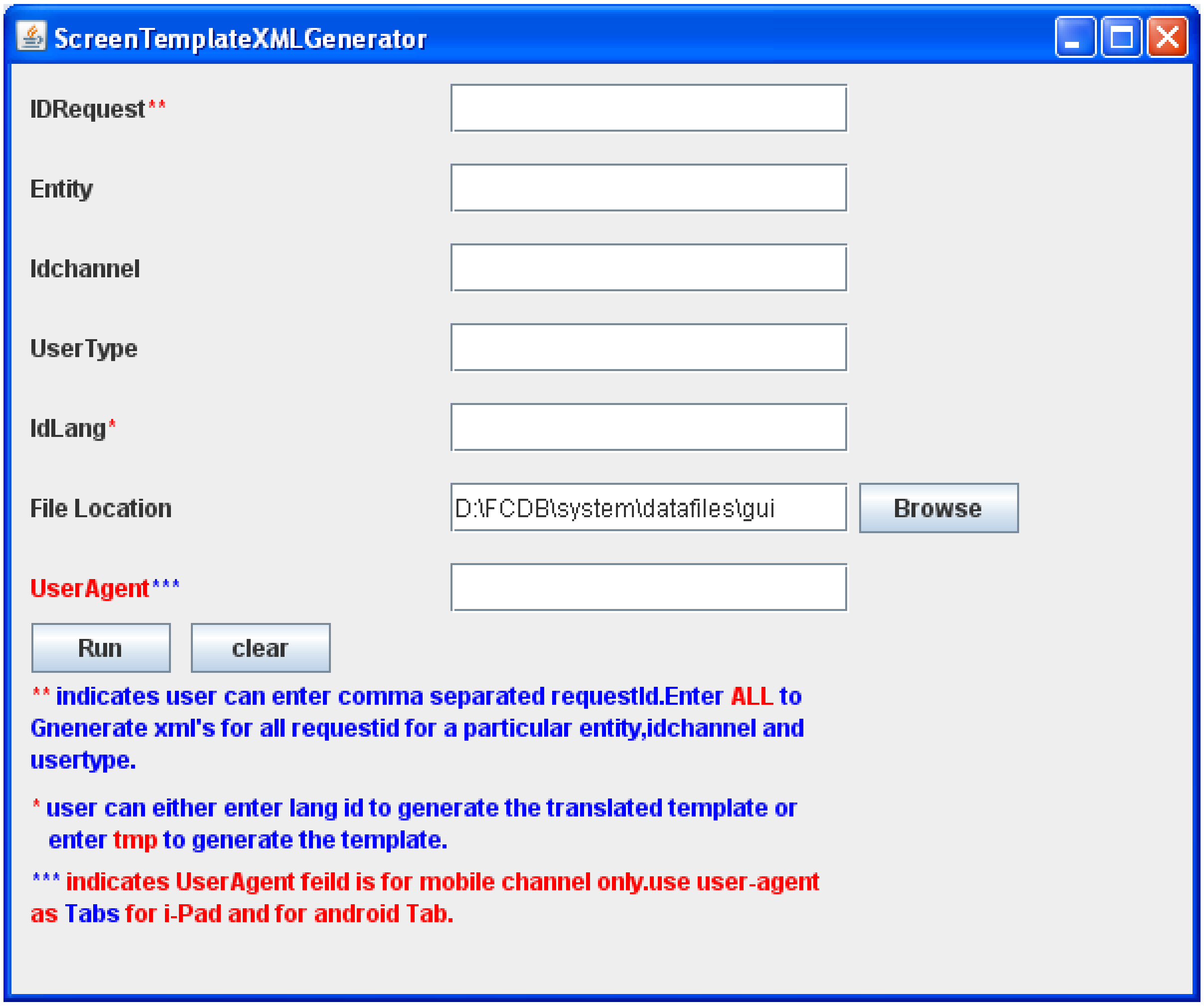


IDREQUEST	RRASM01
TYPPLUGIN	C
AUDITREQUIRED	Y
IDCHANNEL	43
IDTXN	ASM
REQUIRESLOGIN	Y
CONTENTSTYLE	MPXML
NAMRESOURCE	genericscreentemplate.xsl
IDSERVICE	accountsummaryrequest_42.x
IDAPP	RR
FLAGPREPROCESS	
FLAGPOSTPROCESS	2
NAMEOTRESOURCE	eot.xsl
NAMEOSRESOURCE	eos.xsl
ISONLYVALIDATEREQUEST	N
IDSERVICE_EOT	
IDSERVICE_EOS	
IDREQUEST_EOT	
AUTHREQUIRED	Y
TXNPWDREQUIRED	N
FLAGPAGINATION	1
ADTNL_PARAMS	
FLGORCH	C
ISONLYAUTHVALIDATE	D
ISGENERICTEMPLATE	N
HASEXTENDEDRESPONSE	Y
ALERTREQUIRED	N

For pagination support use has to define FLAGPOSTPROCESS as 2 and FLAGPAGINATION as 1

Generating the Screen Template XML

Once developer has defined all the screen components in the desired tables they need to generate a screen template xml. A utility "*ScreenTemplateGUI*" has been provided to generate the XML at a specific location. This utility has been placed in the package com.iflex.fcat.services.tools.internal.



File Location D:\FCDB\system\datafiles\gui

** indicates user can enter comma separated requestId.Enter ALL to Gnenerate xml's for all requestid for a particular entity,idchannel and usertype.

* user can either enter lang id to generate the translated template or enter tmp to generate the template.

*** indicates UserAgent feild is for mobile channel only.use user-agent as Tabs for i-Pad and for android Tab.

The generated XML will be of the format <idrequest>.xml and will be automatically placed at the location as per the usertype and idchannel entered above.

A developer can either generate "language specific" or "template" version of the XML by setting the appropriate language ID in the idlang text box, for template they need to enter *"tmp"* in the text box. Once the template version for the XML is generated it can be build using the Xslbuilder tool used for building the XSL's.

A separate XML can be generated for separate user agents. Currently user agent 'Tabs' is supported. On entering user agent in this tool, generated XML will be of the format <idrequest>_<user agent>.xml and will be automatically placed at the location as per the usertype and idchannel entered above.