Oracle FLEXCUBE Direct Banking

mLEAP Framework(Behavior concept)
Developer Guide for Android and iOS
Release 12.0.3.0.0

Part No. E52543-01

April 2014

ORACLE®

mLEAP Framework(Behavior concept) Developer Guide for Android and iOS

April 2014

Oracle Financial Services Software Limited

Oracle Park

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

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

Introduction

The concept of behavior allows a particular event to take place on click of an item, for instance, show/hide certain form elements, on the screen.

Form of behavior string::

Function1{value1@key1;value2@key2}#Function2{value3@key3}#Function3...and so on.

Functions are '#' separated.

Glossary::

Function - The method triggered on click

Key - The id of the datatype on which action will take place

Value - The value to be applied on a particular datatype

Static Behavior

Static behavior is applied on static datatypes (static dropdown, static segmented button etc).

Leap Entry::

Function column

Comma separated list of **static values** i.e, the items on click of which the event will take place.
Sample::

Block,UnBlock (items of a dropdown, in case of stop/block cheque transaction).

FunctionArgs column (For static parameters)

<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(For dynamic parameters)

<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

Dynamic Behavior

Dynamic behavior is applied on dynamic datatypes (dynamic dropdown, list etc).

Leap Entry::

Function column

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

- Type where the behavior string is common for all items, 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 (For static parameters)

<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(For dynamic parameters)

<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

Values In Function, FunctionArgs, DynamicFunctionArgs

In Order to define values in the function, functionsargs and dynamicfunctionargs of the leap, one needs to consider the following:

1) Behavior is defined by giving value in function column. Behavior is given to a component if:
    a) On click/change of the element certain action is being performed.
    b) On the basis of the value in the component certain action is being performed, etc.
    c) These actions include setting values, clearing values, firing requests, show/hide other components or tables, etc
2) **Function** : For defining function, we need to check for the following:
    a) If the action on the component leads to only a specific behavior **OR** The component is as such that only a single type of action is possible Then we define the function to be **'*'**. E.g.: In case of a button, where action is just click. In case of a dropdown where on any condition or value selected we need to do the same thing, as in set the current value to some hidden component.
    b) If the component is such that it has different values defined for which different actions are to be performed. And these values are known and fixed. Then we define the function of the component as those defined & fixed values, comma separated. E.g.: In case of static segmented button (with two segment Mail & Bulletin) where one can select either mail or bulletin (Defined and fixed) such that selecting Mail defines one set of behavior and selecting Bulletin defines other set of behavior. In this case we define Both Mail and Bulletin with two variables. Let's assume them here to be 'M & B' Thus, Function is given as **'M,B'.**

c) If the component is such that it has different values defined for which different actions are to be performed. But these values are defined and vary on different responses. Then, we define the function of the component as the value in the response that decides the values on which action is to be taken. E.g.: In case of dynamic segmented button where the number of segments being created depends on the response, one can define value of each segment which can be used to uniquely identify them. Suppose the value in response that uniquely identifies segment is 'code'. Thus, function is given as **'code'**.

3) **Functionargs** : Static Values/Components that need certain changes as being shown or hidden, getting set with some values or getting reset etc are Given in functionargs.

   a) In case of a '*' function, the functionsargs is given as the components or tables comma separated. These components are read as static component with their index as the number on which they are placed. Eg: For a button, functionargs can be given as: 'RRXXX62fldcomponent1,RRXXX623' where component and table number being affected are given with the requestid as a prefix. These are read further in mstmobdata entries as S1→RRXXX62fldcomponent1 and S2→RRXXX623.

   b) In case of function that is not a '*', the functionargs is given as PossibleValue1: its functionargs^PossibleValue2:Its functionargs… and so on.

4) **Dynamicfunctionargs** : In certain cases, where we need to use some value that will change according to response, we put these values in the dynamicfunctionargs. For E.g.: On selection of a dropdown value we need to set its account balance in some hidden component. Balance will change with changing value in dropdown. So, we use the x-path of the Balance in dynamicfunctionargs and set the corresponding values dynamically to the component required. It is defined similar to functionargs both in case of '*' and non-'*' function.

Entry in MstMobData

An entry for the behavior of each item needs to be done in the table **Mstmobdata**. For each value on which behavior is applied, a separate entry is required in mstmobdata.

DataName - <REQUEST ID><Field ID>

DataValue - <Static/dynamic Value for which behavior has to be applied> or '*', as applicable.

UserAgent - <UserAgent> which can be ** for all useragents or (iPad, AndTabs, iPhone, AndPhone)

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.

The concept is, that these parameter_place_holders($S1,$D1 etc), will be replaced dynamically, by parameter values provided in functionargs and dynamicfunctionargs column .

Sample::

90	RRADT61fldacctno	*	**	set{$D1@fldacctnum}
604	RRADT62flddetlstype	D	**	change{0@$S1}#observe{RRADT624@$S2}
603	RRADT62flddetlstype	A	**	change{1@$S1}#observe{RRADT624@$S2}
817	RRADT62fldpdfbutton	POPDISMISS	**	clearPopUp{}
815	RRADT62fldsubmit	POPDISMISS	**	clearPopUp{dismiss}
172	RRADT62fldtransdropdown	2D	**	vis{h@$S1}#exp{D@$S2}

*, 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 mstmobdata.

In case of static behavior use $S1,$S2 etc as parameter_place_holders and $D1,$D2 etc for dynamic behavior.

List of Behavior functions

BEHAVIOR	FORMAT	DESCRIPTION
vis	vis{s@fldId;h@fldId;e@fldId;d@fldId}	Used to show,hide,enable,disable a view
visOnly	visOnly{s@fldId;h@fldId;e@fldId;d@fldId}	used to show,hide,enable,disable a view with no change in nextscreenparams
invokeSocial	invokeSocial{fldId@FB}	invoking facebook
genMRL	genMRL{gen@fldId}	for generating QR code
scanMRL	scanMRL{scan@fldId}	for scanning QR code
saveMRL	saveMRL{save@fldId}	for saving QR code
readNFC	readNFC{rec@fldId}	for reading an NDEF

		tag
scanNFC	scanNFC{scan@fldId}	for sending an NDEF tag
clearPopUp	clearPopUp{}	clearing pop ups
reset	reset{tableId} or reset{fldId}	To reset a view
set setT	set{value@fldname} setT{Years,Months,Days@fldId}	setting a value in the view setting tenure in the view
mandatory	mandatory{fldId of mandatory fields@fldId of enable field}	if a mandatory field is filled then only particular field will be enabled
fireRequest	fireRequest{requestId,targetId,actionId}	firing a request
itemCount	itemCount{ListId@fldId}	for counting items in a field and setting the count in another field

matchPassword	matchPassword{fldId of New Password,fldId of Confirm Password@login;requestId}	for matching password of confirm and new password and if matched then fire the request
dealCheck	dealCheck{dealEnabledFlag,fldName of source Ccy,fldName of Destination Ccy@requestId,targetId}	if deal check is true then match of source and destination currency and fire the request
clearTargetTables	clearTargetTables{TableId}	for clearing tables
setAmount	setAmount{value@fldId}	for setting amount in field
clearParams	clearParams{tableId}	clearing nextscreenparams and other data structures
selectSegmented	selectSegmented{fldId@position}	selecting a specific radio or segmented button

disableSegmented	disableSegmented{fldId@position}	De-selecting a specific radio or segmented button
checkVisibiltyCallVis	checkVisibiltyCallVis{fldId^s@tableId;h@tableId}	if a particular field is filled then show or hide other fields or tables
fillCheck	fillcheck{message@fldid;message@fldid;fire@string}	if a field is blank then show a dialog with message and if field is filled then fire the request id

Example Explained

Static Behavior

Leap Entry::

Function- Block,UnBlock

FunctionArgs - Block:val1,val2^UnBlock:val3
DynamicFunctionArgs - Block:Xpath1,Xpath2 ^UnBlock:Xpath3

Mstmobdata Entry::

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

Replaced behavior string::

Block: set{val1@fld1;Xpath1@fld2;val2@fld3}#upd{Xpath2@fld4}
UnBlock: set{val3@fld1;Xpath3@fld2}

Dynamic Behavior

Leap Entry::

Type1: Same behavior string for all items

Function- *

FunctionArgs - val1,val2
DynamicFunctionArgs - Xpath1,Xpath2

Mstmobdata Entry::

DATANAME	DATAVALUE	USERAGENT	VALUESTRING
Mobile_BH_RRSUC61fldstop unstopchq	*	iPad	set{$S1@fld1;$D1@fld2;$S2@fld3}#upd {$D2@fld4}

Replaced behavior string::

*: set{val1@fld1;Xpath1@fld2;val2@fld3}#upd{Xpath2@fld4}

Type2: Different behavior string for different items

Function- ITB,DTB,IFB

FunctionArgs – ITB:val1,DTB:val2,IFB:val3
DynamicFunctionArgs – ITB:Xpath1,Xpath2,DTB:Xpath3,IFB:Xpath4

MSTMOBDATA Entry::

DATANAME	DATAVALUE	USERAGENT	VALUESTRING
Mobile_BH_RRSUC61fldstopunstopchq1	ITB	**	set{$S1@fld1;$D1@fld2}#upd{$D2@fld4}
Mobile_BH_RRSUC61fldstopunstopchq2	DTB	**	set{$S1@fld1;$D1@fld2}
Mobile_BH_RRSUC61fldstopunstopchq3	IFB	**	set{$S1@fld1;$D1@fld2}

Replaced behavior string::

ITB : set{val1@fld1;Xpath1@fld2}#upd{Xpath2@fld4}
DTB : set{val2@fld1;Xpath3@fld2}
IFB : set{val3@fld1;Xpath4}

Behavior for labels in case of dropdown

In case behavior has to be applied to the default label of a dropdown, the entry for behavior is slightly different.

NO entry has to be done on leap side.

Entry in mstmobdata::

The datavalue is of the form **label_beh_<Useragent Type>** ,rest of the entries remaining same.

Where, Useragent Type can be (iPad,And), where iPad is applied to all ios devices and And is for android devices. In case there is some value that is different for android phones and tablets, then, the specific useragent can be defined in Useragent column of the mstmobdata.