Oracle FLEXCUBE Enterprise Limits and Collateral Management ® 12.1
Development of Launch Forms and Others Screens

August 2013

ORACLE
FINANCIAL SERVICES

Contents

1. Preface

This document describes the features of a Launch Form and Others Screen in FLEXCUBE and the process of designing these screens using Oracle FLEXCUBE Development Workbench for Enterprise Limits and Collateral Management.

1.1 Audience

This document is intended for FLEXCUBE Application developers/users that use Development Workbench to develop various FLEXCUBE components.

To Use this manual, you need conceptual and working knowledge of the below:

Proficiency	*Resources*
FLEXCUBE Functional Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Technical Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Screen Development	*04-Development_WorkBench _Screen_Development-I.docx*
Working knowledge of Web based applications	Self Acquired
Working knowledge of Oracle Database	Oracle Documentations
Working knowledge of PLSQL & SQL Language	Self Acquired
Working knowledge of XML files	Self Acquired

1.2 Related Documents

04-Development_WorkBench _Screen_Development-I.docx
05-Development_WorkBench _Screen_Development-II.docx
14-Development_of_Online_Forms.docx
15-Development_of_Call_Form.docx

2. Introduction

2.1 How to use this Guide

The information in this document includes:

3. Launch Forms

Launch Forms are nothing but normal screens; which are called from another function id for view data purpose. **Launch Forms are used for querying (viewing) the data only; no other processing can be done on launch forms unlike call forms.**
Usually, any screen which is used across multiple screens for view purpose is treated as Launch Forms.
Example: Contract Events Screen
Contract Events screen is used across many contract screens for viewing the events that has got fired for the particular contract. So a single screen can be designed for the same and re used across all contract screens
Launch Forms can be launched independently and query operation can be done on the screen independently

3.1 Screen Development

Technically Launch Forms are the same as normal maintenance or transaction screens. There is no difference in development of a launch form from a maintenance or transaction screen.

Launch forms can be designed as of type

i) Maintenance
ii) Transaction

Note that a maintenance launch form can be used if it is invoked from maintenance screens and similarly for transaction launch forms.

Naming Convention:

Launch Form is nothing but a normal function Id. So it has to follow the same naming convention as any other detail screen. Third letter has to be ‘D’ and it should have 8 characters.
Example: CSDEVENT, CSDACCNT are valid names for a Launch Form screen

Menu details has to be provided for the Launch form screen as it is a an independent screen . Entries has to be present in *smtb_function_description* unlike Call Forms

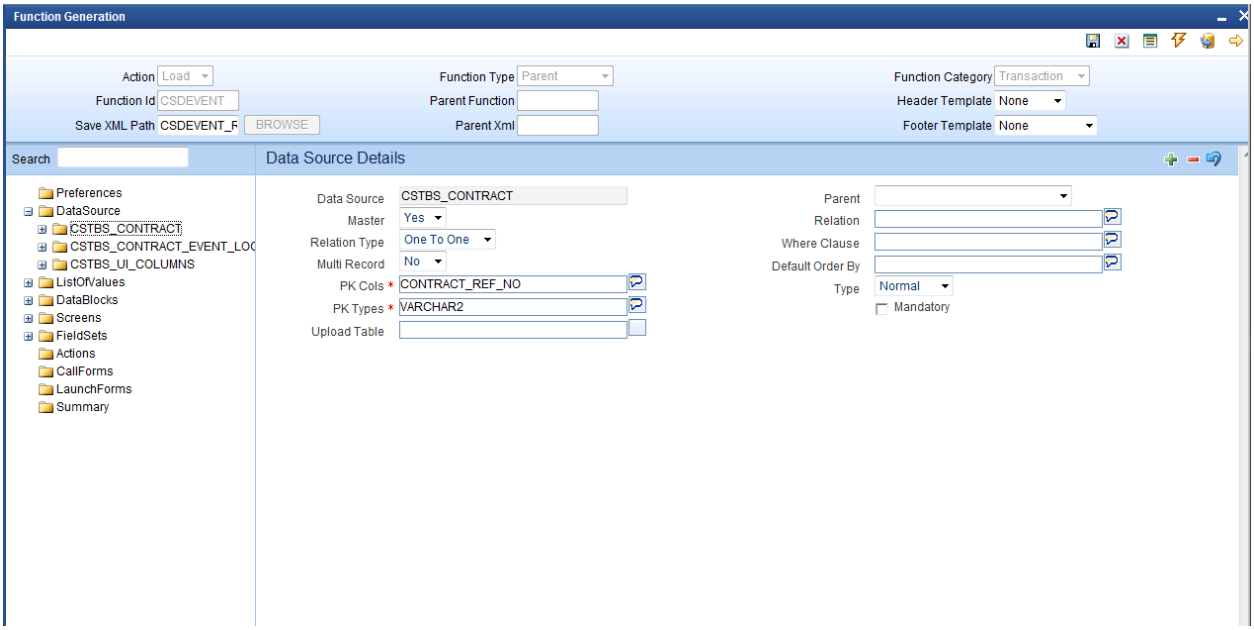

Fig 3.1 A Launch Form function Id showing master Data Source Properties

Refer documents on screen development, development of maintenance and transaction screens for designing a Launch Form screens.

Screen Arguments:

Screen Arguments has to be maintained for the main screen of the launch form.

Launch Forms are used only for querying data. Hence ACTION_CODE has to be passed as a screen argument with argument value as EXECUTEQUERY and the Primary Key values for querying in launch Form screen should be passed as the other parameters

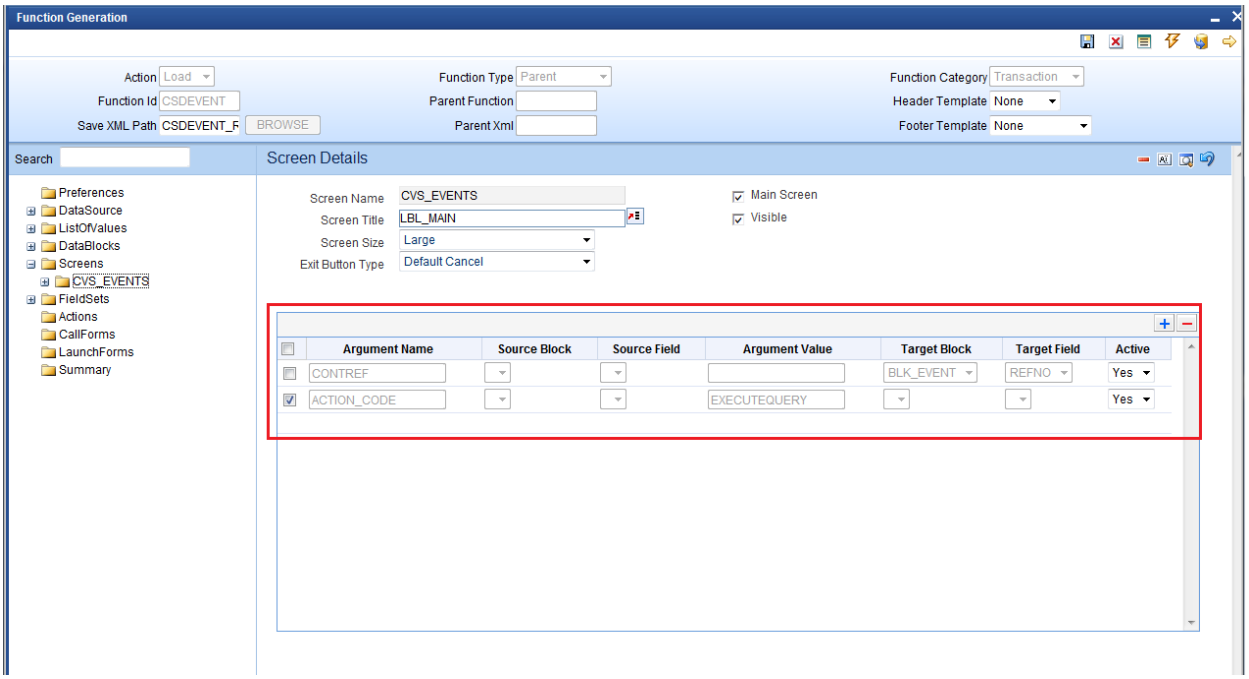

Fig 3.2 Defining Screen arguments for Launch Form Main Screen

Summary screens if required can be designed for the launch form screen

3.2 Generated units

All the units for a normal maintenance or transaction screen will be generated for a Launch Form screen as well.

Note the following while deploying units for Launch Forms

i) Entry has to made manually in CSTB_CALL_FORM_NODES for the launch form. Script won't be generated by the Tool while designing the Launch Form. Hence it has to inserted manually providing the screen arguments as maintained for Launch Form main screen

Screen arguments has to be inserted in SCREEN_ARGS column of CSTB_CALL_FORM_NODES separated by tilde (~)

3.3 Attaching Launch Form to the Main Function Id

Launch Forms has to be attached to the main function Id in Launch Form Node

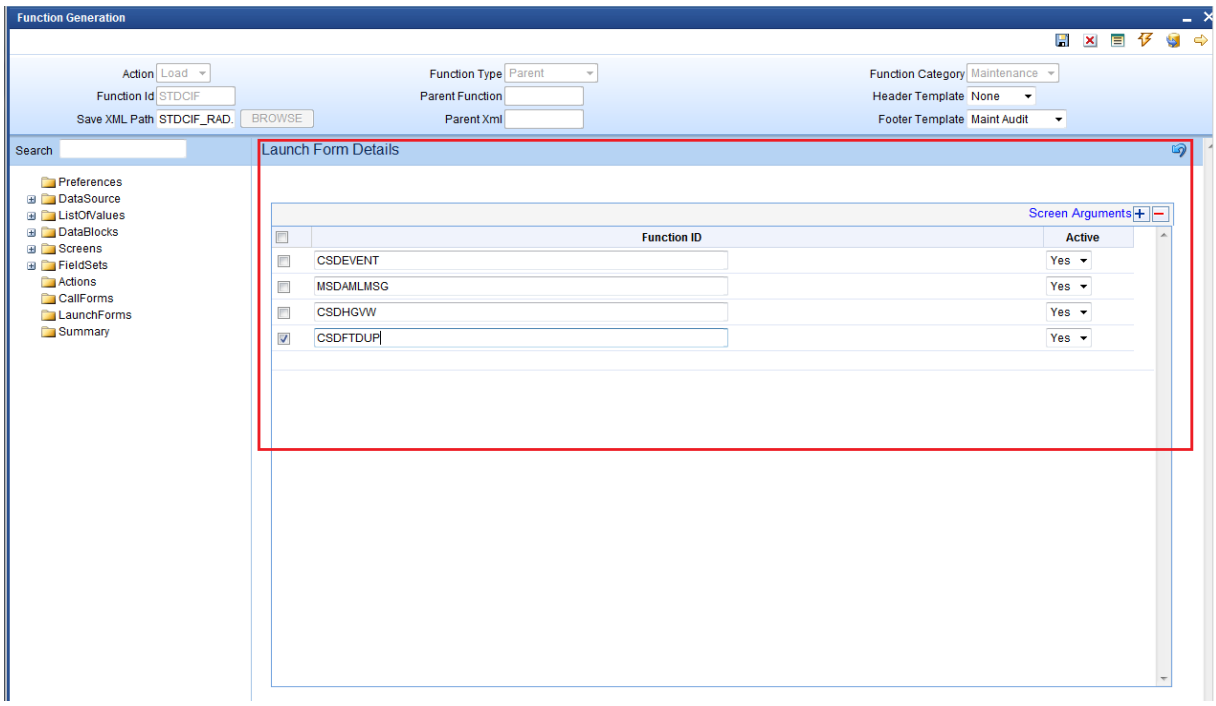

Fig 3.3 Attaching Launch forms to a Function Id

Screen Arguments has to be passed from the main function Id to the Launch Form screen

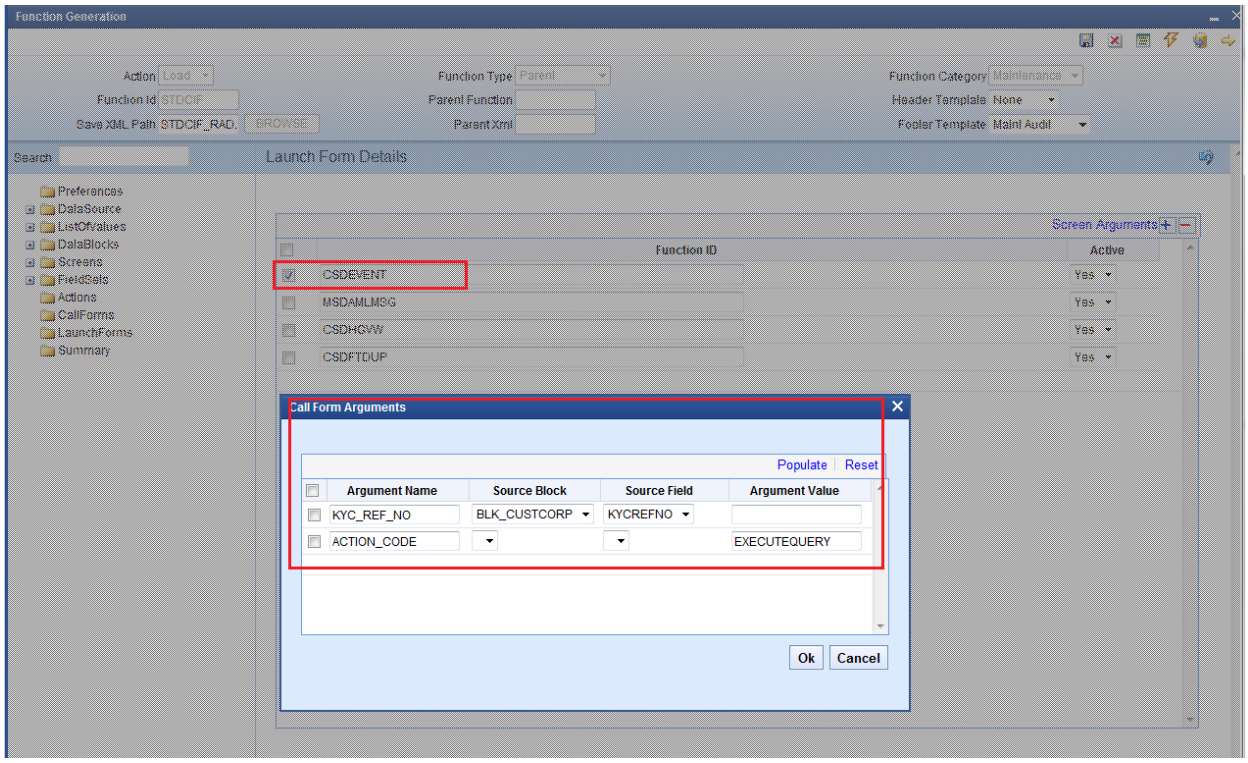

Fig 3.4 Passing Screen Arguments to Launch Form

Launch forms can be launched by clicking on button placed in the main screen.

Button events have to be maintained such that Launch Form will be launched on clicking it.

Refer Launch Form section on *04-Development_WorkBench _Screen_Development-I.docx* for detailed explanation

3.4 Extensible development

Developer can add his code in hook packages and release specific JavaScript file.
This is similar to any other Maintenance or Transaction screen.

Any enhancements or change in query logic for the screen can be code in **fn_post_query** of the Hook package of the particular release

4. Others Screens

If the developer does not want to use the business logic provided by the ODT generated code, he can create the function Id with function type as OTHERS.

In the generated package, code will handle only parsing the Ts list to composite PL/SQL type and vice versa. No other processing logic would be provided by the code. Developer has to write the whole business logic in hook packages.

This can be useful when developer wishes to reduce unnecessary code in main package.

For example: Batch function Ids

Function Id's for processing a batch can designed as an OTHERS screen. No conventional actions (NEW, SAVE, MODIFY etc) is required for a batch processing screen; only batch processing will need to be done. This can be handled better by designing the screen as OTHERS and write the batch processing logic in the Hook packages

4.1 Screen Development

Screen development for an Others screen is similar to a normal maintenance function id

Naming Convention:

Others screen should adhere to the same naming convention as a normal maintenance function id. Name of the function id should be of 8 characters and third letter should be 'D'

Function Type:

Function Type has to be selected as OTHERS

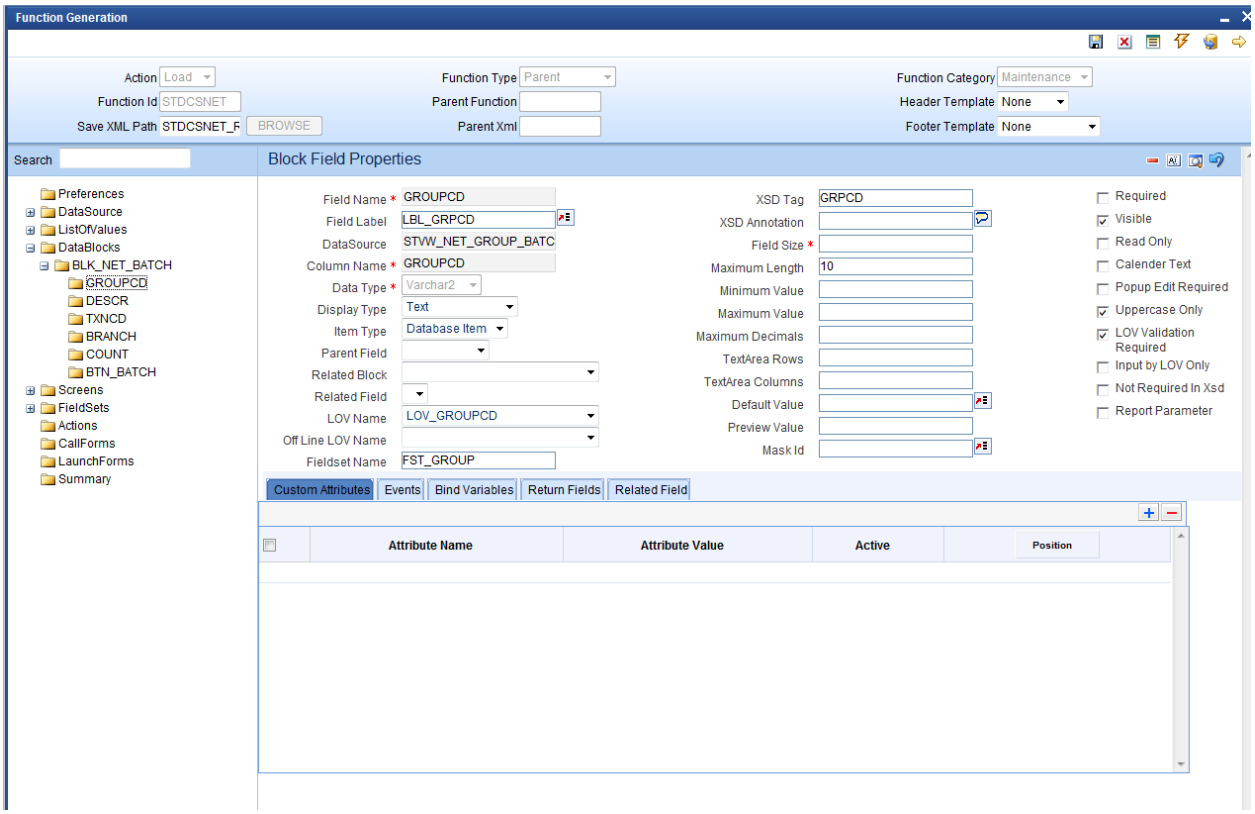

Fig 5.1 Maintaining Block Field Properties

Normally user defined actions would be used in an OTHERS screen .These actions are invoked on click of the buttons placed in the screen.

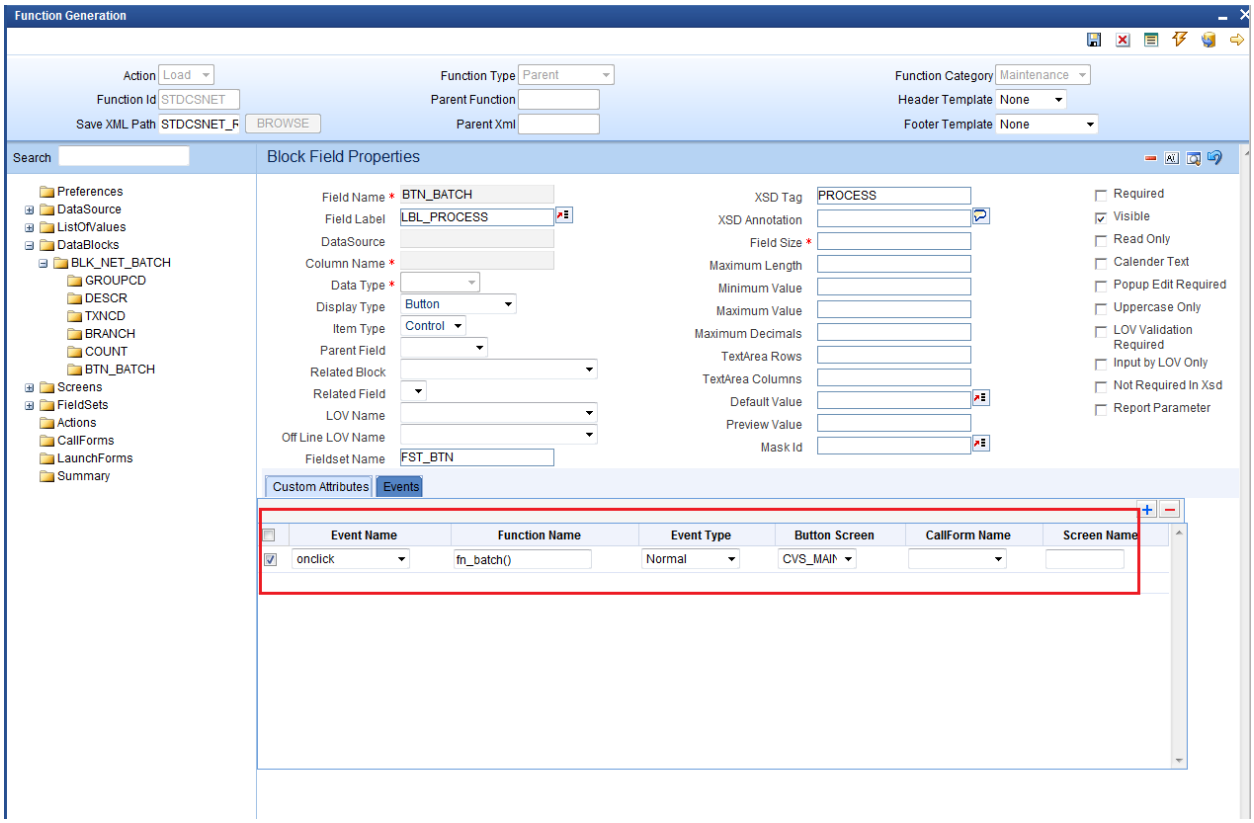

Fig 5.2 Button Field events : invoking a function on click of button

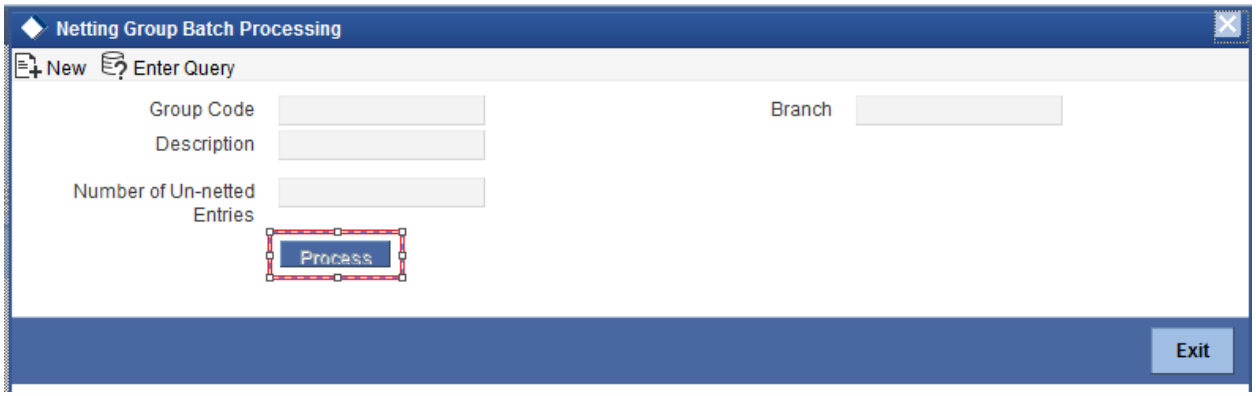

Fig 5.3 Preview of the batch processing screen

4.2 Generated units

All the units for a normal maintenance or transaction screen will be generated for a Launch Form screen as well.

4.3 Extensible development

Developer can add his code in hook packages and release specific JavaScript file.
This is similar to any other Maintenance or Transaction screen.

Coding in JavaScript:

Custom action codes defined by the developer are defined in the JavaScript file.
It is usually defined on click of a button .The action code is defined and the request xml is built and passed to the server.

These codes are written in release specific JavaScript file

Example: Figure below shows a snippet from STDCSNET_CLUSTER.js
Notice the *fn_batch ()* which is invoked on clicking of the *Process* button.

```
function fn_batch()
{
    appendData();
    gAction = 'NET_BATCH';
    fcjRequestDOM = buildUBSXml();
    fcjResponseDOM = fnPost(fcjRequestDOM, servletURL, functionId);
    var msgStatus   =getNodeText( selectSingleNode(fcjResponseDOM,"FCUBS_RES_ENV/FCUBS_HEADER/MSGSTAT"));
    if (!fnProcessResponse())   {
        return true;
    }
}
```

Fig 5.4 Code Snippet from STDCSNET_CLUSTER.js

Here action code is set as 'NET_BATCH' and request xml built and passed to the server and response processed based on response xml

Coding in Packages:

Developer can write the business logic in *fn_main* which will be present in the release specific hook package

Skip Handlers can be used to skip the code in any previous release stages if required.

Processing logic has to be written for the user defined action codes in the release specific hook package

```
--Log#1 change starts
IF p_action_code = 'NET_BATCH' THEN
  IF p_Wrk_stdcsnet.v_stvw_net_group_batch.groupcd = 'ALL' THEN
    IF not stpks_group_netting.fn_group_netting(p_Wrk_stdcsnet.v_stvw_net_group_batch.branch,
                                                p_Err_Code,
                                                p_Err_Params) THEN
      RETURN FALSE;
      dbg('failed in stpks_group_netting');
      dbg('p_err_code' || p_err_code);
    END IF;
  ELSE
    IF not stpks_group_netting.fn_group_netting(p_Wrk_stdcsnet.v_stvw_net_group_batch.branch,
                                                p_Wrk_stdcsnet.v_stvw_net_group_batch.groupcd,
                                                p_Err_Code,
                                                p_Err_Params) THEN
      Pr_Log_Error(p_Function_Id ,'FLEXCUBE',p_Err_Code ,p_Err_Params);--11.2 sfr 92
      RETURN FALSE;
      dbg('failed in stpks_group_netting');
      dbg('p_err_code' || p_err_code);
    END IF;
  END IF;
END IF;
--Log#1 change starts
```

Fig 5.5 Code Snippet from stpks_stdcsnet_cluster.js

The above figure shows the handling for the user defined action code 'NET_BATCH' in *fn_main* of stpks_stdcsnet_cluster.sql

Development of Launch Forms and Others Screens
August 2013

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com/ financial_services/