

Development of Launch Forms and Others Screens  
Oracle Banking Trade Finance  
Release 14.5.2.0.0  
[August][2021]



---

# Contents

|  |    |
|--|----|
| 1. Preface.....  | 2  |
| 1.1 Audience .....                                     | 3  |
| 1.2 Related Documents .....                            | 3  |
| 2. Introduction.....                                   | 3  |
| 2.1 How to use this Guide .....                        | 3  |
| 3. Launch Forms .....                                  | 4  |
| 3.1 Screen Development .....                           | 4  |
| 3.2 Generated units .....                              | 6  |
| 3.3 Attaching Launch Form to the Main Function Id..... | 6  |
| 3.4 Extensible development .....                       | 8  |
| 4. Others Screens.....                                 | 8  |
| 4.1 Screen Development .....                           | 8  |
| 4.2 Generated units .....                              | 11 |
| 4.3 Extensible development .....                       | 11 |

---

## 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 Universal Banking.

## 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:

| <i>Proficiency</i>                          | <i>Resources</i>   |
|---|--|
| FLEXCUBE Functional Architecture            | Training programs from Oracle Financial Software Services. |
| FLEXCUBE Technical Architecture             | Training programs from Oracle Financial Software Services. |
| FLEXCUBE Screen Development                 | <i>04-Development_WorkBench_Screen_Development-I.docx</i>  |
| 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:

- [Chapter 2 , "Introduction"](#)
- [Chapter 3 , "Launch Forms"](#)

- [Chapter 4 , "Others Screen"](#)

### 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

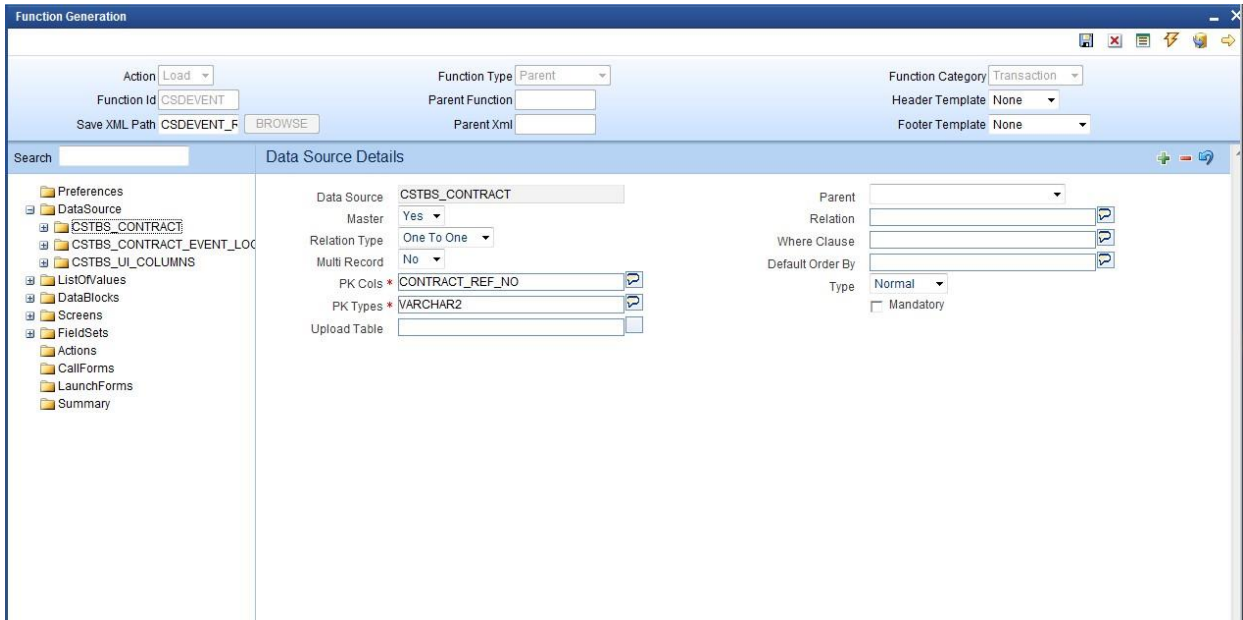


Fig 3.1 A Launch Form function Id showing master Data Source Properties

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

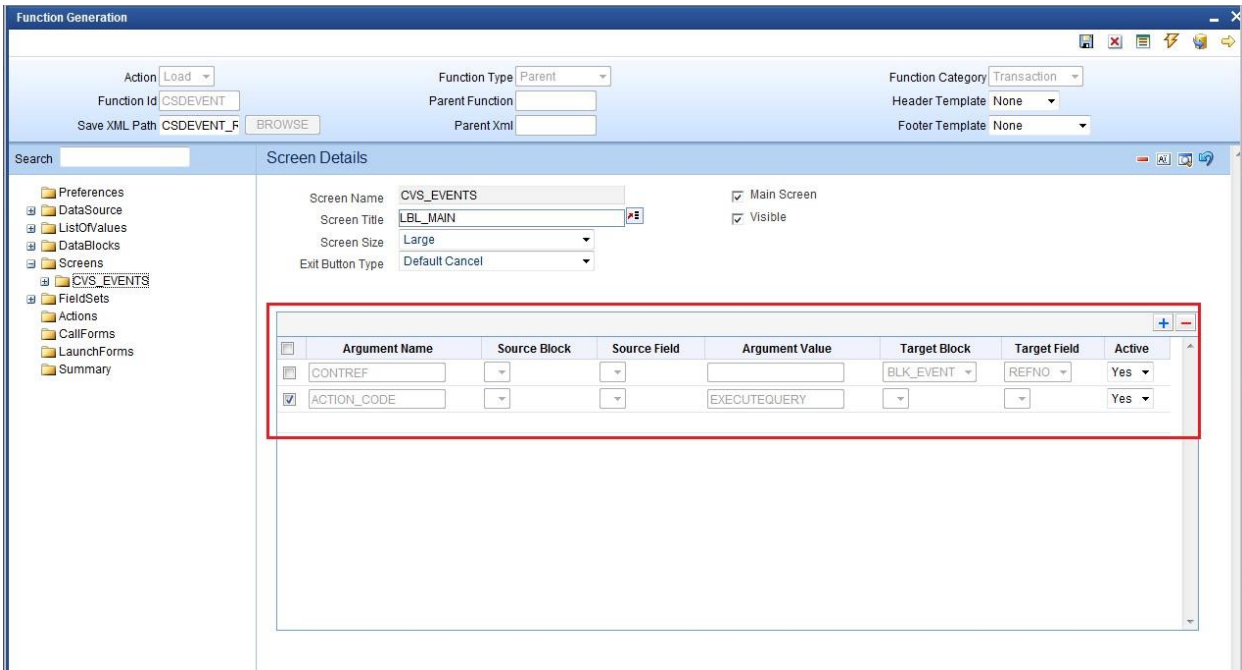


Fig 3.2 Defining Screen arguments for Launch Form Main Screen

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 be 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 be 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

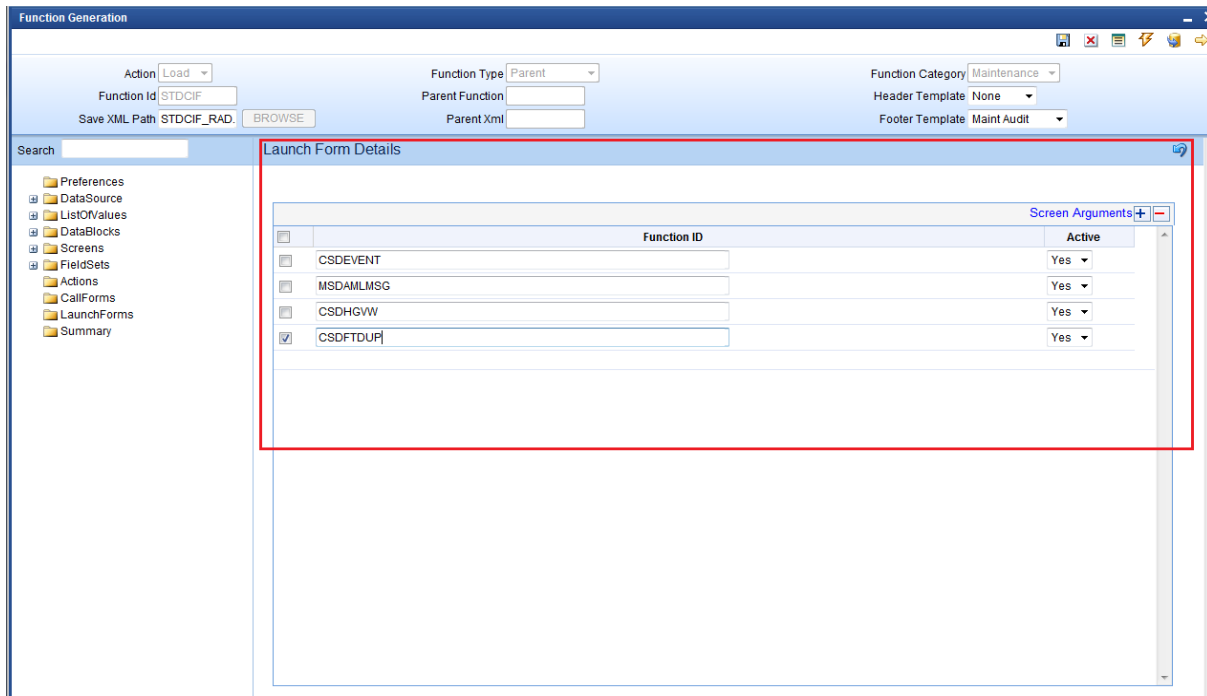


Fig 3.3 Attaching Launch forms to a Function Id

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

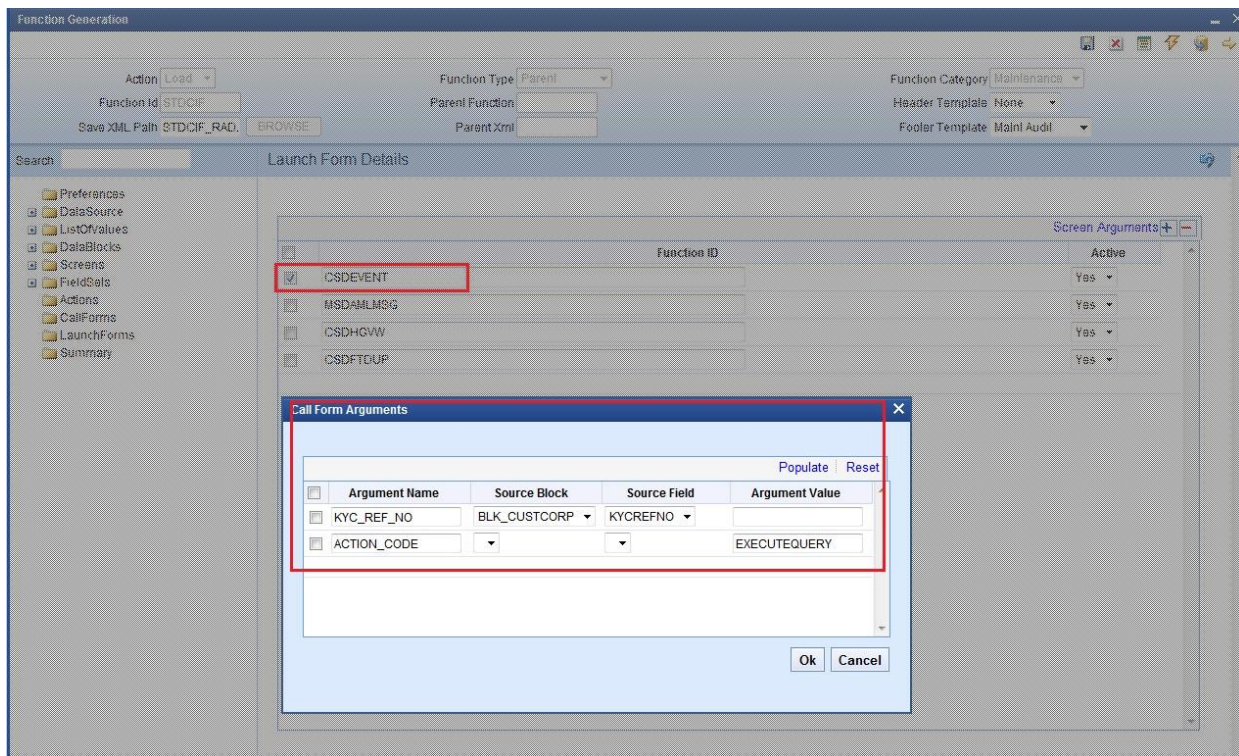


Fig 3.4 Passing Screen Arguments to Launch Form

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



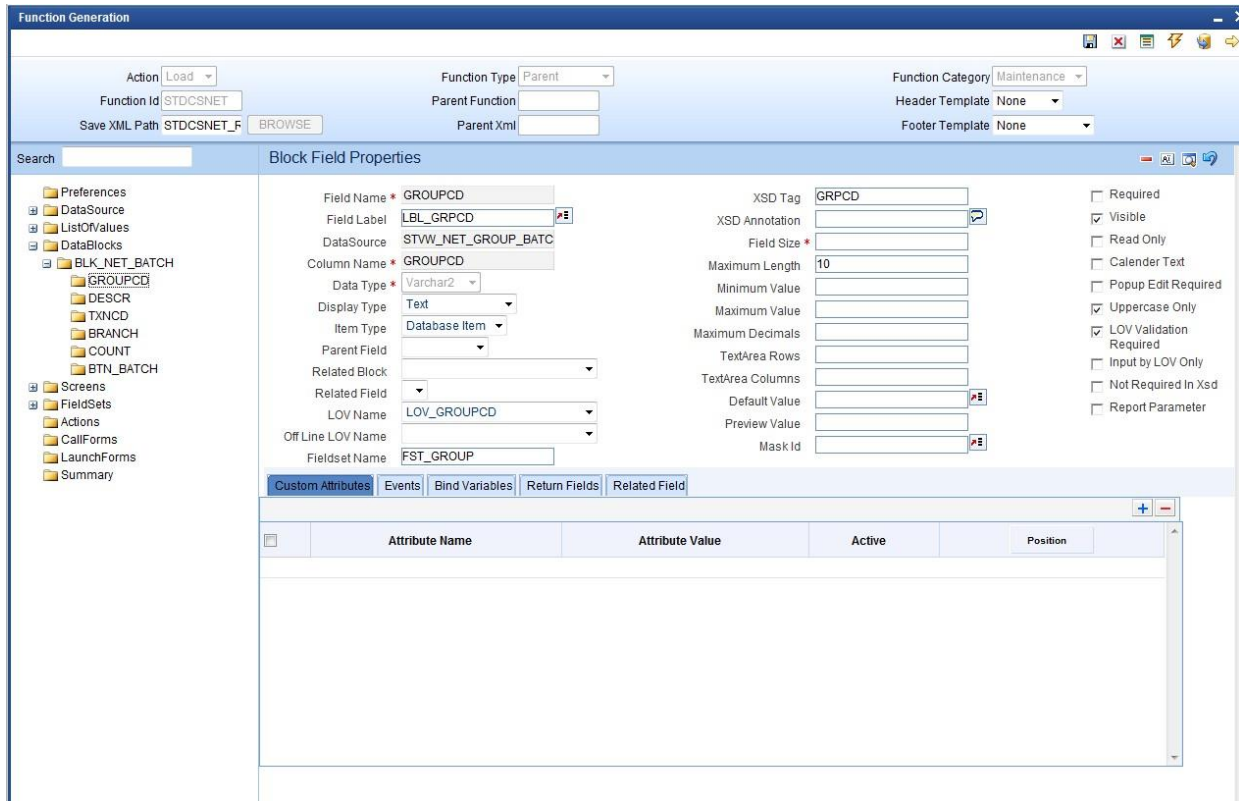


Fig 5.1 Maintaining Block Field Properties

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

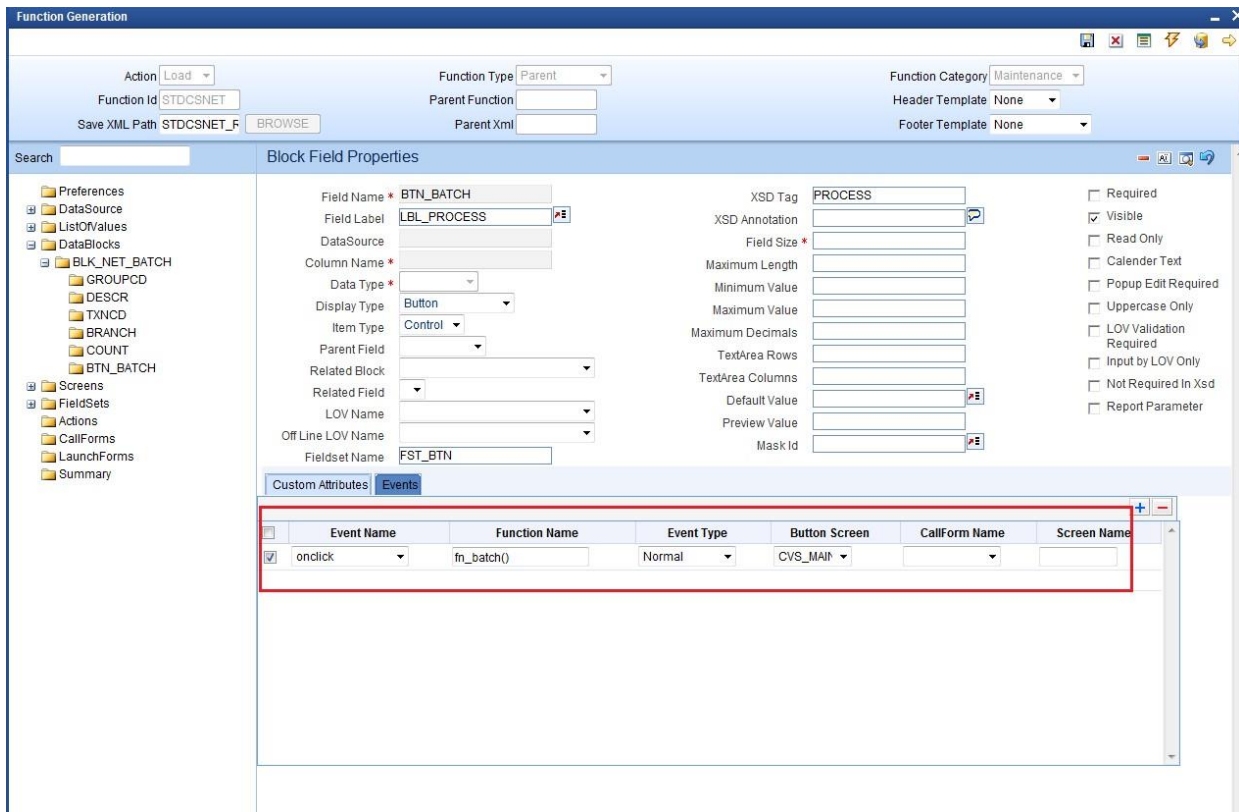


Fig 5.2 Button Field events : invoking a function on click of button

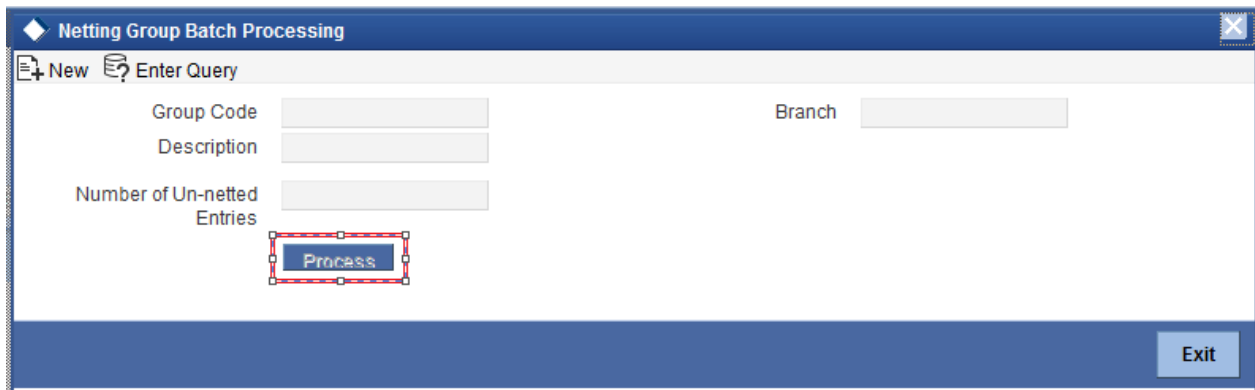


Fig 5.3 Preview of the batch processing screen

## 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 status
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 status

```

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] [2021]  
Version 14.5.2.0.0

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/](http://www.oracle.com/financialservices/)

Copyright © 2021, Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

**U.S. GOVERNMENT END USERS:** Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

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.