Oracle® Banking Treasury Management

Development of Launch Forms and Other Screens

Release 14.8.0.0.0
G28514-01
April 2025

ORACLE

Oracle Banking Treasury Management Development of Launch Forms and Other Screens, Release 14.8.0.0.0

G28514-01

Contents

Preface

This document describes the steps to develop the notification XML and notification trigger using Oracle FLEXCUBE Development Workbench.

Purpose

This manual is designed to help FLEXCUBE Application developers/users to familiarize with ORACLE FLEXCUBE Development Workbench.

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:

Table 1 Proficiency and Resources

Proficiency	Resources
FLEXCUBE Functional Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Technical Architecture	Training programs from Oracle Financial Software Services.
FLEXCUBE Object Naming Conventions	Development Overview Guide
Working knowledge of Web based Applications	Self-Acquired
Working knowledge of Oracle Database	Oracle Documentations
Working knowledge of PLSQL developer	Respective vendor documents

Table 1 (Cont.) Proficiency and Resources

Proficiency	Resources
Working knowledge of PLSQL and SQL Language	Self-Acquired
Working knowledge of XML files	Self-Acquired

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 Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes.

Prerequisite

Specify **User ID** and **Password**, and log in to **Home Screen**.

Related Resources

The functions of ORACLE FLEXCUBE Development Workbench for Investor Servicing system is organized into various guides, each discussing a component.

For more information, see these Open Development Tool documents:

- *Open Development Tool Installation*
- *Development Workbench - Getting Started*
- *Development Workbench - Administration*
- *Development Workbench - Screen Development I*
- *Development Workbench - Screen Development II*
- *Development Workbench - Screen Customizer*
- *Development Workbench - Notifications*
- *Development Workbench - Bulk Generation*
- *Development Workbench - Source Upgrade*
- *Development Workbench - Tracking Changes*
- *Development of Maintenance Form*
- *Development of Online Form*
- *Development of Call Form*
- *Child and Screen Childs - Concept and Design*
- *Development of Dashboard Form*
- *Development Workbench Service XML Development*
- *Development Workbench - Rest Services Development*

1

Launch Forms

This topic describes about the Launch Forms

Launch Forms are nothing but normal screens; which are called from another function id for view data purposes. 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 purposes 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.

This topic contains the following sub-topics:

1.1 Screen Development

This topic describes about the Lunch Forms Screen Development.

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

Launch forms can be designed as of type

1. Maintenance
2. Transaction

> **Note:**
>
> A maintenance launch form can be used if it is invoked from maintenance screens and similarly for transaction launch forms.

This topic contains the following sub-topics:

1.1.1 Name Convention

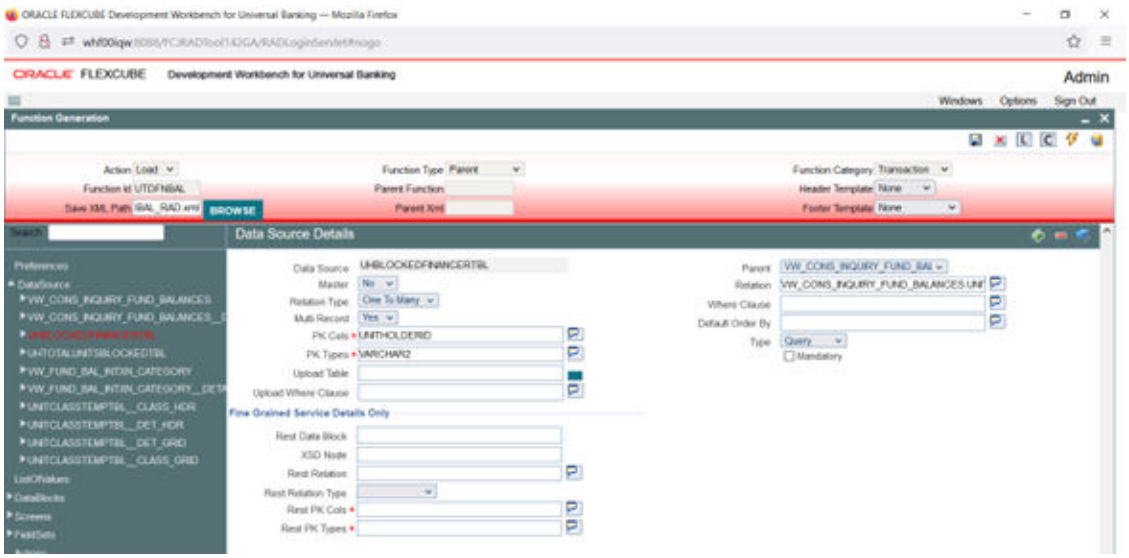
This topic describes about naming convention of launch form screen development.

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: UTDFNBAL is a valid name for a Launch Form screen.

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

Figure 1-1 Data Source Properties



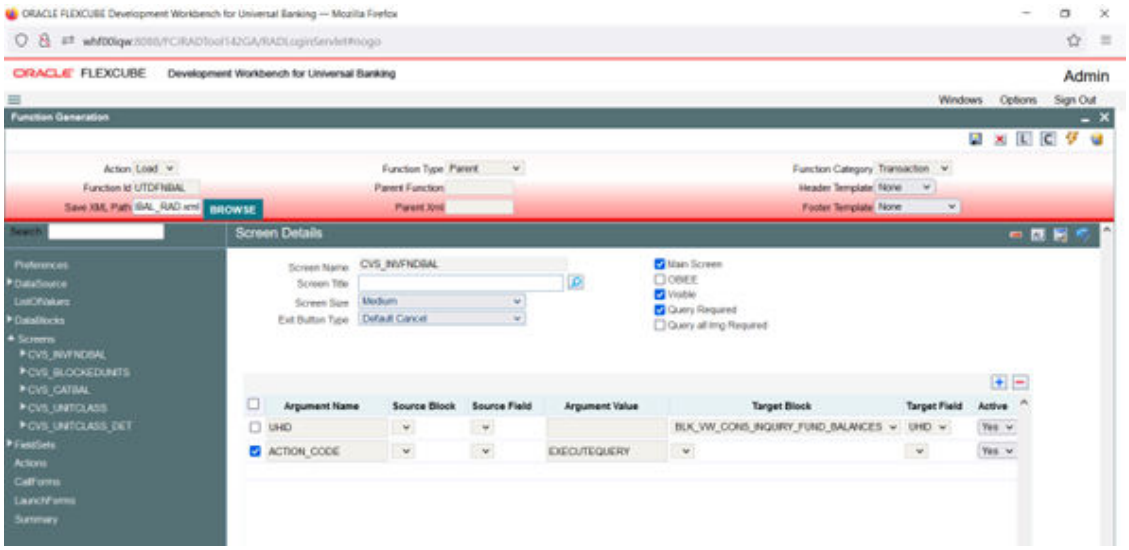
Refer documents on screen development, development of maintenance and transaction screens for designing a Launch Form screens.

1.1.2 Screen Arguments

This topic describes about screen arguments of launch form screen development.

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.

Figure 1-2 Screen Details



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

1.2 Generated Units

This topic describes about generated units of launch form screen development.

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

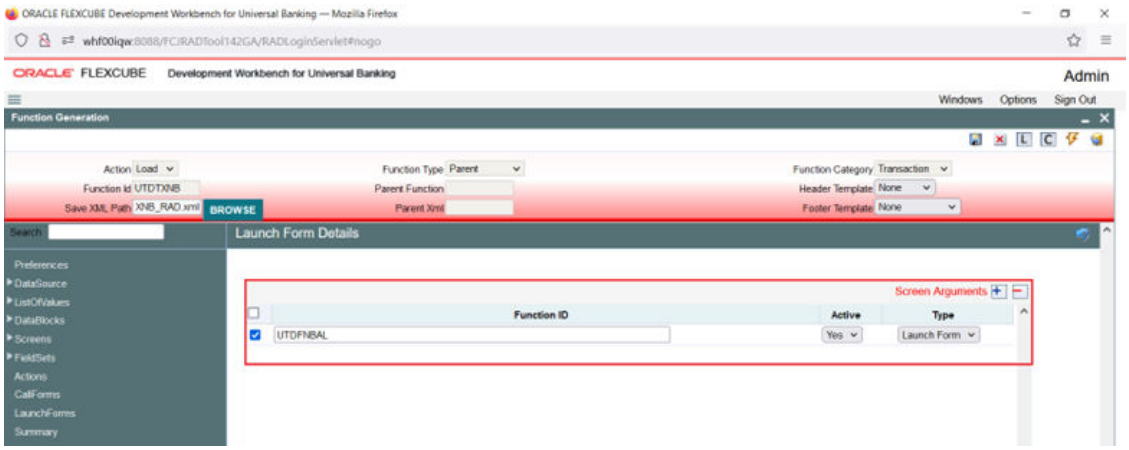
- Entry has to be made manually in CSTB_CALL_FORM_NODES for the launch form. The script won't be generated by the Tool while designing the Launch Form. Hence it has to be inserted manually providing the screen arguments as maintained for Launch Form main screen Screen arguments has to be inserted in the SCREEN_ARGS column of CSTB_CALL_FORM_NODES separated by a tilde (~)

1.3 Attach the Launch Form to the Main Function Id

This topic describes about the attaching launch form to the main function Id.

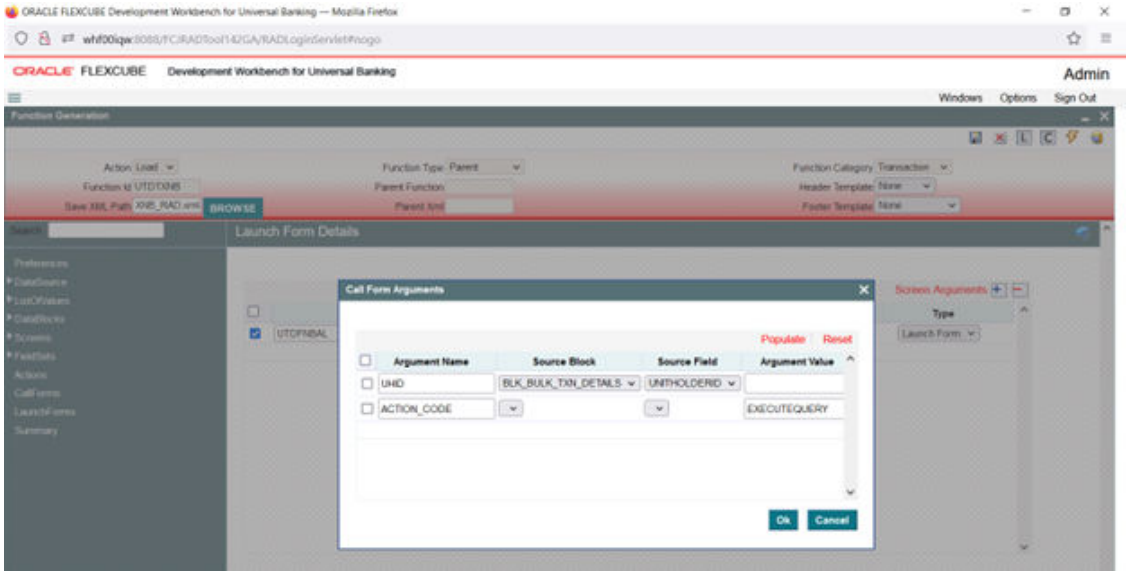
Launch Forms has to be attached to the main function Id in Launch Form Node.

Figure 1-3 Attaching Launch forms to a Function Id



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

Figure 1-4 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 Oracle FLEXCUBE Enterprise Limits and Collateral Management ODT Screen Development for detailed explanation.

1.4 Extensible development

This topic describes about the 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

2

Others Screens

This topic describes about the 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 the function type as **OTHERS**.

In the generated package, the 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. The developer has to write the whole business logic in hook packages. This can be useful when the developer wishes to reduce unnecessary code in the main package. For

Example: Batch function Ids

Function IDs for processing a batch can be designed as an OTHERS screen. No conventional actions (NEW, SAVE, MODIFY etc) are 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 writing the batch processing logic in the Hook packages.

This topic contains the following sub-topics:

2.1 Screen Development

This topic describes about the Other Screen Development.

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

2.2 Name Convention

This topic describe about naming convention.

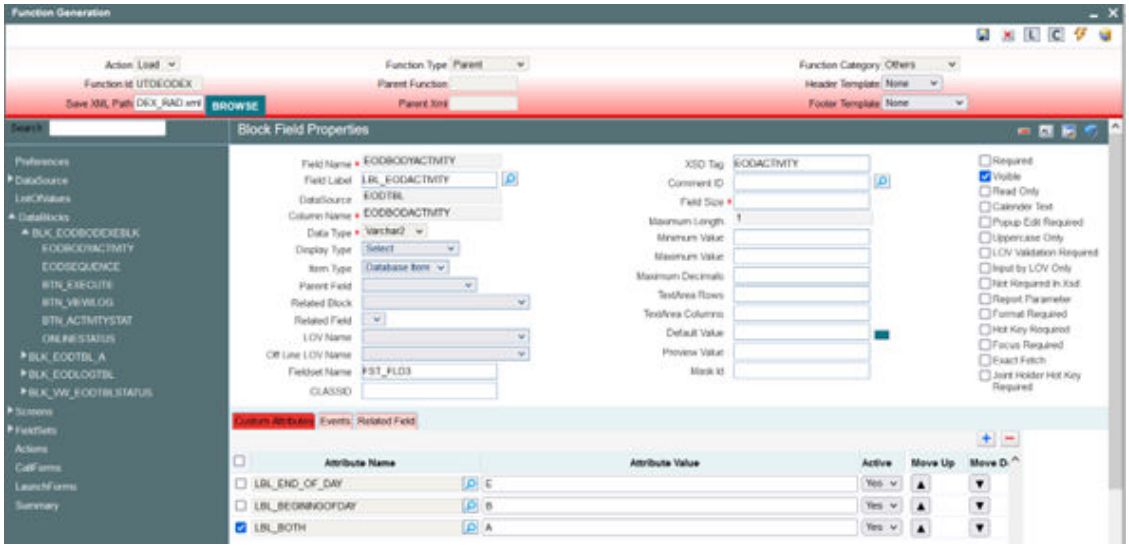
Others screen should adhere to the same naming convention as a normal maintenance function id. Name of the function id should be of eight characters and third letter should be D.

2.3 Function Type

This topic describes about function type.

Function Type has to be selected as **OTHERS**.

Figure 2-1 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.

Figure 2-2 Button Field events

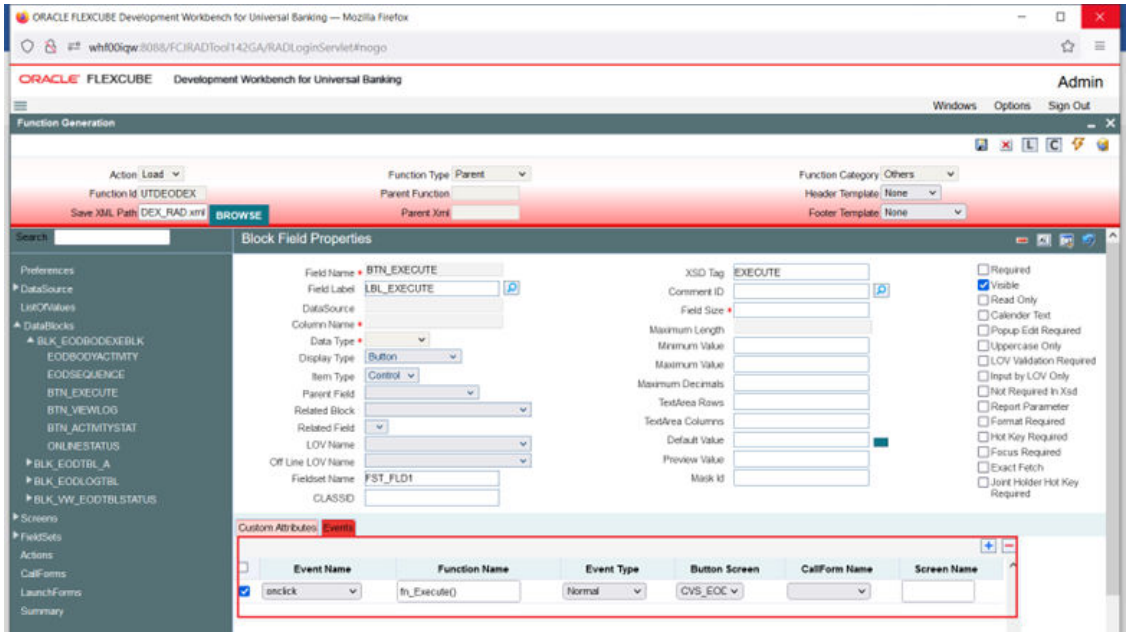
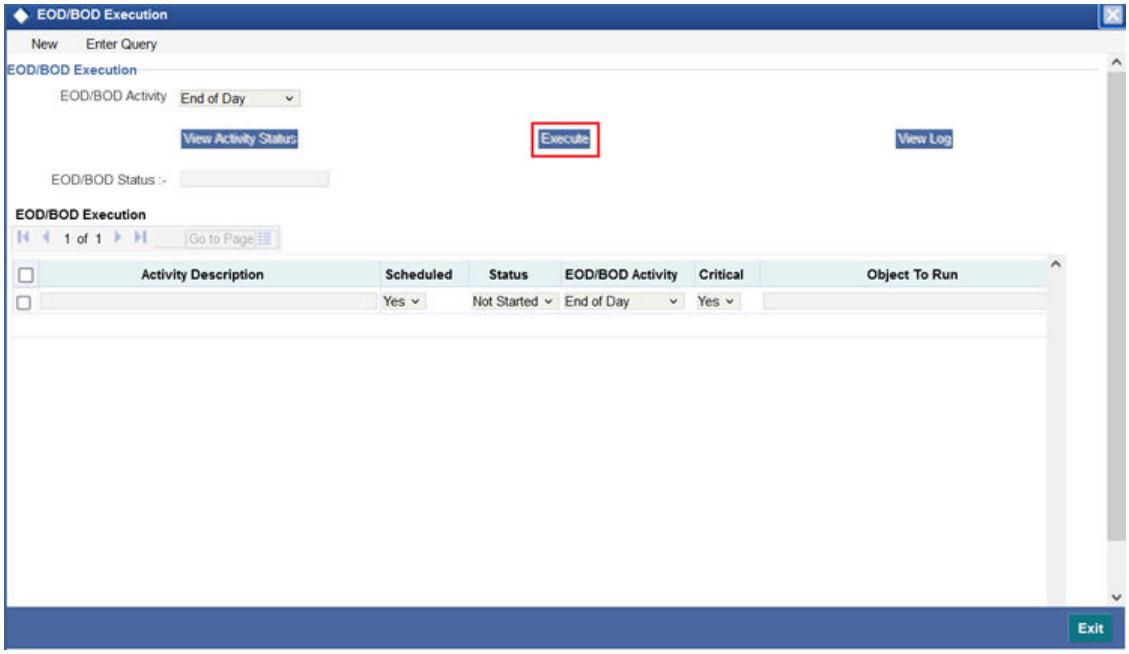


Figure 2-3 Preview of the batch processing screen



2.4 Generated units

This topic describes about the generated units.

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

2.5 Extensible development

This topic describes about the Extensible development.

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

This topic contains the following sub-topics:

- Code in JavaScript
  This topic describes about coding in JavaScript for extensible development.
- Code in Packages
  This topic describes about coding in packages for extensible development.

2.5.1 Code in JavaScript

This topic describes about coding in JavaScript for extensible development.

Custom action codes defined by the developer are defined in the JavaScript file. It is usually defined by the 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

> **Note:**
>
> The `fn_batch()` which is invoked on clicking of the Process button.

Figure 2-4 Coding in JavaScript

```
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;
    }
}
```

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

2.5.2 Code in Packages

This topic describes about coding in packages for extensible development.

The 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

Figure 2-5 Coding in Packages

```
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 ,'PLEXCUBE',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;
```

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

Glossary

Index