Development Workbench - Source Upgrade
Oracle FLEXCUBE Universal Banking
Release 14.8.3.0.0
Part No. G61377-01
October 2026



Contents

1 Preface 3
1.1 Audience 3
1.2 Related Documents 3
2 Introduction 4
2.1 How to use this Guide 4
3 Overview of Refresh functionality in Oracle FLEXCUBE Development Workbench 4
4 Child Refresh 4
4.1 Process Steps 5
4.2 Functionality Demonstration 10
5 Screen Child Refresh 10
5.1 Process Steps 11
5.2 Functionality Demonstration 12
6 Source Refresh 13
6.1 Process Steps 14
6.2 Functionality Demonstration 16
6.3 Source refresh is not possible in below scenarios 17

1 Preface

This document describes the Refresh functionality available in Oracle FLEXCUBE Development Workbench for Universal Banking and guides the developers on how to use this feature

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 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 & SQL Language	Self Acquired
Working knowledge of XML files	Self Acquired

1.2 Related Documents

Development Workbench - Screen Development II
Development of Online Forms
Development of Call Form
Development of Launch Forms and Others Screens
Child and Screen Childs - Concept and Design

2 Introduction

2.1 How to use this Guide

The information in this document includes:

- Chapter 2 , “Introduction”
- Chapter 3 , “Overview of Refresh Functionality in Oracle FLEXCUBE Development Workbench"
- Chapter 4 , “Child Refresh”
- Chapter 5 , "Screen Child Refresh”
- Chapter 6 , "Source Refresh”

3 Overview of Refresh functionality in Oracle FLEXCUBE Development Workbench

Refresh Functionality allows us to upgrade the existing radxml to its later version keeping the *sub version* specific changes intact. Three kinds of refresh can done using the Tool.

1) Child Refresh
2) Screen Child Refresh
3) Source Refresh

4 Child Refresh

Child Refresh allows the developer to upgrade a child radxml with its latest parent radxml .In doing so; the latest changes done in parent functionId would be reflected in the child functionId while retaining all the changes done in the child level

- This process is to be done within a release. i.e. child functionId has to be refreshed it’s the parent function_id from the same release

- It is recommended that this process is done before development cut of the release for all child radxmls within a release. For instance; if development has happened parallel for a child and parent functionId during a release, child refresh should be done before base lining so that child and parent record types are consistent

- All the system units need to be regenerated after Child Refresh. A thorough unit testing is recommended after deployment of all generated units

4.1 Process Steps

Child Refresh process is explained taking two hypothetical function IDs , STDCIF and STDCIFAD as example.

STDCIF – Parent Screen

STDCIFAD – Child Screen

Click on **Refresh** node from Development Workbench landing page.

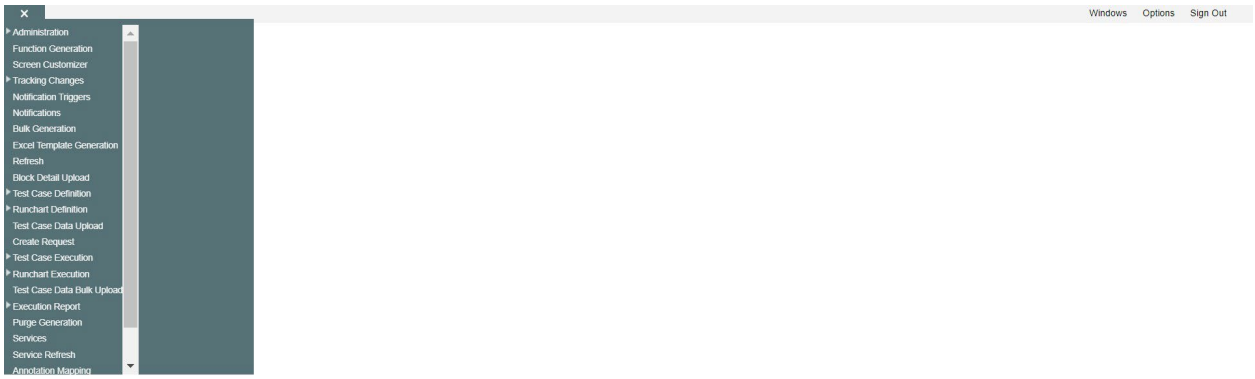


Fig 4.1.1: Development Workbench Landing Page

The following window will be launched.

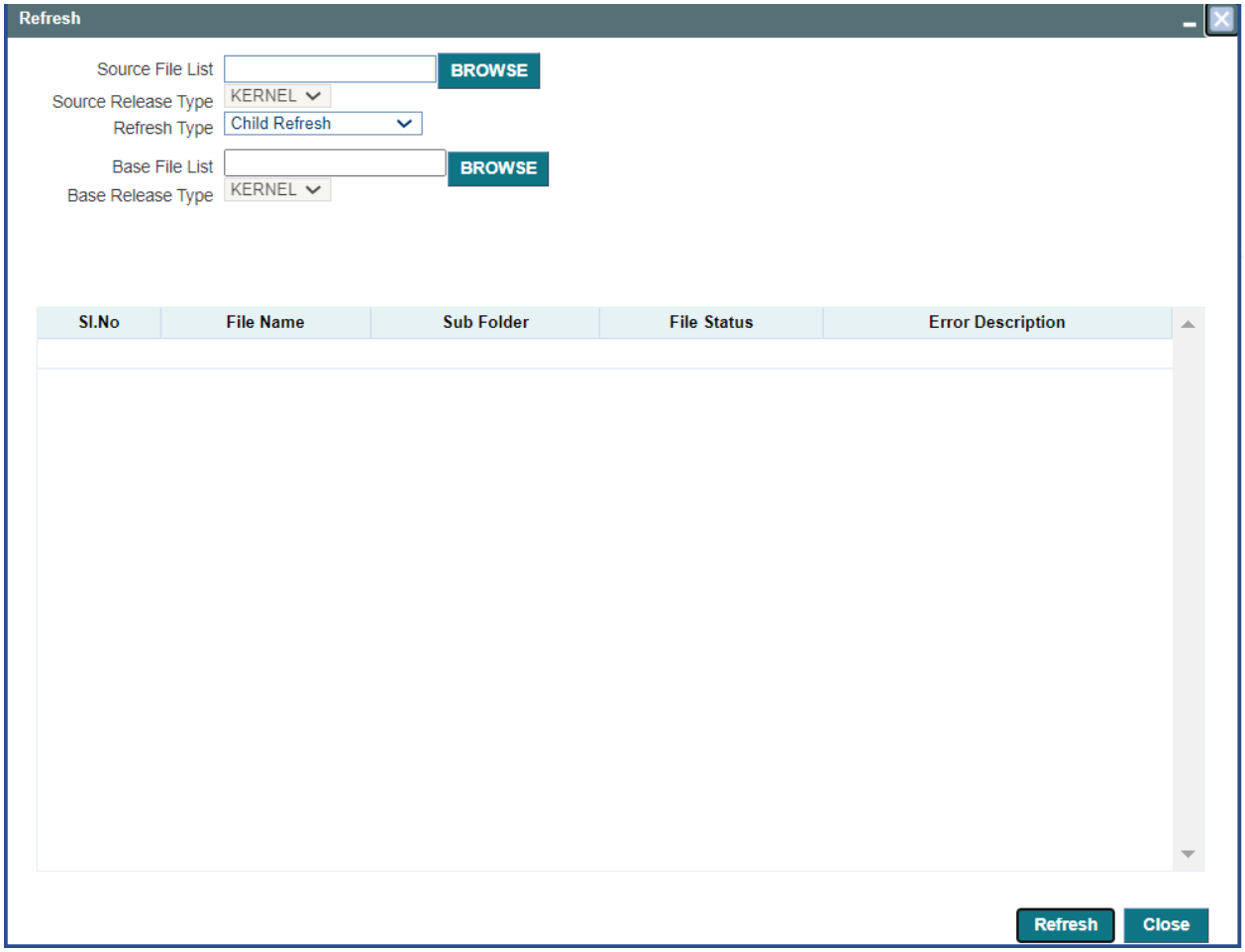


Fig 4.1.2: Workbench Refresh Screen

Source File List: Browse and select the text file containing source file list.

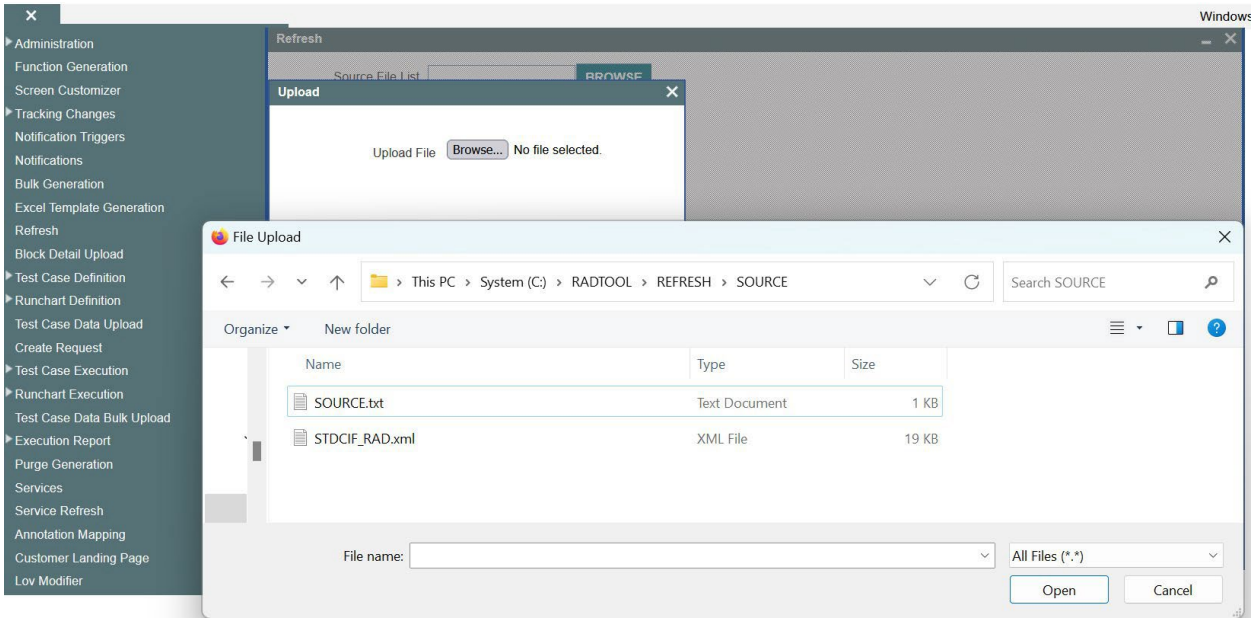


Fig 4.1.3: Selecting source file list text file for Child Refresh

Source File list is a text file which contains the absolute path of all the child radxmls to be refreshed.

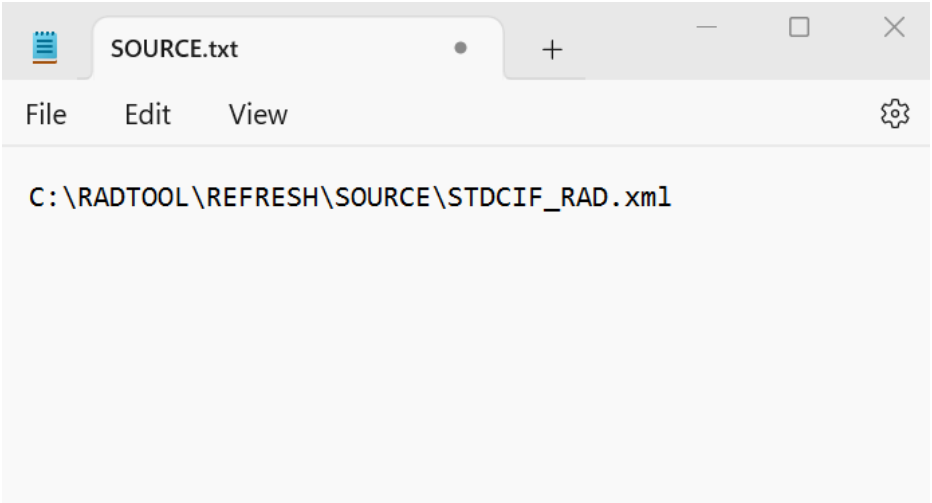


Fig 4.1.4: Content of source.txt file

The figure above shows the content of the source.txt file .Here STDCIFS is the child radxml which has to be refreshed.

If child refresh of more than one functionId is required, absolute path of each child radxmls has to be specified; each in a new line

Base File List: Browse and select the text file containing base file list

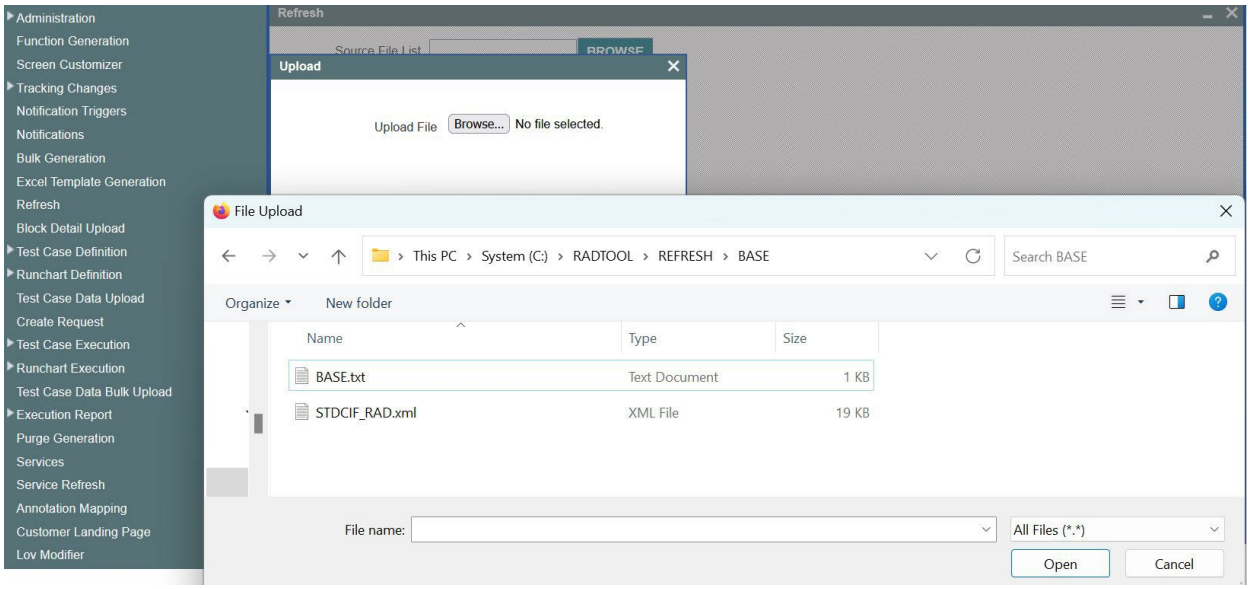


Fig 4.1.5: Selecting base file list text file for Child Refresh

Base File list is a text file which contains the absolute path of all the parent radxmls to be refreshed (here STDCIF is the parent radxml)

If child refresh of more than one functionId is required, absolute path of each parent radxmls has to be specified; each in a new line

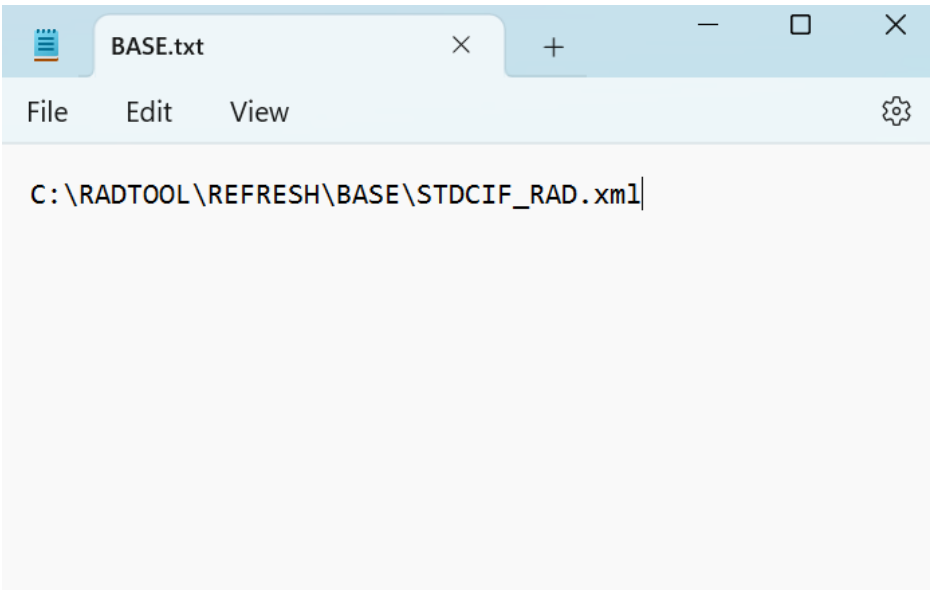


Fig 4.1.6: Content of base.txt file

File Location: Choose file location as client if the path provided is in the client machine .
Refresh Type: Choose Refresh type as Child Refresh
Source Release type and base release type will be disabled for child refresh as the release type of both parent and child is assumed to be same .

Click on Refresh button on lower left portion of the screen and wait for the system to do the process.
Process time will vary depending on the number of files provided, size of each files etc
After Completion of the process, status will be shown in the screen. File status will be successful if refresh is successful.

Save Mode should be either Client Path or Save path for Refresh activity. Zip mode is not supported. Files will be generated in the Work Directory specified.

Generated Files:

1) **Refreshed Radxml :**A folder named RADXML will be created within the source file path which will contain refreshed files for the particular source(child) radxml.
   For instance, if source file path is D:\REFRESH\SOURCE\ STDCIF_RAD.xml;
   refreshed file can be found at D:\REFRESH\SOURCE\RADXML\ STDCIF_RAD.xml
   For child refresh of multiple files, it is recommended to place all source radxmls in one folder so that generated files could be found at a single location
2) **Log Files :** Following log files will be generated

i) **Refresh Log :** This contains the status of all the files refreshed.

ii) **Refresh Report :** This file can be used for troubleshooting .

4.2 Functionality Demonstration

In the above Child Refresh process, STDCIFS is refreshed with the latest STDCIF.

From the screen preview it can be noted that in the child screen many changes has been done which had resulted in a very different layout. Many field sets which were part of the parent screen has been made hidden and new field sets containing new fields has been introduced in the child screen.

Now we load STDCIF in Workbench in the current release and made some modifications to it as required. A new field COUNTRY AND NATION have been introduced

Child Refresh of STDCIFAD is done as explained in previous section.

The system units(main packages, language xml.sys js ,xsd's etc) are regenerated by loading the refreshed radxml and deployed .All the units need to be regenerated.

Here we can find that the field added in parent screen has come in the child screen as well. Meanwhile ,other differences we have noticed between the initial parent and child screens has not come up as they were over ridden in the child functionId.

Hence we find that the changes done in the parent has come up in the child while retaining the changes done in the child .Note that only screen layout changes has been explained in this demonstration for ease of understandability ;this is applicable for all nodes(e.g.: cal form, launch form, lovs etc)

5 Screen Child Refresh

Screen Child Refresh allows the developer to upgrade a screen child radxml with its latest parent radxml .In doing so; the latest changes done in parent functionId would be reflected in the screen child functionId while retaining all the changes done in the screen child level

- This process is to be done within a release. i.e. screen child functionId has to be refreshed with its parent functionId from the same release

- If the parent functionId of the screen child is a child screen ,then it is recommended that child refresh of that screen to be carried out before doing screen child refresh

- It is recommended that this process is done before development cut of the release for all child radxmls within a release. For instance; if development has happened parallel for a screen child and its parent functionId during a release, screen child refresh should be done before base lining so that screen child and parent record types are consistent

- All the system units need to be regenerated after Screen Child Refresh. A thorough unit testing is recommended after deployment of all generated units. Note that only frontend units will be generated for a screen child functionId.

5.1 Process Steps

For explanation purpose two dummy functionId's has been used :

STDCIF: parent screen

STDCIFAD: screen child of STDCIF

Process steps are similar to child refresh. Refer section 4.2 for more detailed explanation

In the Refresh Page, provide input to fields as:

Source File List: Browse and select the text file containing source file list

Source File list is a text file which contains the absolute path of all the screen child radxmls to be refreshed.

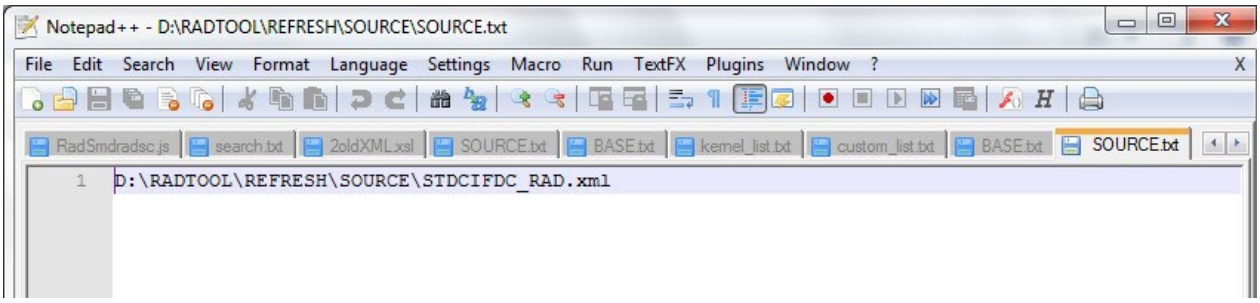


Fig 5.1.1: Content of source.txt file

The figure above shows the content of the source.txt file .Here STDCIFS is the screen child radxml which has to be refreshed.

If screen child refresh of more than one functionId is required, absolute path of each screen child radxmls has to be specified; each in a new line

Base File List: Browse and select the text file containing base file list

Base File list is a text file which contains the absolute path of all the parent radxmls to be refreshed (here STDCIF is the parent radxml)

If screen child refresh of more than one functionId is required, absolute path of each parent radxmls has to be specified; each in a new line

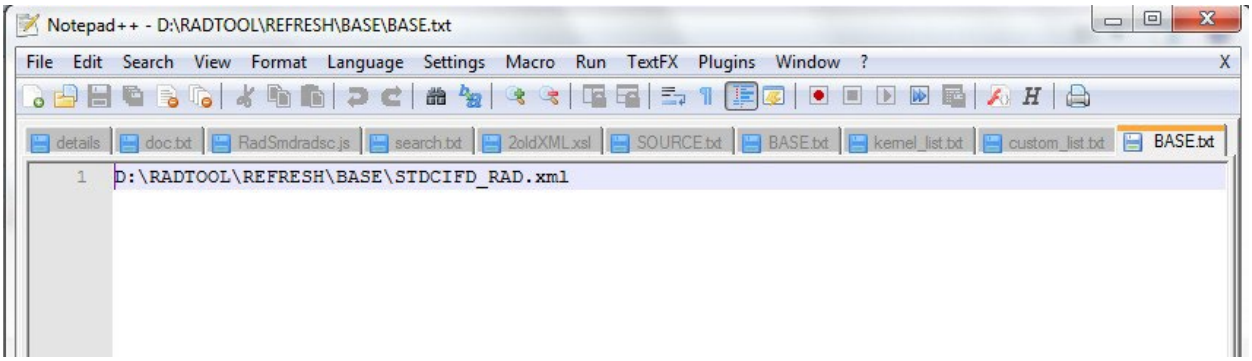


Fig 5.1.2: Content of base.txt file

File Location: Choose file location as client if the path provided is in the client machine.

Refresh Type: Choose Refresh type as Screen Child Refresh

Source Release type and base release type will be disabled for Screen child refresh as the release type of both parent and child is assumed to be same.

Click on Refresh button.

After Completion of the process, status will be shown in the screen. File status will be successful for refresh is successful.

After Completion of the process, status will be shown in the screen. File status will be successful if refresh is successful.

5.2 Functionality Demonstration

In the above Child Refresh process, STDCIFAD is refreshed with the latest STDCIF.

Let us assume that some changes are done in STDCIF as part of the current release. New field has been added and introduced in the screen. Preview of STDCIF main screen after changes is shown below

Find the newly added fields (Nationality and Language) placed in a new field set highlighted in the figure

Do screen Child Refresh for STDCIFAD with the latest parent (i.e. STDCIF with new fields and field set). Regenerate system units for the refreshed radxml and deploy .

Here we can find that new fields and field set added in the parent has come in the screen child while the original screen child changes has also been retained.

6 Source Refresh

Source Refresh allows the customer upgrade his existing release with latest release of Flexcube without affecting his custom changes .By using source refresh option all the extensible radxml's of older version can be updated with latest version changes.

- Source Refresh is possible only for the extensible screens. Hence for non extensible screens customization on the screens can't be retained in case of upgrade
- Source Refresh is done for radxmls in different releases.
- All system units needs to be regenerated after source refresh . A thorough unit testing is recommended after deployment of all generated units
- Child and Screen Child Refresh will be done implicitly during Source Refresh if any child/screen child screens are present .Hence if source refresh of any child/screen child has to be done, include parent radxmls also in the source and base file lists
- Select proper release types for source and base while upgrading in Refresh Page.

It is meaningless to do source refresh between two Kernel versions (or two cluster versions etc) as we can replace the entire source with latest version in such scenario. *Hence Source and Base Release types can never be the same for Source Refresh*

Source release type cannot be **Kernel** it can be either Cluster or Custom. Base Release type options will depend on the source release type selected.

Source Release Type	Cluster	Custom
Base Release Type	Kernel	Kernel, Cluster

- If user selects custom as source release type he has option to upgrade his release based on either cluster pack or Kernel.
- If user selects Cluster as source release type we have only one option as base release type i.e. Kernel.

6.1 Process Steps

Consider a bank which is running on 12.0 version of Flexcube .Bank has done custom developments on top of 12.0 Kernel version .Now bank is upgrading to 12.0 sources Here we consider the case of a single functionId (STDCIF) for demonstration

Process steps are similar to child refresh. Refer section 4.2 for more detailed explanation

In the Refresh Page, provide input to fields as:
Source File List: Browse and select the text file containing source file list

Source File list is a text file which contains the absolute path of all the source release radxmls to be refreshed. Here 11.3 custom radxmls used by bank is the source .

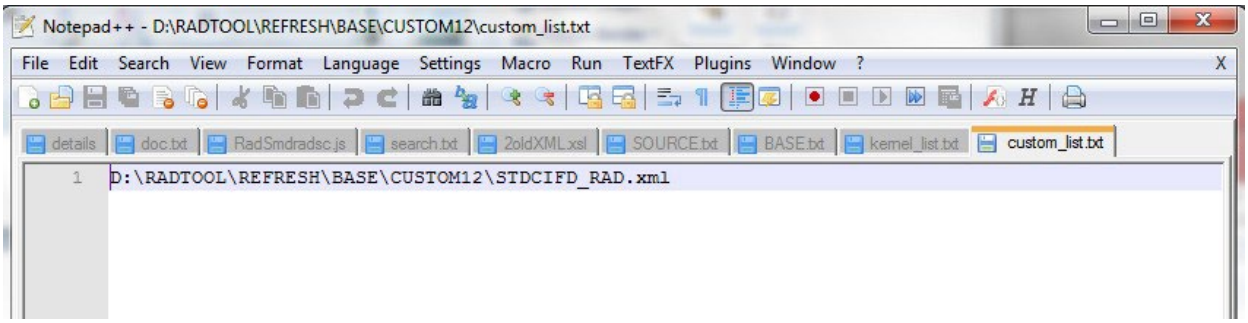


Fig 6.1.1: Content of custom_list.txt file

The figure above shows the content of the source.txt file .Here STDCIF is the 12.0 custom version radxml which has to be upgraded to 12.2 .If source refresh of more than one functionId is required, absolute path of each source radxmls has to be specified; each in a new line

Base File List: Browse and select the text file containing base file list

Base File list is a text file which contains the absolute path of all base version radxmls with which source has to be refreshed. If source refresh of more than one functionId is required, absolute path of each base version radxmls has to be specified; each in a new line

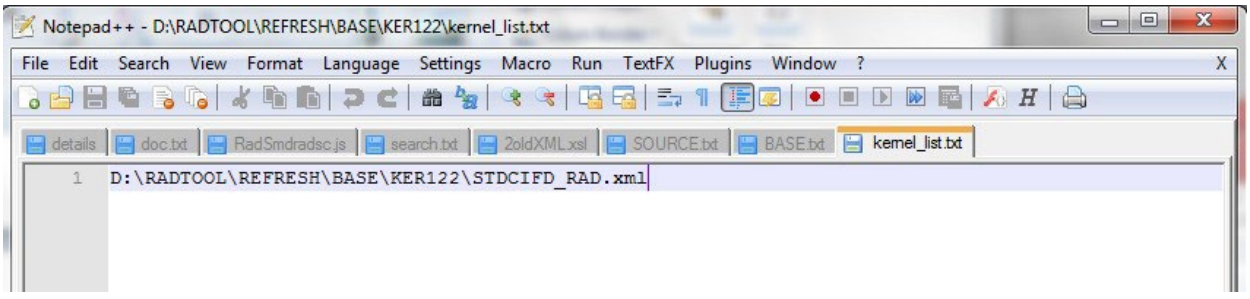


Fig 6.1.2: Content of Kernel_list.txt file

File Location: Choose file location as client if the path provided is in the client machine.
Refresh Type: Choose Refresh type as Source Refresh
Source Refresh Type: Source files are of custom release type (12.0 custom version) ,hence provide source refresh type as custom

Base Refresh Type: Base files are from 12.2 Kernel release. Hence select base release type as Kernel. Click on Refresh button.

6.2 Functionality Demonstration

In the above section process for upgrading a 12.0 custom release functionId (STDCIF) with its 12.2 version is explained

The figure below shows the preview of STDCIF screen as used by the bank;i.e.12.0 custom version

In custom version, Auto Generate button which was present in 12.0 Kernel version was not required; hence made hidden. Highlighted section shows the original position of Auto Generate button in kernel version of 12.0 .

Notice some of the changes done in 12.2 Kernel version highlighted in the figure

1) Country field is added in the body
2) Nationality fields is added in body

Also note that Auto Generate button has been retained in 12.2 Kernel version from 12.0 Kernel

Do Source Refresh as explained in the previous section.

Regenerate all system units (main package, language xml, sys js ,xsds etc) and deploy in Flexcube server

Compile/Deploy Kernel sources (kernel packages, kernel js etc) from the base release (12.2 here) in Flexcube server.

Here we can observe that changes from 12.2 Kernel are now reflected in the custom version also.

1) Country field in body has come in the refreshed file

2 Nationality field of body has also come up in the refreshed screen from the base version

3) Auto Generate button has not come in the Refreshed screen even though it was present in the base screen. This is because it was made hidden in the custom version. Custom changes are retained

6.3 Source refresh is not possible in below scenarios

1. If Parent Function id LOV modified after child refresh if same LOV used in Child function id after next refresh those LOV'S Should be Modified manually

2. If Fields data type are changed (examples Increase/Decreased length, Number to varchar2) in custom layer, Later Kernel changed same fields data type those changes would not be reflected in Custom layer. If these changes requires in Custom layer manually (Using ODT Function generation options to refresh data type) needs to change

3. The Following Custom attributes are changed in custom layer, Later Kernel same custom attributes are changes (Add/Deleted) those changes would not be reflected in Custom layer. If these changes requires in Custom layer (Using ODT Function generation options to refresh data type) manually needs to change

   Radio Button, Static List values

4. There should not be nay naming conflicts in elements across releases (KERNEL, CLUSTER, CUSTOM) or across the parent-child functions For example: same field set should not be created in both Parent and Child Function Hence a standard naming convention has to be followed for each release type/and function type so that names does not conflict. For instance: All Fieldsets in a Custom Parent has to follow convention like FST_U.

5. Upgrade feature in not available for Summary Nodes



Development Workbench - Source Upgrade
October 2026

Version 14.8.3.0.0

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/

Copyright © 2007, 2026, Oracle and/or its affiliates.

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.