Oracle® Banking Corporate Lending

Development Workbench – Source Upgrade

Release 14.8.2.0.0
G53380-02
April 2026

Oracle Banking Corporate Lending Development Workbench – Source Upgrade, Release 14.8.2.0.0

G53380-02

Contents

Preface

Purpose i
Acronyms and Abbreviations i
Audience i
Critical Patches ii
Conventions ii
Diversity and Inclusion ii
Documentation Accessibility ii
Related Resources iii
Screenshot Disclaimer iii

1 Overview of Refresh Functionality in Development Workbench

2 Child Refresh

2.1 Perform Child Refresh 1
2.2 Functionality Demonstration - Child Refresh 5

3 Screen Child Refresh

3.1 Perform Screen Child Refresh 1
3.2 Functionality Demonstration- Screen Child Refresh 4

4 Source Refresh

4.1 Perform Source refresh 2
4.2 Functionality Demonstration - Source Refresh 4
4.3 Source Refresh - Not Possible Scenarios 6

Preface

This topic contains the following sub-topics:

- Purpose
- Acronyms and Abbreviations
- Audience
- Critical Patches
- Conventions
- Diversity and Inclusion
- Documentation Accessibility
- Related Resources
- Screenshot Disclaimer

Purpose

This manual is designed to help acquaint you with the Refresh functionality available in Oracle FLEXCUBE Development Workbench for Universal Banking and guides the developers on how to use this feature.

Acronyms and Abbreviations

Table 1 Acronyms and Abbreviations

Acronyms	Abbreviations
FCUBS	Oracle FLEXCUBE Universal Banking Solution
OBCL	Oracle Banking Corporate Lending
ODT	Oracle Development Tool

Audience

This document is intended for Oracle FLEXCUBE Universal Banking Application developers/ users that use Development Workbench to develop various Oracle FLEXCUBE Universal Banking components. To use this manual, the user needs a conceptual and working knowledge of the below:

Table 2 Proficiency Details

Proficiency	Resources
Oracle FLEXCUBE Universal Banking Technical Architecture	Training programs from Oracle Financial Software Services.
Working knowledge of Web based applications	Self Acquired
Working knowledge of Oracle Database	Oracle Documentations

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.

Conventions

The following text conventions are used in this document:

Table 3 Conventions

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.

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.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Related Resources

For more information on any related features, refer to the following documents:

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

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.

1

Overview of Refresh Functionality in Development Workbench

This topic provides an 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 be done using the Tool:

1. **Child Refresh**
2. **Screen Child Refresh**
3. **Source Refresh**

2 Child Refresh

This topic provides information on Child Refresh.

Child Refresh allows the developer to upgrade a child RADXML with its latest parent RADXML. The latest changes done in parent function id would be reflected in the child function id while retaining all the changes done in the child level.

1. This process is to be done within a release. i.e. child function id has to be refreshed it's the parent function id from the same release.
2. It is recommended that this process is done before the development cut of the release for all child RADXMLs within a release. For instance; if development has happened parallel for a child and parent function id during a release, child refresh should be done before baselining so that child and parent record types are consistent.
3. All the system units need to be regenerated after Child Refresh. A thorough unit testing is recommended after the deployment of all generated units.

   This topic contains the following sub-topics:

- Perform Child Refresh
  This topic provides systematic instructions to perform Child Refresh.
- Functionality Demonstration - Child Refresh
  This topic provides an overview of Child Refresh Functionality.

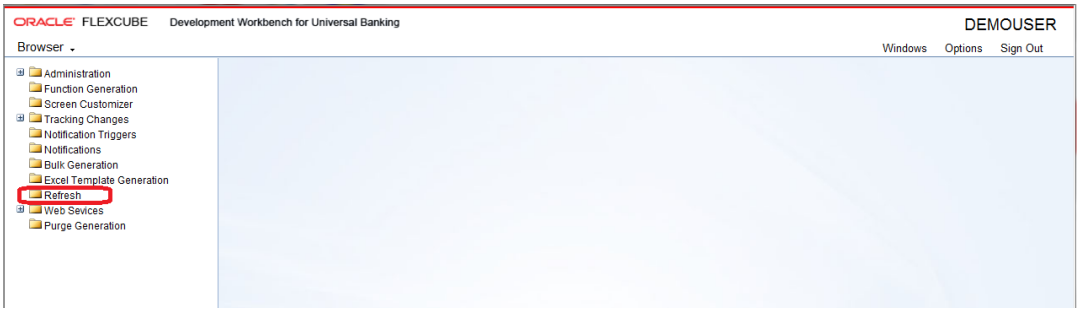
2.1 Perform Child Refresh

This topic provides systematic instructions to perform Child Refresh.

1. On Development Workbench Login page, specify the **Username** and **Password** and log in to the Development Workbench landing page.

   The **Development Workbench For Universal Banking** screen displays.

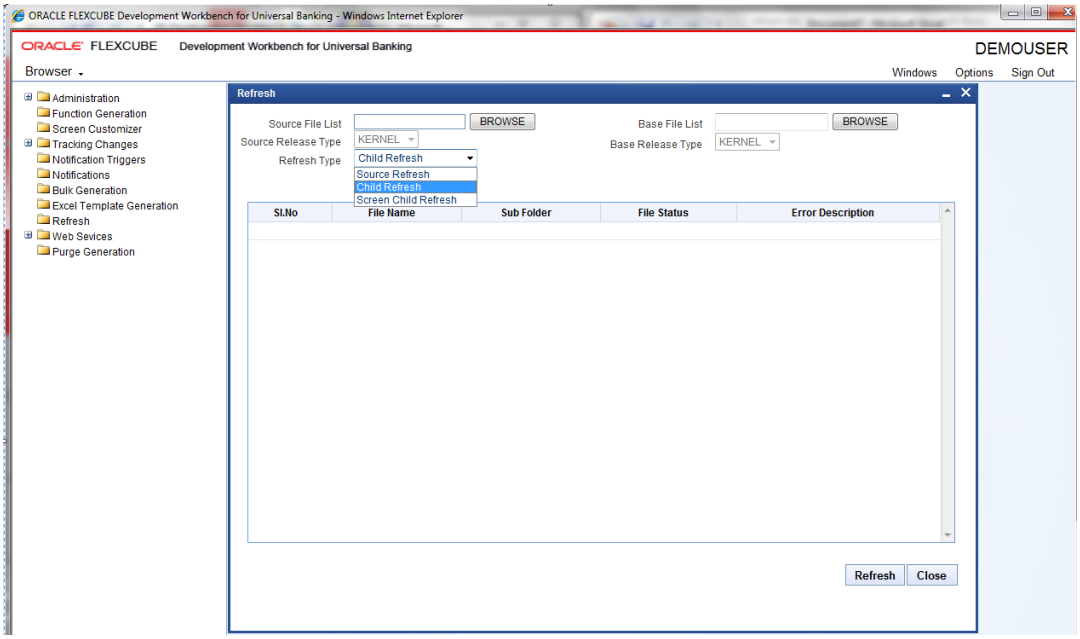
Figure 2-1 Development Workbench For Universal Banking



2. Click on the **Refresh** node under the **Browser** menu.

   The **Refresh** screen displays.

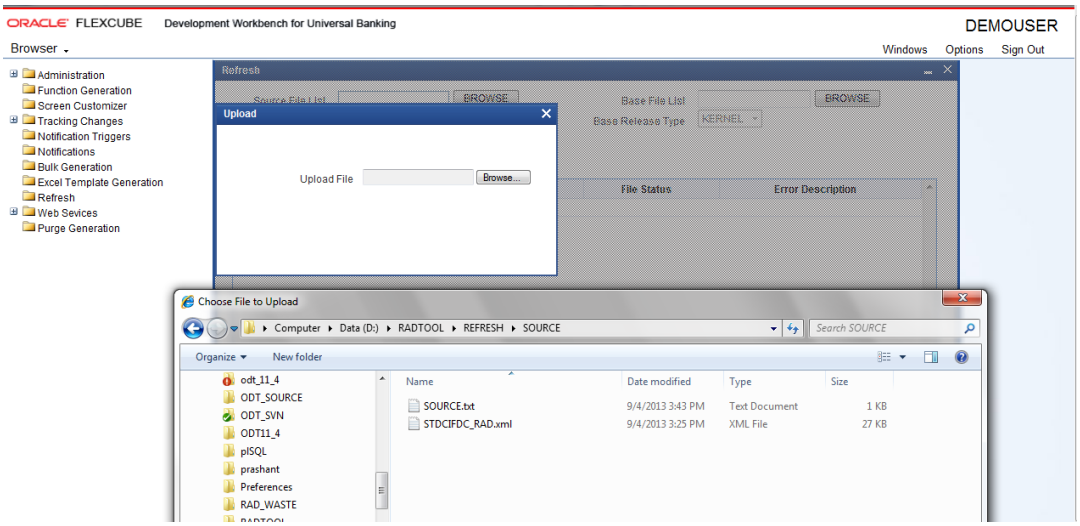
Figure 2-2 Refresh



3. Click the **BROWSE** button at **Source File List** field.

   The **Upload** and **Choose File to Upload** windows displays.

Figure 2-3 Upload- Source File List

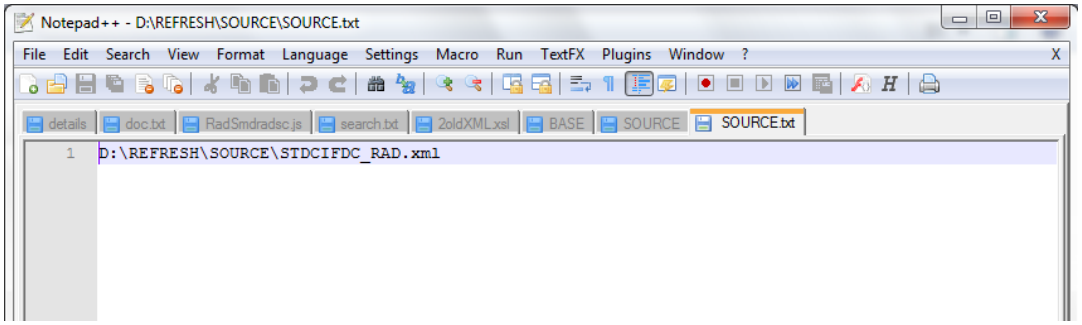


> **Note**
>
> **Child Refresh** process is explained taking two hypothetical function Id's:
>
> - **STDCIFD**: Parent screen
> - **STDCIFDC**: Child screen

4. Select the text file containing the source file list from the folder at **Choose File to Upload** window and click on **Open** button.

   The source file list is a text file that contains the absolute path of all the child RADXMLs to be refreshed.

Figure 2-4 Content of source file



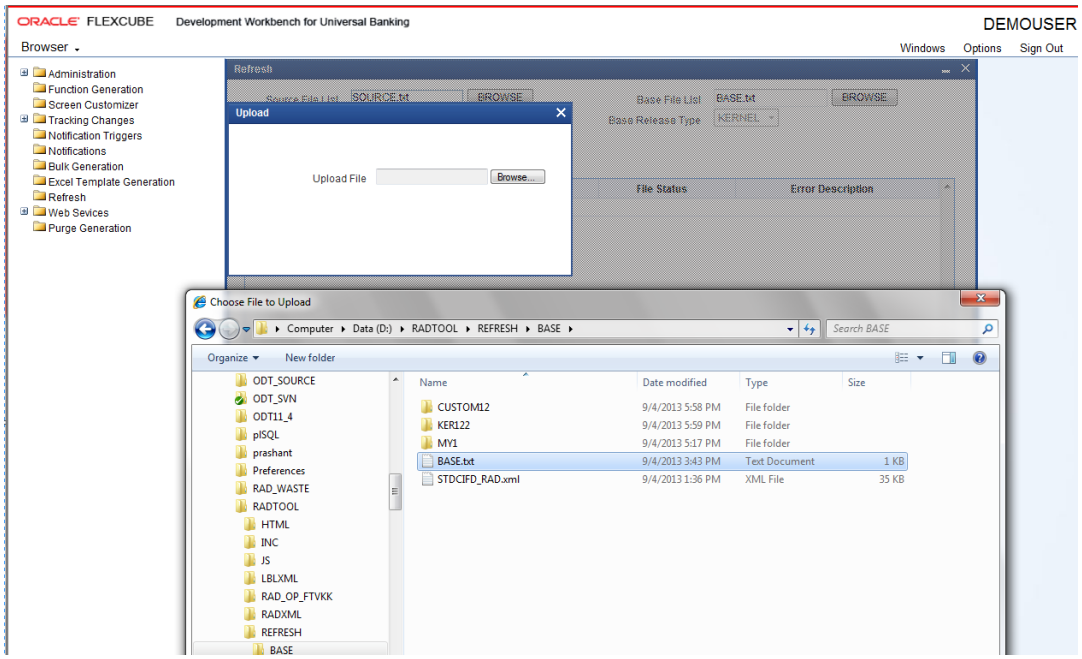
The figure above shows the content of the **source.txt** file. Here **STDCIFDC** is the child RADXML that has to be refreshed. If a child refresh of more than one function id is required, the absolute path of each child RADXMLs has to be specified, each in a new line.

The selected text file displays in the **Source File List** field.

5. Click on the **BROWSE** button at **Base File List** field.

   Upload and **Choose File to Upload** windows displays.

Figure 2-5 Upload- Base File List

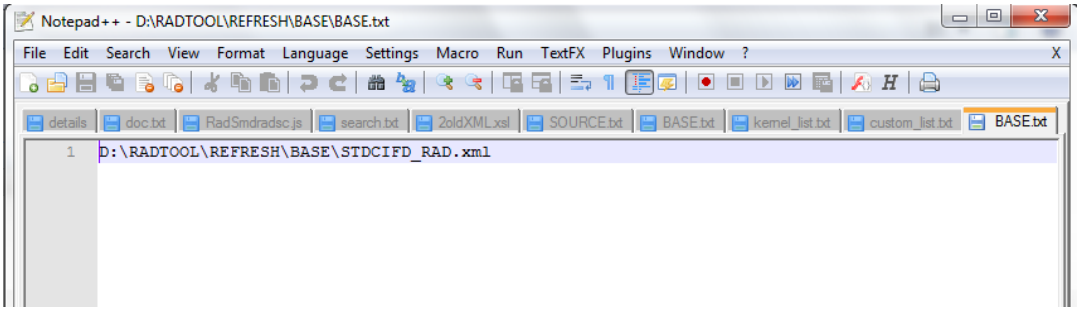


6. Select the text file containing the base file list from the folder at **Choose File to Upload** window and click on **Open** button.

   The base File list is a text file that contains the absolute path of all the parent RADXMLs to be refreshed (here **STDCIFD** is the parent RADXML). If a child refresh of more than one

function id is required, the absolute path of each parent RADXMLs has to be specified; each in a new line.

Figure 2-6 Content of base file



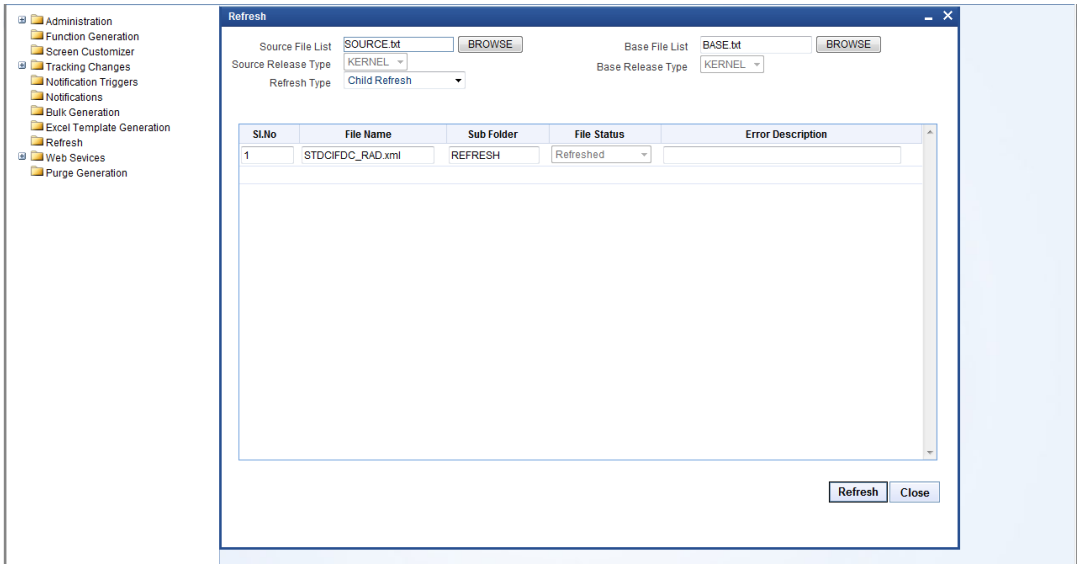
7. Choose **File Location** as a client if the path provided is in the client machine.
8. Choose **Refresh Type** as **Child Refresh**.

   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 the same.
9. Click on the **Refresh** button on the 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 file, etc

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

Figure 2-7 Child Refresh- File Status



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.

Table 2-1 Generated Files

Generated Files	Description
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 the source file path is `D:\REFRESH\SOURCE\ STDCIFD_RAD.xml`; the refreshed file can be found at `D:\REFRESH\SOURCE\RADXML\ STDCIFD_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.
Log Files	Following log files will be generated: a. **Refresh Log**: This contains the status of all the files refreshed. b. **Refresh Report**: This file can be used for troubleshooting.

2.2 Functionality Demonstration - Child Refresh

This topic provides an overview of Child Refresh Functionality.

In the topic *Perform Child Refresh* process, **STDCIFD** is refreshed with the latest **STDCIFD**. The figure below shows the preview of **STDCIFD** and **STDCIFDC** main screens before the refresh.

Figure 2-8 STDCIFD- Before Refresh

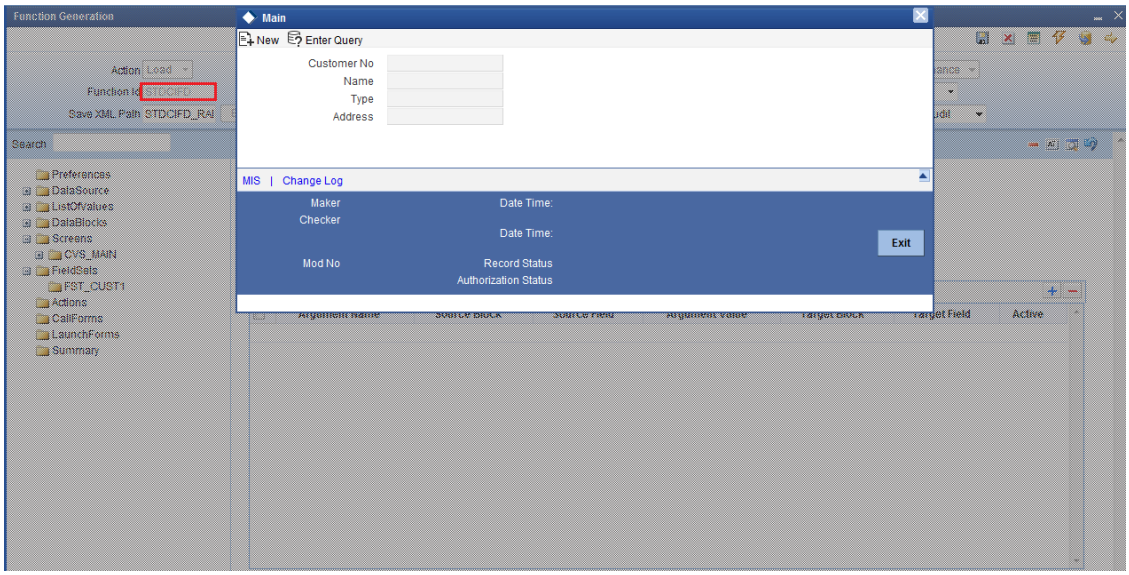
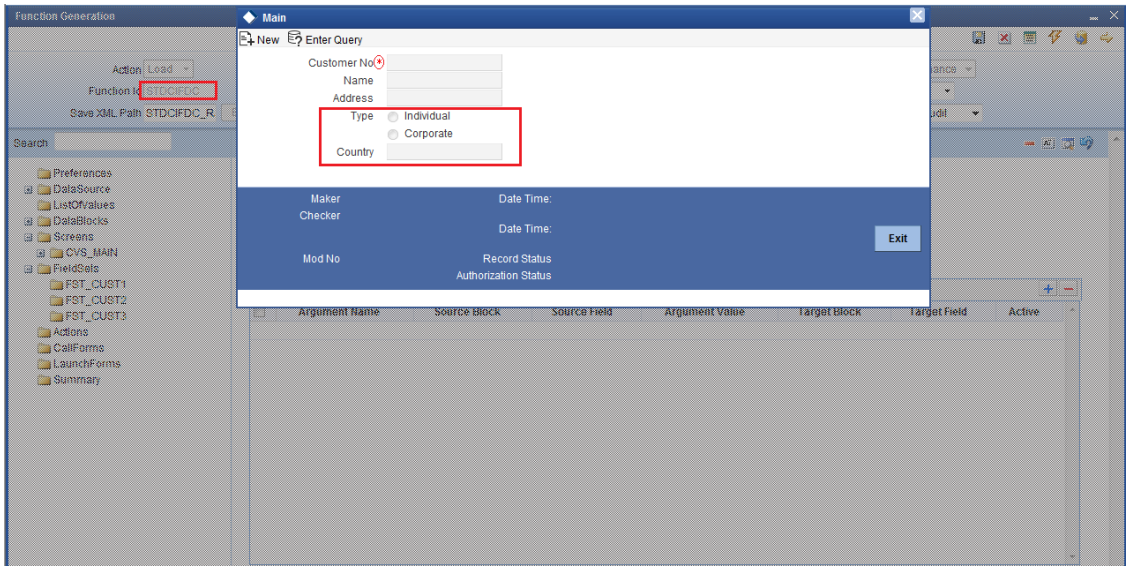


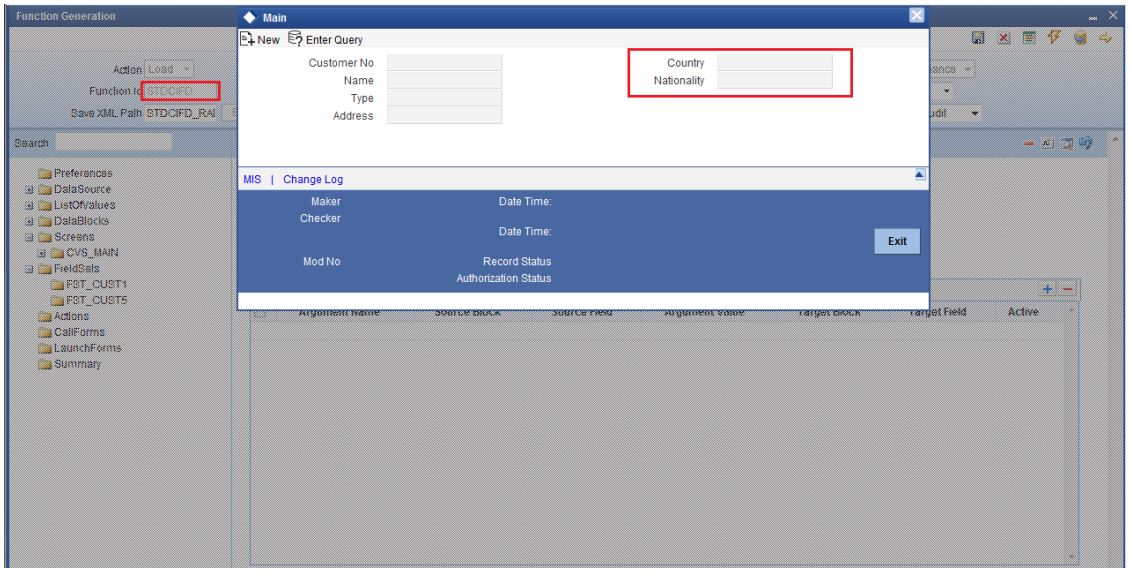
Figure 2-9 STDCIFDC- Before Refresh



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

Now we load **STDCIFD** in Workbench in the current release and made some modifications to it as required. A new field **COUNTRY AND NATION** have been introduced. A preview of **STDCIFD** after the modifications is shown below. Note the newly added field highlighted.

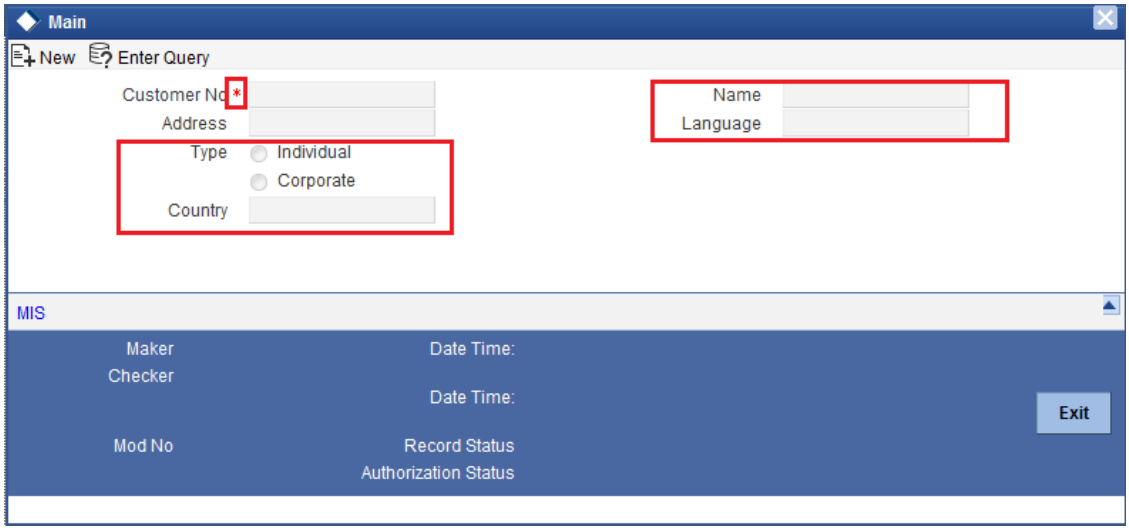
Figure 2-10 STDCIFD - After Modification



Child Refresh of **STDCIFDC** is done as explained in the topic *Perform Child Refresh*. 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. Preview of **STDCIFDC** main screen after the refresh is shown below:

Figure 2-11 STDCIFDC- After Child Refresh



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

Hence we find that the changes done in the parent have come up in the child while retaining the changes done in the child. Note that only screen layout changes have been explained in this demonstration for ease of understandability; this is applicable for all nodes (For example: Call Form, Launch Form, LOVs, etc.).

3
Screen Child Refresh

This topic provides information on Screen Child Refresh.

Screen Child Refresh allows the developer to upgrade a screen child RADXML with its latest parent RADXML. The latest changes done in parent function id would be reflected in the screen child function id while retaining all the changes done in the screen child level.

1. This process is to be done within a release. i.e. screen child function id has to be refreshed with its parent function id from the same release.
2. If the parent function id of the screen child is a child screen, then it is recommended that the child refresh of that screen be carried out before doing screen child refresh.
3. It is recommended that this process is done before the 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 function id during a release, screen child refresh should be done before baselining so that screen child and parent record types are consistent.
4. All the system units need to be regenerated after **Screen Child Refresh**. A thorough unit testing is recommended after the deployment of all generated units. Note that only frontend units will be generated for a screen child function id.

   This topic contains the following sub-topics:

- Perform Screen Child Refresh
  This topic provides systematic instructions to perform Screen Child Refresh.
- Functionality Demonstration- Screen Child Refresh
  This topic provides an overview of Screen Child Refresh Functionality.

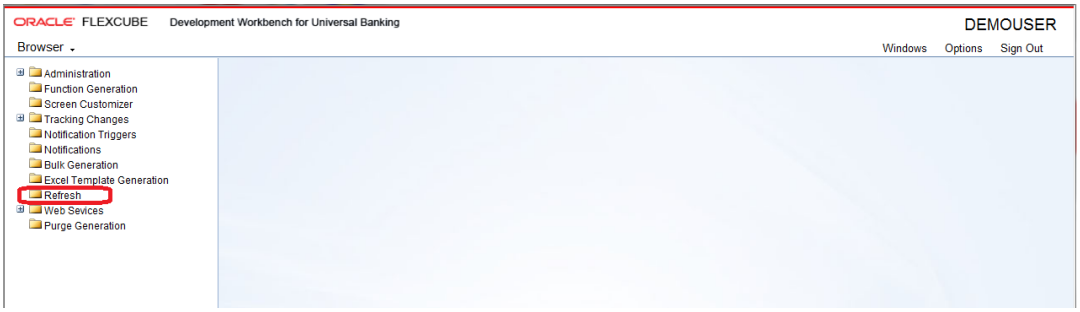
3.1 Perform Screen Child Refresh

This topic provides systematic instructions to perform Screen Child Refresh.

1. On Development Workbench Login page, specify the **Username** and **Password** and log in to the Development Workbench landing page.

   The **Development Workbench For Universal Banking** screen displays.

Figure 3-1 Development Workbench For Universal Banking



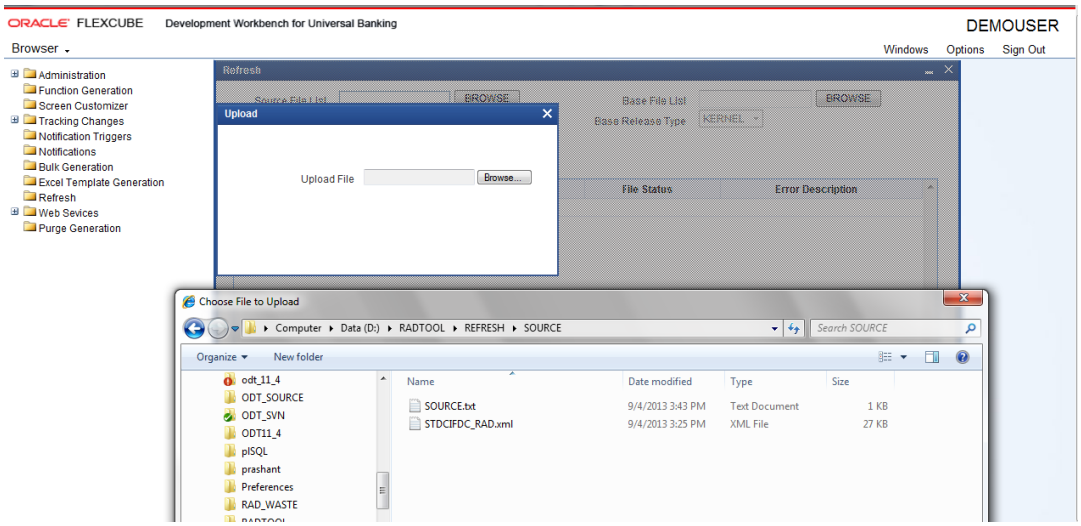
2. Click the **Refresh** node under the **Browser** menu.

Refresh screen displays.

3. Click the **BROWSE** button at **Source File List** field.

   Upload and **Choose File to Upload** windows displays.

Figure 3-2 Upload - Source File List

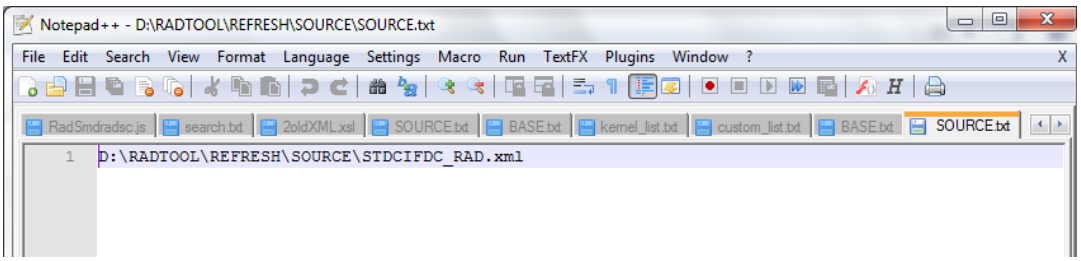


> **Note**
>
> For explanation purpose two dummy function Id's has been used:
>
> - **STDCIFD**: Parent screen
> - **STDCIFDC**: Screen child of **STDCIFDC**

4. Select the text file containing the source file list from the folder at **Choose File to Upload** window and click on **Open** button.

   The source file list is a text file that contains the absolute path of all the child RADXMLs to be refreshed.

Figure 3-3 Content of source file



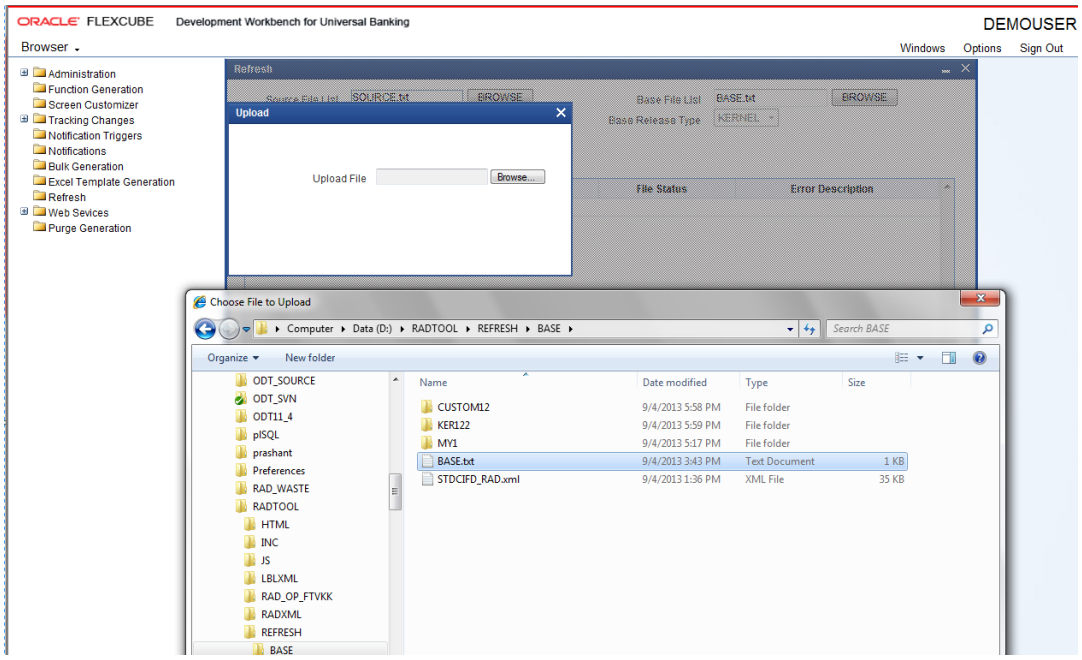
The figure above shows the content of the **source.txt** file. Here **STDCIFDC** is the child RADXML that has to be refreshed. If a child refresh of more than one function id is required, the absolute path of each child RADXMLs has to be specified, each in a new line.

The selected text file displays in the **Source File List** field.

5. Click on the **BROWSE** button at **Base File List** field.

   Upload and **Choose File to Upload** windows displays.

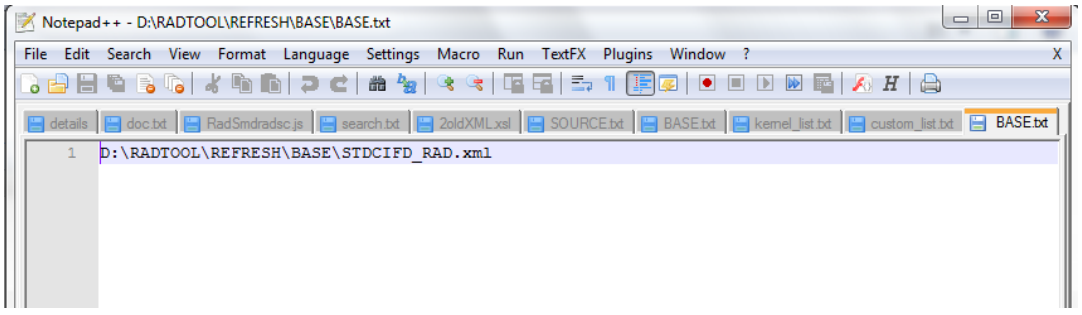
Figure 3-4 Upload- Base File List



6. Select the text file containing the base file list from the folder at **Choose File to Upload** window and click on **Open** button.

   The base File list is a text file that contains the absolute path of all the parent RADXMLs to be refreshed (here **STDCIFDC** is the parent RADXML). If a child refresh of more than one function id is required, the absolute path of each parent RADXMLs has to be specified; each in a new line.

Figure 3-5 Content of Base File

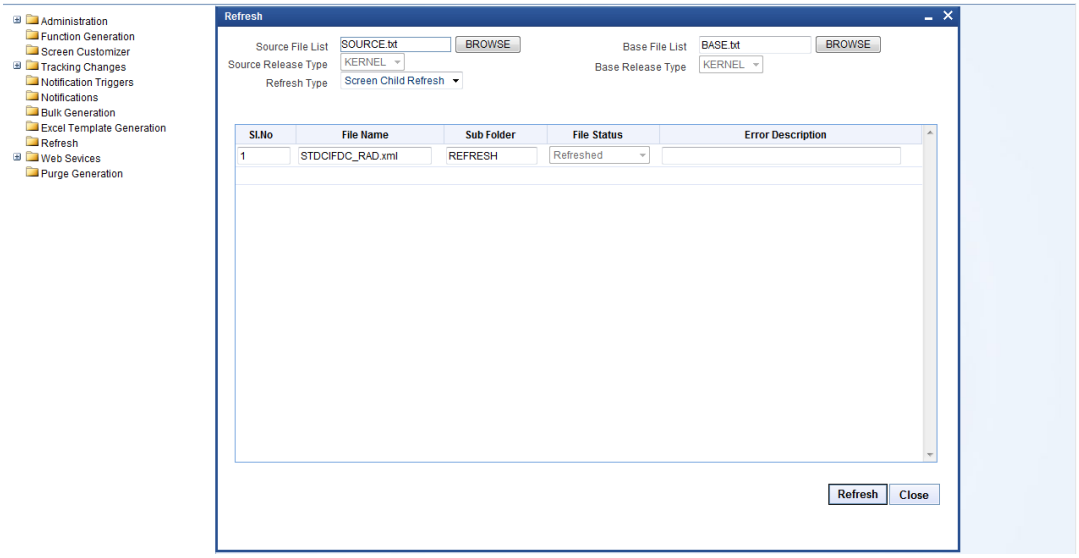


7. Choose **File Location** as a client if the path provided is in the client machine.
8. Choose **Refresh Type** as **Screen Child Refresh**.
9. Click on the **Refresh** button on the lower-left portion of the screen and wait for the system to do the process.

   Process time will vary depending on the number of files provided, size of each file, etc

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

Figure 3-6 Screen Child Refresh - File Status



3.2 Functionality Demonstration- Screen Child Refresh

This topic provides an overview of Screen Child Refresh Functionality.

In the topic *Perform Child Refresh*, **STDCIFDC** is refreshed with the latest **STDCIFD**. The figure below shows the preview of **STDCIFD** and **STDCIFDC** main screens before screen child refresh.

Figure 3-7 STDCIFD - Before Screen Child Refresh

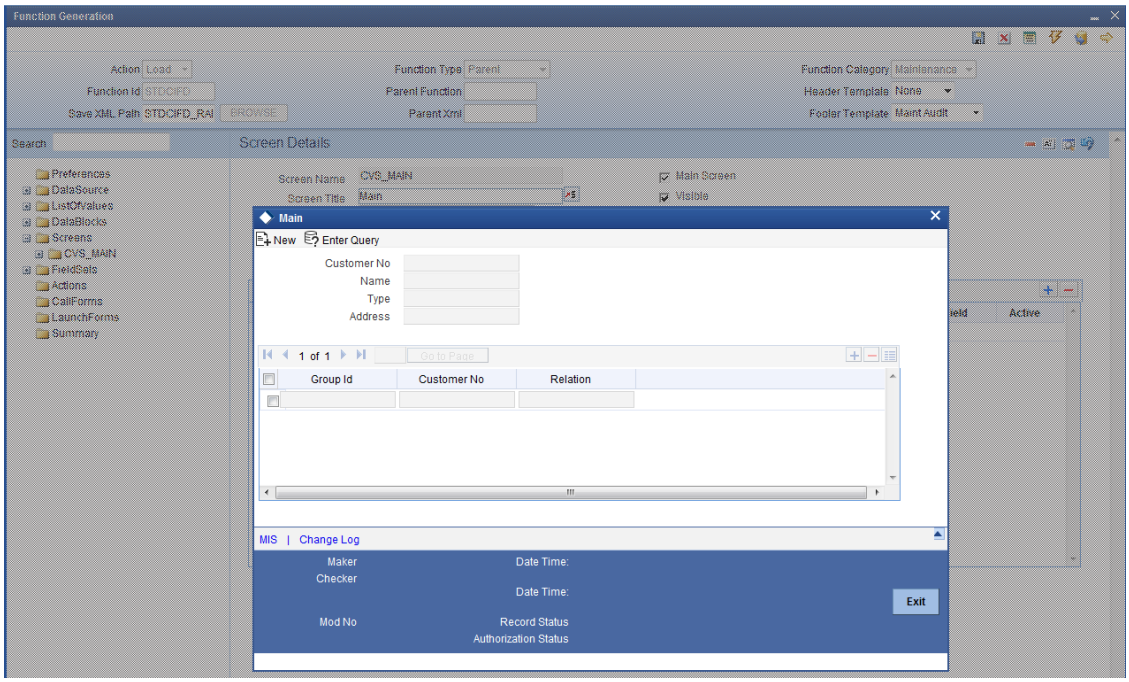
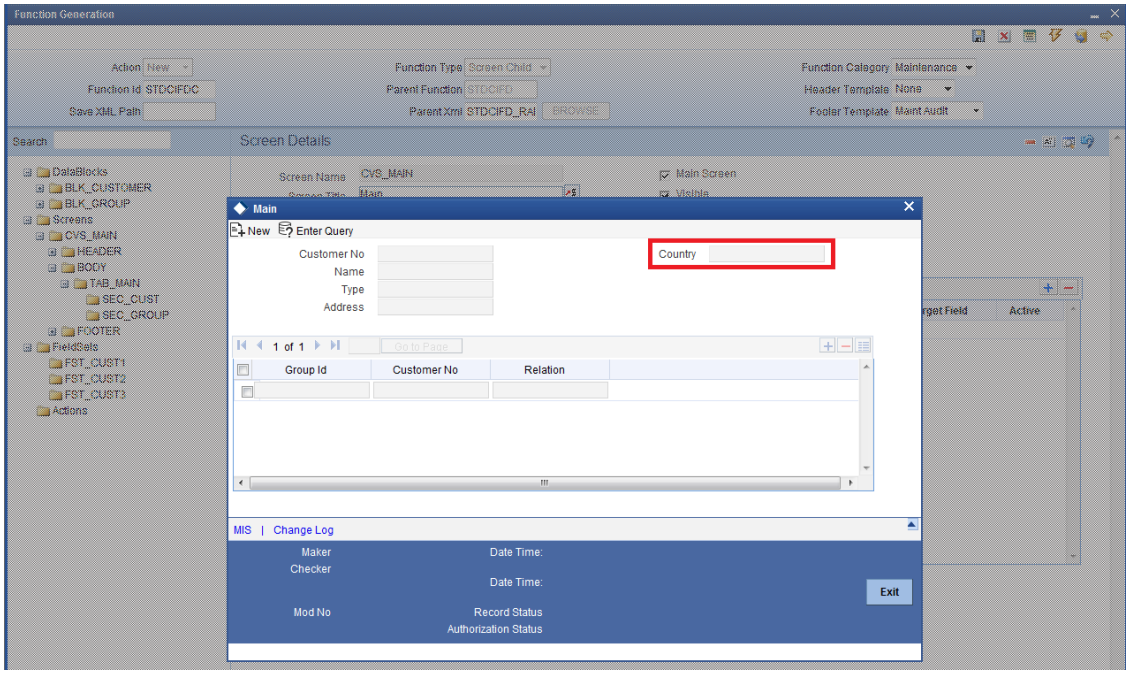
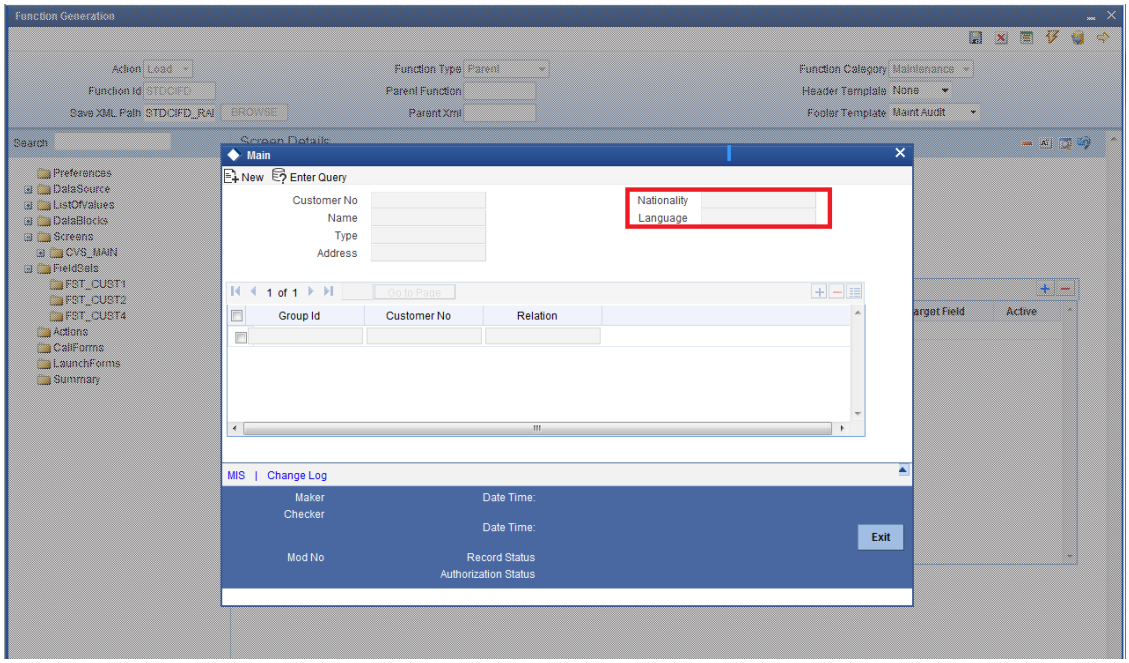


Figure 3-8 STDCIFDC- Before Screen Child Refresh



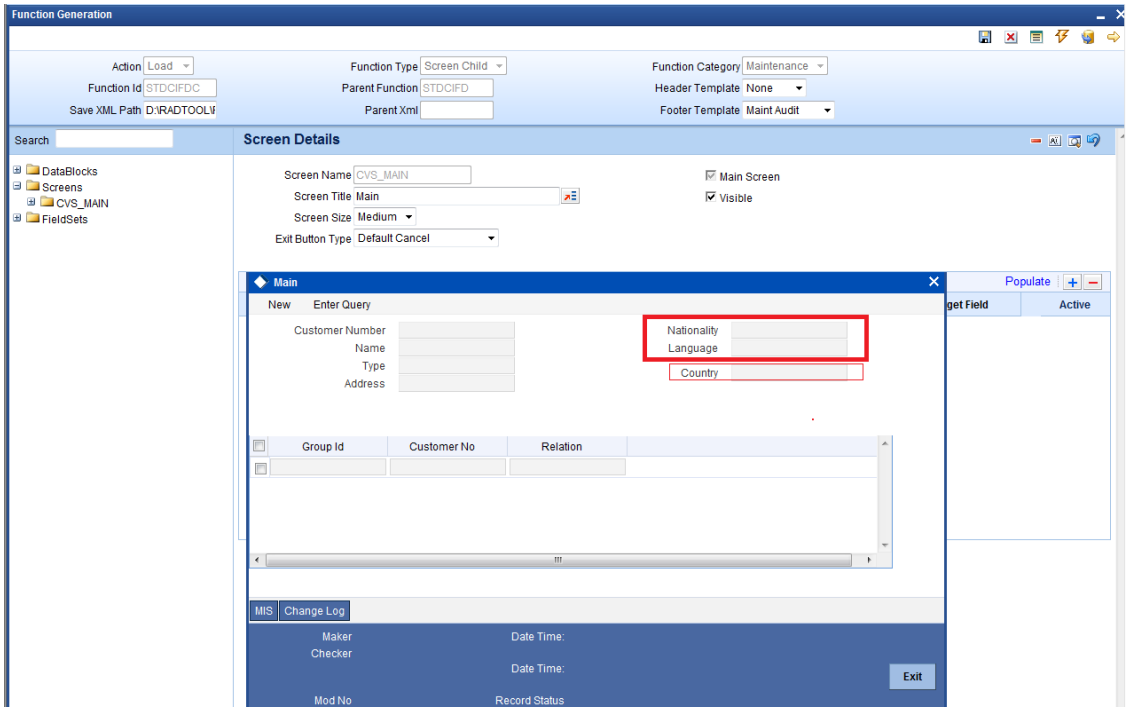
Let assume that some changes are done in **STDCIFD** as part of the current release. The new field has been added and introduced to the screen. Preview of **STDCIFD** main screen after changes is shown below. Find the newly added fields (**Nationality** and **Language**) placed in a new fieldset highlighted in the figure.

Figure 3-9 STDCIFD - After Screen Child Refresh



Do **Screen Child Refresh** for **STDCIFD** with the latest parent (i.e. **STDCIFD** with new fields and fieldset). Regenerate system units for the refreshed RADXML and deploy.

Figure 3-10 STDCIFDC - Post Screen Child Refresh



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

4
Source Refresh

This topic provides information on Source Refresh.

Source Refresh allows the customer to upgrade its existing release with the latest release of FLEXCUBE without affecting its custom changes. By using the **Source Refresh** option all the extensible RADXML‘s of older versions can be updated with the latest version changes.

1. **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 an upgrade.
2. **Source Refresh** is done for RADXMLs in different releases.
3. All system units need to be regenerated after source refresh. A thorough unit testing is recommended after the deployment of all generated units.
4. **Child Refresh** 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.
5. 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 the latest version in such a scenario. Hence Source and Base Release types can never be the same for **Source Refresh**.

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

Table 4-1 Release Type for Source and Base

Release Type	Option 1	Option 2
Source Release Type	**Cluster**	**Custom**
Base Release Type	**Kernel**	**Kernel**, **Cluster**

- If the user selects **Custom** as source release type, the user has the option to upgrade release based on either cluster pack or Kernel.
- If the user selects **Cluster** as source release type, the user has only one option as base release type i.e. **Kernel**.

This topic contains the following sub-topics:

- Perform Source refresh
  This topic provides systematic instructions to perform Source refresh.
- Functionality Demonstration - Source Refresh
  This topic provides an overview of Source Refresh Functionality.
- Source Refresh - Not Possible Scenarios
  This topic provides systematic instructions to Source Refresh - Not Possible Scenarios.

4.1 Perform Source refresh

This topic provides systematic instructions to perform Source refresh.

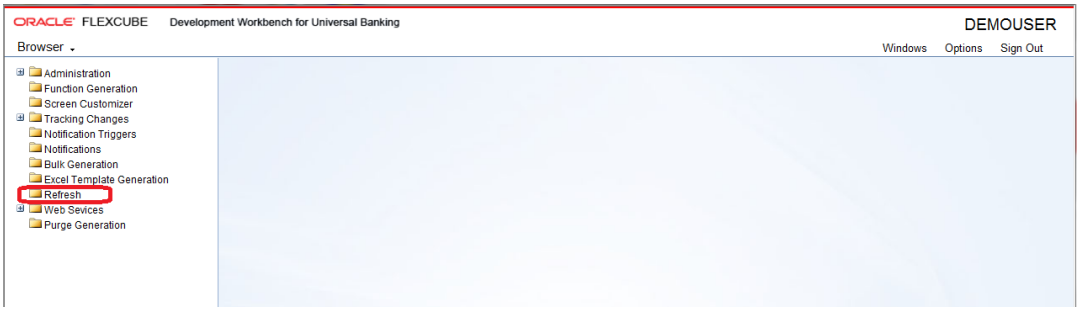
Consider a bank which is running on 12.0 version of Flexcube. Bank has done custom developments on top of 12.0 Kernel version. Now bank is upgrading to 12.0 sources Here we consider the case of a single function ID (UTDCIFD) for demonstration Process steps are similar to child refresh. Refer to the topic *Functionality Demonstration- Child Refresh* for more detailed explanation.

Process steps are similar to child refresh.

1. On Development Workbench Login page, specify the **Username** and **Password** and log in to the Development Workbench landing page.

   The **Development Workbench For Universal Banking** screen displays.

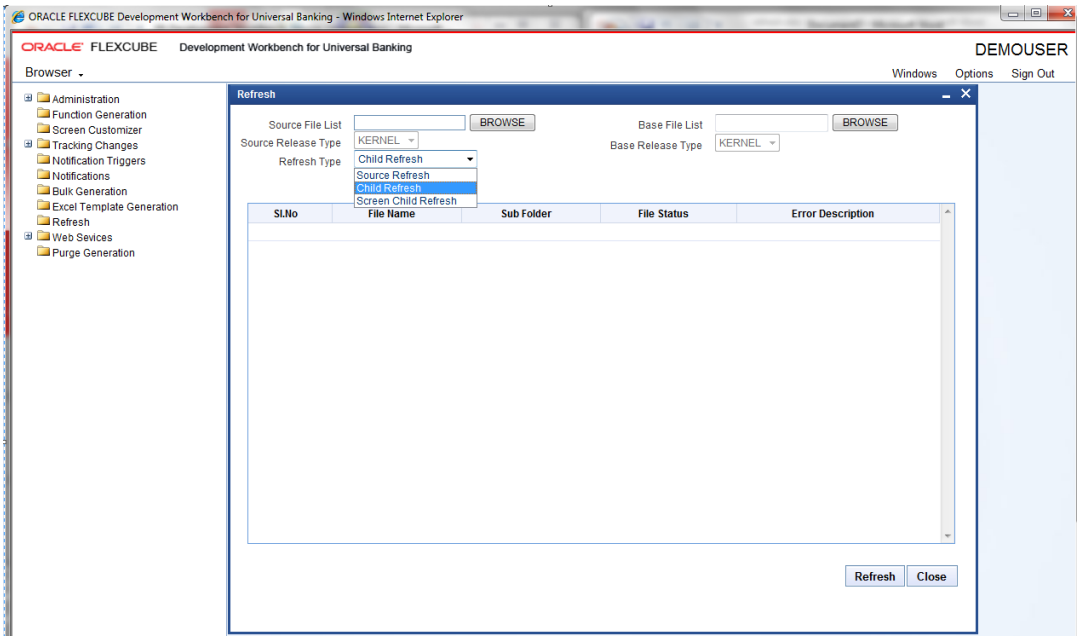
Figure 4-1 Development Workbench For Universal Banking



2. Click on the **Refresh** node under the **Browser** menu.

   Refresh screen displays.

Figure 4-2 Refresh



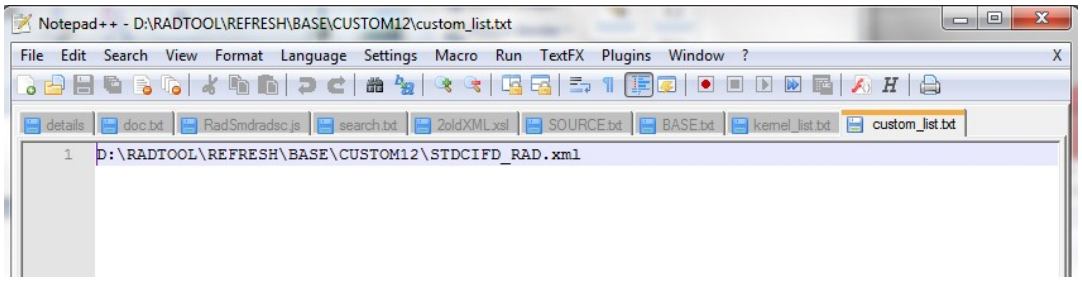
3. Click on the **BROWSE** button at **Source File List** field.

   Upload and **Choose File to Upload** windows displays.

4. In the Refresh Page, provide input to fields as:

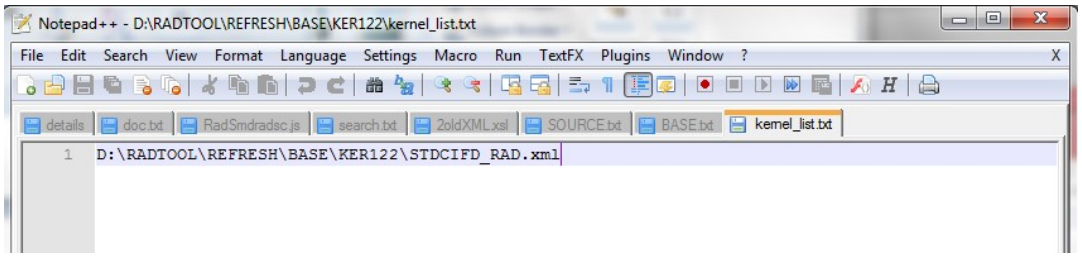
   Source File List: Source File list is a text file which contains the absolute path of all the source release radxmls to be refreshed. Here 11.3 custom radxmls used by bank is the source.

Figure 4-3 Content of Custom List



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 function ID is required, absolute path of each base version radxmls has to be specified; each in a new line.

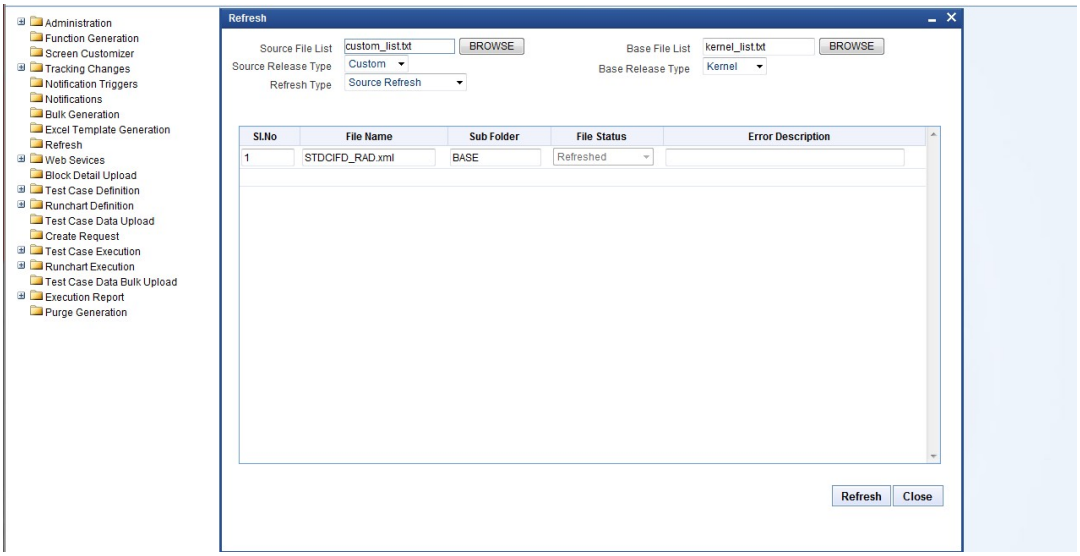
Figure 4-4 Content of Kernel List



5. Select the text file containing the source file list from the folder at **Choose File to Upload** window and click on **Open** button.

   The source file list is a text file that contains the absolute path of all the child RADXMLs to be refreshed.
   After Completion of the process, status will be shown in the screen. File status will be successful for refresh is successful.

Figure 4-5 Source Refresh - File Status



4.2 Functionality Demonstration - Source Refresh

This topic provides an overview of Source Refresh Functionality.

In the above topic, process for upgrading a 12.0 custom release function Id (STDCIFD) with its 12.2 version is explained.

The figure below shows the preview of the **STDCIFD** screen as used by the bank (i.e.12.0 custom version). In the custom version, the **Auto Generate** button which was present in the 12.0 Kernel version was not required; hence made hidden. The highlighted section shows the original position of the **Auto Generate** button in the Kernel version of 12.0.

Figure 4-6 STDCIFD Screen- 12.0 Kernel version

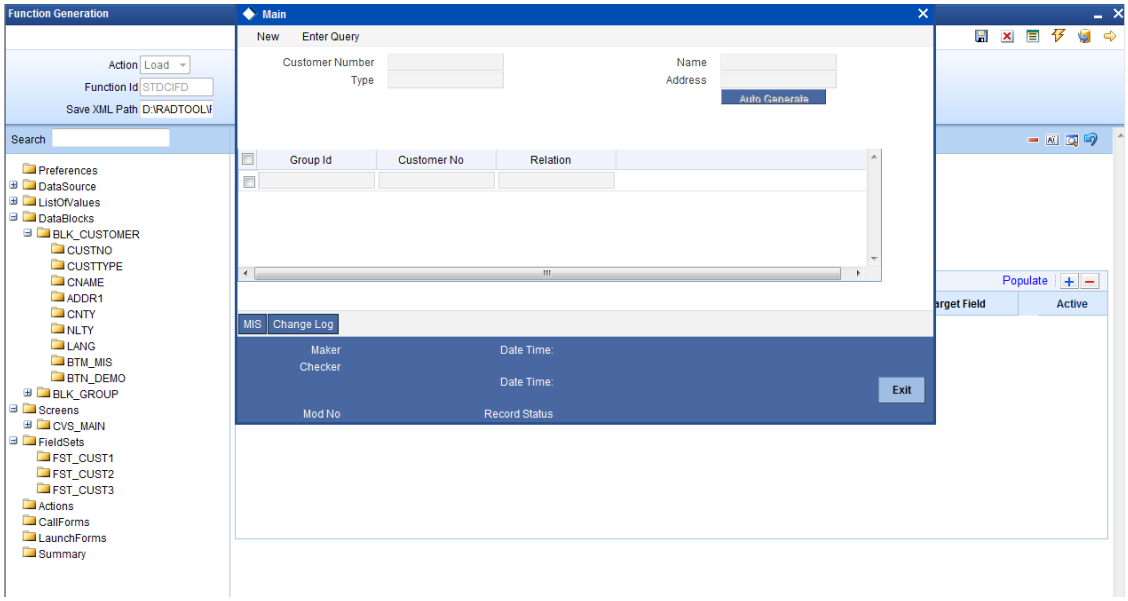
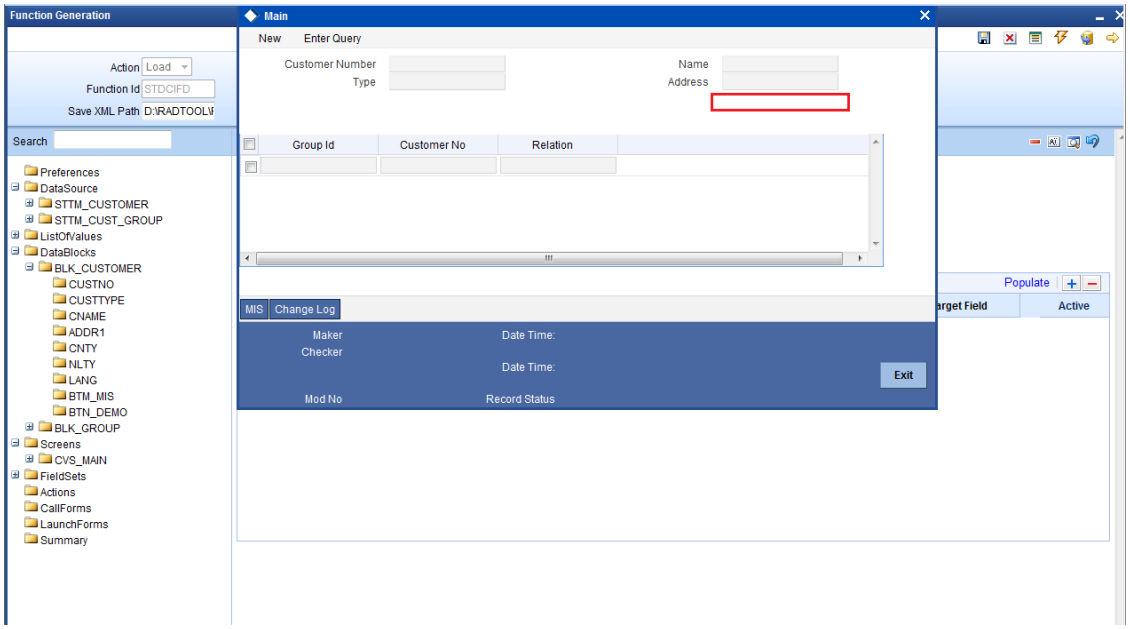
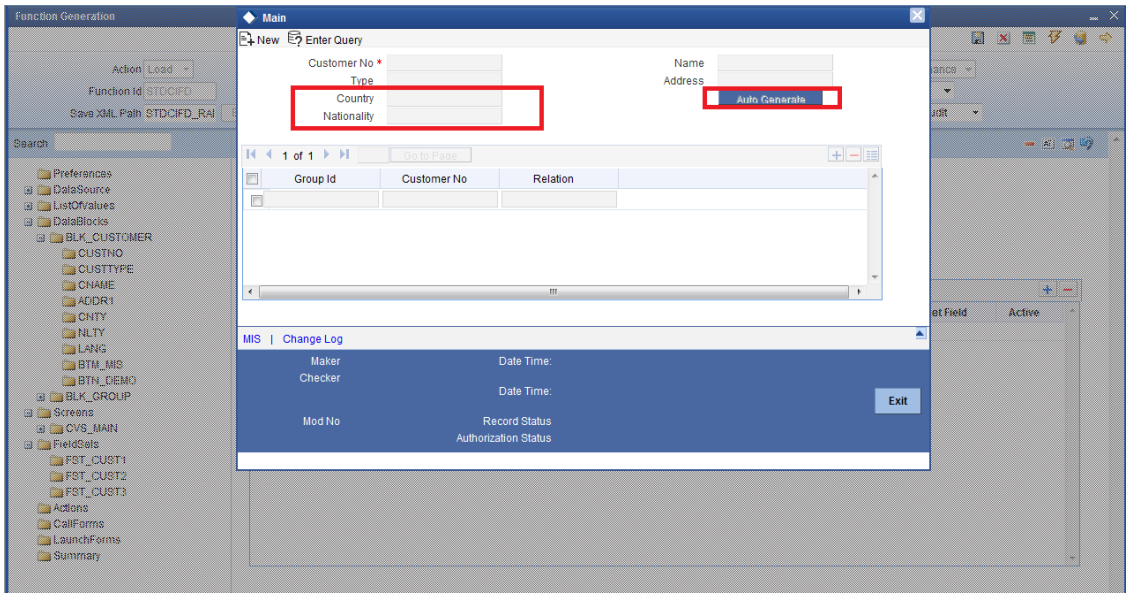


Figure 4-7 STDCIFD Screen- 12.0 Custom Kernel version



The figure below shows the preview of the 12.2 Kernel version of STDCIFD.

Figure 4-8 STDCIFD Screen- 12.2 Kernel version



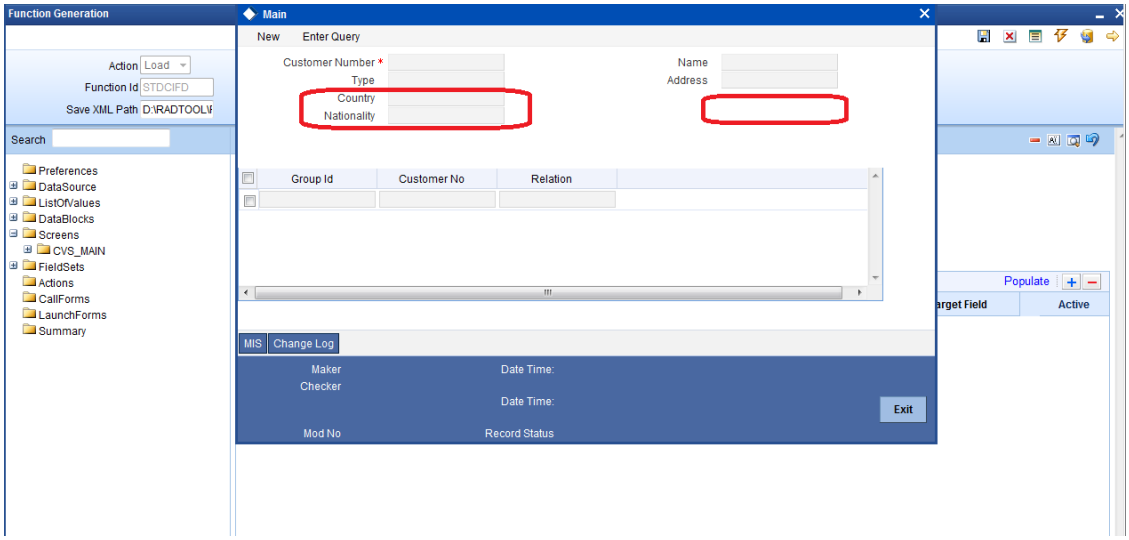
Some of the changes done in the 12.2 Kernel version are highlighted in the above figure:

- **Country** field is added.
- **Nationality** field is added.

> **Note**
>
> The **Auto Generate** button has been retained in the 12.2 Kernel version from the 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 them in the Flexcube server. Compile/ Deploy Kernel sources (kernel packages, kernel js, etc) from the base release (12.2 here) in the Flexcube server. The figure below shows the preview of the screen after **Source Refresh**.

Figure 4-9 STDCIFD Screen- Post Refresh



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

1. **Country** field in the body has come in the refreshed file.
2. **Nationality** field of the 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.

4.3 Source Refresh - Not Possible Scenarios

This topic provides systematic instructions to Source Refresh - Not Possible Scenarios.

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 Field Sets in a Custom Parent has to follow convention like FST_U.
5. Upgrade feature in not available for Summary Nodes.