Oracle® Banking Treasury Management

Child and Screen Child - Concept and Design

Release 14.8.1.0.0
G45878-01
October 2025

ORACLE

Oracle Banking Treasury Management Child and Screen Child - Concept and Design, Release 14.8.1.0.0

G45878-01

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*
- *Child and Screen Childs - Concept and Design*
- *Development of Maintenance Form*
- *Development of Online Form*
- *Development of Call Form*
- *Development of Launch Forms and Other Screens*
- *Development of Dashboard Form*
- *Development Workbench Service XML Development*
- *Development Workbench Performance Tuning Enhancements*
- *Development Workbench - Rest Services Development*

1

Design Child and Screen Child

This topic explains the concept and design of Child and Screen Child screens available in Oracle FLEXCUBE Development Workbench for Universal Banking.

This topic contains the following sub-topics:

- Child screen
  This topic provides an overview on **Child** screen creation.
- Screen Child
  This topic provides an overview on **Screen Child** creation.

1.1 Child screen

This topic provides an overview on **Child** screen creation.

Screens are required in **Oracle FLEXCUBE Investor Servicing** where the base functionality is the same and the differences are either in the layout or in some additional processing that is there are some sets of screens where the majority of the functionality is the same with some variance with existing screens.

Development Workbench supports developing such screens as the child screens of the base function with a facility to upgrade the child screens whenever the base or parent screen changes.

For example: **Term Deposit Account Booking**

The account for term deposit booking will have all the features of a normal customer account with some additional features. Thus it can be designed as the child screen of normal customer account maintenance.

Workbench provides an option to design a child screen as a derivative of a parent screen. Workbench automatically inherits the parent screen and tracks the modifications made in the child screen. Workbench does not allow the developer to delete elements inherited from the parent screen.

For child screens, in addition to release specific hooks for each logical stage, the system also calls the corresponding hook from the parent function's programs. This allows the reuse of the common business logic among all the child functions. **Oracle FLEXCUBE Investor Servicing** does not support multi-level inheritance (that is the child of the Child is not supported).

This topic contains the following sub-topics:

- Develop New Child screen
  This topic provides the systematic instructions to develop a new Child screen.
- Possible Operations in Child screen
  This topic briefs the important considerations to be made when designing a **Child** screen.
- Generation of Files for Child screen
  This topic explains the process of generation of files for a Child screen.
- Extensible Development for Child screen
  This topic provides information on the Extensible Development of a Child screen.

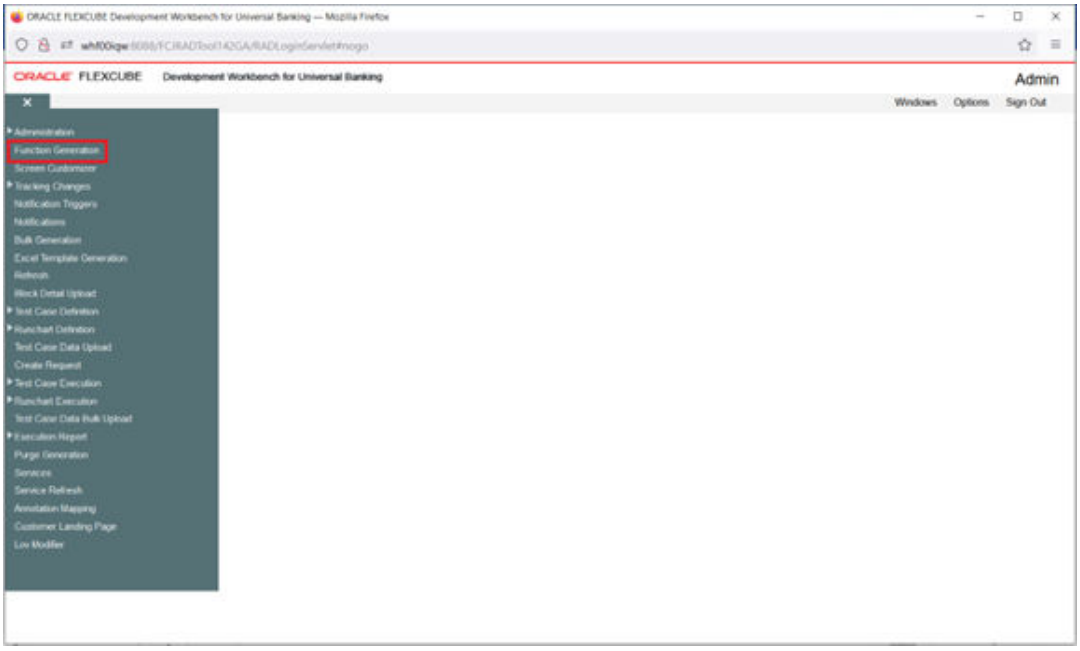
1.1.1 Develop New Child screen

This topic provides the systematic instructions to develop a new Child screen.

1. On **Expand Menu** of the Development Workbench for Universal Banking, click **Function Generation** node.

   The **Function Generation** screen displays.

Figure 1-1 Function Generation



2. On the **Function Generation** screen, specify the fields in the **Header** section.

   For more information on fields, refer to the field description table.

Table 1-1 Function Generation - Field Description

Field	Description
Action	Select **New** from the drop-down list.
Function Type	Select **Child** from the drop-down list.
Parent XML	On entering details for fields **Action** and **Function Type**, the **Parent XML** field is enabled. Click **BROWSE** and select the required parent function from the hard disk. On successful loading, the parent function will get updated with the parent function ID value. Parent function can be either a parent function ID or a child function ID.
Parent Function	The **Parent Function** field defaulted to the name of the function id that has loaded as the parent.
Function Category	The **Function Category** field defaulted with the category of the parent. It cannot be modified.

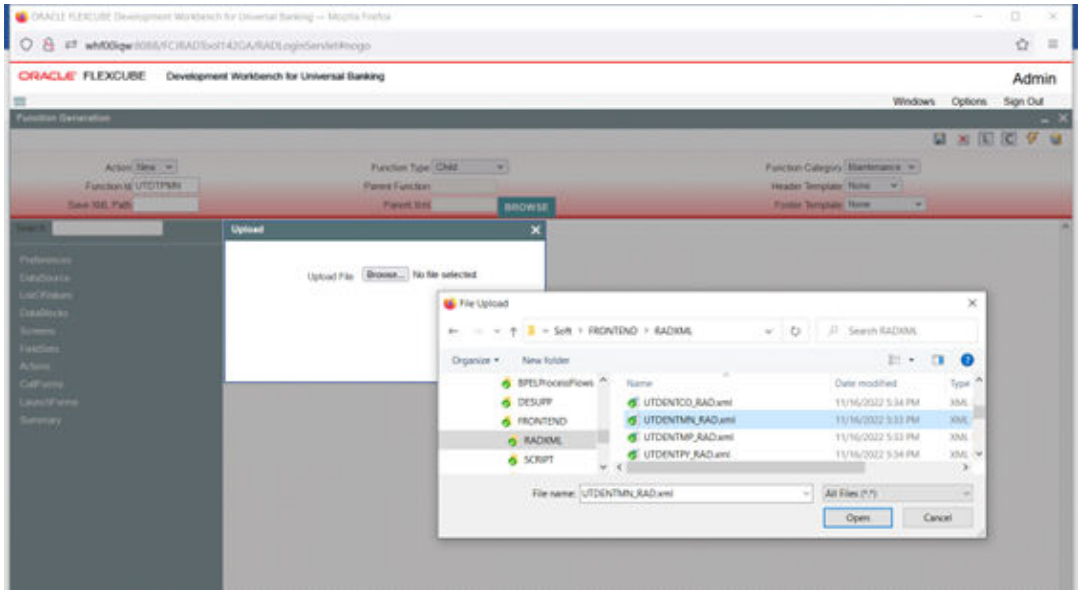
Table 1-1 (Cont.) Function Generation - Field Description

Field	Description
Function ID	Enter Child function ID in the **Function ID** field. Apply standard naming conventions.
Header Template	Select the header template from the drop-down list: • **None** • **Process**
Save XML Path	Specify the XML path.
Footer Template	Select the footer template from the drop-down list. • **None** • **Maint Audit** • **Maint Process** • **Process**

3. On the **Function Generation** screen, Select **Action** as**New** and **Function Type** as **Child**.

   The **Function Generation_Child** screen is displayed.

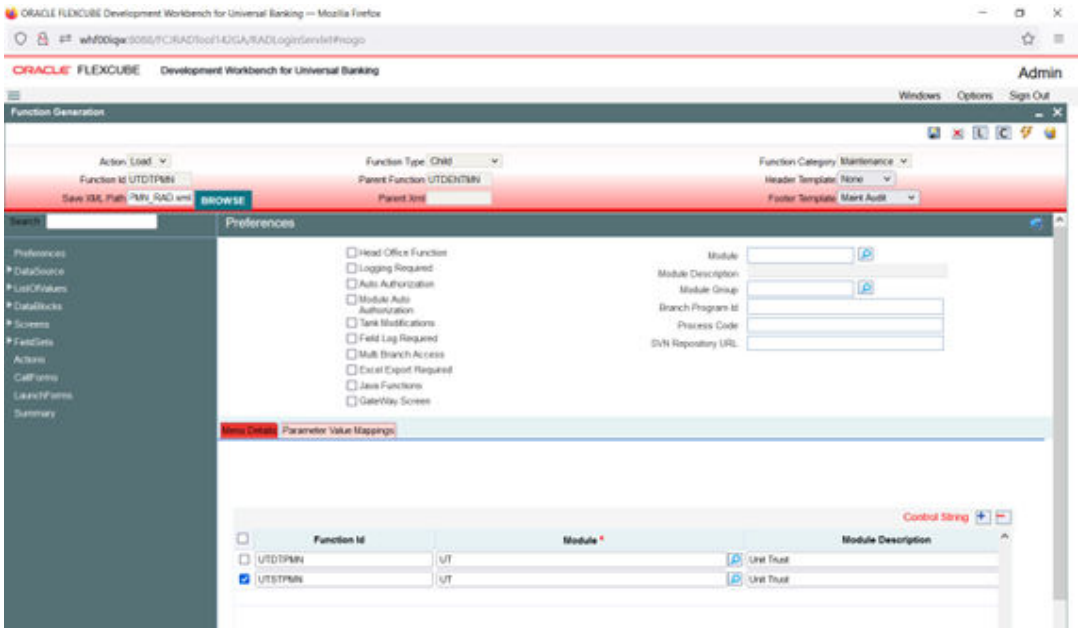
Figure 1-2 Function Generation_Child



4. Click **BROWSE** and select the required parent function.
5. On the **Function Generation** screen, click **Generate**.

   The **Preferences** screen displays.

Figure 1-3 Preferences node of Child screen



In the **Preferences** screen, the function ID name must be of the child screen, and not of the parent screen. On the tab out of the **Preferences** node, the function name gets changed automatically if not changed manually. Hence developer has to visit the **Preferences** node at least once before generating files for the child function ID.

1.1.2 Possible Operations in Child screen

This topic briefs the important considerations to be made when designing a **Child** screen.

Note the following while designing a **Child** screen:

- The developer will not be allowed to delete or rename elements created in the parent function.
- New data sources or adding columns to existing data sources are allowed.
- The addition of new LOVs and modification of existing LOVs are allowed.
- Modifying properties of existing block fields(of the parent) is allowed.
- New Screens, tabs, sections, etc can be added by the developer.
- Deletion and Renaming of screens, tabs, sections, etc are not allowed. Instead, the developer has the option to hide them.
- Addition or removal of fields from fieldset is allowed. Properties of fields can also be modified. Note that the block to which the fieldset is attached cannot be changed at the child level.
- New fieldsets can be defined.
- The developer can add a new Launch Form and Call Form in the child screen. The existing launch Form/Call form can be made inactive, if not required at the child level.
- Amendable fields can be modified.
- The summary screen can also be completely redesigned at the child level.

> **Note**
>
> If enhancement on the parent screen happens parallel to the child screen design developer can use the child refresh option to upgrade the latest changes in the parent screen to the child screen.

The Child screen design is similar to normal screen design. Refer to the *Development Workbench – Screen Development I* document for the detailed information.

1.1.3 Generation of Files for Child screen

This topic explains the process of generation of files for a Child screen.

Generation of Files

The process of generation of files is similar to that of normal function ID. All the units generated for a normal function ID will be generated for the child screen as well.

In the script for **SMTB_MENU**, column **PARENT_FUNCTION** would contain the name of the parent function ID.

1.1.4 Extensible Development for Child screen

This topic provides information on the Extensible Development of a Child screen.

Extensible Development

The developer can add code in hook packages and release specific JavaScript files. The extensible development process is similar to that followed in a maintenance or transaction screen. Refer to respective documents for a detailed explanation. The structure of the system packages will be the same as for a normal maintenance or transaction function ID.

For child screens, in addition to release specific hooks for each logical stage, the system also calls the corresponding hook from the parent function's programs. This allows the reuse of the common business logic among all the child functions.

For example, Assume **UTDENTMN** is a normal maintenance screen and **UTDTPMN** as the child of this screen. Below figure shows the snippet of code from `fn_default_and_validate` of the child main package.

Figure 1-4 Snippet from utpks_utdtpmn_main.sql

```
IF NOT utpks_utdtpmn_Main.Fn_Skip_Master THEN
   Pr_Convert_Child_To_Master(p_utdtpmn,l_UTDENTMN);
   Pr_Convert_Child_To_Master(p_Prev_utdtpmn,l_Prev_UTDENTMN);
   Pr_Convert_Child_To_Master(p_Wrk_utdtpmn,l_Wrk_UTDENTMN);
   Dbg('Calling utpks_utdentmn_Kernel.Fn_Pre_Default_And_Validate ');
   IF NOT utpks_utdentmn_Kernel.Fn_Pre_Default_And_Validate (p_Source,
      p_Source_Operation,
      p_Function_Id,
      p_Action_Code,
      p_function_Id  ,
      l_UTDENTMN,
      l_Prev_UTDENTMN,
      l_Wrk_UTDENTMN,
      p_Err_Code,
      p_Err_Params) THEN
      Dbg('Failed in utpks_utdentmn_Kernel.Fn_Pre_Default_And_Validate ');
      RETURN FALSE;
   END IF;
   Pr_Convert_Master_To_Child(l_Prev_UTDENTMN,p_Prev_utdtpmn);
   Pr_Convert_Master_To_Child(l_Wrk_UTDENTMN,p_Wrk_utdtpmn);
END IF;
```

Here u can find that the developer written defaulting and validation for the parent screen is being called before doing validations specific to child screen. Hence business logic for parent is reused. Developer has the option to skip doing parent validation for the child screens using skip handlers if needed.

Note the functions **Pr_convert_child_to_master** and **Pr_convert_master_to_child**. These procedures have to be used every time a call is made to the parent function id packages. This procedure prepares object types for the parent and child respectively.

1.2 Screen Child

This topic provides an overview on **Screen Child** creation.

Screen Child can be derived from a parent screen or a child screen. The **Screen Child** uses the packages for maintenance functions from the parent itself, only screen-level modifications can be done in a **Screen Child**.

It should be noted that **Screen Child** are not exactly the child of child screens (second level inheritance). It can be the child of a parent or a child screen. The difference between a **Child** and a**Screen Child** is that only screen level changes can be done in a **Screen Child**. Business logic will remain the same as its parent. In the child screen business logic can also be enhanced for the child function ID.

This topic contains the following sub-topics:

- Develop New Screen Child
  This topic provides the systematic instructions to develop a new **Screen Child**.
- Possible Operations in Screen Child
  This topic briefs the important considerations to be made when designing a **Screen Child** screen.
- Generation of Files for Screen Child
  This topic explains the process of generation of files for a Screen Child screen.

- Extensible Development for Child screen
  This topic provides information on the Extensible Development of a Child screen.

1.2.1 Develop New Screen Child

This topic provides the systematic instructions to develop a new **Screen Child**.

1. On **Expand Menu** of the Development Workbench for Universal Banking, click **Function Generation** node.

   The **Function Generation** screen displays.

Figure 1-5 Function Generation

2. On the **Function Generation** screen, specify the fields in the **Header** section.

   For more information on fields, refer to the field description table.

Table 1-2 Function Generation - Field Description

Field	Description
Action	Select **New** from the drop-down list.
Function Type	Select **Screen Child** from the drop-down list.
Parent XML	On entering details for fields **Action** and **Function Type**, the **Parent XML** field is enabled. Click **BROWSE** and select the required parent function from the hard disk. On successful loading, the parent function will get updated with the parent function ID value. Parent function can be either a parent function ID or a child function ID.
Parent Function	The **Parent Function** field defaulted to the name of the function ID that has loaded as the parent function.
Function Category	The **Function Category** field defaulted with the category of the parent. It cannot be modified.
Function ID	Enter **Screen Child** function ID in the **Function ID** field. Apply standard naming conventions.
Header Template	Select the header template from the drop-down list: • **None** • **Process**
Save XML Path	Specify the XML path.
Footer Template	Select the footer template from the drop-down list. • **None** • **Maint Audit** • **Maint Process** • **Process**

3. On the **Function Generation** screen, Select **Action** as**New** and **Function Type** as **Screen Child**.

   The **Function Generation_Screen Child** is displayed.

Figure 1-6 Function Generation_Screen Child

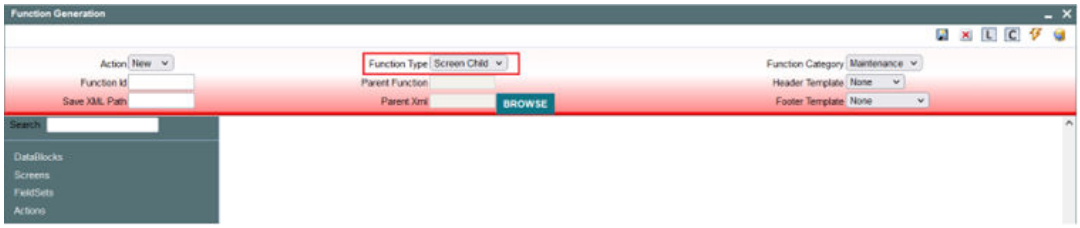


Figure 1-7 Function Type - Screen Child

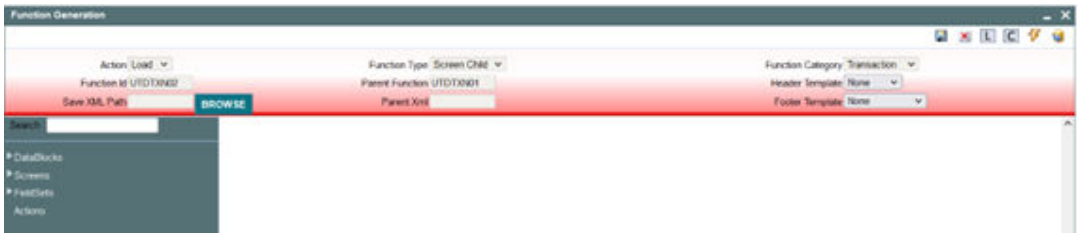
4. Click **BROWSE** and select the required parent function.
5. On the **Function Generation** screen, click **Generate**.

   The following 4 nodes of the parent can only be modified in **Screen Child** creation:

   - **Data Blocks**
   - **Screens**
   - **Field Sets**
   - **Actions**

   The Node options in Screen Child displays.

Figure 1-8 Node options in Screen Child



1.2.2 Possible Operations in Screen Child

This topic briefs the important considerations to be made when designing a **Screen Child** screen.

Below are the possible operations that can be done on a **Screen Child** in Oracle FLEXCUBE Universal Banking Development Workbench:

- Existing elements cannot be deleted.
- New blocks or block fields cannot be added.
- Only certain properties of the fields in the block can be modified.
- Properties of screens and fieldsets can be modified.
- New screens and fieldsets can be added.
- Tabs, sections, and partitions can be added to the existing screens.
- Web Service and operation can be configured for the screen child, but amendable fields cannot be modified.

Field-level properties that can be modified are as follow:

- **Field Label**
- **Field Size**
- **Maximum Decimals**
- **Default Value**
- **Preview Value**
- **Visible**
- **Read Only**
- **Calendar Text**
- **Popup Edit Required**
- **Uppercase Only**
- **Input by LOV only**
- **Not Required in XSD**

Screen level modification can be done according to the requirement. Most of the properties at the screen level can be modified. Fieldset level modification also can be done as per requirement. The block to which the field set is attached cannot be changed at **Screen Child** for existing fieldsets. However, all properties can be modified for new fieldsets.

If parent screen enhancements are happening parallel to **Screen Child** design, developer can use the **Screen Child** refresh option to upgrade the latest parent changes to the **Screen Child**.

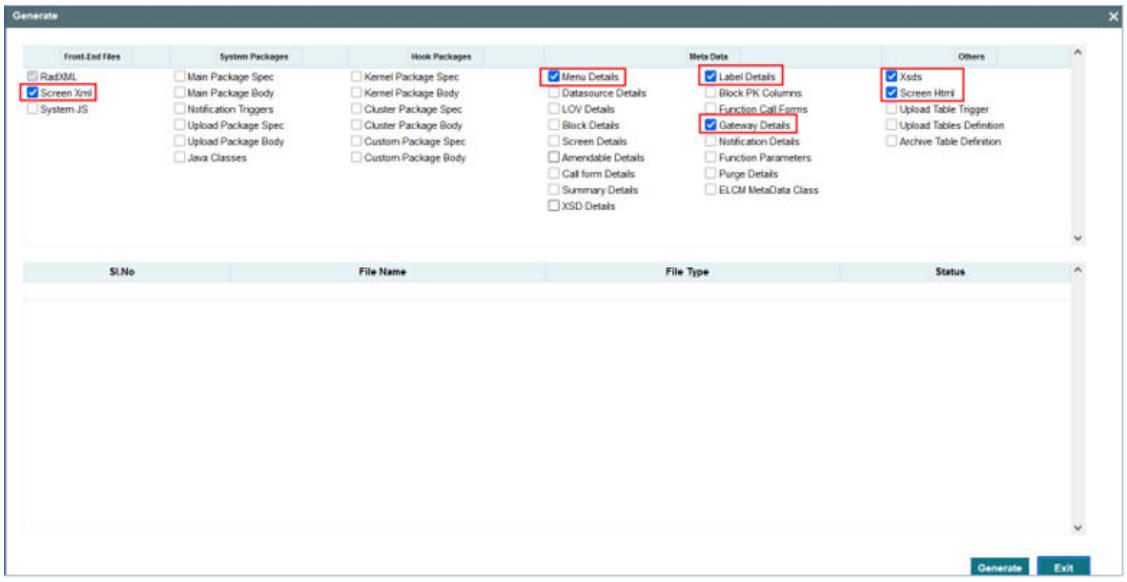
1.2.3 Generation of Files for Screen Child

This topic explains the process of generation of files for a Screen Child screen.

Generated Files

Only Screen XML (language-specific XML), Menu Details, Label Details, XML Schema Definitions, and Screen HTML will be generated from the Workbench for **Screen Child** Screen.

Figure 1-9 Generation of Files for Screen Child



1.2.4 Extensible Development for Child screen

This topic provides information on the Extensible Development of a Child screen.

Extensible Development

The screen child uses the packages and JavaScript files of the parent itself.