

Oracle® Banking Digital Experience

UX Plugins Guide



Release 25.1.1.0.0
G43925-01
October 2025

ORACLE®

Copyright © 2015, 2025, Oracle and/or its affiliates.

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.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. 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 fail-safe, 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.

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

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Purpose	i
Before you Begin	i
Pre-requisites	i
Audience	i
Documentation Accessibility	ii
Critical Patches	ii
Diversity and Inclusion	ii
Related Resources	ii
Conventions	ii
Screenshot Disclaimer	iii
Acronyms and Abbreviations	iii
Post-requisites	iii

1 The UX Plugin

2 Plugin Setup

3 List of Components

4 Creating Business Components

4.1	When to Create a Business Component	1
4.2	How to Create Business Components	1
4.3	Handling Device - Specific Variants	1
4.4	Handling Other Variants	2
4.5	Exporting Business Components	3

5 Supported Variables in Figma

6	List of Templates	
6.1	Transaction Template	1
6.2	Widgets	4
6.3	Overview	6
7	Auto Layout to Flexbox Mapping	
7.1	Direction (flex-direction)	2
7.2	Wrap (flex-wrap)	2
7.3	Align Items (align-items)	3
7.4	Justify Content (justify-content)	4
7.5	Gap (gap, row-gap, column-gap)	6
7.6	Padding (padding)	6
7.7	Flex Sizing (Child Behavior)	7
8	Create New Design	
9	Modify Component	
9.1	Add Grid	1
9.2	Add Conditions	4
9.2.1	Add Logic	11
10	Export to Toolkit	
10.1	Steps to Export Design to Toolkit	1
	Next Steps	5
11	FAQ	

Preface

- [Purpose](#)
- [Before you Begin](#)
- [Pre-requisites](#)
- [Audience](#)
- [Documentation Accessibility](#)
- [Critical Patches](#)
- [Diversity and Inclusion](#)
- [Related Resources](#)
- [Conventions](#)
- [Screenshot Disclaimer](#)
- [Acronyms and Abbreviations](#)
- [Post-requisites](#)

Purpose

This guide is designed to help acquaint you with the Oracle Banking Digital Experience application. This guide provides answers to specific features and procedures that the user need to be aware of the module to function successfully.

Before you Begin

Kindly refer to our **Getting Started User Guide** for common elements, including Symbols and Icons, Conventions Definitions, and so forth.

Pre-requisites

Specify **User ID** and **Password**, and login to **Home** screen.

Audience

This document is intended for the following audience:

- Customers
- Partners

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 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.

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.

Related Resources

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

- Oracle Banking Digital Experience Installation Manuals
- Oracle Banking Digital Experience Licensing Manuals

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
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; actual screens that appear in the application may vary based on selected browser, theme, and mobile devices.

Acronyms and Abbreviations

The list of the acronyms and abbreviations used in this guide are as follows:

Table 1 Acronyms and Abbreviations

Abbreviation	Description
OBDX	Oracle Banking Digital Experience

Post-requisites

After finishing all the requirements, please log out from the **Home** screen.

1

The UX Plugin

This topic provides information on **the UX Plugin**.

The UX plugin is a Figma-based tool designed to facilitate the seamless conversion of Figma designs into Oracle JET-compliant code. This plugin helps ensure consistency between design and development by eliminating discrepancies between the visual design and the final product.

By automating this translation process, the tool significantly reduces the time and effort required to develop user interfaces, accelerating the overall UI implementation.

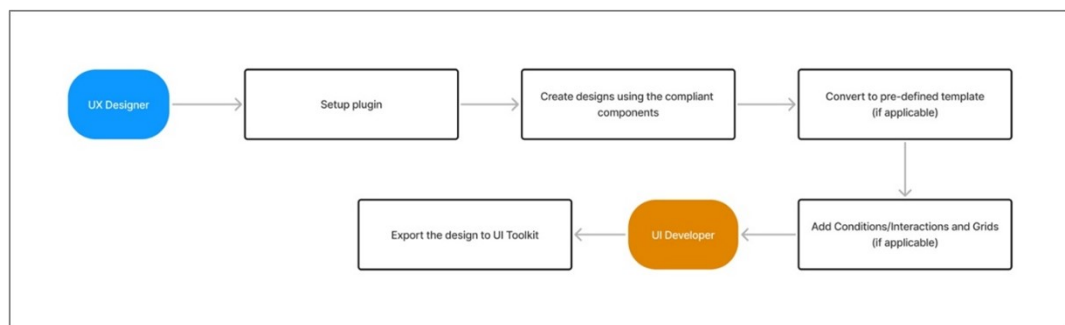
Pre-requisites:

- Basic understanding of Figma
- Basic understanding of Oracle's Redwood Design System
- Basic understanding of Oracle JET
- Basic understanding of the UI Toolkit
- An active Figma license
- UI Toolkit installed and running on the system (latest version)

Workflow

The process to create a screen and export the design in the UI toolkit is as follows:

Figure 1-1 Workflow



Step 1: Setup Plugin

The plugin can be easily setup on your system by following the instructions outlined in the **Plugin Setup** section. Once done you can run the plugin and start with your design journey.

Step 2: Create designs using compliant components

Designers can either drag and drop the compliant components to create their designs on Figma or can explore the **Create New** feature of the plugin to get started.

Step 3: Convert to pre-defined template

In case if the designer wants to structure their designs into a pre-defined template, they can do so using the **Convert to Template**.

Step 4: Add Conditions/Interactions

In case the designer wants to add an **On Click** or **On Change** functionality on components they can do so by either using the Figma's prototyping feature or using the **Add Conditions** feature available in the **Modify Component** option.

Step 5: Export to UI Toolkit

At this stage the UX Designer hands over the design to the UI Developer, who will then take over the procedure to export this design into the toolkit. The UI developer will have to ensure the toolkit is up and running in the background before starting the export procedure. Detailed explanation of the export procedure is mentioned in the **Export to Toolkit** section.

Note

The workflow outlined above provides an overview of the plugin usage.

A detailed explanation of each feature is provided in the sections below.

Usage Guidelines

- Use components exclusively from the Oracle JET or Redwood Library
- Ensure that the UI toolkit is successfully installed with the latest version
- Ensure that the Figma components are not detached as those components may not be recognized by the plugin.
- Create business components only if they will be reused multiple times across your screens. Avoid creating excessive business components to maintain efficiency and manageability.
- Ensure that the UI toolkit is up and running in the background while exporting to avoid any issues.
- For optimal plugin functioning, keep the original layer names of components unchanged, as modifying them may result in recognition errors.

2

Plugin Setup

This topic provides information on **Plugin Setup**.

1. Install the latest version of **Node.js** supported for your device specifications, before downloading the plugin zip file.

Note

This is a one-time step, need not to be repeated.

2. Download the zip file and extract it in the desired destination folder.
3. Upon extraction, there will be two folders:
 - a. **Backend:**
 - i. Open the command terminal and navigate to the Backend Folder.
 - ii. In the terminal, enter the following command:

```
npm install
```

(For Mac Users, append 'sudo' before the command)

- iii. Now, enter the following command in the same terminal:

```
npm run build
```

- b. **UI:**

- i. Open the command terminal and navigate to the UI Folder.
- ii. In the terminal, enter the following command:

```
npm install
```

(For Mac Users, append 'sudo' before the command)

- iii. Now, enter the following command in the same terminal:

```
npm link '../Backend/'
```

Note

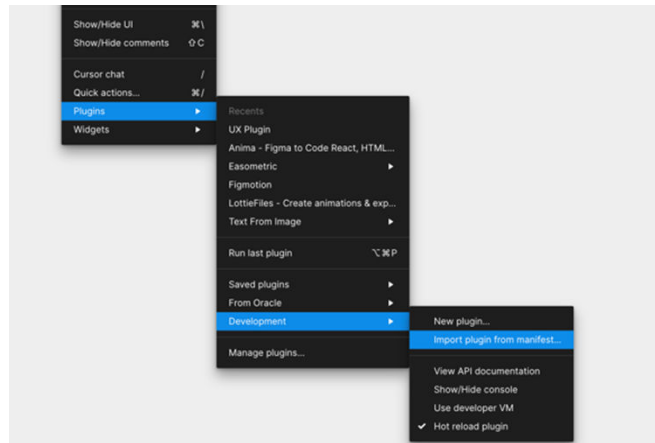
The '../Backend/' is the path to the Backend folder.

- iv. Enter the following command to build the UI of the plugin:

```
ojet build
```

4. Move to Figma, right click on any page, Click **Plugin**, then click **Development** , and then click **Import plugin from manifest**.

Figure 2-1 Import plugin from manifest



5. Navigate to the UI folder, and click on the **manifest.json** file inside the UI folder, that needs to be imported.

The plugin is successfully imported inside Figma and is ready to use.

3

List of Components

This topic provides information on **List of Components**.

The plugin supports a range of components that streamline the design and development process. Components from the following component libraries are supported by the UX Plugin:

- Redwood Design System (RDS) - Oracle's flagship design system offering consistent UI patterns and components, enabling seamless plugin integration
- Oracle JET - JavaScript Extension Toolkit offering responsive templates and interactive components, optimized for data-intensive enterprise applications.
- OBDX Components Library - Oracle Banking Digital Experience (OBDX) components tailored for financial applications, featuring specialized UI elements for banking interfaces

List of RDS supported components:

1. Avatar
2. Badge
3. Button
4. Checkbox set
5. File Picker
6. Input Date
7. Input Number
8. Input Password
9. Input Select Single (supports Figma interaction in case of static dropdown values)
10. Input Text
11. Input Text Area
12. Meter Bar
13. Radio Button
14. Rating Gauge
15. Slider
16. Switch
17. Tab Bar
18. Table
19. Toggle Button

List of Oracle JET supported components:

1. **Action Card (JET Version)**
An Action Card is an actionable container rendering related information. An action card renders children in a styled rectangular area and has support for an action event when clicked.

It has a single property of instance swap that allows the designers to place their content in the body slot of the card.

[Link to JET Component](#)

2. Charts

A chart displays information graphically, making relationships among the data easier to understand.

Plugin supports the following chart types:

- Bar Chart
- Line Chart
- Pie Chart

Following are the properties of this component:

- Chart Type: Bar, Line, Pie
- Legend Position: Bottom, Top, Start, End
- Show Legend: Boolean

3. Collapsible (JET Version)

A collapsible displays a header that can be expanded to show its content.

Following are the properties of this component:

- Variant – Basic, Horizontal Rule
- Header – Instance Swap
- Body – Instance Swap

[Link to JET Component](#)

4. Column Share

The Column Share component is designed to allocate and display data in a column-based format. It enables the division of content into proportional sections for better layout and organization.

Following are the properties of this component:

- Columns – 2,3
- Field Size – 25-75, Equal, 40-60, 75-25, 60-40

Note

To create more than 3 column layout the designer can combine two form layouts and achieve the desired column structure.

5. Conveyor Belt

A conveyor belt manages overflow for its child elements and allows scrolling among them.

Following are the properties of this component:

- Orientation – Horizontal, Vertical
- Arrow Visibility – Auto, Visible, Horizontal
- Item Template – Instance Swap

[Link to JET Component](#)

6. Drawer (JET Version)

A Drawer Popup is a panel that slides into the viewport. It can be placed at the start, end or **bottom** edge and it always overlays the page.

Following are the properties of this component:

- Edge – Start, End
- Body – Instance Swap
- Header – Boolean
- Header Text – Text Input
- Sub Header – Boolean
- Sub Header Text – Text Input
- Hide Close Icon – Boolean
- Show Footer – Boolean

Link to JET Component

7. Link

The Link component is used to create clickable text elements that navigate users to external URLs, internal pages, or trigger specific actions within the application.

Following are the properties of this component:

- State – Default, Primary, Secondary, Disabled
- Type – Standalone, Embedded
- Label – Text Input

Note

Use Figma's native prototyping feature for links that trigger components within the same page. For links that redirect to separate pages or external locations, handle them using the UI Toolkit.

8. List Item Layout (JET Version)

A List Item Layout represents layout used for list view item elements. To create a list view combine all list items and add them in a frame.

Following are the properties of this component:

- Leading – Boolean & Instance Swap
- Default – Instance Swap
- Secondary – Boolean & Instance Swap
- Tertiary – Boolean & Instance Swap
- Overline – Boolean & Instance Swap
- Metadata – Boolean & Instance Swap
- Trailing – Boolean & Instance Swap
- Action – Boolean & Instance Swap
- Bottom Slot – Boolean
 - Quaternary – Boolean & Instance Swap
 - Navigation – Boolean & Instance Swap

Note

Designers are recommended to use this component instead of RDS Component as this has been re-created specifically for the purpose of making it more compatible with the plugin.

[Link to JET Component](#)

9. List View (JET Version)

A list view displays data items as a list or a grid with highly interactive features.

Following are the properties of this component:

- Gridlines – Hidden, All, In Between
- Drilldown – True, False
- Item Template – Instance Swap

Note

Designers are recommended to use this component instead of RDS Component as this has been re-created specifically for the purpose of making it more compatible with the plugin.

10. Modal Dialog (JET Version)

A dialog displays a popup window that provides information and gathers input from the application user. Modal dialogs require interaction before control can be returned to the outer window.

Following are the properties of this component:

- Hide Header – Boolean
- Body – Instance Swap
- Close Button – Boolean
- Footer – 3 buttons, 2 buttons, 1 button

[Link to JET Component](#)

11. Progress Bar

- a. A progress bar allows the user to visualize the progression of an extended computer operation.

Following are the properties of this component:

- Value: 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%

[Link to JET Component](#)

12. Progress Circle

A progress circle allows the user to visualize the progression of an extended computer operation.

Following are the properties of this component:

- Value: 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%

[Link to JET Component](#)

13. Spark Chart

A spark chart displays information graphically, typically highlighting the trend of a data set in a compact form factor.

Following are the properties of this component:

- Status – Neutral, Success, Critical
- Direction: Up, Down, No Change
- Type – Line, Line with Area

[Link to JET Component](#)

14. Table

A table displays data items in a tabular format with highly interactive features.

Following are the properties of this component:

- Number of Columns: From 2, up to 12
- Text Only: True, False

[Link to JET Component](#)

1. Account Input

This is a custom input field created specifically to select an account from a list of accounts fetched from a pre-configured API. It has two display modes: dropdown view and card view.

Following are the properties of this component:

- Display: Dropdown, Card
- Label: Text Input
- Read Only: Boolean
- Hide Label: Boolean
- Value: Boolean

2. Address

The Address component is designed for entering address information within a form. It includes two elements: address input and address selection.

Following are the properties of this component:

- Read Only – Boolean
- Type – Saved Address, Branch Near Me, Custom
- Address Enabled – Boolean

3. Amount Input

The Amount Input component is specifically designed for entering monetary values in a form. It supports input fields for currency formatting and precise value entry.

Following are the properties of this component:

- Label Hidden – Boolean
- Read Only – Boolean
- Currency List – Boolean
- Mandatory – Boolean
- Label – Text Input
- View Limits – Boolean

4. Icon

This component was created specifically to identify the correct icon from the Redwood icon repository. It is recommended that designers use this component when incorporating icons independently.

It has only one property of instance swap for selecting the correct icon.

5. Internal Account Input

The Internal Account Input component is designed for entering account numbers in a format commonly used in the banking domain. It consists of two input fields: a primary field for the account number and a secondary field for confirming the account number.

Following are the properties of this component:

- Required – Boolean
- Mask – Boolean
- State – Default, Filled
- Read Only – Boolean
- Confirm Style Account – Boolean (for the secondary account number field)
- Confirm Label – Text Input
- Label – Text Input

6. Page Section

The **Page Section** component creates distinct sections on a page, allowing for clear organization and layout. It is especially useful for structuring forms, such as those needed for transaction screens.

Following are the properties of this component:

- Page Heading – Boolean
- Header Template – Boolean (for hiding the controls present on the header)

7. Phone Input

Phone Input allows users to input mobile number along with a dropdown for country code

Following are the properties of this component:

- Read Only – Boolean
- Without Label – Boolean
- Required – Boolean

8. Quick Action Button

The **Quick Action Button** provides a shortcut to specific tasks or actions related to modules or transactions, streamlining user interaction. It consists of an avatar, primary text and a custom slot.

Following are the properties of this component:

- Primary Text – Text Input
- Custom Slot – Boolean & Instance Swap
- Avatar – Dedicated Component Properties (defined by Redwood)

9. Row

The **Row** component displays read-only data in a label-value format, allowing designers to customize how the information is presented.

Following are the properties of this component:

- Type – Primary, Secondary, Disabled, Success, Warning, Danger

- Size – 2XS, XS, SM, MD, LG, XL
- Label – Text Input

4

Creating Business Components

This topic provides information on **Creating Business Components**.

Business components are custom-built components created to meet specific module or design needs. Use them when you want to build something reusable across your designs.

- [When to Create a Business Component](#)
This topic provides information on **When to Create a Business Component**
- [How to Create Business Components](#)
This topic provides information on **How to Create Business Components**.
- [Handling Device - Specific Variants](#)
This topic provides information on **Handling Device - Specific Variants**.
- [Handling Other Variants](#)
This topic provides information on **Handling Other Variants**.
- [Exporting Business Components](#)
This topic provides information on **Exporting Business Components**.

4.1 When to Create a Business Component

This topic provides information on **When to Create a Business Component**

Create a business component only if:

- It will be **used** multiple times in your design
- It is a reusable unit of UI

4.2 How to Create Business Components

This topic provides information on **How to Create Business Components**.

- Always build business components using components from the supported libraries.
- The plugin reads Boolean and Text properties directly from Figma and configures them automatically in the toolkit.

4.3 Handling Device - Specific Variants

This topic provides information on **Handling Device - Specific Variants**.

To support different devices (e.g., Desktop and Mobile):

1. Add a variant **property** named **Device**
2. Add variant **values**: **Desktop** and **Mobile**

Note

The property name and value are case-sensitive and users are advised to adhere to the mentioned standards.

4.4 Handling Other Variants

This topic provides information on **Handling Other Variants**.

For other types of variants:

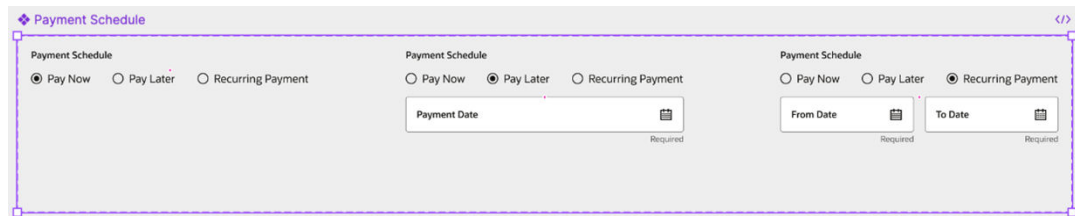
1. Create a master component named `masterComponent`
2. Add all possible variant combinations as individual components inside it
3. Use conditional UI logic to control the visibility or behaviour of each variant based on the selected property

For example:

Let's say you want to create a **Payment Schedule component** with the following variants:

1. **Pay Now** – No additional fields
2. **Pay Later** – Includes a single date input
3. **Recurring Payment** – Includes a date range (From and To date inputs)

Figure 4-1 Payment Schedule component



To make this work with the plugin:

- Create a **fourth variant** called '`masterComponent`'
- Inside `masterComponent`, include **all the UI elements** from all three variants (i.e., the date input and both date range fields)
- Apply **conditional logic** to show or hide specific elements based on the selected variant (e.g., only show the date range when '`Recurring Payment`' is selected)

This setup ensures the plugin can map and configure all possible states of the component correctly.

Note

The property name and value are case-sensitive and users are advised to adhere to the mentioned standards

Tip: Make sure the variant property (e.g., Schedule Type) is clearly named and consistent across all variants.

Here's the detailed procedure on how to apply conditional logic: **Add Conditions**.

4.5 Exporting Business Components

This topic provides information on **Exporting Business Components**.

The procedure to export a business component is the **same** as exporting an '**Individual Page**'. The user just needs to ensure that they check the **Re-usable Component** checkbox.

5

Supported Variables in Figma

This topic provides information on **Supported Variables in Figma** .

The plugin supports predefined variables from the Oracle JET and OBDX framework. These variables are essential for maintaining consistency across the application and are defined in the OBDX Component Library Figma file. Users must use variables from this library for the plugin to recognize and process them correctly.

Types of Supported Variables

The plugin supports the following types of variables:

- Colours
- Spacing (including padding and gap)

How Variables Work

The plugin reads only the variable name, not its actual value. This ensures that the application theme, set by the admin, determines the final appearance.

For example:

- If a text element is assigned a colour variable brand-120, which defaults to deep lilac, the plugin will recognize brand-120 but not its colour value. The application's theme will define the actual colour displayed.
- Similarly, for spacing and padding, the plugin identifies the variable names without interpreting their numerical values.

By following this approach, the plugin ensures design consistency while allowing flexibility through theme customization.

6

List of Templates

This topic provides information on **List of Templates**.

UI Templates are pre-designed layouts that provide a structured framework for creating user interfaces. They help streamline the design process by offering standardized, reusable components and styles, ensuring consistency and efficiency in the development of applications.

Predefined templates of OBDX Application that are supported by the plugin include:

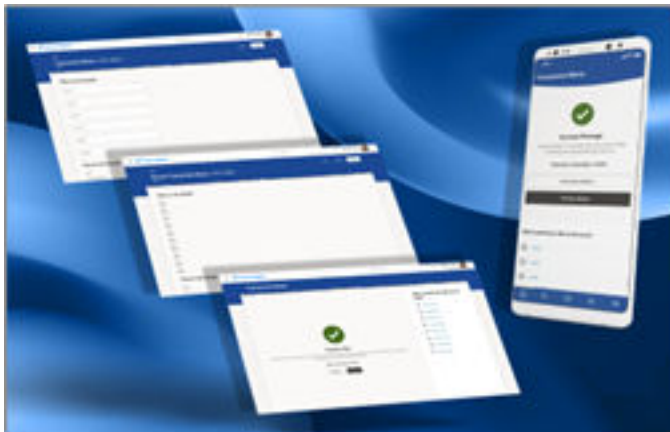
- Transaction Template
- Widgets
- Overview Template
- [Transaction Template](#)
This topic describes the systematic instruction to **Transaction Template** option.
- [Widgets](#)
This topic provides information on **Widgets**.
- [Overview](#)

6.1 Transaction Template

This topic describes the systematic instruction to **Transaction Template** option.

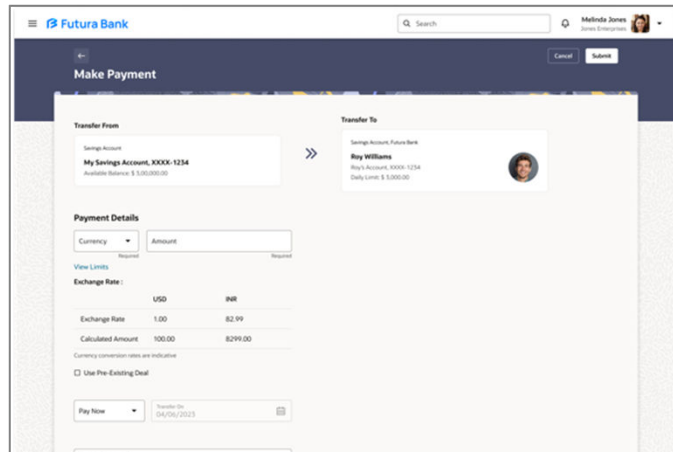
A streamlined three-step process designed for accuracy and ease of use, consisting of an initiation screen, a review screen, and a confirmation screen.

Figure 6-1 Transaction Template



Initiation Screen

The starting point of the process, where users input or select initial information to begin the workflow.

Figure 6-2 Initiation Screen

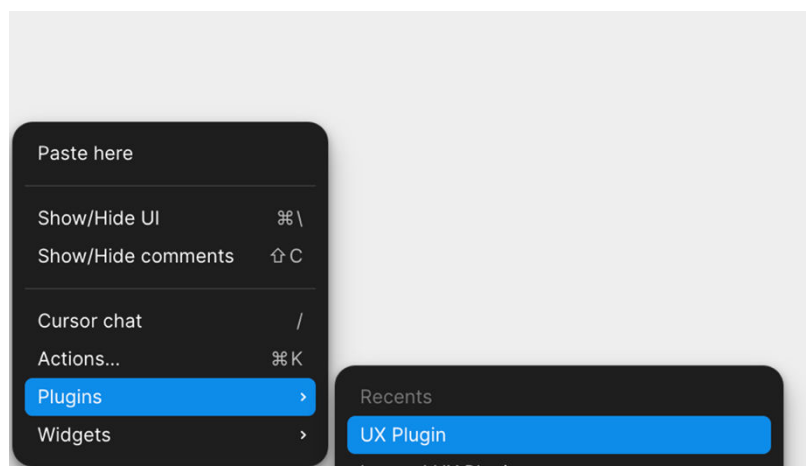
Following are the properties of the initiation page:

- Title – Text Input
- Context Switcher – Boolean
- Show Subtitle – Boolean
- Action Slot – Boolean
 - Secondary Button Type – Default, Menu Button
 - Primary Button – Boolean
 - Secondary Button – Boolean
 - Tertiary Button – Boolean
- Footer Tabs – Boolean

How to create a transaction experience using the UX Plugin:

1. Step 1: Launch the Plugin

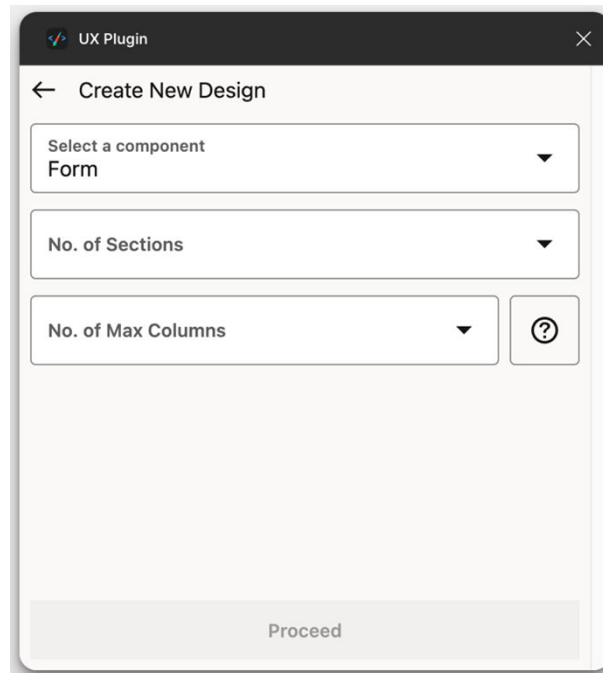
Right-click anywhere on the Figma design board and navigate to: Plugins → UX Plugin

Figure 6-3 Launch the Plugin

2. Step 2: Generate a Boilerplate Form

Use the Create New Design feature in the plugin to create a base form layout.

Figure 6-4 Generate a Boilerplate Form



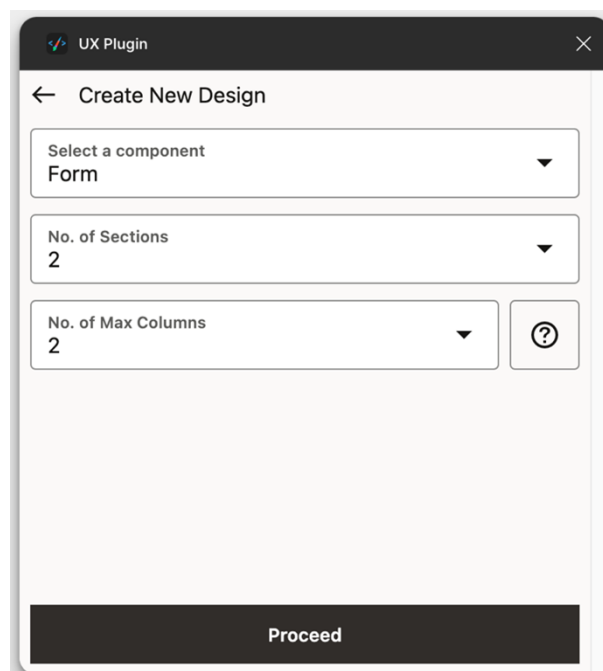
The screenshot shows the 'UX Plugin' window with the 'Create New Design' tab selected. It contains three dropdown menus: 'Select a component' (set to 'Form'), 'No. of Sections' (empty), and 'No. of Max Columns' (empty). A help icon (?) is next to the 'No. of Max Columns' dropdown. A 'Proceed' button is at the bottom.

3. Step 3: Specify Layout Details

Input the number of sections and the Max Columns.

Max Columns refers to the maximum number of columns displayed on the largest screen size (e.g., desktop layout).

Figure 6-5 Specify Layout Details



The screenshot shows the 'UX Plugin' window with the 'Create New Design' tab selected. The 'Select a component' dropdown is set to 'Form'. The 'No. of Sections' dropdown is set to '2'. The 'No. of Max Columns' dropdown is set to '2'. A help icon (?) is next to the 'No. of Max Columns' dropdown. A 'Proceed' button is at the bottom.

4. Step 4: Customize the Form

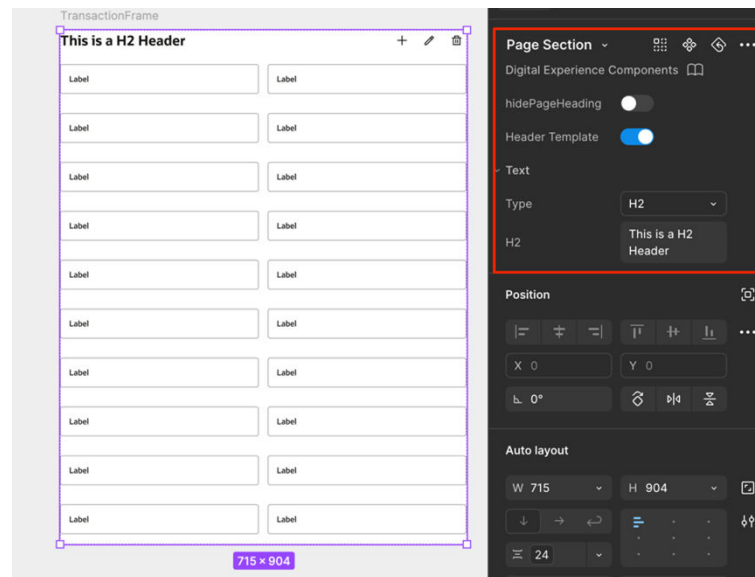
The plugin will generate a boilerplate form with random input fields.

Replace these fields with the ones needed for your use case.

You can also make additional modifications such as:

- Hiding the page header
- Toggling header template actions
- Adjusting layout as per design guidelines

Figure 6-6 Customize the Form



Next Step

Once the form is ready, proceed to convert it into a transaction template using the process outlined in the **Convert to Template – Transaction** section.

6.2 Widgets

This topic provides information on **Widgets**.

The dashboard page enables users to create and display widgets, offering graphical representations of information for quick insights and a personalized experience.

Figure 6-7 Widgets

Following are the properties of widgets:

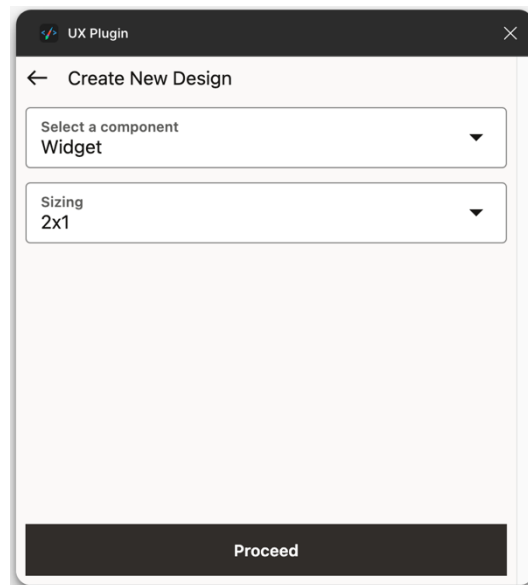
- Full Body – Boolean (a property that hides the header and footer)
- Size: 8 pre-defined sizes supported as per the Masonry Layout of Oracle JET
- Body – Instance Swap
- Body Slot – Instance Swap (applicable on from **Custom** Body)
- Heading – Boolean & Text Input
- Description – Boolean & Text Input
- Header Slot – Boolean & Instance Swap (action slot)
- Custom Description Slot – Boolean & Instance Swap
- Footer – Boolean & Instance Swap

How to create widgets using the UX Plugin:

1. Step 1: Create the Widget

Use the plugin's Create New feature to generate a pre-built widget. You can select the widget size during this step.

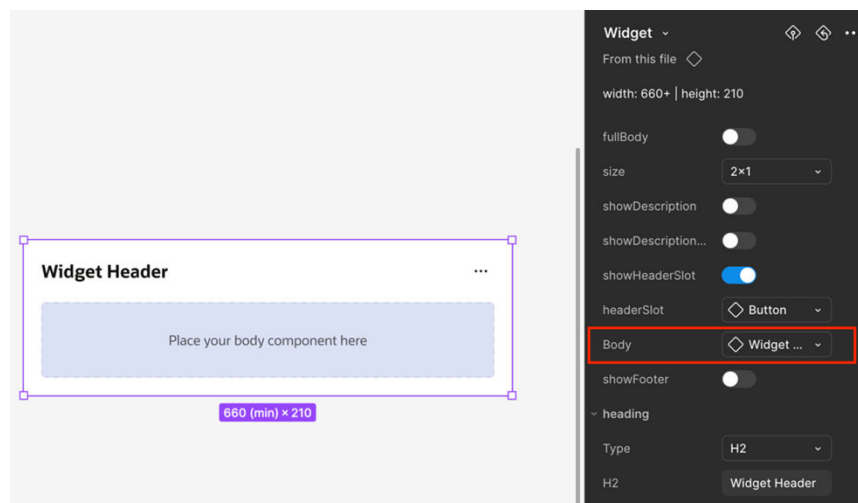
Figure 6-8 Create the Widget



2. Step 2: Design the Body

Design the widget's body, convert it into a component, and place it into the Body Slot using the Instance Swap property.

Figure 6-9 Create the Widget



3. Step 3: Configure Properties

Give your widget a name and set the remaining properties as needed.

Next Step

Once your widget design is ready, user can export the widget – [Link to Export Widget procedure](#)

6.3 Overview

The Overview Template is used to display detailed information about a specific **product** or **listing**. It is always paired with a relevant product or listing context and is typically navigated to from a **Listing Page**.

Figure 6-10 Overview Template

Structure

The template is divided into two main sections:

1. **Side Panel**
A compact section that displays key summary details of the product or listing.
2. **Overview Body**
A larger, scrollable area that presents in-depth and dynamic content such as:
 - Tables
 - Graphs
 - Stats
 - Other associated details

Listing Page

This page typically includes a **Tabular, Grid View or List View** component. It fetches and displays data based on the attached **business service**.

Figure 6-11 Listing Page

Properties of the Overview Template:

This template includes properties of **'Listing Page'** and **'Overview Page'**

Following are the properties of this component:

- Empty State (Listing Page): Boolean
- Listing Page
 - Listing Menu: Boolean
 - Show Bottom Footer: Boolean
 - Listing Body: Instance Swap
- Listing Header
 - Title: Text Input
 - Show Subtitle: Boolean
 - Action Slot: Boolean
- Listing Body: Instance Swap
- Overview Page
 - Overview Menu: Boolean
 - Show Bottom Footer: Boolean
 - Side Panel: Instance Swap
- Overview Header
 - Title: Text Input
 - Context Switcher: Boolean
 - Show Subtitle: Boolean
 - Action Slot: Boolean
- In App Navigation (Bottom Footer): Nested Instance Properties as provided by Redwood Design

How to design an overview template using the UX Plugin:

1. Step 1: Design the Main Components

Create the **Overview Body** and **Side Panel** components in Figma.

These form the two primary sections of your overview layout—the body for detailed content and the side panel for key summary info.

2. Step 2: Export as Business Components

Export both components individually as **Business Components**.

This ensures they can be reused and referenced when building the final template.

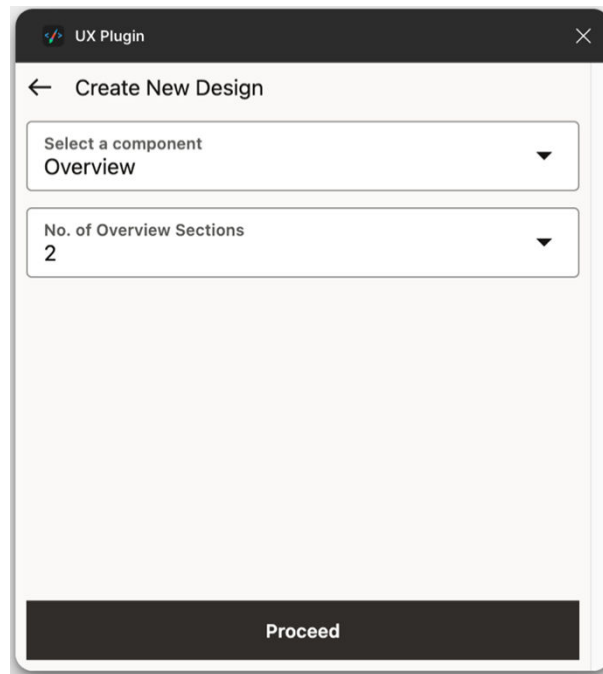
Note

The procedure to export business components is the same as exporting individual pages. Link to [Export to Toolkit – Individual Page](#).

3. Step 3: Generate Overview Layout

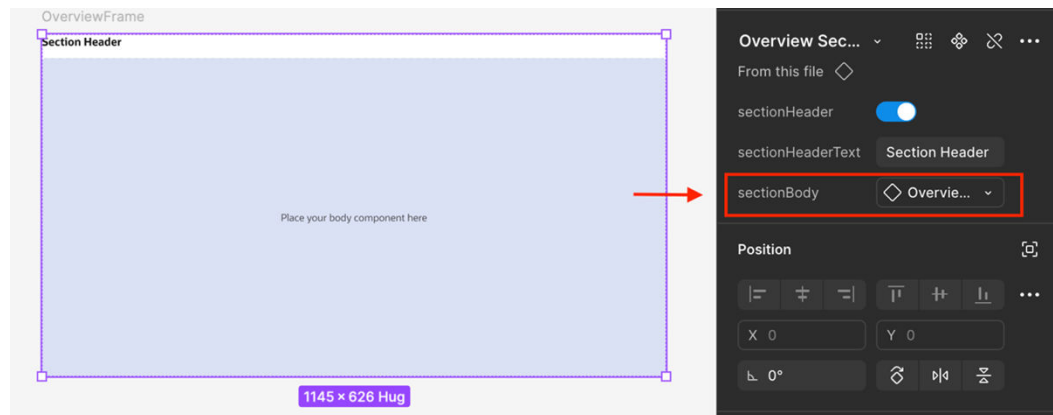
Run the plugin and use the **Create New** feature to create a pre-structured overview layout.

Specify how many overview sections you need.

Figure 6-12 Generate Overview Layout**4. Step 4: Replace Placeholder Content**

After the layout is generated, replace the dummy overview body with your actual **Overview Body** component.

This lets you retain the plugin's structure while using your real design.

Figure 6-13 Replace Placeholder Content**Next Step**

After this, convert the design into Overview Component – Link to the procedure of Convert to Template – Overview.

7

Auto Layout to Flexbox Mapping

This topic provides information on **Auto Layout to Flexbox Mapping**

The core principle:

An Auto Layout frame in Figma directly maps to a Flex container (display: flex) in CSS. The layout properties you set on the Auto Layout frame and its children determine how the plugin translates them into Flexbox properties.

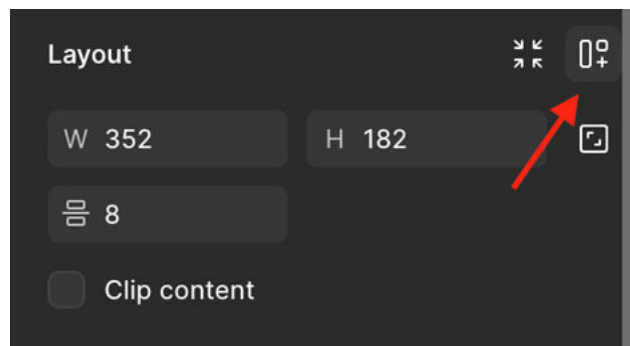
Prerequisites

To use these features, the parent frame must have Auto Layout enabled.

How to Enable Auto Layout:

- Select the frame and click the **Add Auto Layout button** next to **Layout** in the right-hand panel

Figure 7-1 Layout



OR

Use the shortcut Shift + A (after selecting the required frame)

Note

This document only explains how the plugin maps Figma's Auto Layout properties to their corresponding CSS Flexbox properties. It does not cover other Figma layout features or general CSS flex concepts outside this mapping context.

- [Direction \(flex-direction\)](#)
This topic provides information on **Direction (flex-direction)**.
- [Wrap \(flex-wrap\)](#)
This topic provides information on **Wrap (flex-wrap)**.
- [Align Items \(align-items\)](#)
This topic provides information on **Align Items (align-items)**.

- [Justify Content \(justify-content\)](#)
This topic provides information on **Justify Content (justify-content)**.
- [Gap \(gap, row-gap, column-gap\)](#)
This topic provides information on **Gap (gap, row-gap, column-gap)**.
- [Padding \(padding\)](#)
This topic provides information on **Padding (padding)**.
- [Flex Sizing \(Child Behavior\)](#)
This topic provides information on **Flex Sizing (Child Behavior)**.

7.1 Direction (flex-direction)

This topic provides information on **Direction (flex-direction)**.

CSS Equivalent:

flex-direction: row; (horizontal) or flex-direction: column; (vertical)

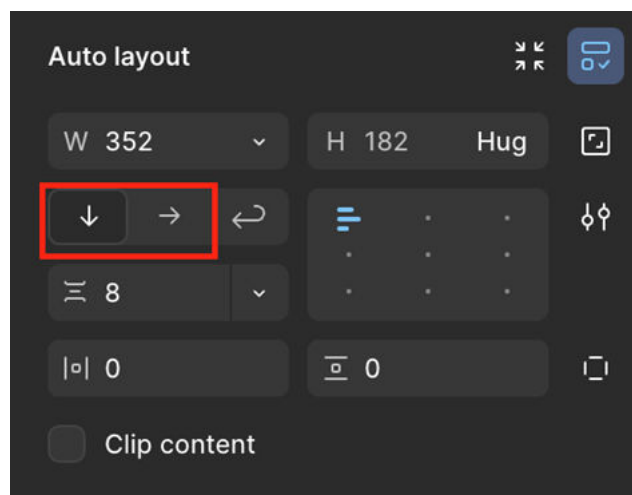
Figma Concept:

Controls the primary axis along which items are arranged in the Auto Layout frame.

How to Set in Figma:

1. Select the Auto Layout frame
2. In the Auto Layout section:
 - a. Click → for flex-direction: row.
 - b. Click ↓ for flex-direction: column.

Figure 7-2 Direction (flex-direction)



7.2 Wrap (flex-wrap)

This topic provides information on **Wrap (flex-wrap)**.

CSS Equivalent:

- flex-wrap: wrap; – Items wrap to the next line
- flex-wrap: nowrap; – Items stay on a single line (default)

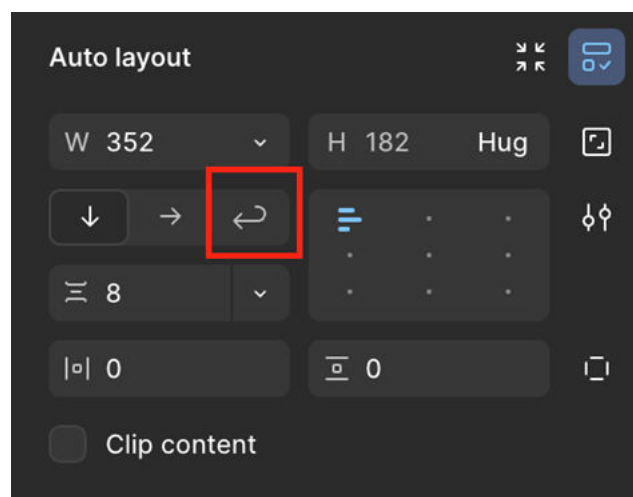
Figma Concept:

Figma allows similar behaviour using the “Wrap” toggle in Auto Layout.

How to Set in Figma:

1. Select the Auto Layout frame
2. Enable the Wrap toggle in the Auto Layout panel
3. Items will wrap to the next line when space runs out

Figure 7-3 Wrap (flex-wrap)



7.3 Align Items (align-items)

This topic provides information on **Align Items (align-items)**.

Controls alignment along the cross axis

CSS Equivalent:

align-items: flex-start | center | flex-end | stretch;

Figma Concept:

Aligns children perpendicular to the Auto Layout direction.

How to Set in Figma:

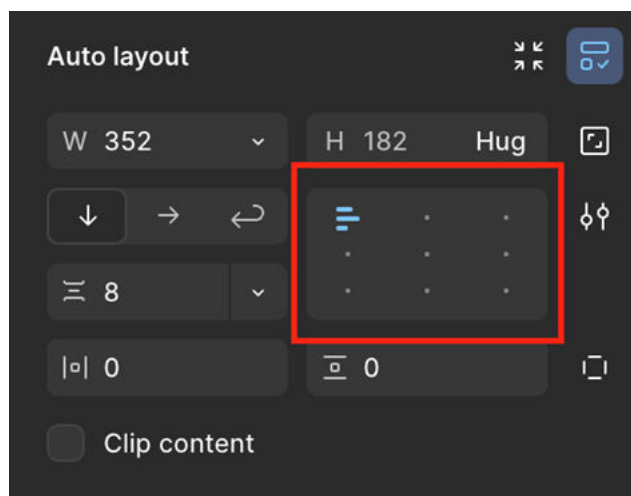
1. Select the Auto Layout frame
2. Use the Alignment Grid (3x3 dot grid) in the design panel
3. Mapping:
 - a. For Horizontal (→) direction:
 - i. Top → flex-start

- ii. Center → center
- iii. Bottom → flex-end
- b. For Vertical (↓) direction:
 - i. Left → flex-start
 - ii. Center → center
 - iii. Right → flex-end

Note

Stretch behaviour – If children are set to **Fill** container, they behave like **align-items: stretch**

Figure 7-4 Align Items (align-items)



7.4 Justify Content (justify-content)

This topic provides information on **Justify Content (justify-content)**.

Controls distribution along the main axis

CSS Equivalent:

justify-content: flex-start | center | flex-end | space-between | space-around;

Figma Concept:

Managed via the Spacing Mode in Auto Layout.

How to Set in Figma:

1. Select the Auto Layout frame
2. Open the Spacing Mode dropdown
3. Options:

- a. Packed → flex-start
- b. **Space Between** → space-between
- c. **Center** → center

Note

on space-around:

Figma doesn't directly support space-around, but the plugin maps it if:

The container's padding equals **twice the gap between items**

Figure 7-5 Justify Content (justify-content)- (reference for space-between)

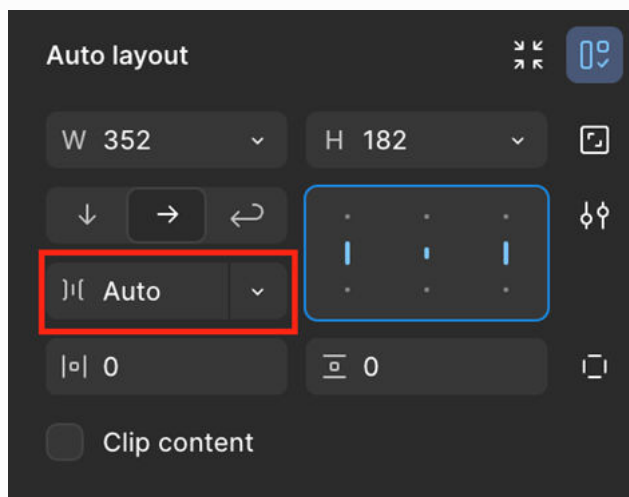
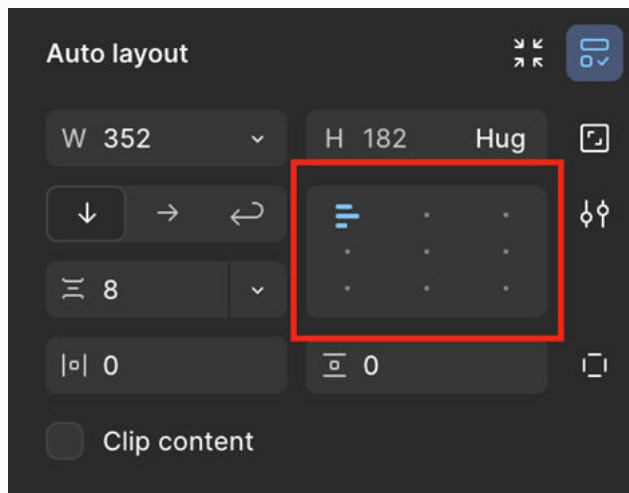


Figure 7-6 Justify Content (justify-content)- (reference for flex-start)



7.5 Gap (gap, row-gap, column-gap)

This topic provides information on **Gap (gap, row-gap, column-gap)**.

CSS Equivalent:

gap, row-gap, column-gap – Spacing between items

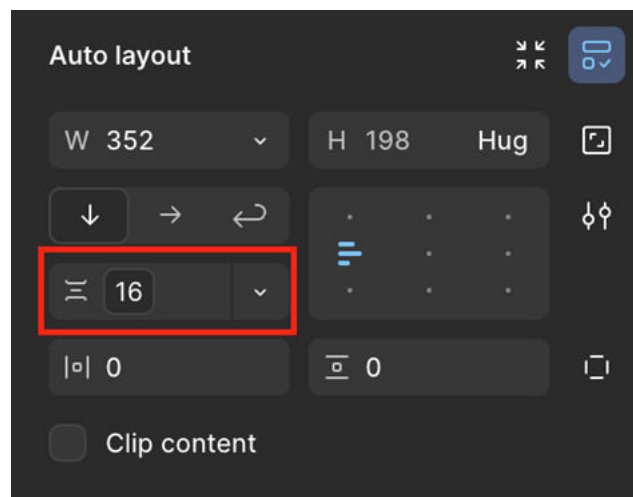
Figma Concept:

Set via the Spacing Between Items input in Auto Layout.

How to Set in Figma:

1. Select the Auto Layout frame
2. Enter a value in **Spacing Between Items**
 - a. For → (Row): Maps to column-gap
 - b. For ↓ (Column): Maps to row-gap
3. Some versions of Figma may show separate controls for horizontal and vertical gaps

Figure 7-7 Gap (gap, row-gap, column-gap)



7.6 Padding (padding)

This topic provides information on **Padding (padding)**

CSS Equivalent:

padding, padding-top, padding-right, etc.

Figma Concept:

Space inside the container pushing children inward.

How to Set in Figma:

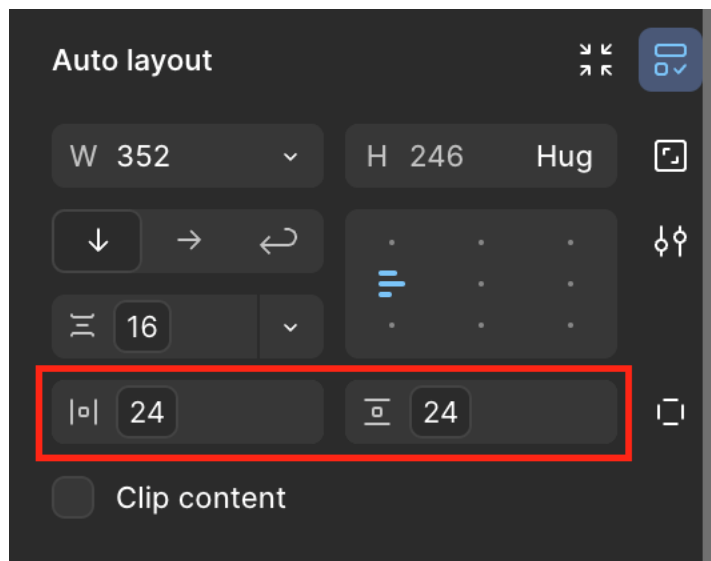
1. Select the Auto Layout frame

2. Use Padding Around Items in the Auto Layout section
3. Options:
 - a. Uniform padding → One value (e.g. padding: 16px;)
 - b. Vertical & Horizontal padding → Use “Alignment & padding” icon
 - c. Individual sides → Use “Independent paddings” icon for full control

Note

Please ensure that you **use padding variable** instead of explicit value.

Figure 7-8 Padding (padding)



7.7 Flex Sizing (Child Behavior)

This topic provides information on **Flex Sizing (Child Behavior)**.

CSS Equivalent:

flex: initial, flex: 1, align-self, width, height

Figma Concept:

Set via the Resizing settings on child elements inside an Auto Layout frame.

1. Hug Contents → flex: initial

CSS Equivalent:

flex: 0 1 auto – No grow, size to content

How to Set in Figma:

1. Select the child element

2. Identify parent's main axis:
 - a. Horizontal (→) = Main Axis: Horizontal
 - b. Vertical (↓) = Main Axis: Vertical
3. Set the child's resizing for main axis to Hug Contents

Fill Container → flex: 1 (Grow)

CSS Equivalent:

flex: 1 1 0% or width: 100% depending on axis

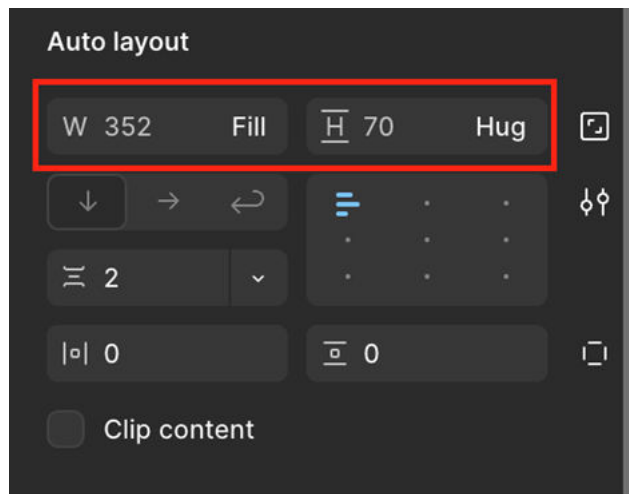
2. How to Set in Figma:

1. Main Axis Grow (e.g., make item fill remaining space):
 - Set resizing to Fill Container along the main axis
2. Cross Axis Stretch (e.g., full width/height):
 - a. Set resizing to Fill Container along the cross axis
 - b. Requires parent's align-items to allow stretching

Result:

The child fills available space. Perfect for flexible layouts and shared columns.

Figure 7-9 Flex Sizing (Child Behavior)



8

Create New Design

This topic provides information on **Create New Design**.

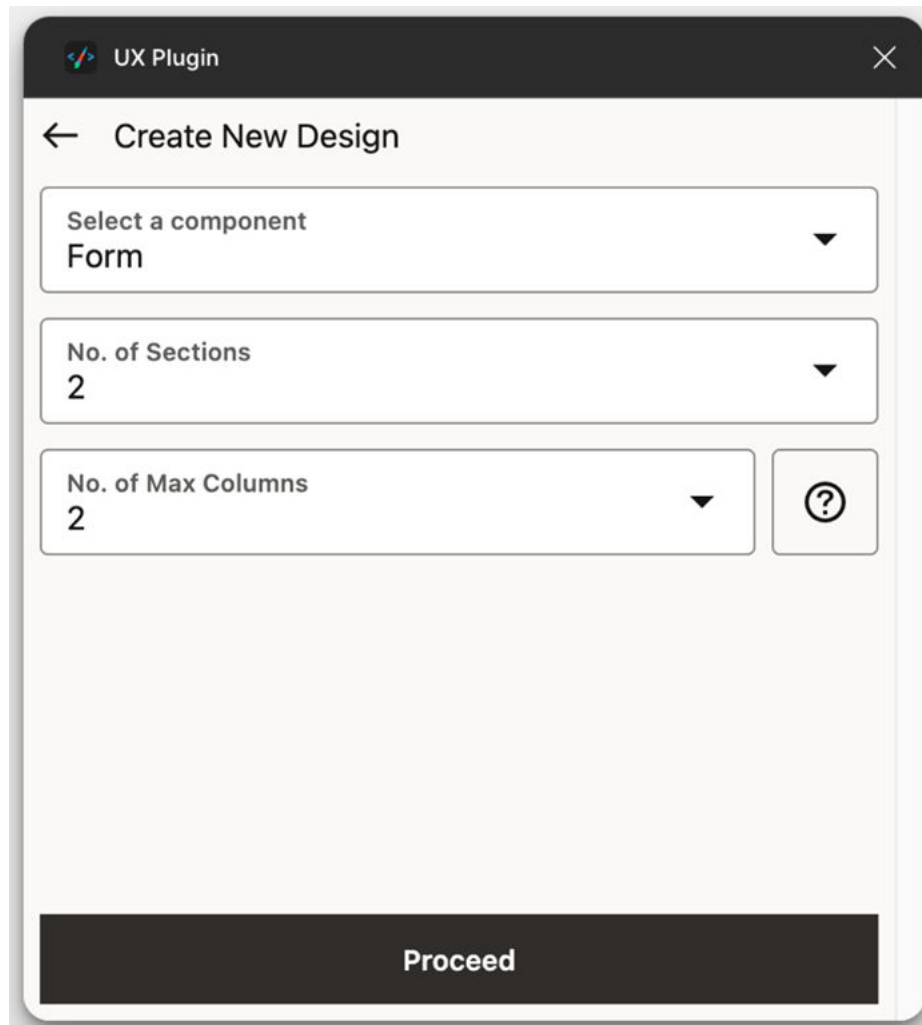
This feature allows for the automatic creation of design components in Figma based on predefined specifications. It streamlines the early stages of the design process by providing standardized elements, enabling designers to efficiently begin their work with consistent and compliant components.

The Create New Design feature supports:

Forms

- Auto-generates a boilerplate form with random input fields.
- Fields can be replaced to suit your design requirements.
- Users can:
 - Specify the number of form sections.
 - Define the maximum number of columns, which refers to how many columns appear on large screens (e.g., desktops).

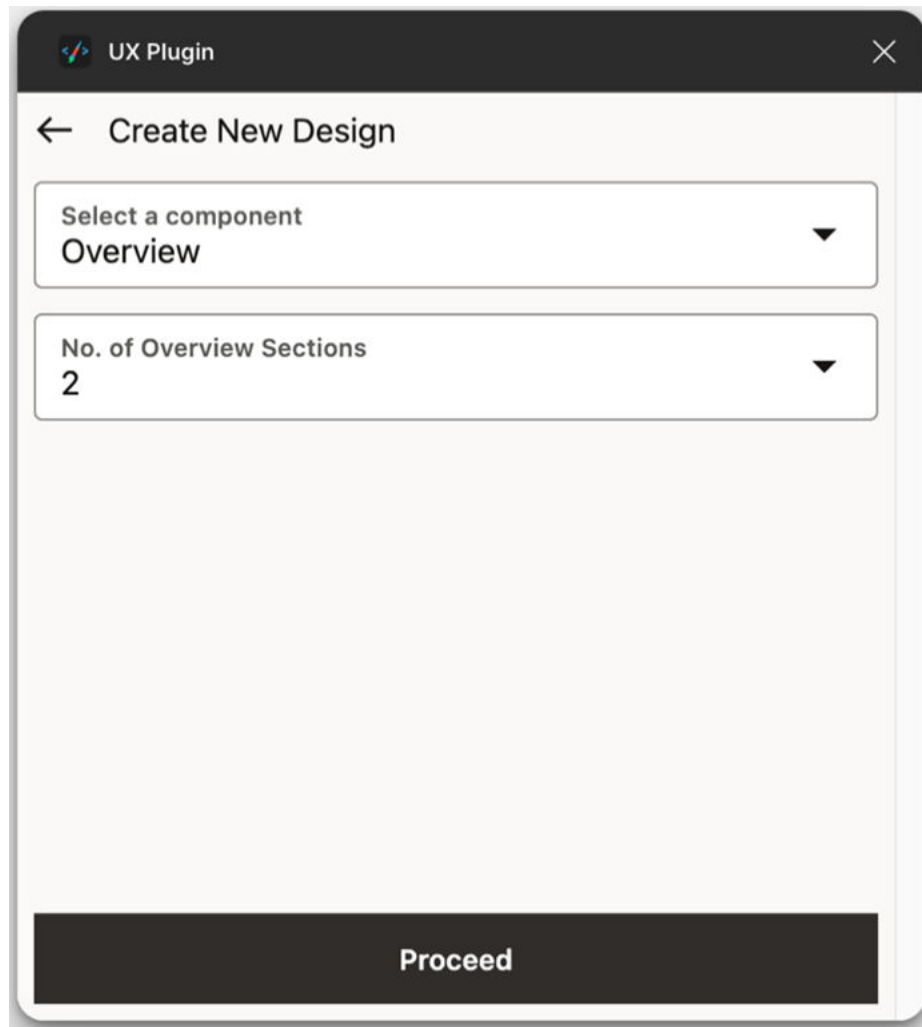
Figure 8-1 Forms



The image shows a 'UX Plugin' window titled 'Create New Design'. It contains three dropdown menus for configuration: 'Select a component' (set to 'Form'), 'No. of Sections' (set to '2'), and 'No. of Max Columns' (set to '2'). A help icon (?) is located next to the 'No. of Max Columns' dropdown. A 'Proceed' button is at the bottom.

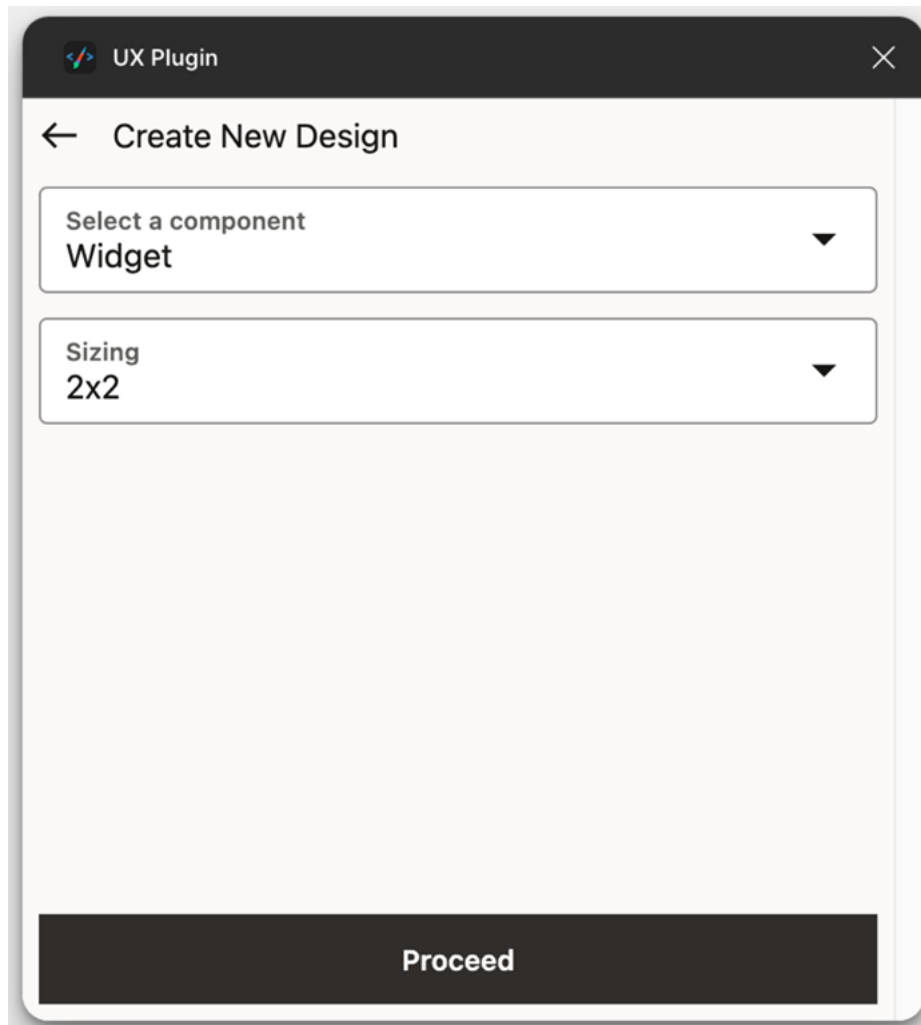
Overview Sections (Body)

- Generates dummy overview sections used in the body of an overview template.
- Users can define the number of overview sections needed.
- The dummy section can be replaced with a user-designed business component using the instance swap property.

Figure 8-2 Overview Sections (Body)**Widgets**

- Creates a pre-built widget in the specified size.
- Includes a placeholder body that can be swapped with a user-designed component using the instance swap property.

Figure 8-3 Widgets



9

Modify Component

This topic provides information on **Modify Component**.

This feature enables designers to refine and enhance existing components, simplifying the process of adjusting components to meet specific design and interaction requirements efficiently.

This feature has two capabilities:

- Add Grid
- Add Conditions
- [Add Grid](#)
This topic describes the systematic instruction to **Add Grid** option.
- [Add Conditions](#)
This topic describes the systematic instruction to **Add Conditions** option.

9.1 Add Grid

This topic describes the systematic instruction to **Add Grid** option.

This feature allows the designers to ensure the correct grid structure will be implemented to the designed components when the screens are exported into the UI toolkit.

For example:

If designers want to ensure the grid specification of the below design is translated identically,

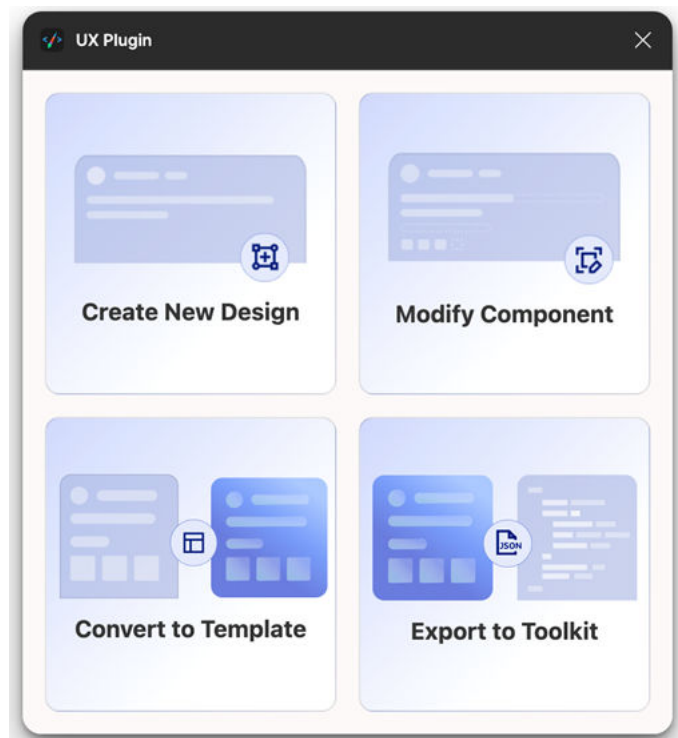
Figure 9-1 Grid Specification



they should:

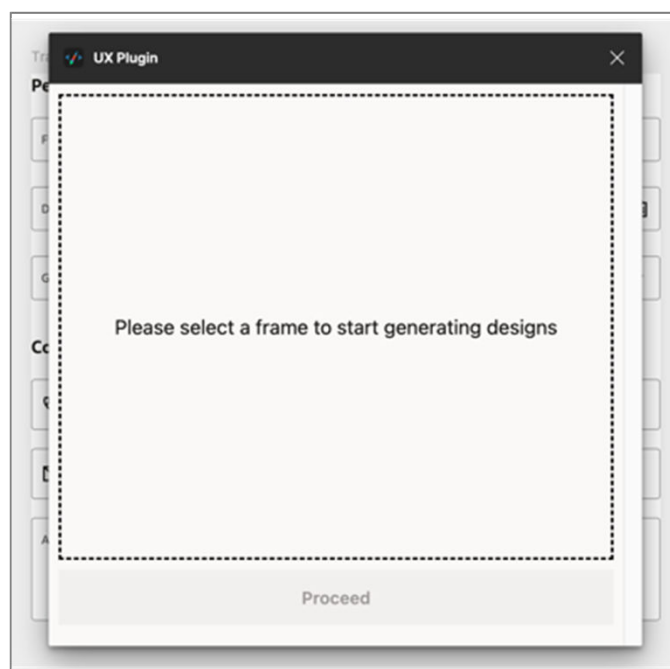
1. Run the plugin and choose the **Modify Component** option.

Figure 9-2 Modify Component



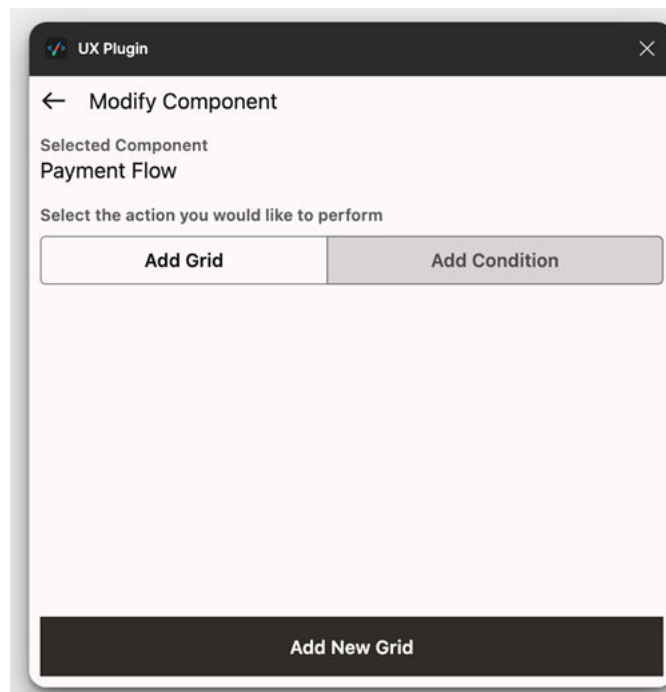
2. Select the frame on which the grid specifications need to be added.

Figure 9-3 Frame



3. After selecting the frame, select the **Add Grid** option and then click on the **Add New Grid** button located at the bottom of the window.

Figure 9-4 Add New Grid



On the **Add New Grid** page, the plugin will automatically detect and list the child components of selected frame.

4. Specify the grid size, its alignment and click **Save** to save the specifications.

Figure 9-5 Adding Grid Specifications



The new grid gets created.

9.2 Add Conditions

This topic describes the systematic instruction to **Add Conditions** option.

This capability allows designers to define complex interaction handling for supported components. It establishes a relationship between two components:

- **Variable component:** The dependent component that changes based on input from the controlling component.
- **Controlling component:** The component that determines what is displayed or how the variable component behaves.

This feature simplifies managing interactions within a particular component ensuring dynamic design behavior.

This capability has two use cases:

- a. Show/Hide of variable component based on the explicit value, state or selection of the controlling component.
- b. Show/Hide of variable component based on the defined properties of the controlling component.

Note

- For cases where there are multiple variable components controlled by a single controlling component, place all the variable components in a single frame.
- For cases where the interaction leads an external component, please use Figma's native prototyping feature.

Case 1: Show/Hide of variable component based on the explicit value, state or selection of the controlling component.

For example:

To hide and show the relevant fields on the click of the below radio button:

Figure 9-6 Variable Component

Investment Period
Investment Period

☒ Specify Tenure ☐ Specify End Date

Years Months

Required Required

Investment Period
Investment Period

☐ Specify Tenure ☒ Specify End Date

End Date

Required

1. Ensure that all the components are present in one single frame.

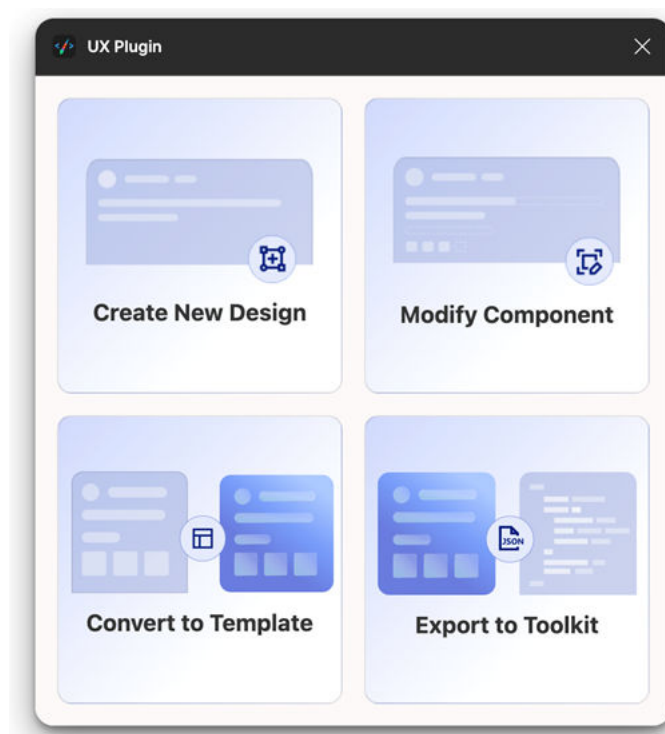
Figure 9-7 Variable Component

Investment Period
Investment Period
☒ Specify Tenure ☐ Specify End Date
Years Months
Required Required

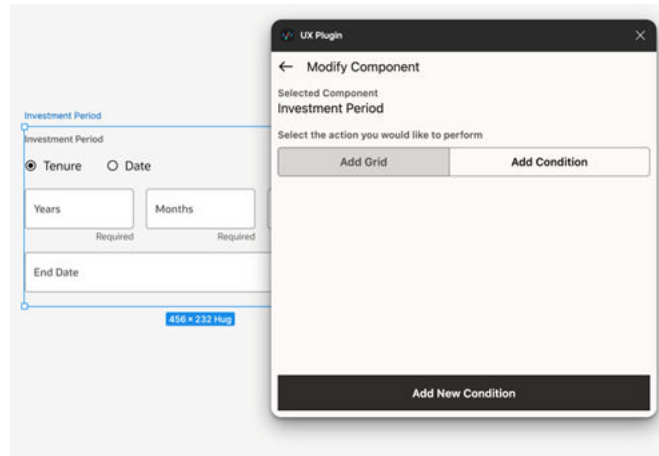
Investment Period
Investment Period
☐ Specify Tenure ☒ Specify End Date
End Date
Required

2. Select the designed frame, run the plugin, and select the **Modify Component** option.

Figure 9-8 Modify Component

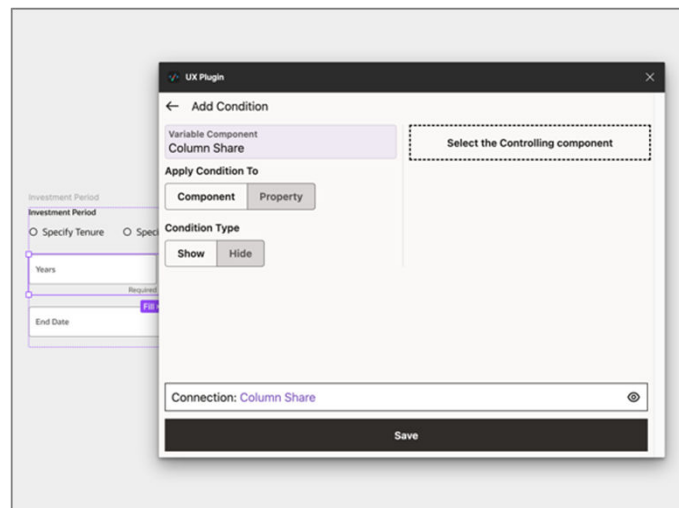


3. Select the variable component (in this case, the Input fields), then choose the **Add Condition** option. Next, click on the **Add New Condition** button located at the bottom of the window.

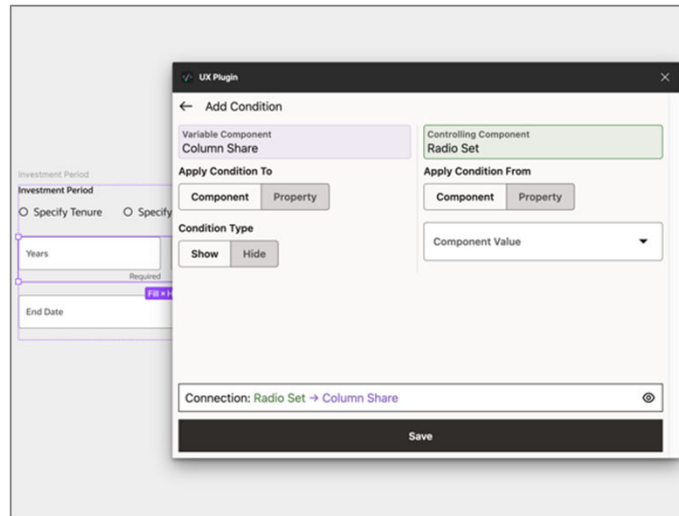
Figure 9-9 Add Condition

The next page appears.

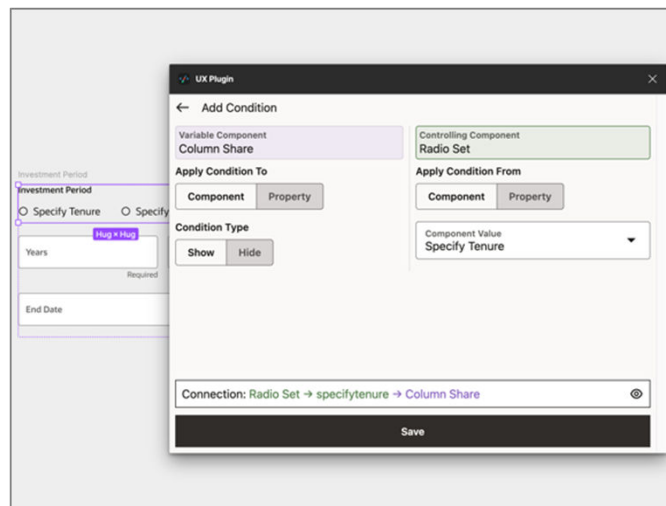
4. Choose which aspect of the variable component the controlling component will affect.
 - The entire variable component
 - A property of the variable component
 - In this case, show the entire variable component.

Figure 9-10 Controlling Component

5. Tap on the button **Select the Controlling Component** and choose the controlling component **Radio Set**.

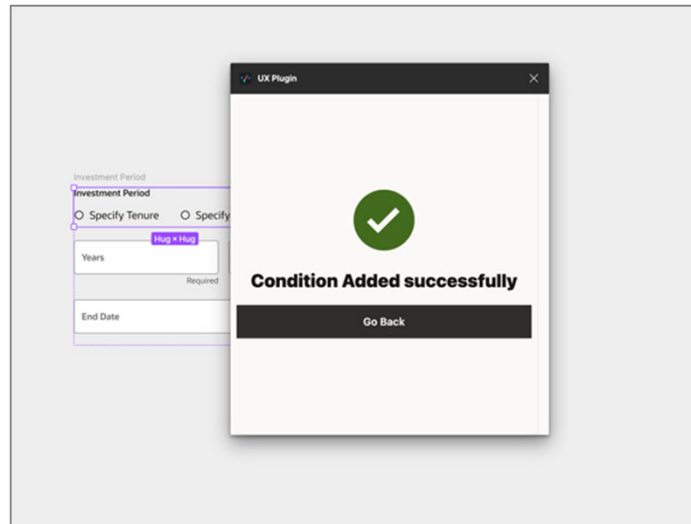
Figure 9-11 Controlling Component

6. Select the controlling radio button **Specify Tenure** from the **Component Value** drop-down to control the visibility of the input fields on tap of the **Specify Tenure** radio button.

Figure 9-12 Controlling Component

7. Review the condition through the connection flow located at the bottom of the window and click **Save** to save the condition.

The success message of condition added appears.

Figure 9-13 Success message

8. Repeat the same procedure for the other radio button **Specify Duration** and ensure that the connection reads:

Figure 9-14 Connection

Connection: Radio Set → specifyenddate → Input - Date (Text)

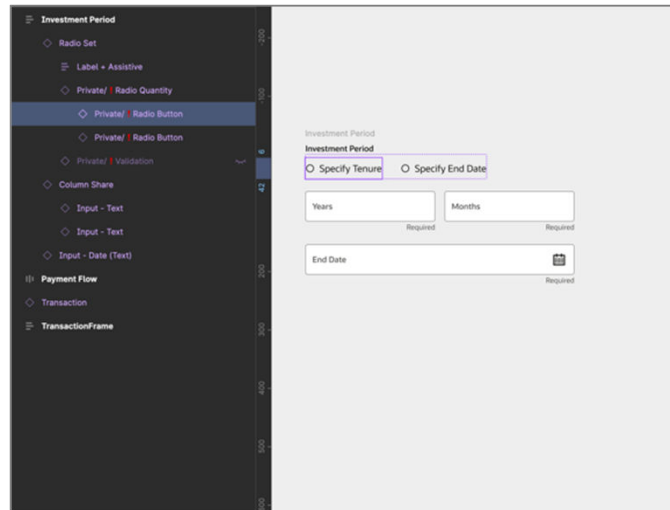
Case 2: Show/Hide of variable component based on the defined properties of the controlling component.

9. Repeat the steps 1-4 of the Case 1.
The variable component is set.
10. In order to control the selected property of the **Specify Tenure** radio button, the radio button's layer must be selected.

Note

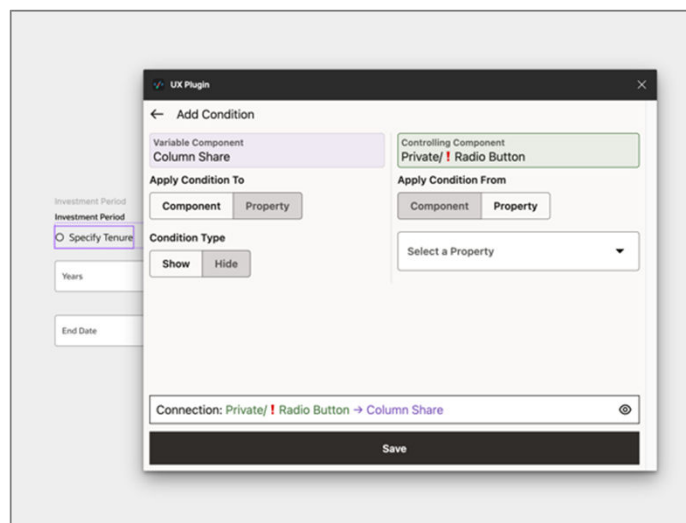
For convenience use Figma's layer panel to select that specific layer.

Figure 9-15 Specify Tenure



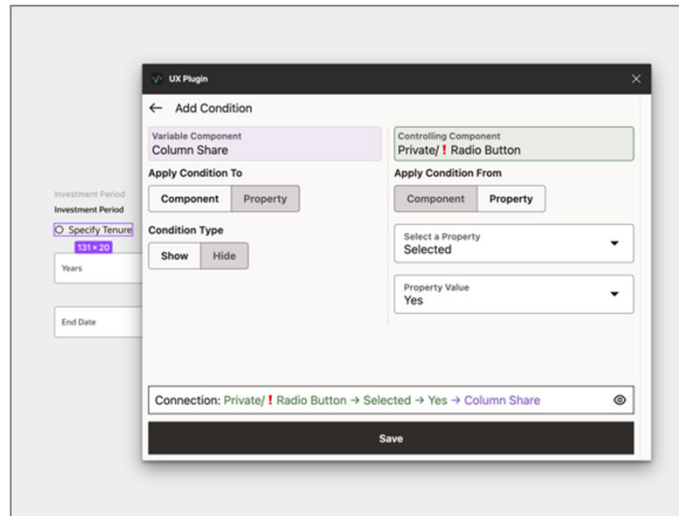
11. After selecting the appropriate layer, switch back to the plugin and apply the desired condition from the **Property** options.

Figure 9-16 Apply Condition



12. Choose the selected property followed by the **Yes** property value from their respective drop-downs.

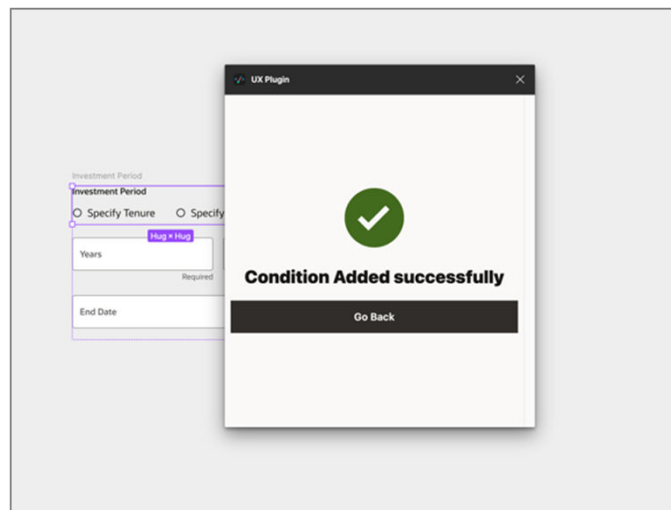
Figure 9-17 Property Selection



13. Review the condition through the connection flow located at the bottom of the window and click **Save** to save the condition.

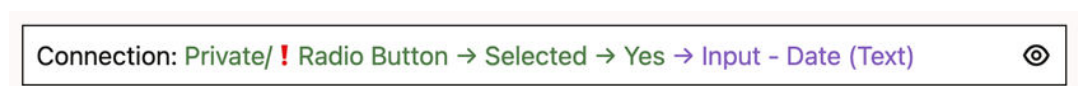
The success message of condition added appears.

Figure 9-18 Success Message



14. Repeat the same procedure for the other radio button **Specify Duration** and ensure that the connection reads:

Figure 9-19 Connection



- [Add Logic](#)
This topic describes the systematic instruction to **Add Logic** option.

9.2.1 Add Logic

This topic describes the systematic instruction to **Add Logic** option.

This capability allows designers to enhance conditions by adding and/or logic, enabling the creation of multiple, complex conditions for supported components. It provides greater flexibility in controlling component interactions and behavior.

There are two types of Logics that can be added:

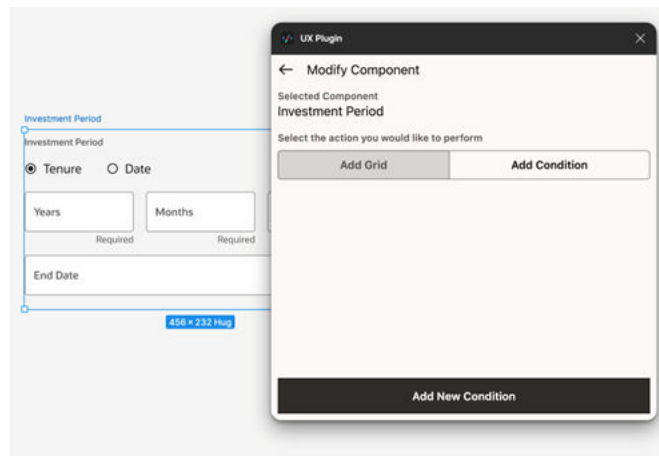
- OR Logic: The case is successful if it meets at least one of the conditions.
- AND Logic: The case is successful only if it meets all conditions.

If there are multiple conditions added to a component, they are in an 'OR' logic.

To add conditions in an AND logic:

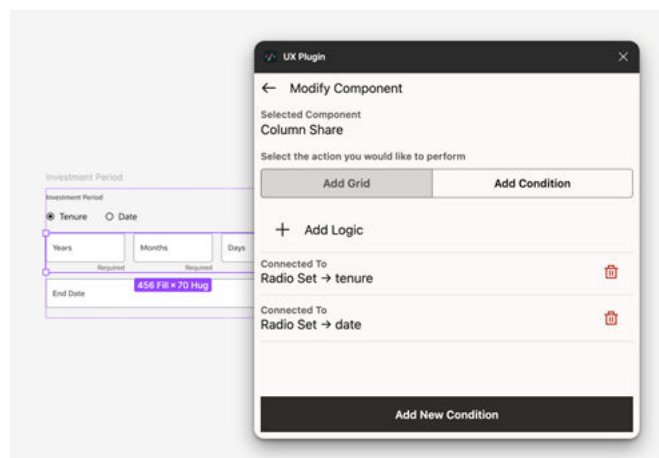
1. Select the component with multiple conditions, then run the plugin and click **Modify**.

Figure 9-20 Component with multiple conditions



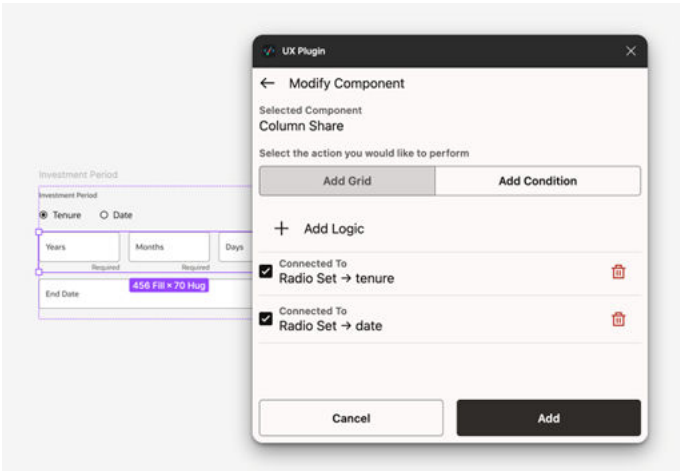
2. Select the **Conditions** button where the multiple conditions are listed and tap on **Add Logic**.

Figure 9-21 Add Logic



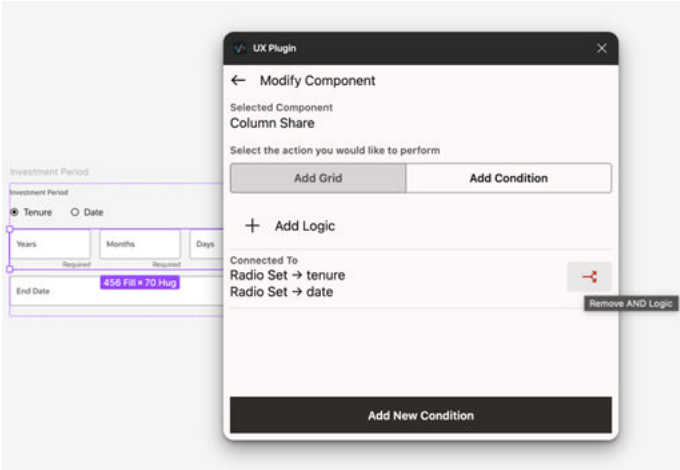
3. Select the conditions where the **AND** logic is to be applied and click **Add**.

Figure 9-22 AND Logic



Removing the **AND** logic will default the logic to **OR**.

Figure 9-23 AND Logic



10

Export to Toolkit

This topic provides information on **Export to Toolkit**.

The Export to Toolkit feature enables you to import a Sigma design directly into the UI Toolkit using three key identifiers:

1. **Access Token** - Provides secure access to your Sigma design, enabling the toolkit to read the file.
2. **Page ID** - Identifies the specific page within the file that contains the design to be exported.
3. **Component Node ID** - Identifies the exact design or component being exported from the page.

These identifiers allow the toolkit to read and process your design, ensuring it is imported accurately for further development and configuration.

Note

Before starting the export procedure, ensure that the entire design is placed within a single, dedicated Sigma section. This section must include:

- The main design
- Responsive variants (if applicable)
- External interactions (if applicable)

- [Steps to Export Design to Toolkit](#)

This topic provides information on export design for the toolkit.

10.1 Steps to Export Design to Toolkit

This topic provides information on export design for the toolkit.

1. Run the UI Toolkit.

Once your Figma section is ready, launch the UI Toolkit.

- a. In the first step, **Folder Creation**, enter all necessary details of your design.
- b. Ensure that the toggle **Import from Figma** is set to **Yes**.

Figure 10-1 Create a new module and component

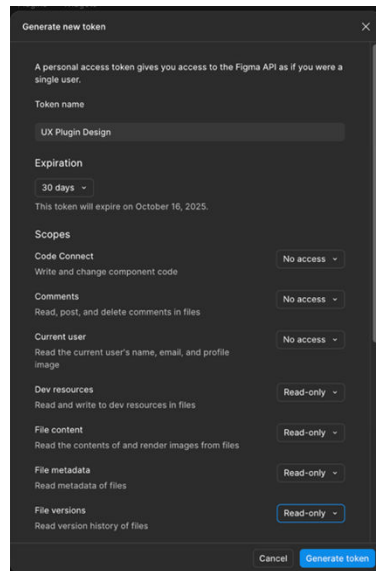
The screenshot shows a web form titled "Create a new module and component". The form is divided into several sections:

- Component Type:** A dropdown menu with "Flow" selected. Below it is a link that says "Know More".
- Import from Figma:** Two buttons labeled "Yes" and "No".
- Save in Extension?:** Two buttons labeled "Yes" and "No".
- Review Type:** A dropdown menu with "Default" selected.
- Flow name:** A text input field containing "figma-toolkit-demo".
- Number of stages:** A text input field containing "1".
- Next:** A button at the bottom left.

2. Click **Next** to provide the **Access Token**, **Page ID**, and **Component Node ID** on the screen prompted.
3. Generate Access Token.

To generate an Access Token:

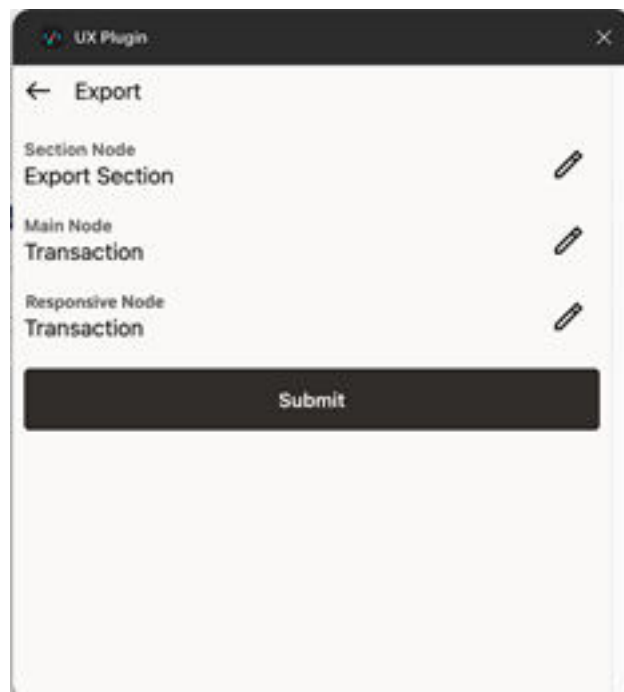
- a. Navigate to **User Profile**, click **Settings**, and then click **Security Tab**.
- b. Under **Personal Access Tokens**, create a new token.
- c. Specify the details such as **Name** and **Duration**.
- d. For **Scope**, select **Read-only access** for the following permissions:
 - Dev Resources
 - File Content
 - File Metadata
 - File Versions
 - Library Assets
 - Library Content
 - Team Library Content
 - Variables
 - Webhooks
- e. Copy the generated token and paste it into the **Access Token** field in the Toolkit.

Figure 10-2 Generate new token

4. Click **Generate token**.
5. Generate Page ID and Component Node ID.

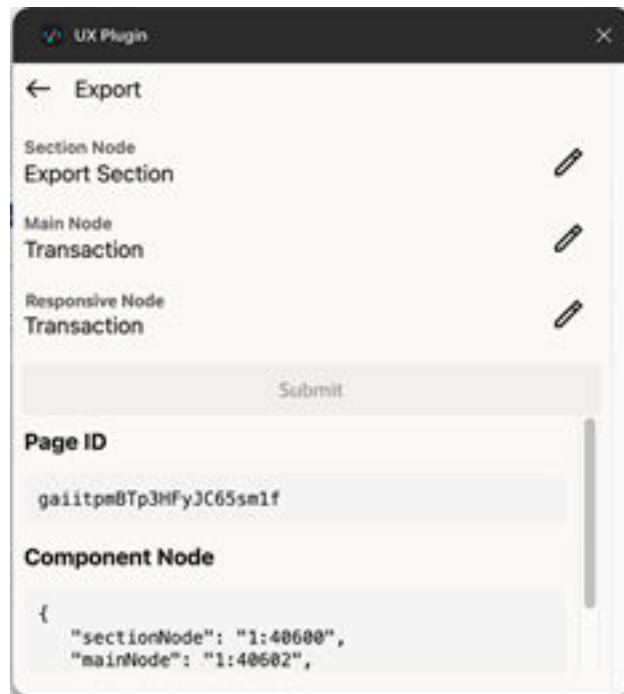
To generate the identifiers:

- a. Run the plugin in Figma.
- b. Click **Export**.
- c. Select the Figma section that contains the complete design.
- d. Select the main design and responsive designs (if applicable).

Figure 10-3 UX plugin

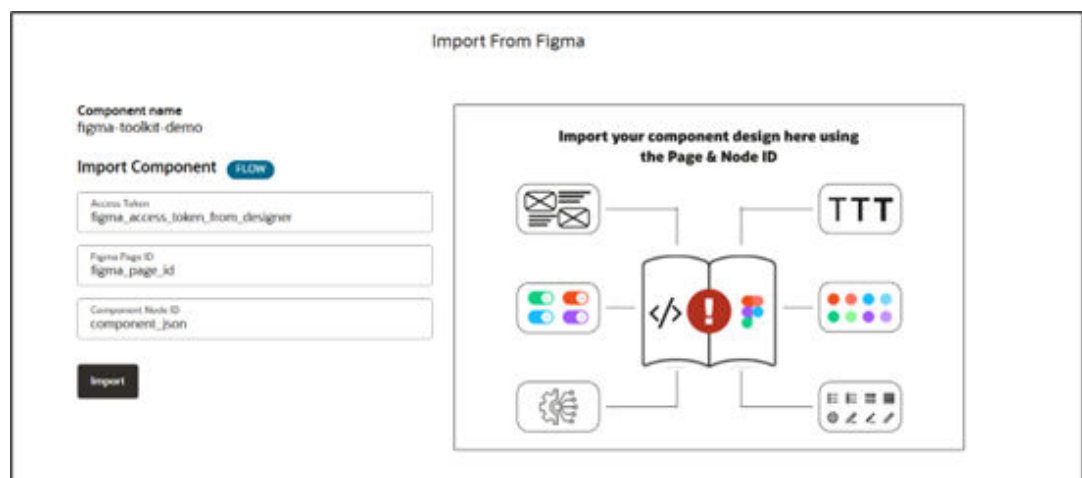
6. Click **Submit** to generate the **Page ID** and **Component Node ID**.

Figure 10-4 UX plugin 1



7. Copy and Paste IDs in Toolkit.
Copy the **Page ID** and **Component Node ID** provided by the plugin.
 - Paste them into the respective fields in the Toolkit interface.

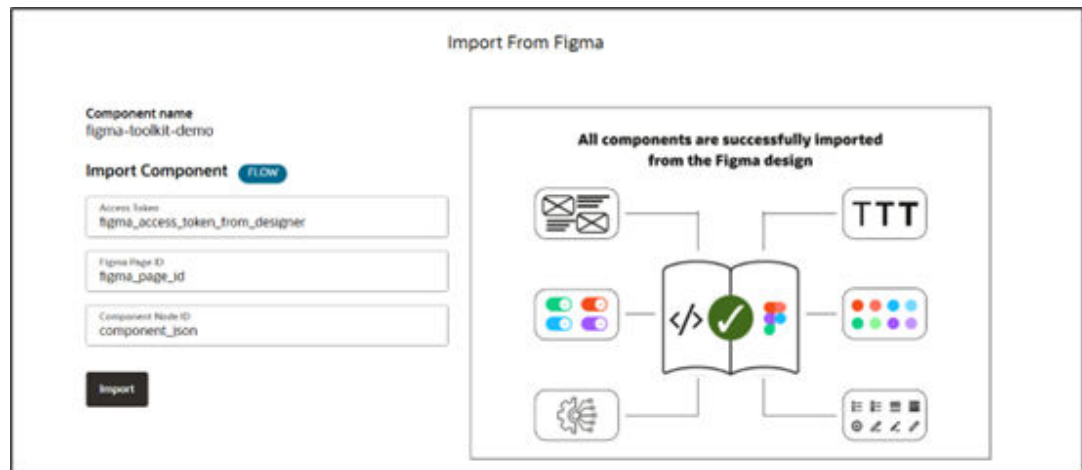
Figure 10-5 Import from Figma



8. Click **Import** to import the design.
9. **Confirmation.**
 - a. If the import is successful, a confirmation message will appear.
All components are successfully imported from the Figma design.

- b. If unrecognized components are detected, they will be listed in the right panel.

Figure 10-6 Import from Figma - Confirmation



Next Steps

After completing the import, proceed with the standard Toolkit workflow starting **from Step 3: Layout Selection**. For detailed guidance on subsequent steps, refer to the **UI Toolkit documentation**.

- 1. What are the prerequisites to use the UX plugin?**

You'll need a basic understanding of Figma, Oracle's Redwood Design System, Oracle JET, and the UI Toolkit. Additionally, an active Figma license and the latest version of the UI Toolkit installed on your system are required.
- 2. How do I set up the plugin?**

Follow the steps outlined in the Plugin Setup section of this manual to install and configure the plugin. Once set up, you can start designing with the plugin.
- 3. Can I create new components using the UX Plugin?**

Yes, the plugin allows designers to kick-start their designs by automatically generating components based on the inputs provided. This is done through the Create New feature, which streamlines the process of setting up design components according to specified requirements.
- 4. How do I enhance existing components?**

The Generate from Existing feature allows you to refine and enhance existing components. This includes capabilities such as converting to transactions, adding grids, and adding conditions.
- 5. What does the Convert to Transaction feature do?**

This feature converts a series of page sections into a 3-step transaction template, which can then be further customized by the designer.
- 6. How do I add a grid to my design components?**

Use the **Add Grid** capability under the **Generate from Existing** option to apply a grid structure to your components. Simply select the frame, specify the grid size and alignment, and save the specifications.
- 7. What is the purpose of the Add Conditions feature?**

The **Add Conditions** feature allows designers to manage interactions between components by defining how one component (controlling component) affects another (variable component), such as showing or hiding elements based on user inputs.
- 8. What if I need to apply multiple conditions to a component?**

You can use the **Add Logic** feature to apply **and/or** logic to your conditions, enabling the creation of more complex interaction handling.
- 9. Can I modify layer names in Figma?**

It is recommended to keep the original layer names of components unchanged, as modifying them may result in recognition errors by the plugin.
- 10. What components are supported by the plugin?**

The plugin supports components exclusively from the Oracle JET and Redwood libraries. Other components are not recognized by the plugin.
- 11. How do I export my design to the UI Toolkit?**

Once the design is complete, the UI developer can export the design to the UI Toolkit. Ensure that the toolkit is running in the background during the export process to avoid any issues.
- 12. Can I update an already exported design?**

Yes, the plugin allows you to overwrite or update an existing design when exporting to the UI Toolkit. Details on this process can be found in the **Export to Toolkit** section.

