USER GUIDE

Bridgetek

BRIDGING TECHNOLOGY

EVE Screen Designer 4.10

Document Version: 1.0

Date: 03-12-2020

Neither the whole nor any part of the information contained in, or the product described in this manual, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. This product and its documentation are supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied. Bridgetek Pte Ltd will not accept any claim for damages howsoever arising as a result of use or failure of this product. Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Bridgetek Pte Ltd, 178, Paya Lebar Road, #07-03, Singapore 409030. Singapore Registered Company Number 201542387H. © Bridgetek Pte Ltd.

Contents

A. Preface 5
Purpose 5
Intended Audience 5
Document Reference 5
Feedback 5
B. Overview 6
Introduction 6
Key Features 6
What's new in ESD 4.10 6
Known Issues & Limitations 7
Terms & Description 8
Credits 8
C. Setup & Installation 9
System Requirements 9
Hardware Requirements 9
Dependencies / Pre-Requisites 10
Installing ESD 4.10 Package 11
Installation Folder 17
D. Working with ESD 4.10 18
Major features 18
Support full BT81X features 18
Support for ASTC bitmap format 19
Support extended for font format and Unicode font 20
Generate Flash file and write Flash file to device 21
Restructured EVE HAL Library 21
Additional layouts and widgets 21
UI enhancements 21
Impacts of ESD 4.10 updated features 22
Migrating from ESD 3.0 to ESD 4.10 project 22
Setting Platform Specific Properties 23
Migrating existing projects to new platform 24
E. Getting Started 24
The Graphical User Interface 24
Menu bar 24
Toolbar 26
Project Explorer 29
Screen Layout Editor 30
Logic Node Editor 31
Library Browser 32
Error List 32
Output 33
Property Editor 33
Status bar 34
Application Project Structure 34
Application Workflow 36
Design Layout using Layout Editor 37
Custom Widget 37
Add Bitmap Resource 37

Copyright © Bridgetek Pte Ltd

Configure Bitmap Resource 38
Design Screen Logic using Logic Node Editor 39
Simulation 40
Target selection 40
Build & Upload 42
Export 43
Exported Folder Structure 44
Bitmap Resource 45
Layout Editor 45
Page File 45
"Active" Property 46
Switch Page 46
Page Persistence 47
User Defined Function for Page File 48
Zoom In & Out 48
Main File 49
Logic File 49
C File 50
Logic Note Editor 50
Basic Logic Node 50
Composite Logic Node 51
Page Node 52
Widget Node 52
Layout Type Widget 54
Actor Node 55
Logic Object 56
Connections 56
Rendering Order 56
Logic Note Editor – User Interaction 57
Library Browser 59
ESD Layouts 60
Utilities 60
Switch Page 65
Dynamic Switch Page 66
Auto Switch Widget 67
Fixed Positioning 68
Fill Layout 69
Linear Layout 69
Stretch 71
Scroll Layout 72
Scroll Switch Page Layout 73
ESD Render Functions 75
Elements 75
Primitives 77
Display List 80
ESD Theme 80
Built In Themes 81
ESD Utilities 83
ESD Idle Checker 84
Logic Flow 84
Condition 85
Binary Condition 85
Local Method 86
Binary Operator 87
Ternary Operator 88
Unary Operator 88
Sequence 89
Switch Node 89

Copyright © Bridgetek Pte Ltd

Switch Value 90
Set Variable 90
Watch Variables 91
Logic Interface 91
Input 92
Output 92
Signal 93
Slot 93
Variable 94
Writer 95
Widget Interface 95
Property Editor 97
Common Properties 97
Name 98
Depth Sort 98
Active 99
Programming Features 99
Macros 99
ESD_TYPE 99
ESD_ENUM 100
ESD_FUNCTION 101
ESD_METHOD 102
ESD_VARIABLE 103
ESD_GLOBAL 104
ESD_INPUT 104
ESD_OUTPUT 104
ESD_UPDATE 104
Pre-compiler options 104
ESD_SIMULATION 105
FT900_PLATFORM 105
Add User Functions 105
Creating Source File 105
Editing the Source File 105
F. Appendix A – List of Figures 107
G. Appendix B – List of Tables 109
H. Appendix C – Revision History 110

Copyright © Bridgetek Pte Ltd

A. Preface

Purpose

This document describes the functionality and procedures involved in using the **E**VE **S**creen **D**esigner (ESD) 4.10

Intended Audience

The intended audience shall be any GUI application developer working with EVE products.

Document Reference

Document Name	Document Type	Document Format
FT81x Series Programmers Guide	Programming Guide	PDF
FT81x Datasheet	Datasheet	PDF
FT9xx Toolchain Installation Guide	Installation Guide	PDF
BT81X Series Programming Guide	Programming Guide	PDF
BT81x Datasheet	Datasheet	PDF

Feedback

Every effort has been taken to ensure that the document is accurate and complete. However any feedback on the document may be emailed to docufeedback@brtchip.com. For any additional technical support, refer to http://brtchip.com/contact-us/.

Copyright © Bridgetek Pte Ltd

B. Overview

Introduction

EVE **S**creen **D**esigner (ESD) is the next generation of smart IDE for EVE, making EVE-based GUI development much easier to accomplish. This tool enables users to build one GUI application using a **visual programming**[1] method without needing to know any EVE-specific display list commands.

ESD provides a **WYSIWYG** ("**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et") environment for editing graphics, designing visual effects, and defining GUI application user logic, generating **ANSI C code** for the targeted hardware platform. Users can also choose to simulate the whole design to experience the UI before compiling and downloading the generated source code. Furthermore, ESD has the capability to work seamlessly with Bridgetek FT9XX tool chain. Users can compile, link the generated source code with FT9XX tool chain and upload it to the targeted platform without leaving ESD.

A layout mechanism is introduced to manage widgets and pages in a more generic way. The layout mechanism will enable users to create more dynamic UI much easier than before. In addition, ESD 4.10 dramatically enhances the functionality of the logic nodes editor, layout editor and project browser, for better user experience.

Key Features

The following are some of the key features of **E**VE **S**creen **D**esigner:

- **WYSIWYG** GUI
- High level widgets
- No EVE display list knowledge required
- Widget based GUI construction
- Drag and drop widget to create screen layout
- Inter widget communication
- Screen logic creation without coding
- Simulation of screen logic and user touch input using mouse
- Building and downloading the generated "C" code (if FT9XX Toolchain is installed)

What's new in ESD 4.10

New Features:

- Adds support for EVE GPU BT817/8 based platforms (ME817-WH10C, ME817-WH10C)
- Enhanced UI of New Target Dialog
- Allow users to configure the default empty project
- Extend panning feature of Logic editor with Alt key and Left mouse press
- Re-factor ESD Framework libraries
- Sample project for video player added
- Updated user guide with new platform details

[1] https://en.wikipedia.org/wiki/Visual_programming_language

Copyright © Bridgetek Pte Ltd

Bug Fixes:

- Rectified build failure issue of exported Eclipse project.
- Rectified build failure issue of emulator based MSVC exported project.
- Fixed Build error when MM900X platform is selected
- Fixed code generator for local method nodes.
- Fixed screen widget bounding box for auto switch layout
- Fixed bugs in example projects

Known Issues & Limitations

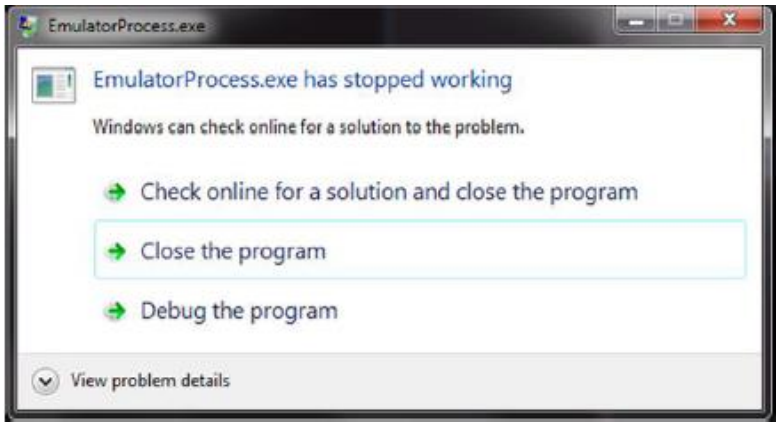
The following are some known issues and limitations of ESD:

- Only the BT81X/FT81X series EVE is supported. FT80X Series EVE is **NOT** supported
- The C code editor in ESD does not support all the features of a code editor.
- If the project file path is too long (i.e. more than 512 bytes), ESD may have a problem opening it. The typical error message is shown below:

  `"Unable to generate output files, check directory permission at: C:\Users\xxxx.xxxx\Project\ESD\......"`

  Where

  `"C:\Users\xxxx.xxxx\Project\ESD\......"` refers to the project folder.
- Logic node editor background goes white after system hibernates.
- In some unusual cases, users may encounter a dialog box (shown below) which will not affect the functionality. In this case, just ignore the dialog box by closing it.



- In few cases, users may need to click **[Recompile]** at the toolbar to update the frozen simulation result.

Copyright © Bridgetek Pte Ltd

Terms & Description

Abbreviations/Term	Description
Actor	One type of logic node which is regularly updated but without visual appearance
DLL	Dynamic Link Library is a collection of small programs, any of which can be called when needed by a larger program that is running in the computer
ESD	EVE Screen Designer
EVE Emulator	Bridgetek behaviour-modelling software for EVE Series chip
HAL	Hardware Abstraction Layer is a software subsystem providing hardware abstraction.
IDE	Integrated Development Environment
Layout type widget	One type of widget which has no visual appearance rendered by EVE , but manages the associated widget
Logic Node	The node expressing certain logic. (Also referred to simply as "node" in this document)
Logic Node Editor	The place where you create logic by connecting the logic node
Page	One single screen in design
Simulation	Preview the project or page by running the generated C code on PC
Widget	One type of logic node which has visual appearance rendered by EVE

Credits

Open Source Software

- Qt: http://doc.qt.io/qt-5/licensing.html under LGPL.
- TinyCC: http://bellard.org/tcc/ and http://repo.or.cz/tinycc.git available under LGPL.
- Errorlist module: https://github.com/kaetemi/errorlist available under MIT license
- QScintilla: part of PyQt available under PyQt commercial license
  https://www.riverbankcomputing.com/commercial/license-faq

Icons Copyright

Some of the icons used in ESD 4.10 are from:
http://p.yusukekamiyamane.com/icons/search/fugue/
They are used in compliance with the Creative Commons Attribution 3.0 License.

Copyright © Bridgetek Pte Ltd

C. Setup & Installation

System Requirements

To install ESD 4.10 application, ensure that your system meets the requirements recommended below:

- ✓ Ideally Windows 10; alternatively Windows 8 or 7 with the latest windows updates
- ✓ 1.6GHz or faster processor
- ✓ 1GB of RAM (1.5GB if running on a virtual machine)
- ✓ Multi-Core CPU is highly recommended
- ✓ At least 512MB hard disk space
- ✓ Display resolution 1080x800 pixels or higher
- ✓ "Write" permission to the installation folder

Hardware Requirements

The exported C source code from ESD 4.10 is targeted at **FT90X MCU Module** and **EVE Module** platforms.

The supported platforms are listed as below:

Platform Name	Compatible EVE Module	Compatible FT90X Module Name	Compatible PC based Module
ME812A WH50R	ME810A-WH70R(800x480) ME811A-WH70R(800x480) ME812A-WH50R(800x480) ME813A-WH50C(800x480)	MM900EV-LITE MM900EV1A MM900EV2A MM900EV3A	NA
ME810A HV35R	ME810A-HV35R(320X480)		NA
VM816CU50A	VM816C (800x480)	NA	FT4222 (libFT4222.dll)
ME813AU WH50C	ME813AU WH50R(800x480)	NA	FT4222 (libFT4222.dll)
VM816C50A	VM816C (800x480)	NA	MPSSE (libMPSSE.dll)
ME817EV WH70C	ME817EV (1024X600)	MM900EV-LITE MM900EV1A MM900EV2A MM900EV3A	FT4222 (libFT4222.dll)
ME817EV WH10C	ME817EV (1280X800)		FT4222 (libFT4222.dll)

While working on hardware platform (FT90X MM modules), users must ensure that an SD card is inserted into the SD Slot since ESD 4.10 will try to detect the presence of an SD card. If the SD card cannot be detected, then the application may fail.

Copyright © Bridgetek Pte Ltd

Dependencies / Pre-Requisites

- **Visual C++ Redistributable for Visual Studio 2015**

  If the PC does not have Microsoft Visual Studio 2015 installed, Visual C++ Redistributable is required. Users can download this from:

  https://www.microsoft.com/en-sg/download/details.aspx?id=48145

- **Windows 10 Universe C Runtime**

  ESD 4.10 has run-time dependency on Windows 10 Universe C Runtime (CRT). You may download it from https://www.microsoft.com/en-us/download/details.aspx?id=48234 and install on your PC should the following problem be encountered:

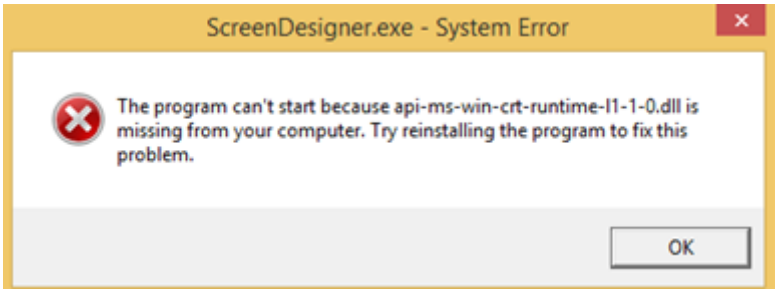


Figure 1 - Screen Designer - System Error

- **FT9XX Tool Chain Version 2.4.0 or later**

  To compile and build projects, the FT9XX Tool Chain 2.4.0 must be installed on the PC. It is downloadable from http://brtchip.com/ft9xx-toolchain/.

  Please ensure that the Tool Chain executable path is defined by the system PATH environment variable.

  Users are advised to check the known issues and limitations (of FT9XX Toolchain) while building the ESD 4.10 project with FT9XX Toolchain. The respective FT9XX Toolchain package version release note contains the list of known issues and limitations.

  For ESD 4.10, we recommend users to install FT9XX Tool Chain version 2.5.0 for the best results.

Copyright © Bridgetek Pte Ltd

Installing ESD 4.10 Package

The following steps will guide you through the ESD 4.10 *Setup/Installation* process.

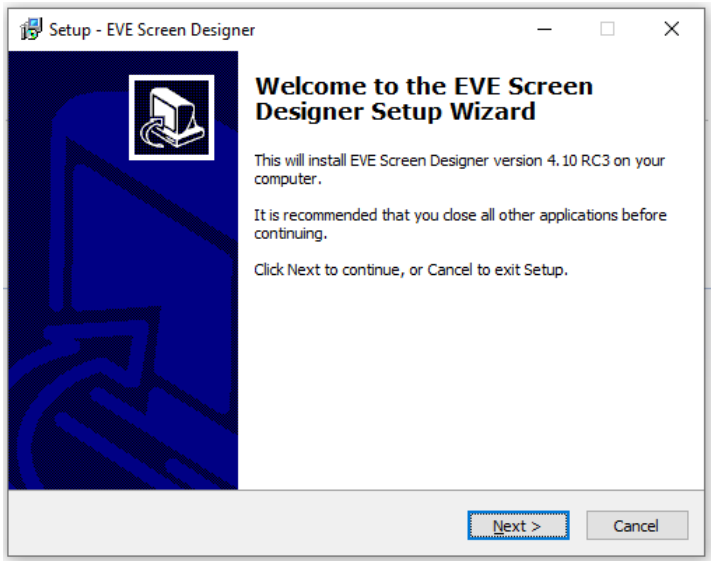
i. Download the package from www.brtchip.com.

ii. When prompted with a download dialog box. Click **[Save]**.

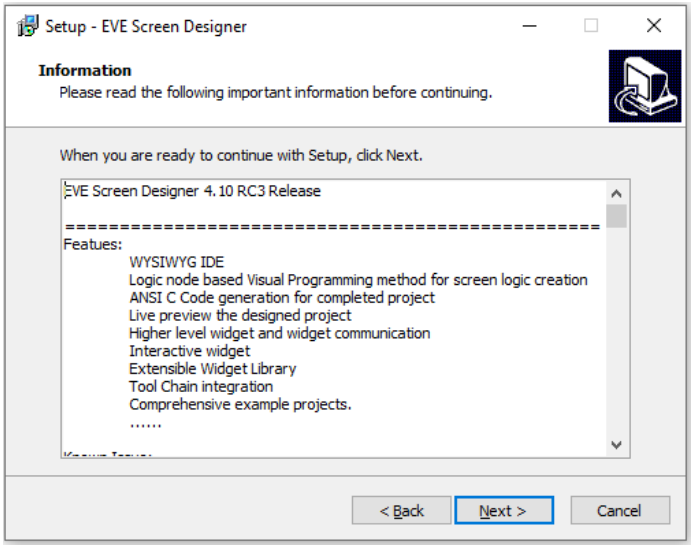
iii. Navigate to the folder under which the package files are downloaded.

iv. Extract the zip file contents. Double click on the executable file – **EVE Screen Designer 4.10.exe**

v. The EVE Screen Designer 4.10 Setup Wizard is displayed along with a Welcome message.

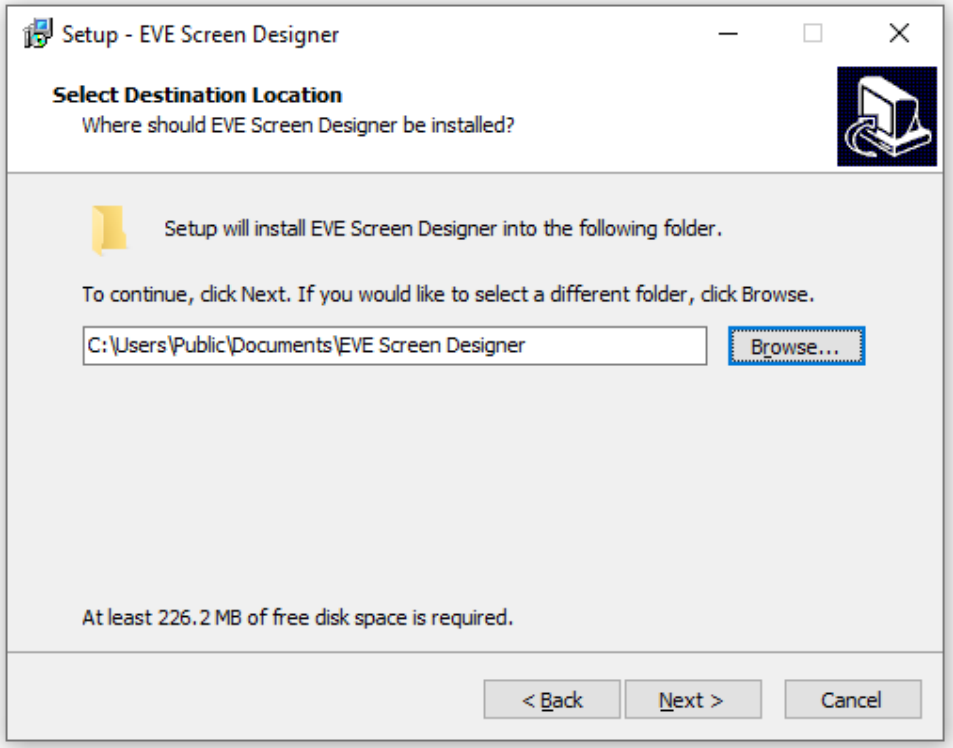


vi. Click **[Next]** to view the end user information window.



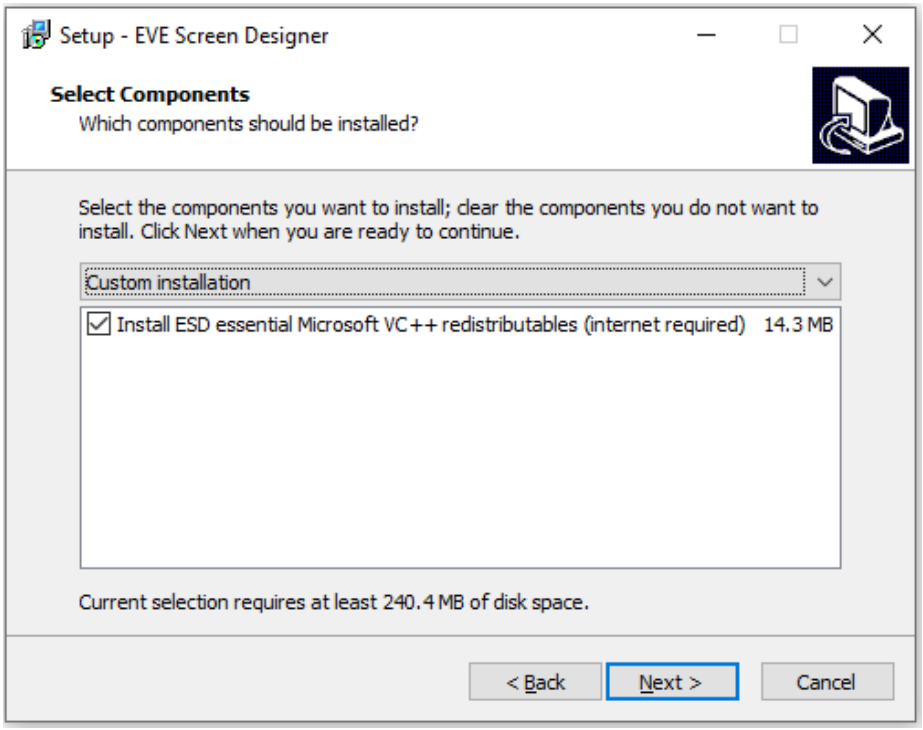
Copyright © Bridgetek Pte Ltd

vii. Click **[Next]** to select a "Destination Folder" for installing the files. Accept the default folder or click **Browse** to specify a different location. Click **[Next]** to confirm the destination folder and continue.

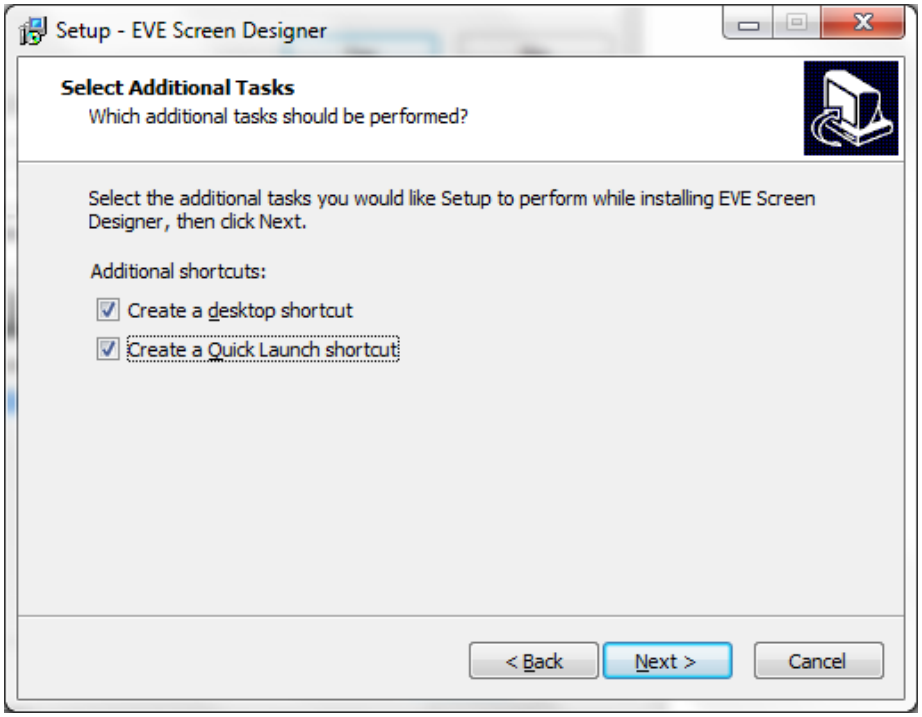


viii. In the Select **Components** window, check **"Install ESD essential Microsoft VC++ redistributables (internet required)"** boxes, to install runtime components of Visual C++ Libraries. Click **[Next]** to prepare for the installation.

Copyright © Bridgetek Pte Ltd

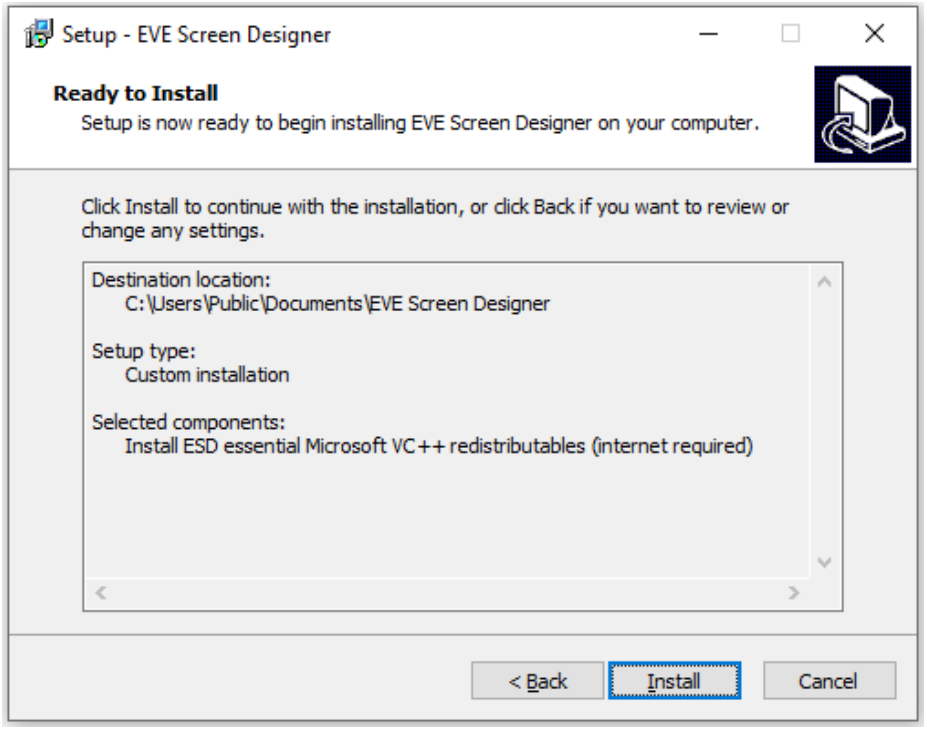


ix. In the **Select Additional Tasks** window, check **"Create a desktop / Create Quick Launch shortcut"** boxes, to have the ESD 4.10 icon and Quick Launch shortcut displayed on the desktop if required. Click **[Next]** to prepare for the installation.

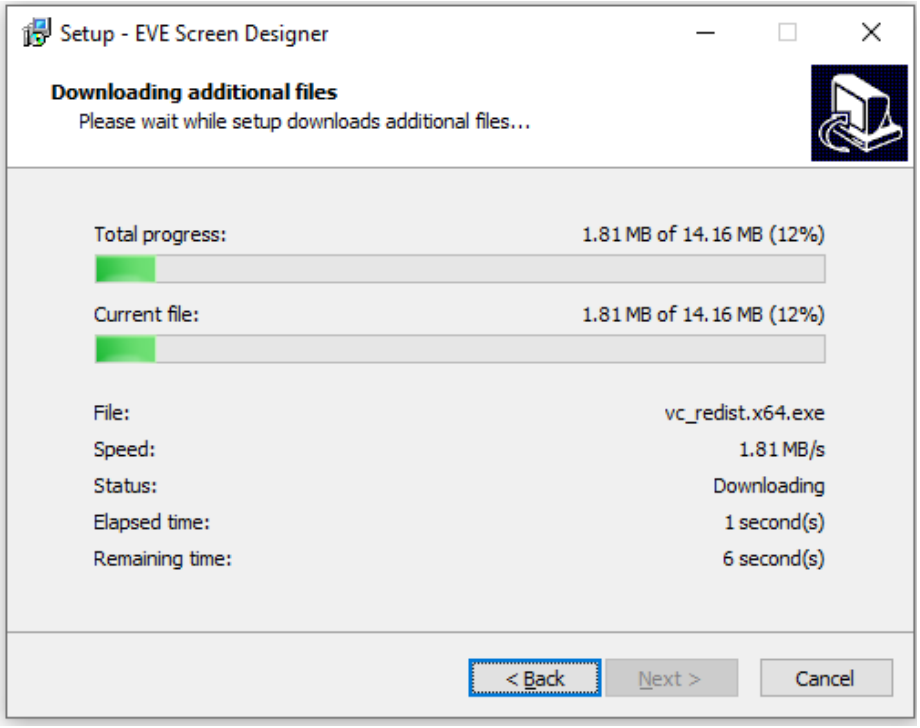


Copyright © Bridgetek Pte Ltd

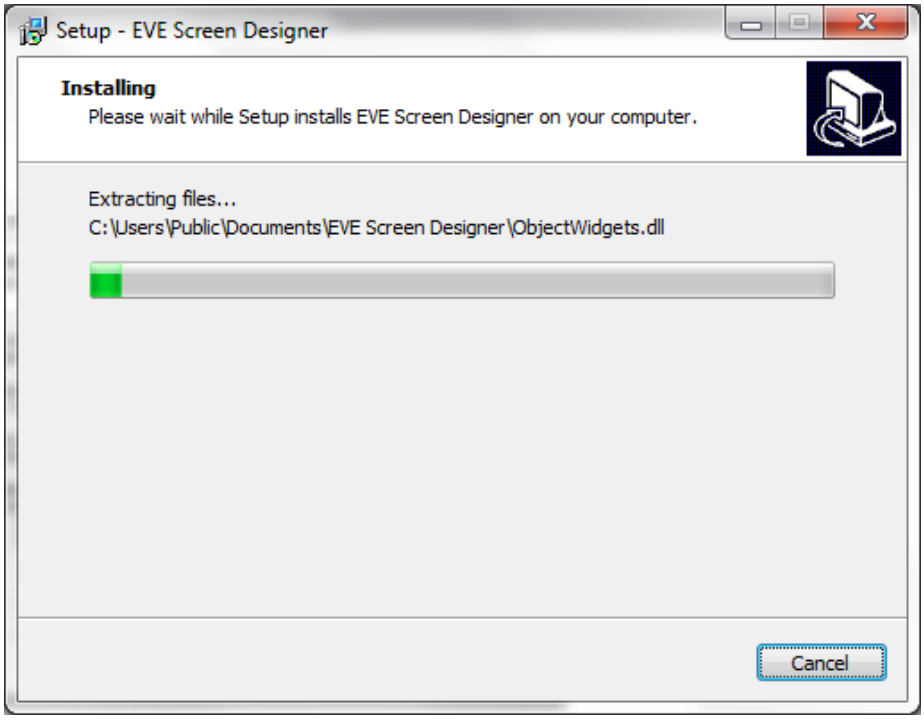
x. The initial setup is completed and the application is ready to be installed.



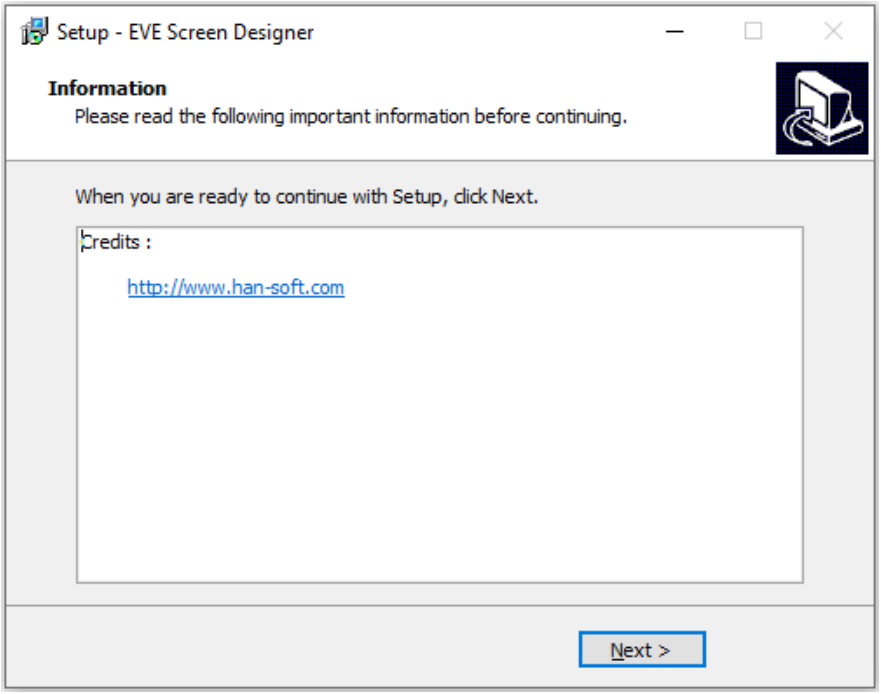
xi. Click **[Install]** to start the installation. First page progress bar indicates the progress of the download of the MSVC C++ redistributable and the second page progress bar indicates that the installation is in progress.



Copyright © Bridgetek Pte Ltd

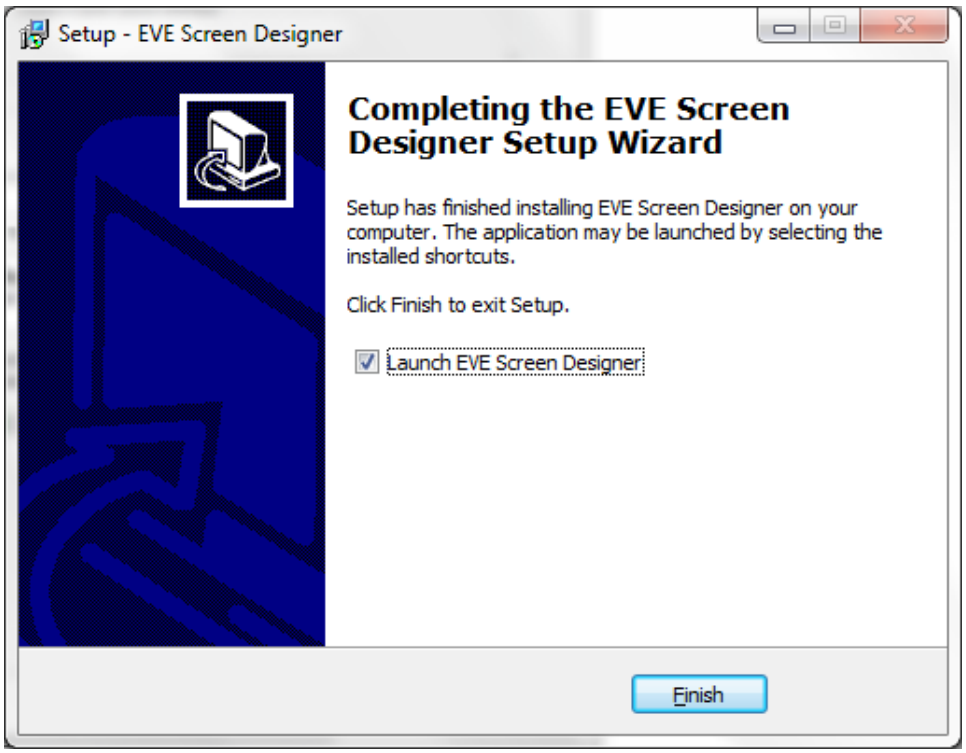


xii. In the **Information** page, Click **[Next]** to proceed ahead



Copyright © Bridgetek Pte Ltd

xiii. Upon successful installation, click **[Finish]**.



Copyright © Bridgetek Pte Ltd

Installation Folder

The following table provides a list of folders that can be found under the installation path upon successful installation of ESD 4.10.

Folder Name	Description	Permission
Examples	The example projects created by ESD 4.10	Read/Write
Imageformats	Qt run-time DLLs for image format supporting	Read-Only
Styles	Qt run-time DLLs for UI styles	Read-Only
Libraries	Widget library, application framework and Hardware abstraction layer	Read-Only
Log	Stores the runtime logs for debug purpose	Read/Write
Manual	Contains this document	Read-Only
platforms	Qt platform specific run-time DLLs	Read-Only
Settings	Configuration files for third-party utilities and tool chains as well as ESD settings. The files are in XML format.	Read-Only. Reserved for advanced users to change.
Templates	The template files used by ESD	Read-Only
TinyCC	TinyCC run-time used for ESD simulation purpose.	Read-Only
Tools	The third-party utilities used in ESD 4.10 for bitmap conversion purpose	Read-Only

Table 1 - Installation Folder

Copyright © Bridgetek Pte Ltd

D. Working with ESD 4.10

Major features

Support full BT81X features

Users can refer to EVChargePoint examples in the path *<ESD Folder>/Examples/Advanced/EVChargePoint*. Opening it may take more than 5 minutes because of ASTC, bitmap and font conversion. This example contains all of the new BT81X features. Here are the details:

1. Apply ASTC Bitmap Format for images and fonts.

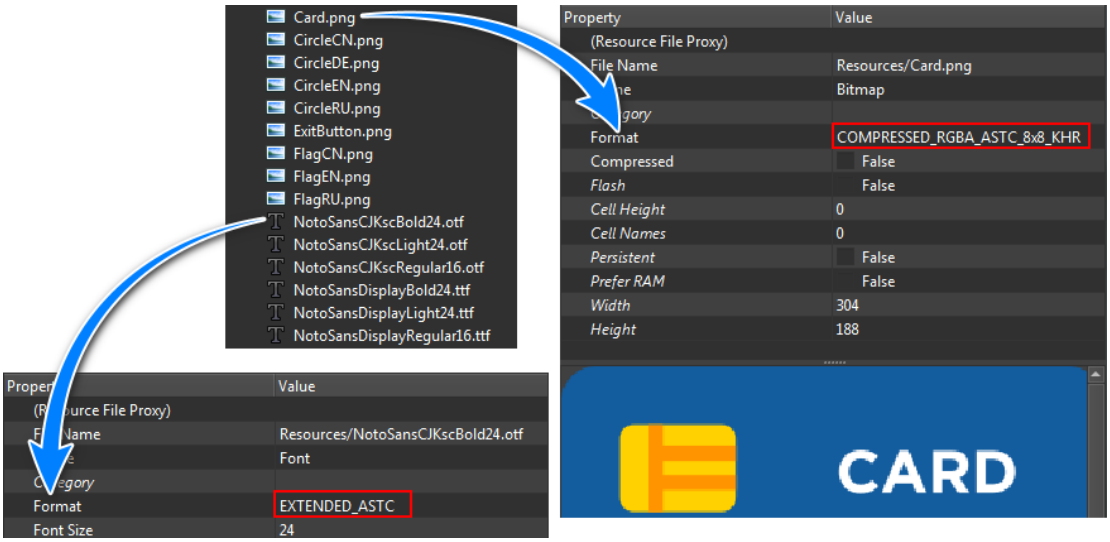


Figure 2 - ASTC applied for images and fonts

2. Resources are stored and loaded from flash.

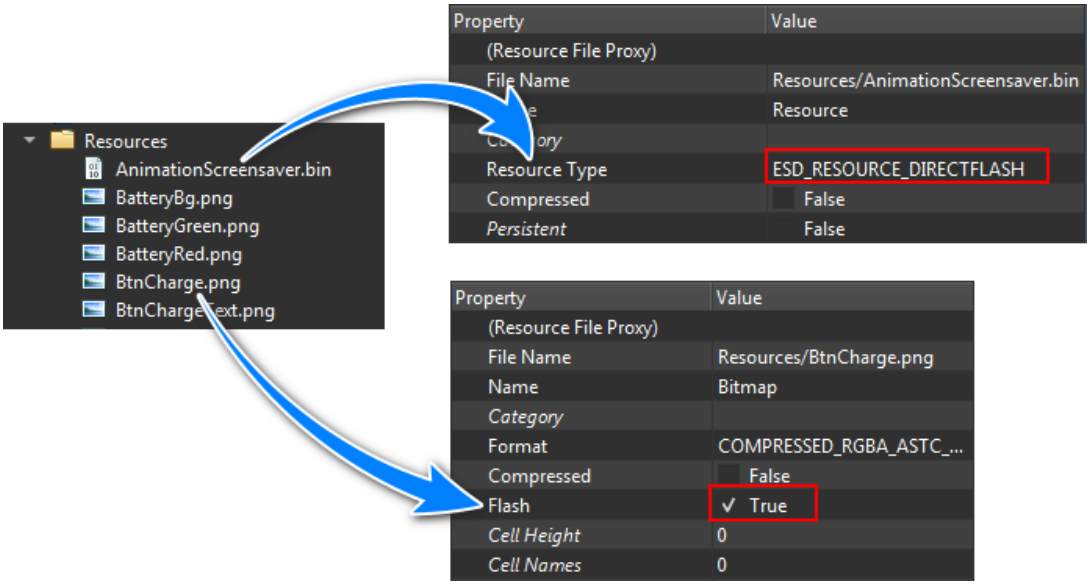


Figure 3 - Load resources from flash

Copyright © Bridgetek Pte Ltd

3. Use Unicode font with ASTC format and predefined character set

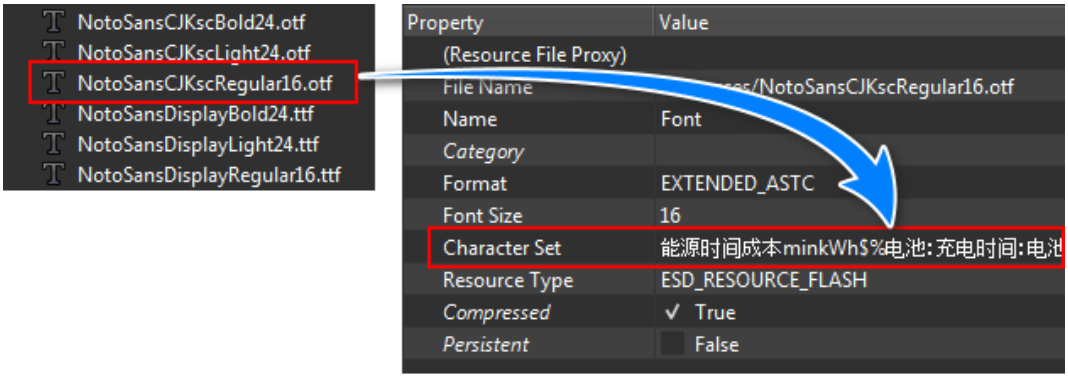


Figure 4 - Use Unicode font with pre-defined character set

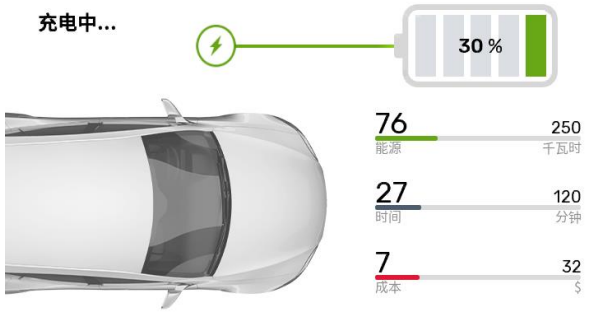


Figure 5 - Chinese font used in application

Support for ASTC bitmap format

In ESD 4.10, images can be converted to ASTC bitmap format to save a lot of storage space while image quality is still retained.

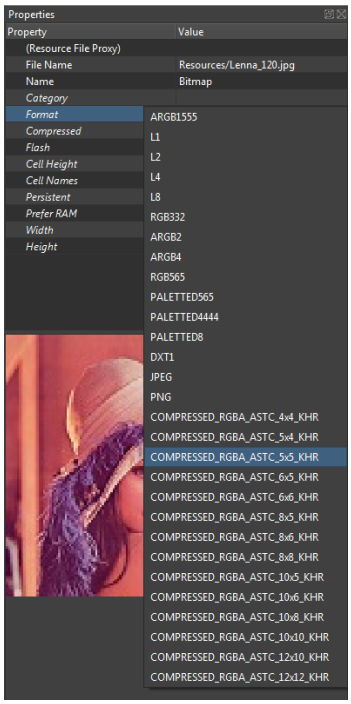


Figure 6 - ASTC Format

Copyright © Bridgetek Pte Ltd

Support extended for font format and Unicode font

Unicode font file can be added to a project by clicking ***File -> Add*** from the menu bar and in the *Property* window, input font size and character set as desired.

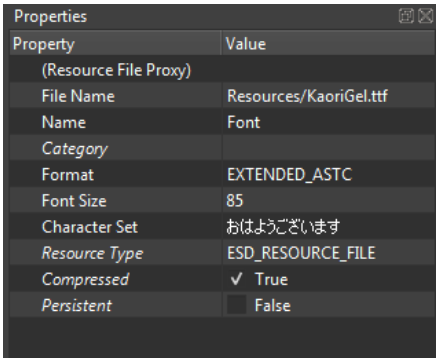


Figure 7 - Font size and Character set

Once the font resource is ready, add a widget, such as ESD Label, ESD Button to project. In *Property* window, select appropriate *Font Resource*.

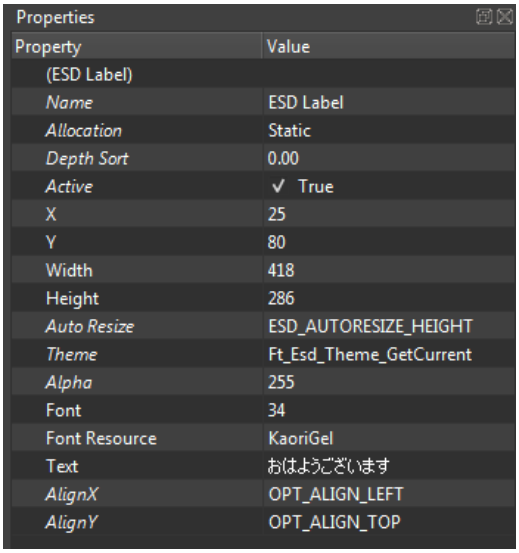


Figure 8 - Select Unicode Font Resource

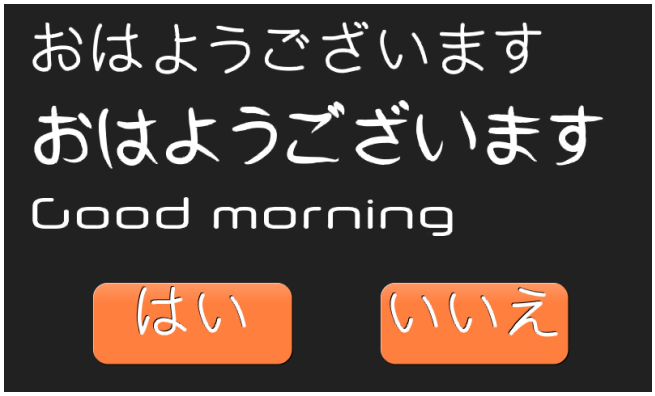


Figure 9 - Unicode font applied for Label and Button

Copyright © Bridgetek Pte Ltd

Generate Flash file and write Flash file to device

Resources can be loaded from flash or RAM_G. ESD 4.10 can generate Flash file for BT81X projects, then uploads the Flash file to the device by pressing the "Build and Upload to Hardware" button.

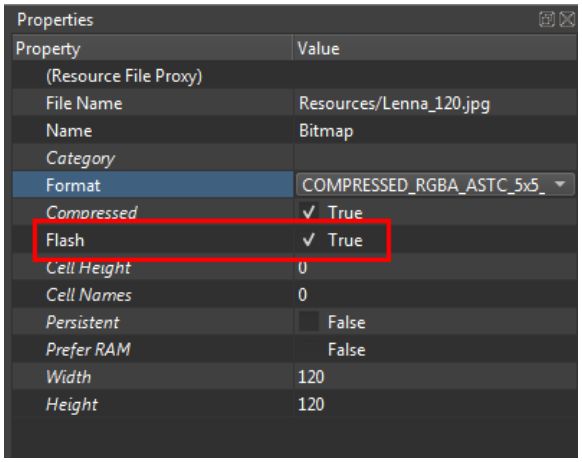


Figure 10 - Mark resource as storage in Flash

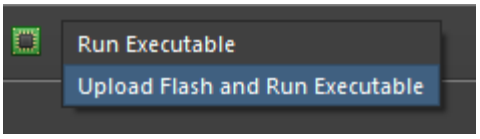


Figure 11 - Upload Flash file to device

Restructured EVE HAL Library

EVE HAL Library is simplified. It contains C source files, and a sub-folder "Hdr" that comprises of the C header files.

Additional layouts and widgets

The following layouts and widgets have been added –

- ESD Arc Slider, ESD Circular Slider
- ESD Ring, ESD Partial Ring

UI enhancements

The following User Interface related enhancements has been added to ESD 4.10 –

- Filter for Library Browser and Project Browser window
- More comprehensive toolbar buttons for platform description
- New menu to clean up unused/generated files in project
- Simplify the process of adding new resources into project
- Added context menu for document tab

Copyright © Bridgetek Pte Ltd

Impacts of ESD 4.10 updated features

Since the new/updated features of ESD 4.10 may cause API changes, existing ESD 3.0 projects are not fully compatible with ESD 4.10. Although ESD 4.10 has the utility inside to detect and migrate opening ESD 3.0 project, it may still not work as expected.

- *ESD Layouts:* A new category of widgets "ESD Layout" are introduced into the Library Browser, which enables the layout feature.

- *ESD Widgets:* A new set of widgets are introduced into "ESD Widgets" category in the library. These set of widgets are to be used in constructing the pages. The nodes under the "ESD Render Functions" are not supposed to construct page as it has no widget interface and the layout widget is unable to manage them.

Migrating from ESD 3.0 to ESD 4.10 project

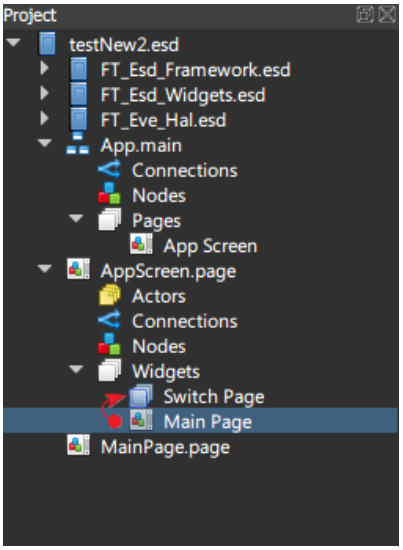
When ESD 4.10 opens an existing ESD 3.0 project, it will prompt users to migrate to the latest version. If user chooses not to migrate, ESD 4.10 will not open ESD 3.0 project.

MIGRATION NOTES

1. Please back up your project if you are not sure about the migration. The migration process will **overwrite the project and it cannot be reversed**.
2. The migrated project file will be renamed from **"*.esd3"** to **"*.esd"**.
3. The target module of the opened project will be redirected to two built-in platforms after migration. So if the target module is in 320 x 480 resolutions, users may select the platform **"ME810A HV35R"** from "Build Target" combo box from the toolbar and make a platform switch.

In some cases, if an existing ESD 3.0 project cannot be migrated to ESD 4.10, then users are required to migrate manually.

- The variable *"Parent"* of logic is renamed to *"Owner"* in ESD 4.10.
- *X, Y, Width, Height* properties in widgets is now accessible through the widget variable.
- Instead of pointer, struct *"Ft_Esd_BitmapCell"* shall be used by value in ESD 4.10.
- In order to restore the default behaviour of application logic *"App.main"*, users may need to manually move the existing pages into layout widget *"Switch Page"* in the Project Browser.



Copyright © Bridgetek Pte Ltd

Setting Platform Specific Properties

ESD 4.10 users are able to create one project supporting multiple platforms with various screen size. Therefore, on different platforms, the same widget may have a different property value. For instance, for a platform with screen size 480 x 320, the button size may shrink to a smaller size from the original value for a platform with screen size 800 x 480.

Users can make a property value specific to the current target through the context menu of the property browser. This can be done by right clicking the selected property of the current widget. This is illustrated in the picture given below –

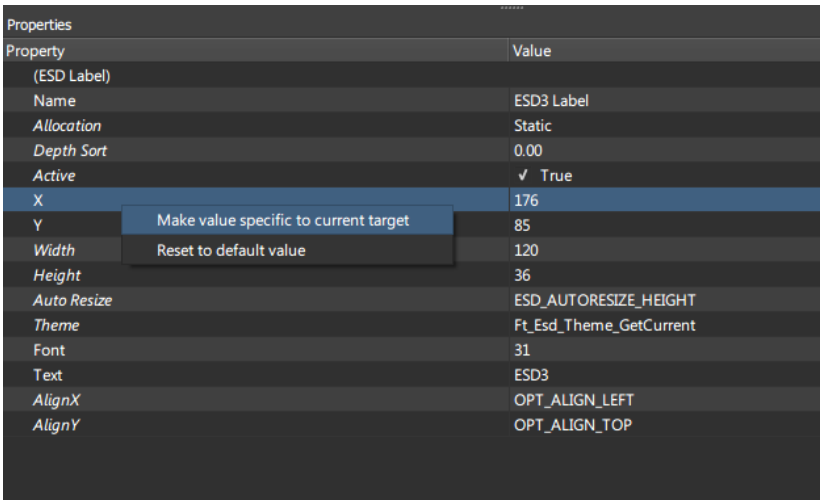


Figure 12 - Making Value Specific to Current Target

User can remove value specific to current target through context menu in property browser by right clicking the selected property of current widget. This is illustrated in the picture given below -

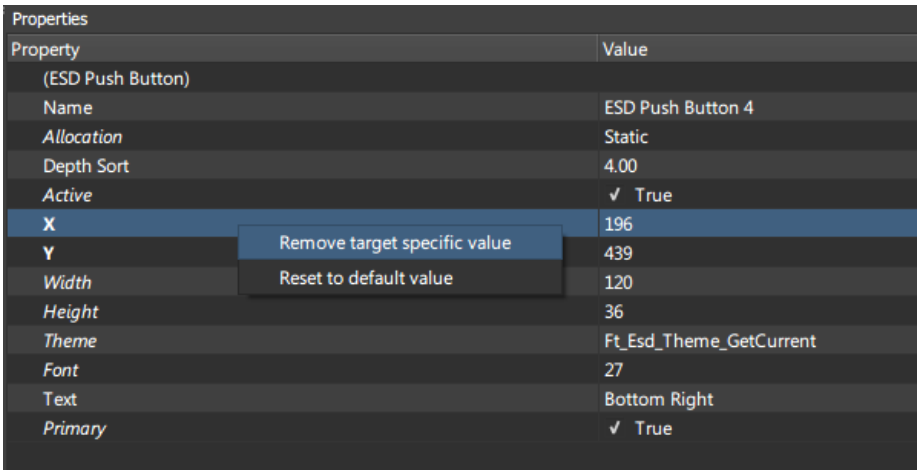


Figure 13 - Removing Target Specific Value

Users can refer to the "ScreenResolution" project under the "Installation/Example" folder.

Copyright © Bridgetek Pte Ltd

Migrating existing projects to new platform

All the existing projects may have been developed using a specific platform. Users can choose to migrate projects to a different platform by selecting the appropriate "Build Target" combo box. If the desired target is not available in the "Build Target" Combo box, then the user can configure the individual specifications ("Eve Platform", "Display Resolution" and "Host Platform") of the target by selecting the appropriate combo boxes.

E. Getting Started

The Graphical User Interface

ESD 4.10 user interface has the following components:

1 *Menu Bar*
2 *Toolbar*
3 *Project Explorer*
4 *Screen Layout Editor*
5 *Logic Node Editor*
6 *Library Browser*
7 *Error List/Inspector*
8 *Output* window
9 *Property Editor*
10 *Status Bar*

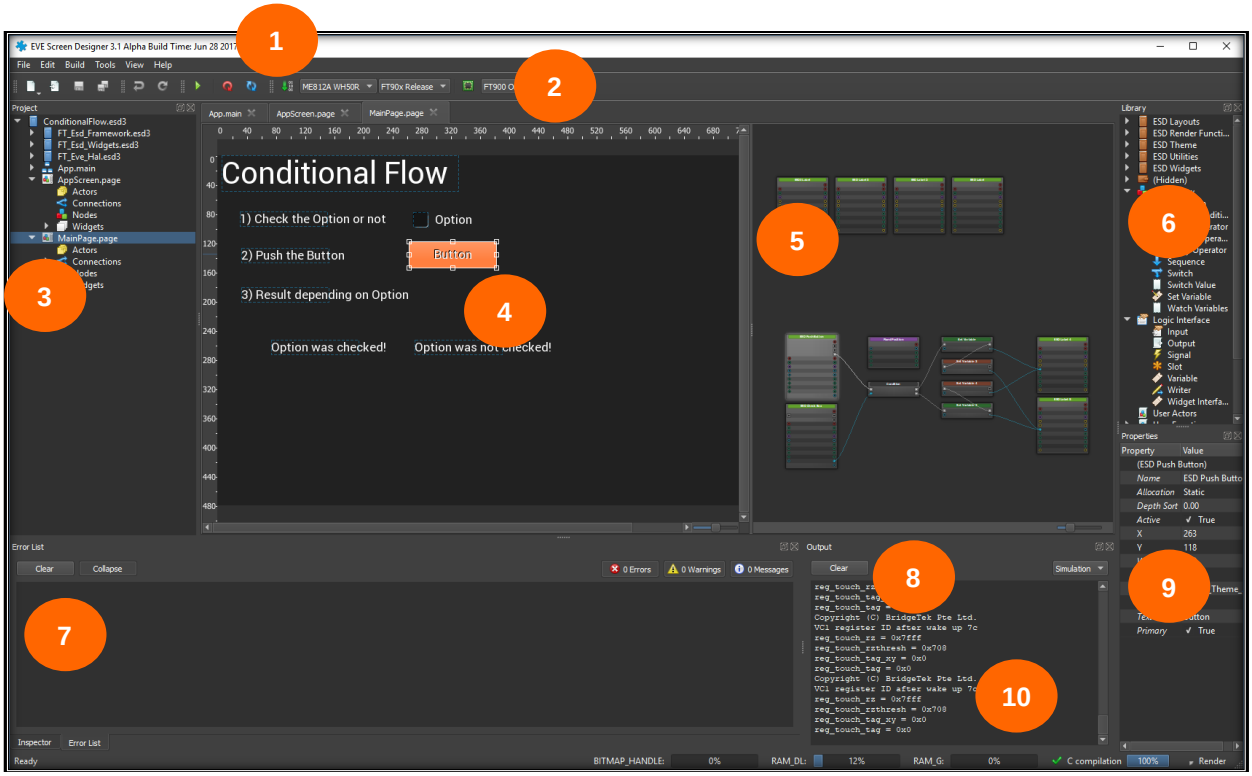


Figure 14 - EVE Screen Designer 4.10 User Interface Components

Menu bar

Copyright © Bridgetek Pte Ltd

The ESD 4.10 *Menu bar* displays the headings for each drop-down menu. According to the function, the commands are grouped under each of these menu headings.

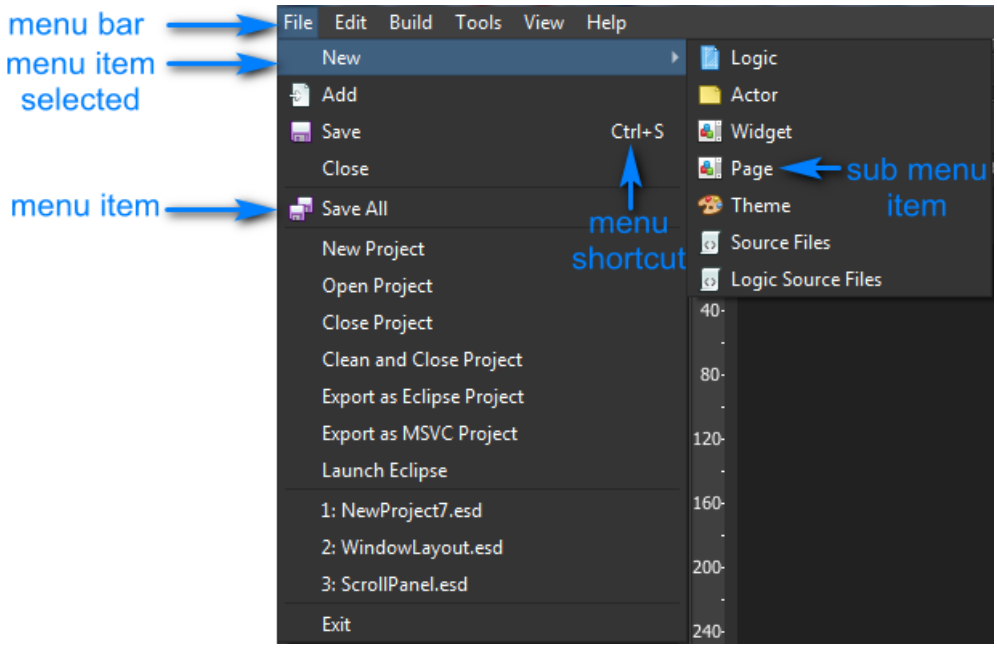


Figure 15 - Menu bar

The following table provides the list of Menu/Submenu and its description –

Menu	Submenu		Description
File	New	Logic	Creates new file for the current project
		Actor	
		Widget	
		Page	
		Theme	
		Source Files	
		Logic Source Files	
	Add		To add existing file(s)/resource (such as bitmap, C source file etc.) to the project
	Save		To save the current file
	Close		To close the current file
	Save All		To save all the files in the current project
	New Project		To open a new project
	Open Project		To open an existing project
	Close Project		To close a current project
	Clean And Close Project		To clean current project then close it
	Export as Eclipse Project		To export as Eclipse project
	Export as MSVC Project		To export as MSVC project
	1:<Project		To open recent project

Copyright © Bridgetek Pte Ltd

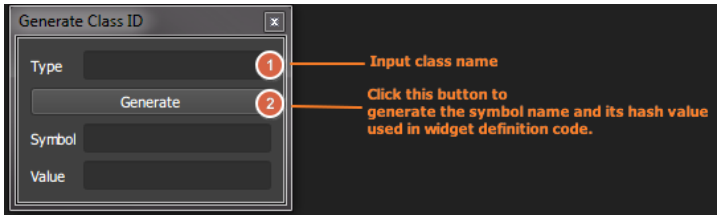
	Path>		
	Exit		To close the ESD 3.1 application
Edit	Undo		To reverse the action of an recently performed action
	Redo		To revert the effects of the undo action
	Cut		Cut the selected node(s) to clipboard
	Copy		Copy the selected node(s) to clipboard
	Paste		Paste the node(s) from clipboard
	Find		Find a specific text string in source editor
	Replace		Replace a specific text string in source editor
Build	Build Executable		To generate a executable file
	Build and Upload to Hardware		To generate a executable file and upload to hardware
	Browse to Executable		To navigate to the folder under which the executable file is located
Tools	Generate Class ID		To generate a Class ID for user widget or layout. It is a hash from the widget name and generally useful when user prefers to write code in "C" language. 
View	Project		To display or hide a component view. By default all the components are displayed.
	Library		
	Properties		
	Inspector		
	Error List		
	Output		
Help	User Guide		Opens the ESD 4.10 User Guide with respect to current version
	About		Displays the version details
	3rd Party		Displays the copyright, disclaimer and license information of the 3rd party software

Table 2 - Menu & Description

Toolbar

The *Toolbar* provides easy access to common functions (in the form of icons) such as *new file, save file, undo, redo* etc.

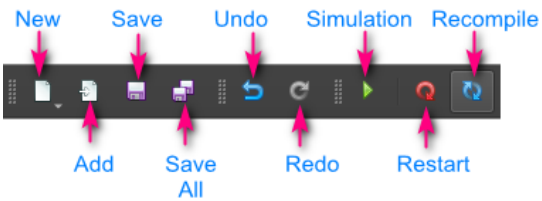


Figure 16 - Toolbar 1

Copyright © Bridgetek Pte Ltd

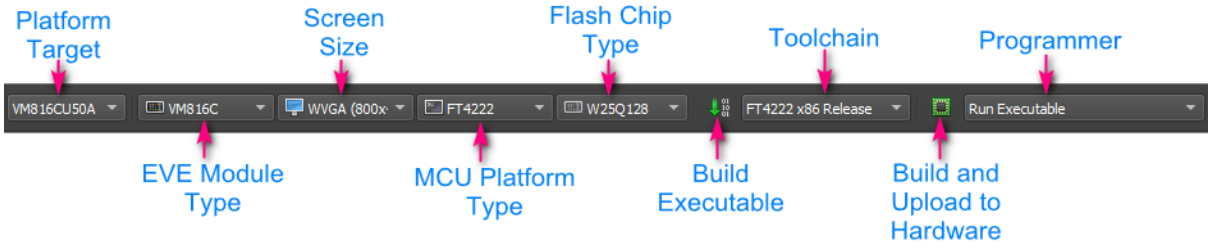


Figure 17 – Toolbar 2

The following table provides the list of toolbar functions and its description –

Toolbar Function	Description
New	To open new *Logic/Actor/Widget/Page/Theme/Source Files/Logic Source Files*
Add	To add existing resource (such as bitmap, C source file etc.) to the project
Save	To save the currently open file
Save All	To save all the files in the current project
Undo	To reverse the action of a recently performed action
Redo	To revert the effects of the undo action
Simulation	A toggle button which starts or stops the simulation mode. - This state indicates that the ESD 4.10 is in simulation mode. Clicking this button stops the simulation. - This state indicates that the ESD 4.10 is out of simulation mode. Users can drag/drop widgets or edit them safely.
Restart	To automatically restart the EVE emulator. Clicking this button will force the simulated screens to be re-drawn.
Recompile	To recompile the whole project's source code using ESD built-in TinyCC compiler. It is a mandatory procedure for simulation.
Platform Target	Select the hardware platform as source code building target
EVE Module Type	To select EVE module type
Screen Size	To set screen size
MCU Platform Type	Select the building configuration when toolchain is invoked to build
Flash Chip Type	To select flash chip type
Build Executable	Build the project files
Toolchain	Select toolchain to build
Build and Upload to Hardware	Upload Flash file to device and/or run executable file
Programmer	Select executable mode

Table 3 - Toolbar & Description

Copyright © Bridgetek Pte Ltd

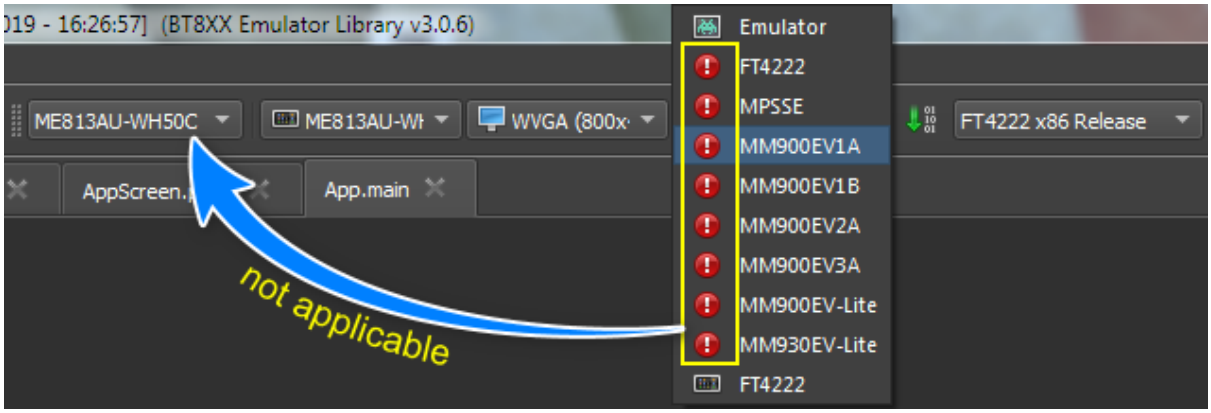


Figure 18 - MCU Type is not applicable to selected platform

The red exclamation mark in the image above means those MCU types are not applicable to the currently selected platform.

Copyright © Bridgetek Pte Ltd

Project Explorer

The *Project Explorer* window organizes all the files used in the project in a tree view. It also lists out all the resources used by each file. Project explorer allows users to navigate each page of the project.

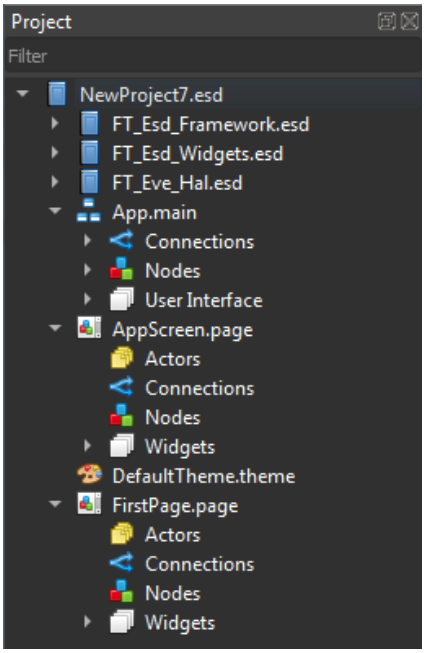


Figure 19 - Project Explorer window

In case of many items existing in a project, user can use filter textbox in order to get a sort list of items. Text that matches the filter is highlighted. Regular expressions are supported.

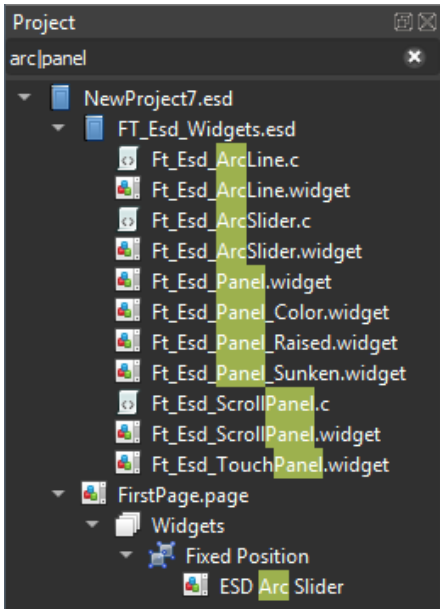


Figure 20 - Project Explorer Filter

Copyright © Bridgetek Pte Ltd

Screen Layout Editor

The *Screen Layout Editor* displays the rendering output of EVE and allows users to edit C source code. The page/widget/main files can be opened and edited in this editor.

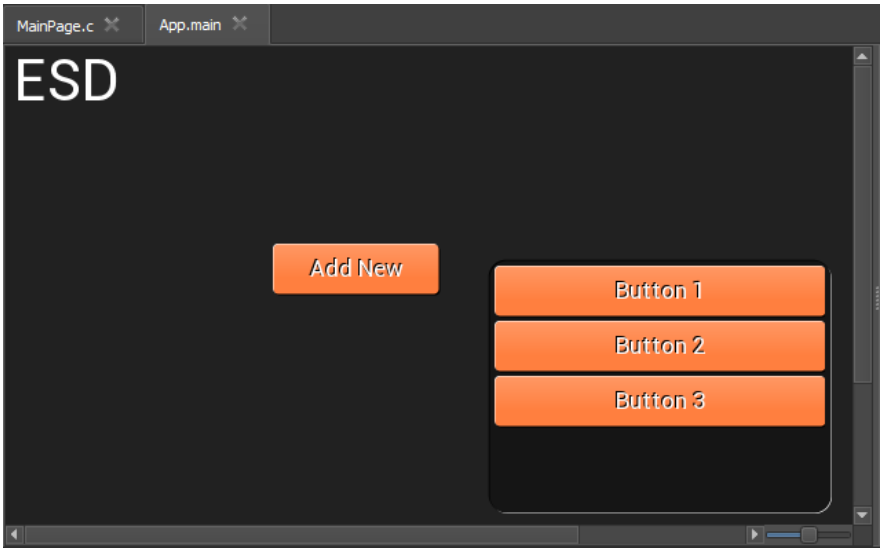


Figure 21 - Screen Layout Editor – App.main

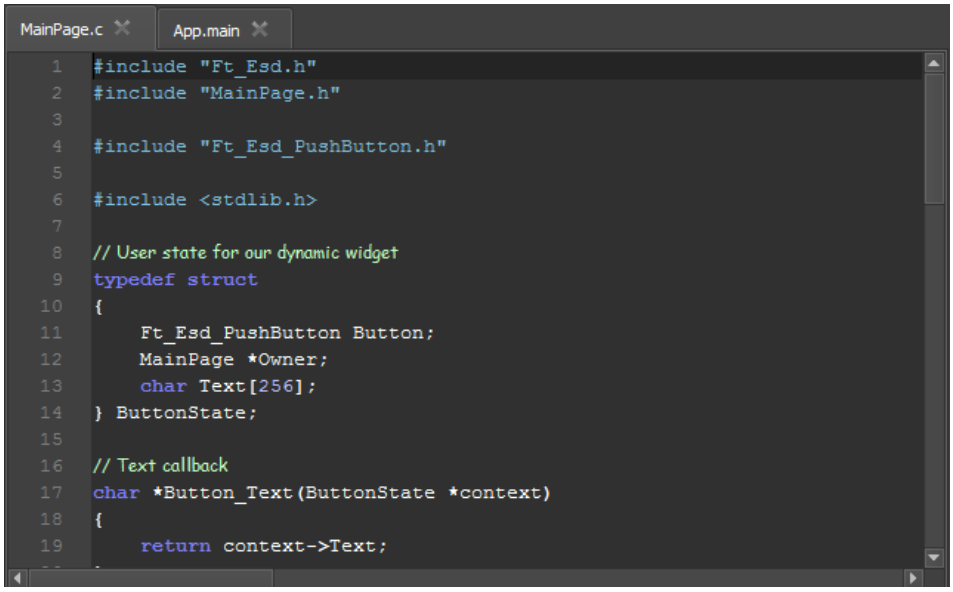


Figure 22 - Screen Layout Editor - MainPage.c (C Source Code)

It shares one view port with the logic node editor. Users can adjust the size of the screen layout editor by dragging the splitter handle.

Users can drag and drop the widgets from the library browser to form the layout when simulation mode is *"off"*. For the other kinds of logic nodes, if the EVE rendering process is not defined, screen layout editor does not allow them to be dropped in.

Copyright © Bridgetek Pte Ltd

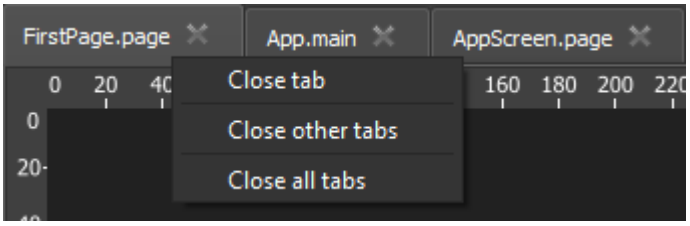


Figure 23 - Context menu on Tab bar

Users can right click on tab bar to active a context menu. Those menu items are described in table below.

Menu	Description
Close tab	Close the current tab
Close other tabs	Close all tabs except the current tab
Close all tabs	Close all tabs that were opened

Logic Node Editor

The *Logic Note Editor* allows users to layout logic nodes and to establish connections to create logic maps. Users can drag and drop a logic node from the Library Browser component to the Logic Note Editor in order to create connections.

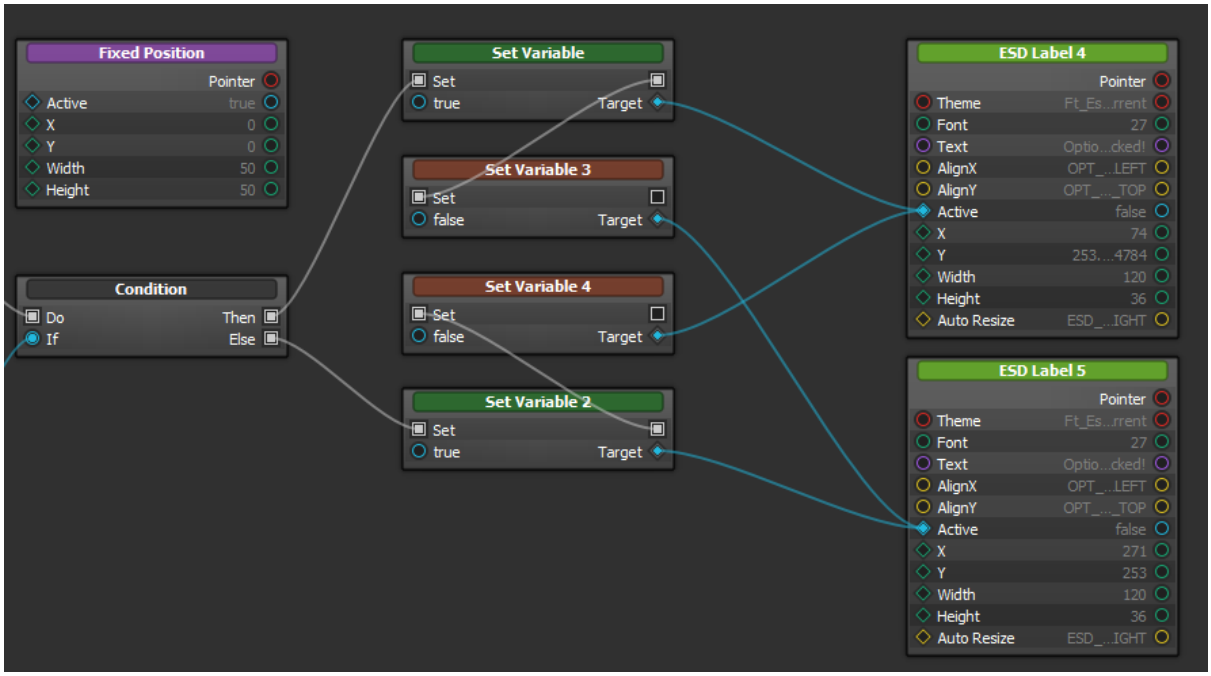


Figure 24 - Logic Node Editor

Copyright © Bridgetek Pte Ltd

Library Browser

The *Library Browser* allows the storing of all the available logic nodes and resources in ESD 4.10 that includes both built-in and user-defined ones. Users can view these logic nodes by category and select one for their project.

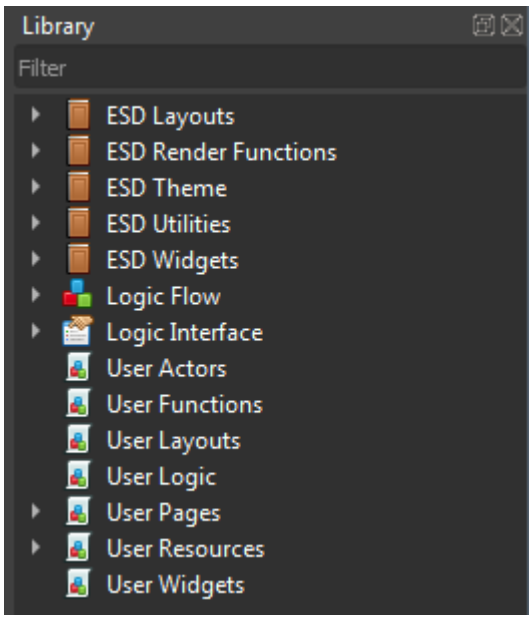


Figure 25 - Library Browser

The following table provides the ESD library name and its description –

Library Name	Description
ESD Layouts	ESD built-in layout widgets and relevant utilities
ESD Render Functions	ESD built-in Render functions
ESD Theme	ESD built-in basic theme manipulation functions
ESD Utilities	ESD built-in utilities
ESD Widgets	ESD built-in widgets
Logic Flow	ESD built-in logic node for control flow
Logic Interface	ESD built-in logic node for interface
User Actors	User defined actor logic node. It is empty by default.
User Functions	User defined functions. It is empty by default.
User Layouts	User defined layouts. It is empty by default.
User Logic	User defined logics. It is empty by default.
User Pages	Pages added by user. It is empty by default if no page is created.
User Resources	Resources added by users (For example: bitmap). It is empty by default.
User Widgets	Widgets created and added by users. It is empty by default.

Table 4 - ESD 4.10 Libraries

Error List

Copyright © Bridgetek Pte Ltd

The *Error List* is a dock window which shows the message output from ESD 4.10, while saving and recompiling the project files. Any error message displayed in this window indicates that the generated source code for the current project is unable to be executed successfully. Users need to double check the logic defined in *logic node editor* or the user-defined source code in the project.

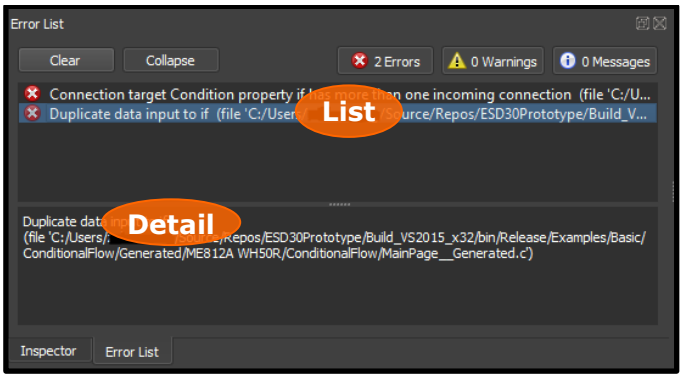


Figure 26 - Error List Window

Output

The *Output* component is a docked window which shows the message output from the EVE emulator and Toolchain.

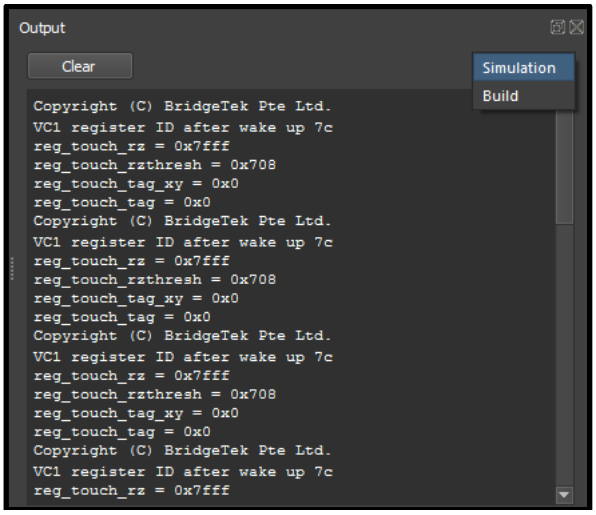


Figure 27 - Output Window

To check a message from the EVE emulator, click **"Simulation"** from the drop down list. To check a message from the EVE Toolchain while building generated source code, click **"Build"** from the drop down list. This window is automatically updated during simulation or building.

Property Editor

The *Property Editor* allows user to edit the properties of selected logic nodes. The sample screenshot given below shows the property editor of an ESD Push button.

Copyright © Bridgetek Pte Ltd

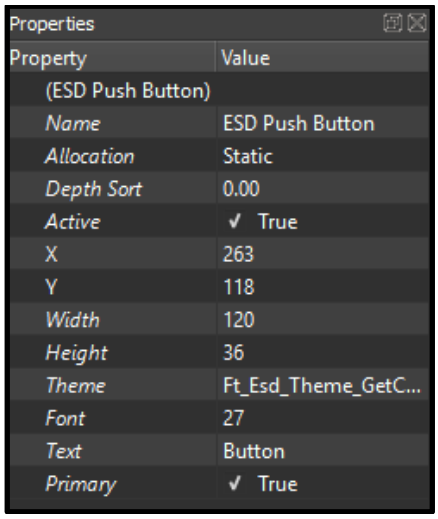


Figure 28 - Property Editor (ESD Push Button)

Status bar

The *Status bar* is found at the bottom of the user interface. It primarily displays the current status of any of process or job that is being handled by the application. For example, if the user is performing a C compilation, then the compilation status of the generated C code is displayed.

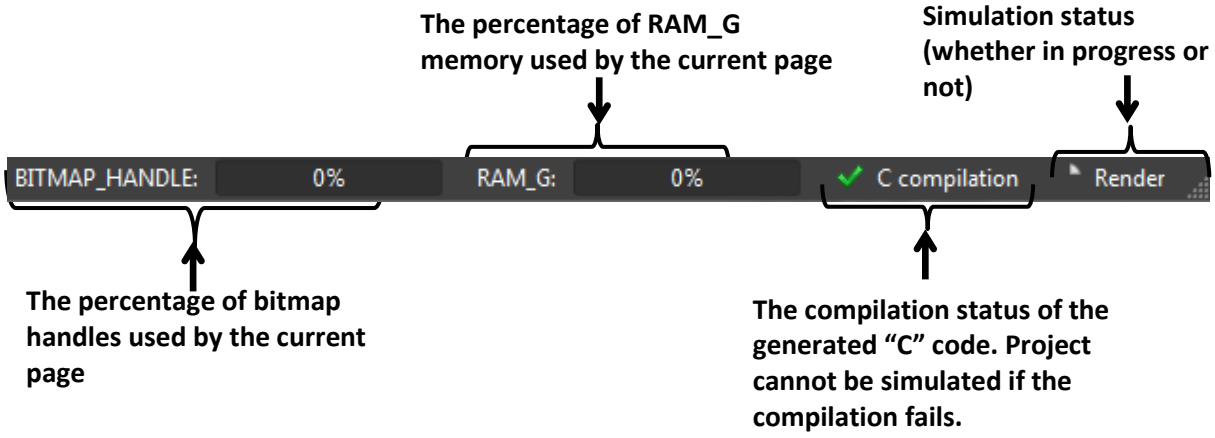


Figure 29 - Status bar

Application Project Structure

Copyright © Bridgetek Pte Ltd

An Application consists of one or more pages. Each page in turn may contain one or more widgets. Widgets and pages are connected via connection lines so that the screen logic is defined.

The application model follows a strict hierarchical structure where all the pages are owned and managed by the application, and all the widgets are owned by their respective pages. The visibility and memory lifetime of the widgets are thus managed on a page-by-page basis.

By design, all widgets and pages are generated to be entirely modular and self-sufficient. Any interaction between the widgets and pages is therefore required to be routed through the hierarchical chain. (Only node connections exist between the directly connected nodes in the hierarchy.)

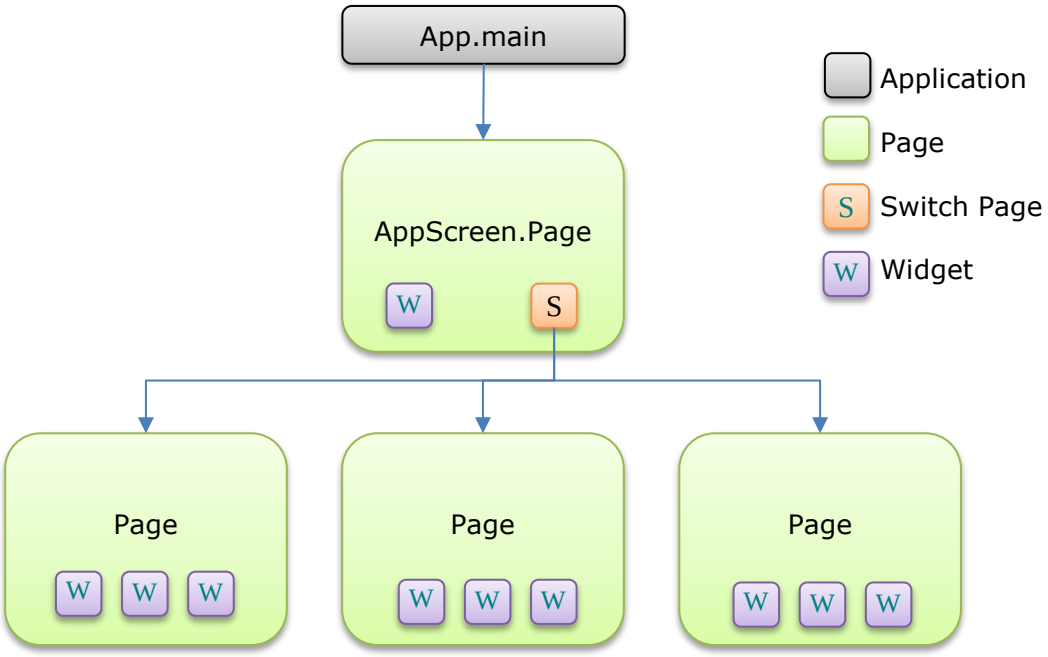


Figure 30 - Application Project Structure

File Name	Description
*.esd	Project file - This XML file describes what files are included in this project. This file is unique for the whole project.
*.main	Logic file - This XML file describes the application level logic. By default, it is named as "App.main" and is unique project wide.
*.page	Page Node Definition file - This XML file describes what widgets are contained and connected in each page. By default: "AppScreen.Page" is added, where a switch page may be included for control on page transition. A page can contain other pages, but switch page is required for page transition control.
*.widget	Widget Node Definition file. This XML file describes a widget detail. A widget can contain other nodes, which may include widget node, actor node and logic node.
*.actor	Actor Node Definition file. This XML file describes the actor node details.
*.logic	Logic Object Definition file. This XML file describes the logic node details.
*.theme	Application Theme Definition file. This XML file describes the theme detail.
*.h	C language source header file
*.c	C language source body file

Copyright © Bridgetek Pte Ltd

*.png, *.jpg, *.jpeg	Image resource files

Note: The file names listed above shall **NOT** be conflicting with the C standard library function as well as windows header files. Here are some examples: *fopen, windows, winbase, math, printf, realloc.* ESD 4.10 simulation process may give errors and fail, if user gives the file name of their widgets, logics, images, or any resources with these reserved names. Therefore, ESD 4.10 enforces the 'Usr_' prefix to all the new/renamed widgets, pages, functions, actors, as well as source file, theme, logic source file etc, in order to ensure that the simulation process succeeds without error.

Application Workflow

The workflow given below illustrates how to create an EVE based GUI application using ESD 4.10.

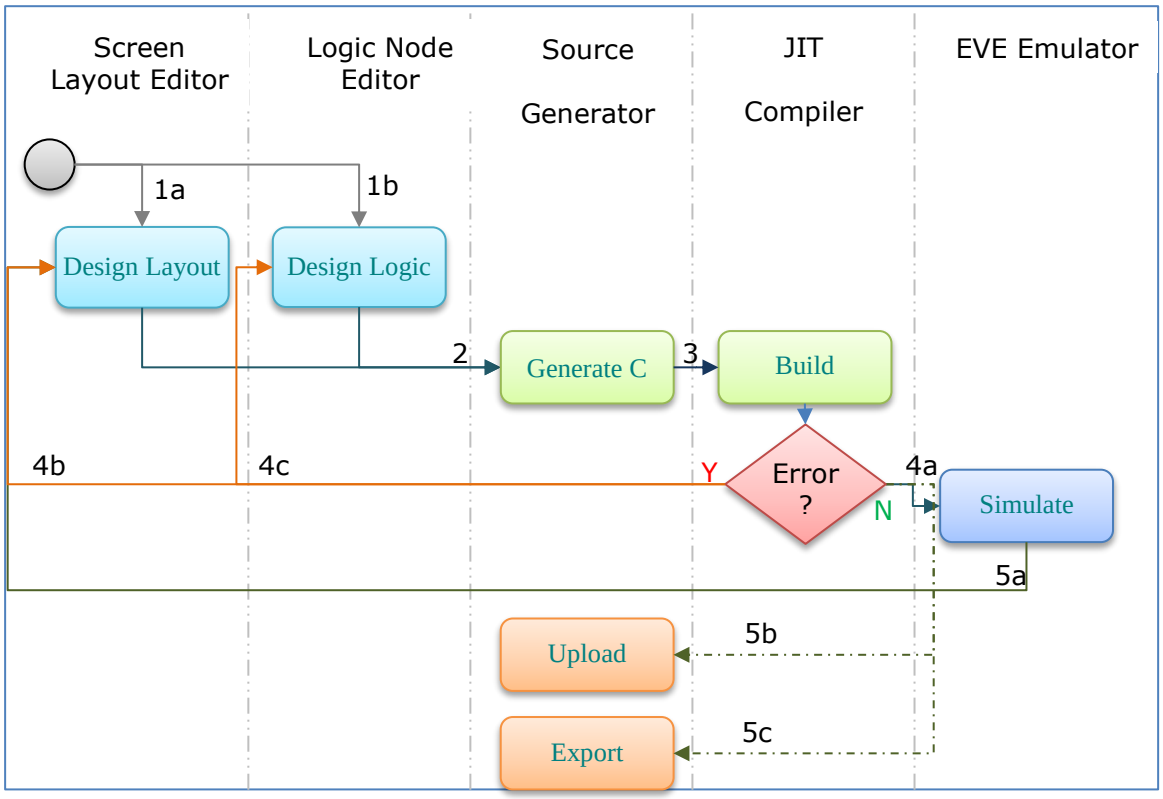


Figure 31 - Application Workflow

The 5 main modules that comprise the application workflow are the *Screen Layout Editor, Logic Node Editor, Source Generator, JIT Compiler* and *EVE Emulator*. Among these, *Source Generator*, *JIT Compiler* and *EVE Emulator* are the back-end modules which have no UI exposed to end user. The compiler is a JIT style C compiler, i.e. TinyCC compiler, which compiles and links the generated code into executable. The executable is invoked by the emulator module for simulation.

The application workflow starts by design layout (1a) or design logic (1b); Upon user saving the changes, it will trigger C code generation (2), then the compiler will try to build the generated c code; if any error encountered during building, then user needs to

Copyright © Bridgetek Pte Ltd

continue on design layout (4b) or design logic (4c); or if no error found, then the emulator will start simulating (4a) the project. Users will be able to upload (5b) the binary to the EVE chips or export (5c) the project as eclipse C project.

Design Layout using Layout Editor

Users need to determine how many pages are included in the project and what widgets are to be contained in each page. After a page is created, users can drag and drop the widgets to design the appearance of the page in the screen layout editor. This process determines how many screens are shown in the project and how they look like together.

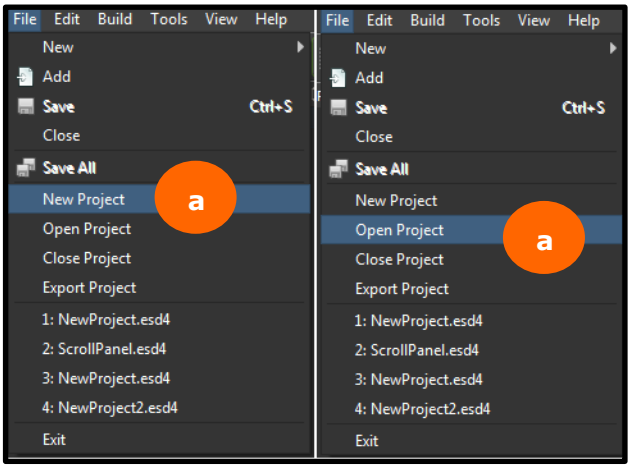
Custom Widget

ESD 4.10 provides a set of built-in widgets for users to construct screen layouts. Additionally, if the built-in widgets does not meet the design requirement, users may define and design custom widgets.

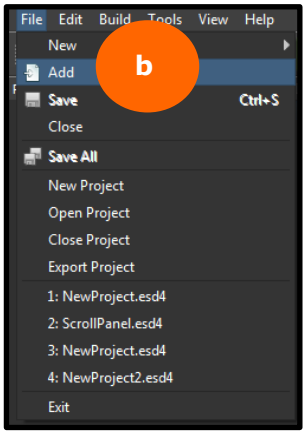
Add Bitmap Resource

To display a bitmap on the page, users need to add a bitmap resource into the project. The following steps will provide guidance on how to add a bitmap resource:

a. Create a New Project (***File → New Project***) or open an existing Project (***File → Open Project***).

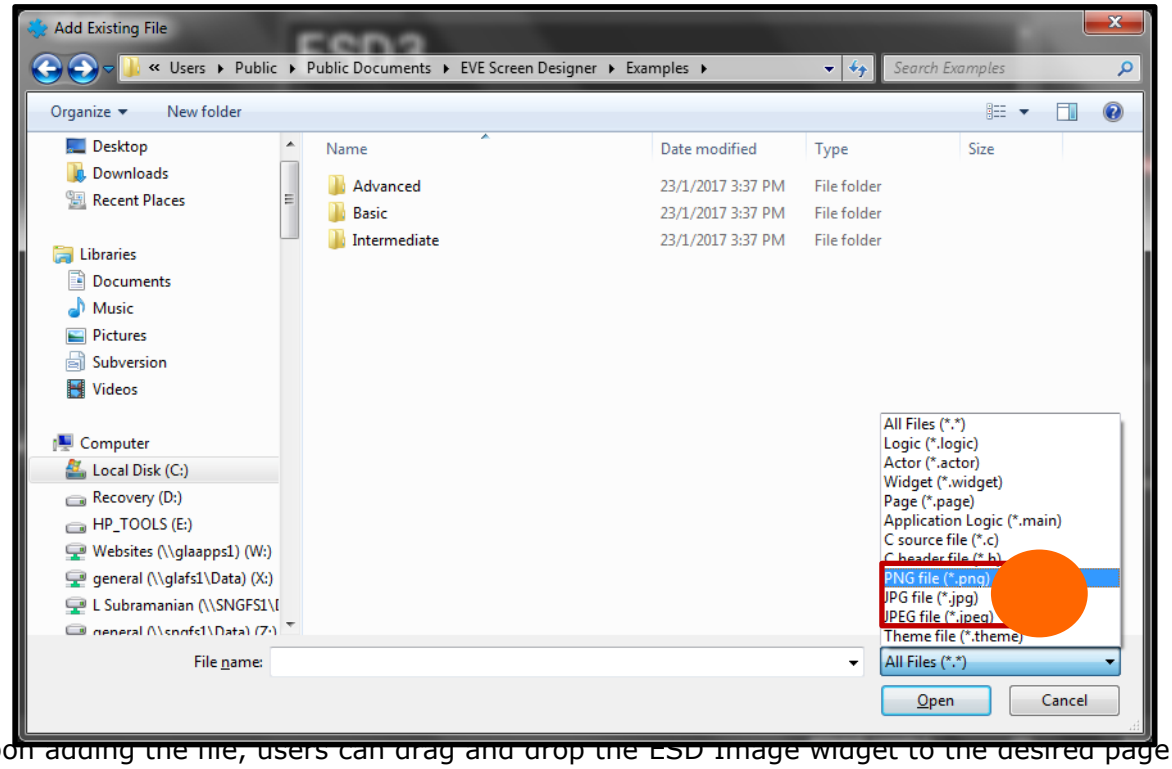


b. Click **File → Add** from the menu.



c. Browse and select the image file(s) from the file explorer window.

Copyright © Bridgetek Pte Ltd



Upon adding the file, users can drag and drop the ESD Image widget to the desired page and select the bitmap cell (which is default at cell 0) accordingly.

Configure Bitmap Resource

Users can configure a bitmap resource via the *Property Editor*. Select the image file from the project browser and check the Property Editor.

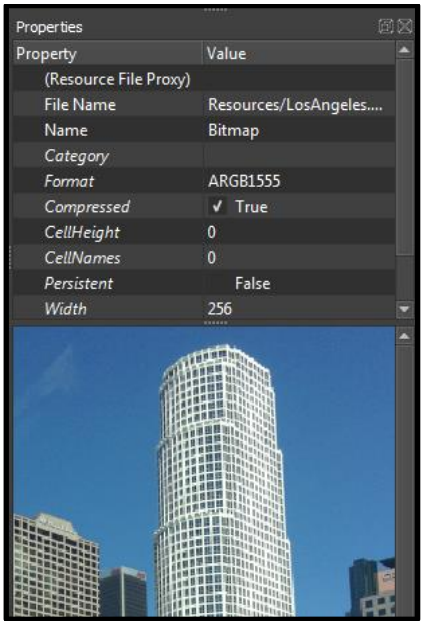


Figure 32 - Property Editor

Copyright © Bridgetek Pte Ltd

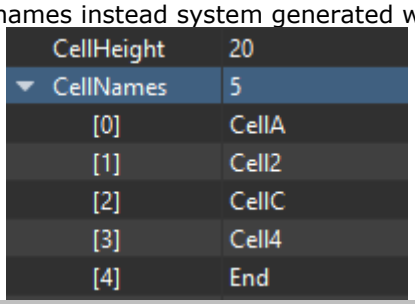
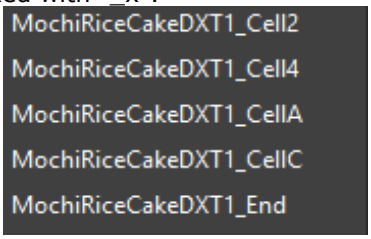
Property	Value/Description
File Name	Name of the image file
Name	Resource Type Name. In this case, the selected resource type is Bitmap.
Category	Category of the selected object
Format	Image Format – The target format of bitmap can be – • L1/L2/L4/L8 • RGB332/RGB565 • ARGB1555/ARGB2/ARGB4 • PALETTED4444/PALETTED565/PALETTED8 • DXT1 • JPEG (Decoded into RGB565 format only) • PNG (Decoded into RGB565 format only) • COMPRESSED_RGBA_ASTC_4x4_KHR • COMPRESSED_RGBA_ASTC_5x4_KHR • COMPRESSED_RGBA_ASTC_5x5_KHR • COMPRESSED_RGBA_ASTC_6x5_KHR • COMPRESSED_RGBA_ASTC_6x6_KHR • COMPRESSED_RGBA_ASTC_8x5_KHR • COMPRESSED_RGBA_ASTC_8x6_KHR • COMPRESSED_RGBA_ASTC_8x8_KHR • COMPRESSED_RGBA_ASTC_10x5_KHR • COMPRESSED_RGBA_ASTC_10x6_KHR • COMPRESSED_RGBA_ASTC_10x8_KHR • COMPRESSED_RGBA_ASTC_10x10_KHR • COMPRESSED_RGBA_ASTC_12x10_KHR • COMPRESSED_RGBA_ASTC_12x12_KHR
Compressed	Set bitmap compression on/off
CellHeight	The pixel number of one cell. For example, if the bitmap width is 240 and height is 240, then it can be defined as two cells with "CellHeight" - 120 or four cells with "CellHeight" – 60. Cell Height default at 0 which means there will be only one cell no matter how large the bitmap it is.
CellNames	This field will enable system to create a list of n number of user defined cell names instead system generated which is suffixed with "_x".  
Persistent	Keeps the memory persistent even if the bitmap is not displayed when it is true.
Width	The width of bitmap image in pixels
Height	The height of bitmap image in pixels

Table 5 - Bitmap Resource Properties

Design Screen Logic using Logic Node Editor

Users can define the dynamic behaviour of an application in this stage. The Logic Node Editor employs the innovative visual programming idea to help users define logic flows without coding.

Copyright © Bridgetek Pte Ltd

During this stage, users can decide the behaviour mode of both inner-page and inter-pages. For inner-page behaviour, users can connect the widgets on the page via the logic node editor adding more logic nodes from the library browser, if necessary.

Inter-page logic behaviour is captured in the application logic, which is normally named as "AppScreen.page". Users are required to drag and drop the predefined pages into the logic node editor and connect these pages via the logic interface. This will define the inter-pages behaviour like screen logic.

Simulation

To validate the behaviour of the screen logic, users may need to simulate the project on the PC before compiling and downloading the generated C code into the target device. When application logic (AppScreen.page or App.main) is selected, users can simulate the whole project by clicking **[Simulation]** button on the toolbar. When the application is in simulation mode, users cannot drag and drop the widgets into the layout editor. The PC mouse will act as a touch stylus on the touch panel to interact with the application directly.

To validate the behaviour of single page, users need select the page file and open it with logic node editor before clicking the **[Simulation]** button on the toolbar. Similarly, users can simulate the behaviour of other logic node, such as widget, actor by opening it with logic node editor and clicking **[Simulation]** button on the toolbar.

Before performing a simulation, it is strongly recommended to save the current project by clicking **[Save all]** button. Once the simulation is completed, ensure to switch off the simulation mode by clicking the **[Simulation]** button again.

Simulation State	Description
	ON State - Indicates that the simulation is in progress. Clicking on it will switch off the simulation. In this state, users will not be able to edit the page or project.
	OFF State - Indicates that currently simulation is not in progress. Clicking on it will switch on the simulation.

Target selection

ESD 4.10 provides support for the following built-in build targets -

1. ME812A WH50R
2. ME810A HV35R
3. ME813AU WH50C
4. VM816CU50A
5. VM816C50A
6. ME817EV-WH10C
7. ME817EV-WH70C

These build targets typically come with standard configuration which the user can modify within certain limits as per the requirement.

ESD 4.10 also provides a mechanism to create new custom targets for new ESD projects. For example, if the user wants to create a new build target "PC_BT815EMU_WVGA" with PC as Host platform, BT815 as EVE platform, Screen Resolution as WVGA (800 X 480) and Flash set to 16MB MX25L16. Refer to the screenshots given below –

Copyright © Bridgetek Pte Ltd

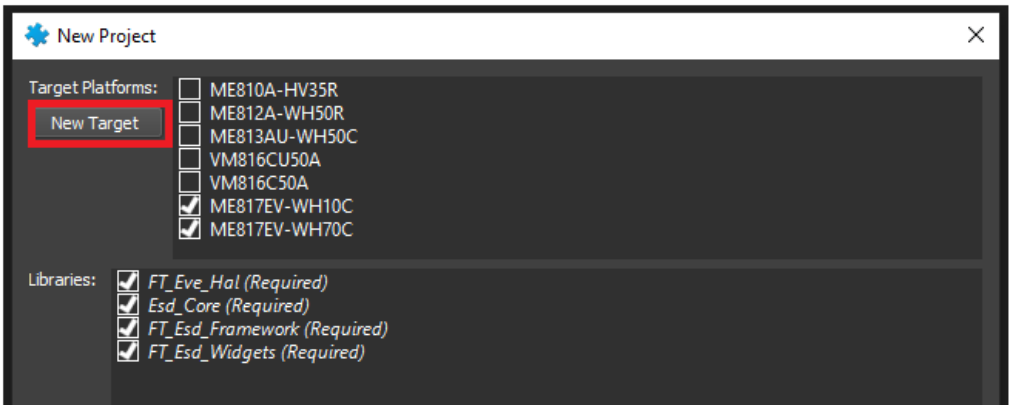


Figure 33 : New Target button

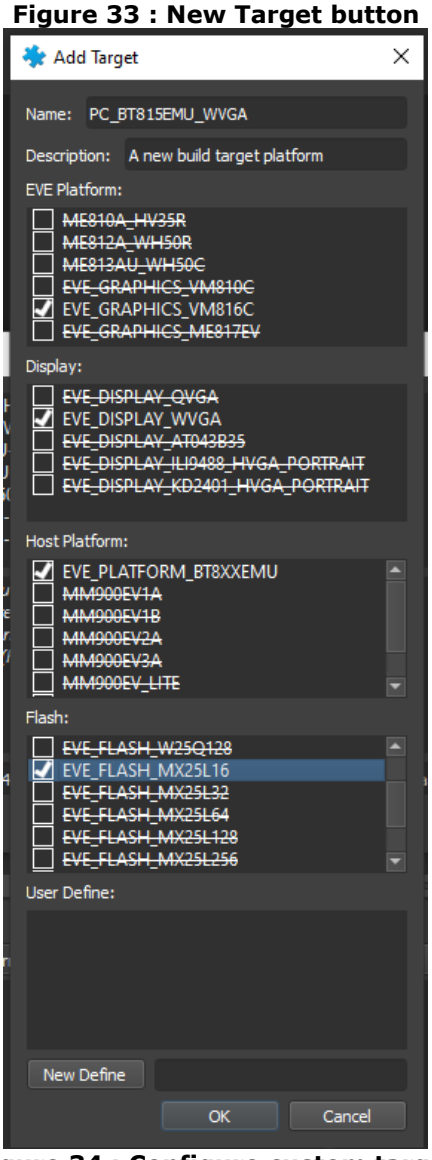


Figure 34 : Configure custom target

The newly created target can then be set via build target. Refer to the screenshot given below:

Copyright © Bridgetek Pte Ltd

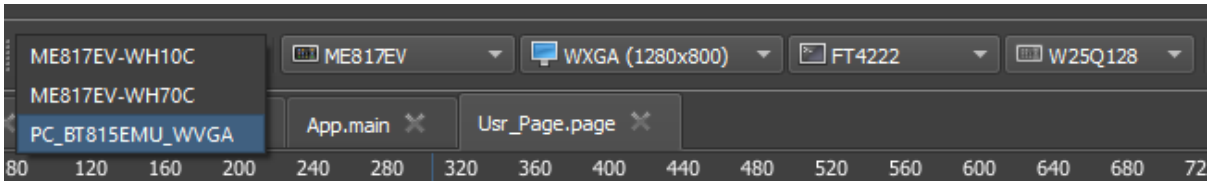


Figure 35 : Select custom build target

Build & Upload

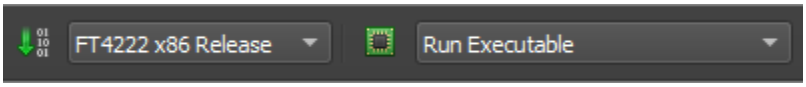


Figure 36 - Build & Upload

"**Build Executable**" button builds a project.

"**Build and Upload to Hardware**" button for a FT90X toolchain will invoke the compiler and programmer, so that the current project can be build and uploaded into the target hardware. If user selects x64 toolchain, this button will build and generate an '.exe' application which can be run on a PC as host environment.

In ESD 4.10, new build targets "ME817-WH10C" and "ME817-WH70C" have been added to the build target list. The respective output files generated by "Build Executable" for different build configuration and host platforms are shown in Table 6.

Build Targets	Host Platform	Output files
ME812A WH50R	FT90x toolchain	*.bin; *.elf
	X64 toolchain (FT4222, MPSSE)	*.exe
ME810A HV35R	FT90x toolchain	*.bin; *.elf
	X64 toolchain (FT4222, MPSSE)	*.exe
ME813AU WH50C	FT90x toolchain	*.bin; *.elf
	X64 toolchain (FT4222, MPSSE)	*.exe
VM816CU50A	FT90x toolchain	*.bin; *.elf
	X64 toolchain (FT4222, MPSSE)	*.exe
VM816C50A	FT90x toolchain	*.bin; *.elf
	X64 toolchain (FT4222, MPSSE)	*.exe

Copyright © Bridgetek Pte Ltd

ME817EV-WH10C	FT90x toolchain	*.bin; *.elf
	X64 toolchain (FT4222, MPSSE)	*.exe
ME817EV-WH70C	FT90x toolchain	*.bin; *.elf
	X64 toolchain (FT4222, MPSSE)	*.exe

Table 6 - Output files for different Build Configuration & Host Platforms

Export

Upon completing the screen design, it's time to move the project to the next phase of the workflow – export project. There are two ways of exporting project. They are **"Export as Eclipse Project" and "Export as MSVC Project"**. This function is used to prepare all the necessary files for the MCU tool chain so that users can make further enhancements such as connecting sensors or external hardware to a full HMI solution.

On exporting, ESD (4.10) copies the generated "C" source code, the necessary bitmap resources and the ESD 4.10 application framework code into a user defined folder. In ESD, all the generated files are named as "*_generated.c". For eclipse export projects, ".cproject" and ".project" files are generated so that the users can open the project in FT90X eclipse IDE[2]. For MSVC export projects, "Project" folder is generated containing the MSVC compatible project. Table 7 gives more information on the Build Targets, Host platform and the corresponding export projects that can be created.

Build Target	Host Platform	Export Project
ME812A WH50R	Emulator, FT4222, MPSSE	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
ME810A HV35R	Emulator, FT4222, MPSSE	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
ME813AU WH50C	Emulator, FT4222, MPSSE	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	Eclipse Project
VM816CU50A	Emulator, FT4222, MPSSE	MSVC Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A,	Eclipse Project

[2] http://brtchip.com/ft9xx-toolchain/

Copyright © Bridgetek Pte Ltd

	MM900EV-Lite	
VM816C50A	Emulator, FT4222, MPSSE	Eclipse Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	MSVC Project
ME817EV-WH10C	Emulator, FT4222, MPSSE	Eclipse Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	MSVC Project
ME817EV-WH70C	Emulator, FT4222, MPSSE	Eclipse Project
	MM900EV1A, MM900EV1B, MM900EV2A, MM900EV3A, MM900EV-Lite	MSVC Project

Table 7 - Build Targets, Host platform and the corresponding export projects

Exported Folder Structure

Table 8 provides the list of folders and files that are created upon exporting the project:

Folder / Files	Description
	(Common export files)
Esd_Core	Contains the application core files. **It is recommended not to make any changes to this folder/ files**
FT_Esd_Framework	Contains the application framework files. **It is recommended not to make any changes to this folder/ files**
FT_Esd_Widgets	Contains the widget files **It is recommended not to make any changes to this folder/ files**
FT_Esd_Hal	Contains the hardware abstraction layer files. These files may be changed to support a different MCU.
Data	Contains the converted bitmap resource data and the flash data which is used by the current project
$(ProjectName)	Contains the application specific source code
	(Eclipse export files)
.cproject (for Eclipse export)	Eclipse project file
.project (for Eclipse export)	Eclipse project file
ThirdPartyLib	Contains third party files such fat filesystem library handler files
	(MSVC export files)
Project/$(ProjectName)_MSVC/ $(ProjectName)_MSVC.sln	MSVC project solution file
Project/$(ProjectName)_MSVC/ $(ProjectName)_MSVC /$(ProjectName)_MSVC.vcxproj	MSVC visual studio C++ project file

Table 8 – List of Folder & File created upon exporting project

Copyright © Bridgetek Pte Ltd

Bitmap Resource

The bitmap resources used in an ESD 4.10 project can be saved in a SD card or to flash based on the selection of flash flag configuration in the ESD 4.10.

SD card

If Bitmap resources meant for SD card are used in a project, it is converted to *.astc (for ASTC file format)/*.raw(for other file formats) or *.bin file format (if compressed flag is selected). Users are required to download the converted bitmap resource into an SD card **root** directory and insert the SD card into the hardware module.

The converted bitmap resource is located at:

$(ExportFolder)\data

The bitmap resource file with .bin extension is called to decompress using the EVE engine command *CMD_INFLATE*.

As per the Hardware Platform requirements, the SD card must be formatted as a FAT32 file system.

Flash

If bitmap resources meant for flash are used in a project, it is converted to a Flash.bin file. Users are required to download the converted bitmap resource into flash memory of the hardware module.

The converted bitmap resource is located at:

$(ExportFolder)\data

The bitmap resource file with .bin extension is called to decompress using the EVE engine command *CMD_INFLATE*.

Layout Editor

The *Layout Editor* allows users to preview a single page as well as the whole project. ESD 4.10 employs the EVE emulator to render an EVE display onto the layout editor. It provides users, the interface to view the result of the screen design and its logic.

The following file types are displayed in ESD layout editor:

- *Application logic file* - generally with ".main" extension
- *Page file* - generally with ".page" extension

Page File

Page stands for a single screen which will be rendered. One application consists of at least one page. Page may have its own life cycle and manages all the widgets on it. Only widgets can be dragged and dropped into pages at Page layout editor. However, all the nodes from a library can be dragged and dropped into the logic node editor.

Copyright © Bridgetek Pte Ltd

"Active" Property

The *"Active" property* is a boolean value to control the page, and is created or released while the application is updating the pages. By default, the "Active" property is True. The page is created and shown, unless "Switch" end is connected.

Switch Page

Users can define multiple pages within the project. *"Switch Page"* ESD layout should add to manage the pages. User can drag and drop the pages into this switch page from project browser. The picture below displays how the logic editor view looks like for the switch page logic.

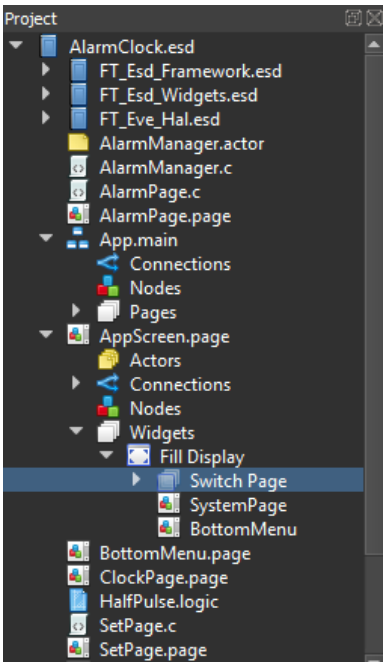


Figure 37 – Switch Page

Copyright © Bridgetek Pte Ltd

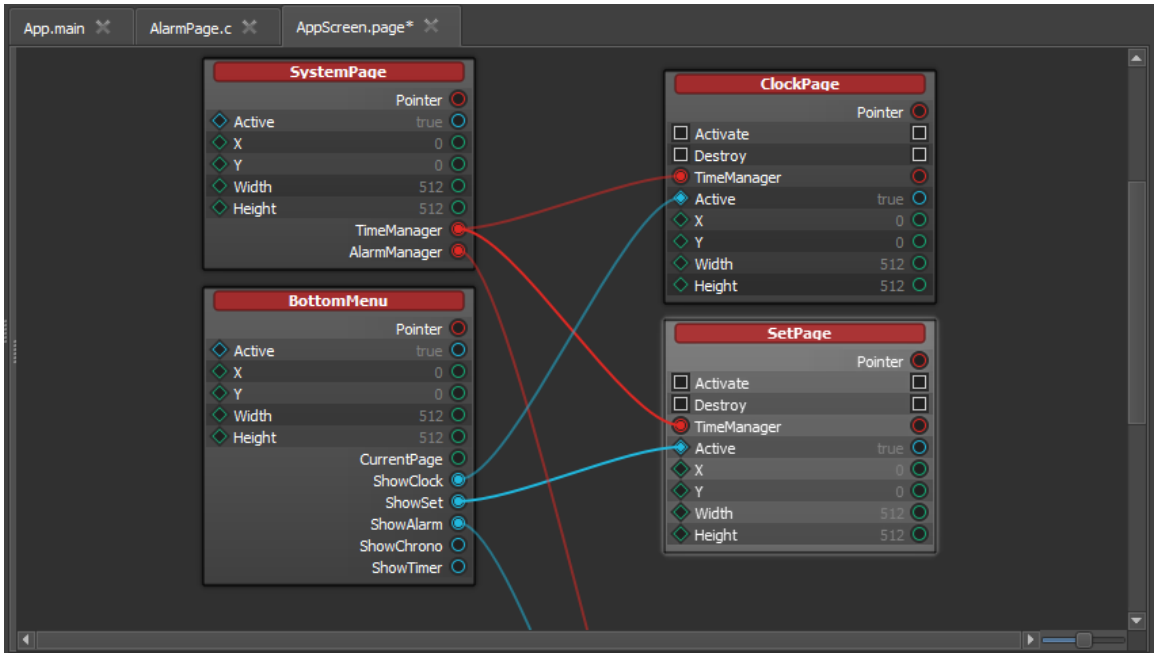


Figure 38 - Sample Switch Page UI

Page Persistence

To allow a page to remain in memory with its current state even when it is made inactive, set the allocation mode to "Lazy Persistent" or "Static" from the application logic in the page's Properties. This feature can help users to reserve the information of a page even when it is not visible.

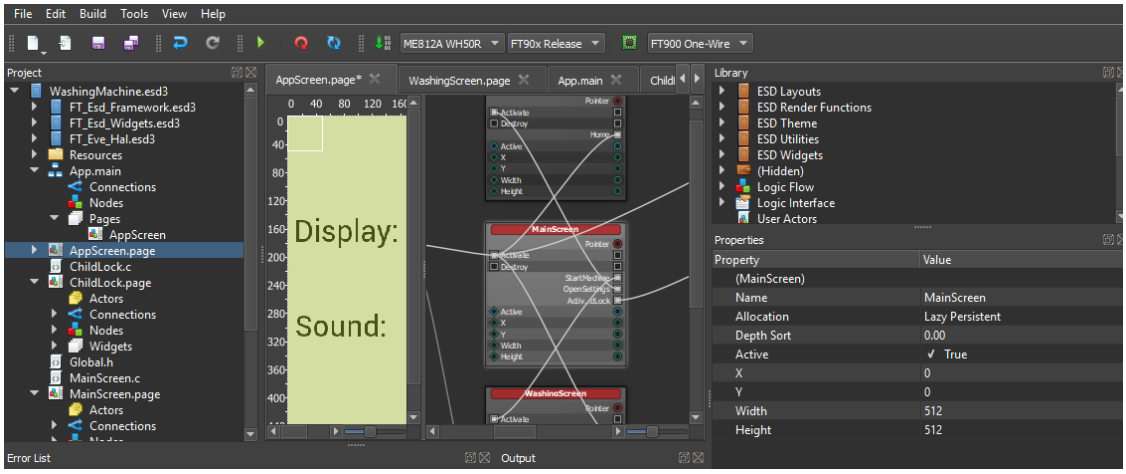


Figure 39 - Properties Editor – Page Allocation

The page allocation modes supported in ESD 4.10 are listed in the table given below:

Allocation Mode	Description
Static	Page is always persistent in memory after application starts
Lazy	Page is created when it is in "active" state and destroyed when it is inactive
Lazy Persistent	Page is created when it is in "active" state but persistent in memory when it is "inactive" until application is closed.

Copyright © Bridgetek Pte Ltd

User Defined Function for Page File

Users can write their own code to handle the "Pushed" signal of the "ESD Push Button" widget. Users are required to select the "ESD Push Button" widget in the logic node editor and double click it. A function will be generated to replace the user defined function.

```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context = MainPage)
void MainPage_ESD_Push_Button_Pushed(MainPage *context)
{
        // Users can write their code here
}
```

The code can be located at ***$(PageFileName).c*** file.

MainPage.page* | App.main | MainPage.c

```
1  #include "Ft_Esd.h"
2  #include "MainPage.h"
3
4  ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context = MainPage)
5  void MainPage_ESD_Push_Button_Pushed(MainPage *context)
6  {
7      // …
8      printf("This push button is pushed\n");
9  }
10
```

Figure 40 - Sample Source File

Zoom In & Out

The visual area of the screen layout editor can be zoomed when the current file is a page file, i.e., "*.page*" file. The mouse wheel button provides *zoom-in* and *zoom-out* functionality, allowing users to view more details about the screen design. Refer to the sample picture given below:

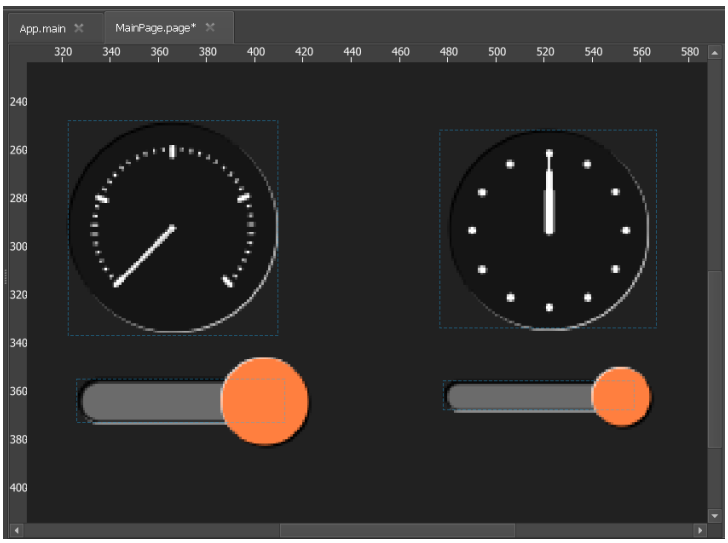


Figure 41 - Zoom In & Out

Copyright © Bridgetek Pte Ltd

Main File

To view the application logic file (*.main) file, double click the file within the project explorer. The screen layout editor will show the first page defined in the logic node editor. To view how the application logic is defined, click the **"Simulation"** button and check it in the screen layout editor.

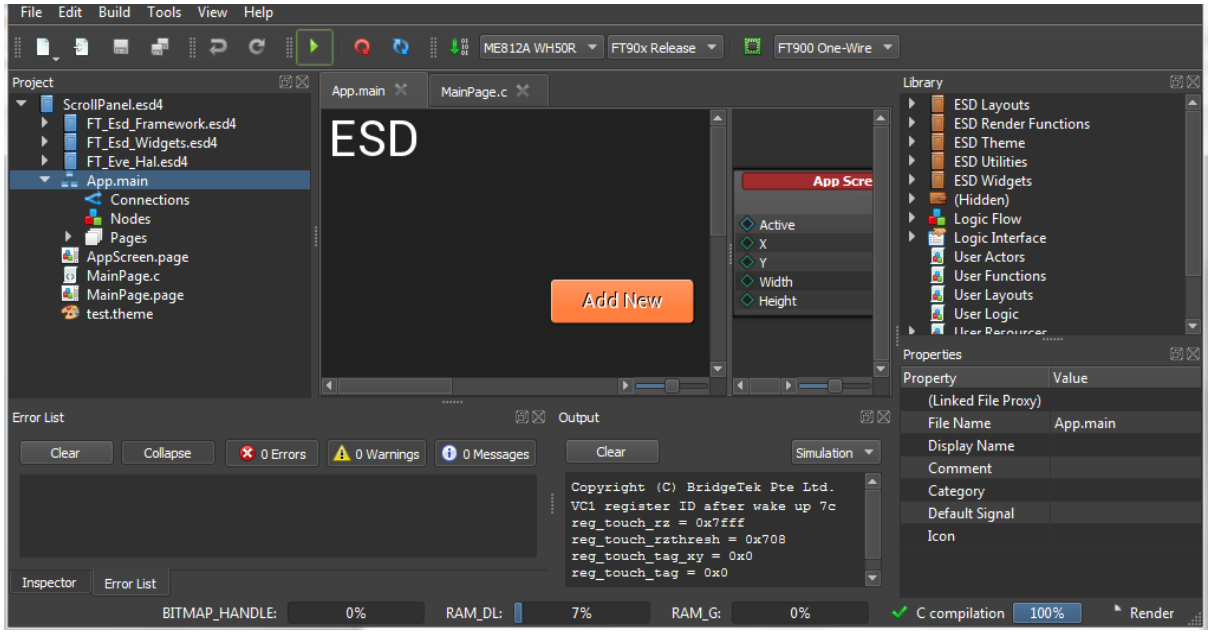


Figure 42 - Main File

Logic File

The logic file contains certain logic that is used to encapsulate the user defined logic. It works similar to a C function. A logic file does not render any user interface.

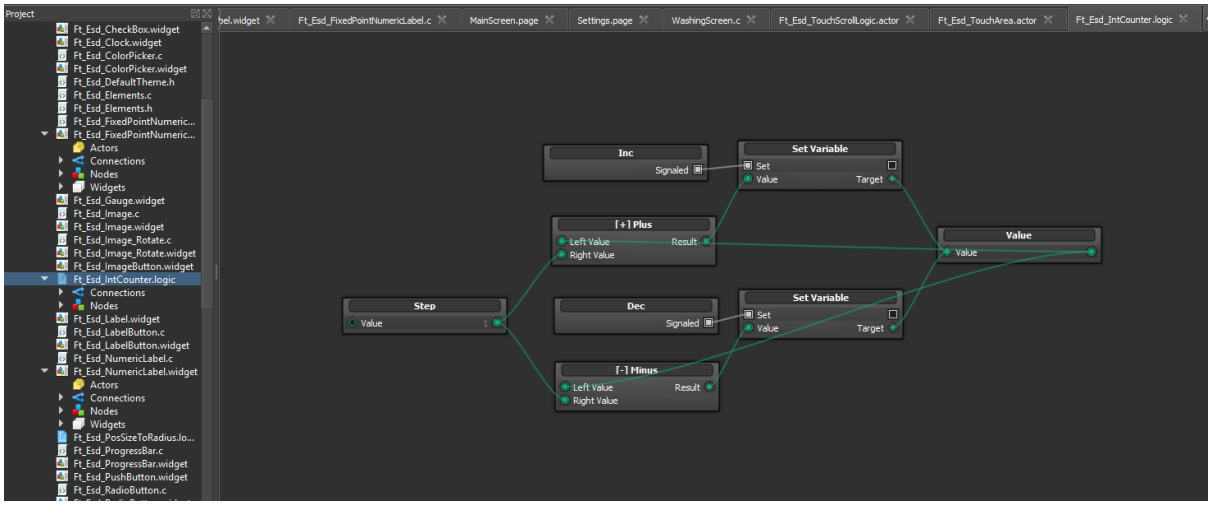


Figure 43 – Sample Logic File

Copyright © Bridgetek Pte Ltd

C File

The screen layout editor provides a simple C editor when users double click a ".c" file in the project browser. Note that it is not a full-fledged C editor and many features may be implemented in the future.

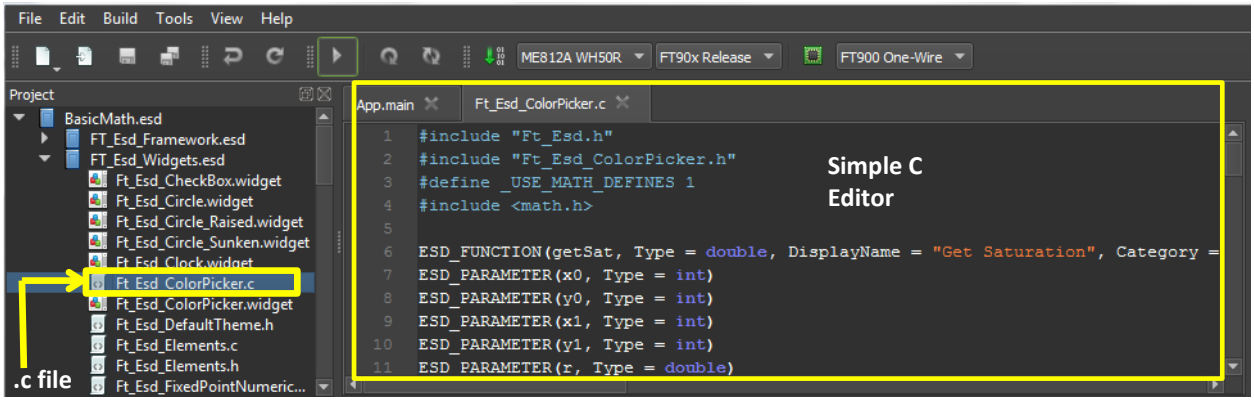


Figure 44 - C File / C Editor

To effect the changes made, ensure that all the changes are saved to the project by clicking **"File → Save All"** from the menu or from the toolbar.

Logic Note Editor

Using the *Logic Node Editor* (LNE), users compose screen logic by connecting logic nodes. The Logic Note Editor is designed to open the following file formats:

- Application logic file , *.main file
- Screen logic file, *.page file
- Widget file, *.widget
- Actor file, *.actor
- User defined logic file, *.logic file

Basic Logic Node

ESD 4.10 has some built-in predefined *basic logic nodes*. These basic logic nodes provide basic control flow and logic interfaces as well as basic functions. A typical basic logic node is represented in the logic node editor as below, the name of which is shown in white.

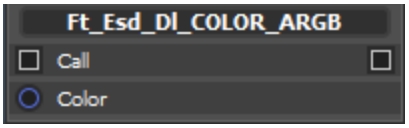


Figure 45 – Logic Node Editor - Basic Logic Node

Within the library browser, the basic logic nodes are located at *"Logic Flow"*, *"Logic Interface"*, *"ESD Utilities"* and *"EVE Render Functions"*.

Copyright © Bridgetek Pte Ltd

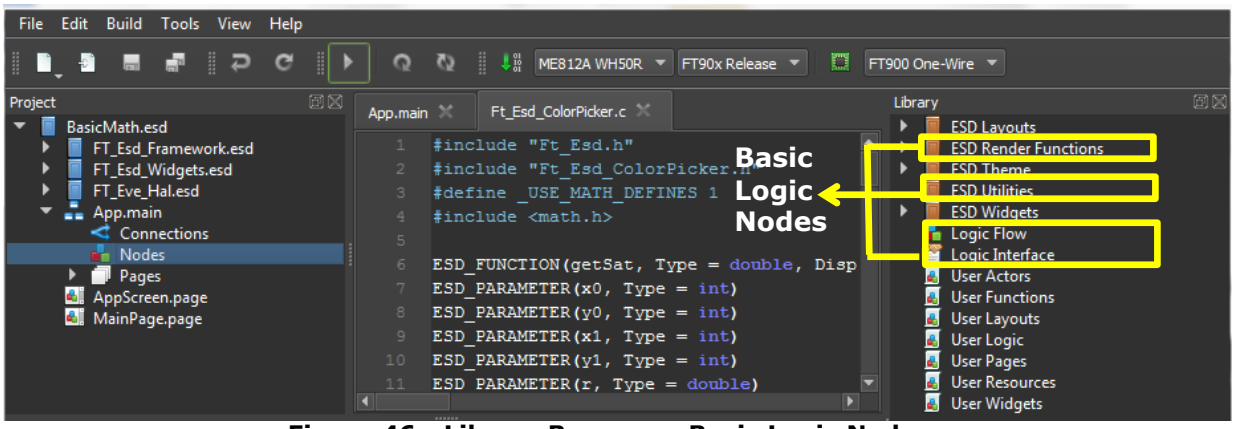


Figure 46 - Library Browser - Basic Logic Nodes

Composite Logic Node

A *Composite Logic Node* is made up by connecting multiple basic logic nodes. A typical composite logic node is defined in a standalone document in XML format and is shown in the logic node. Refer to the picture given below -

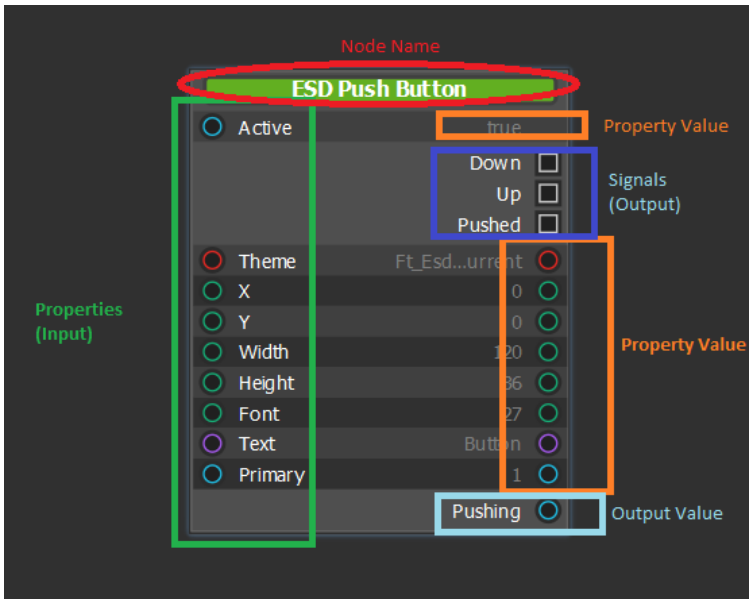


Figure 47 - Composite Logic Node

The following are the different types of composite logic nodes defined in ESD 4.10:

- *Page Node*
- *Widget Node*
- *Layout Widget*
- *Actor Node*
- *Logic Object*

Copyright © Bridgetek Pte Ltd

Page Node

When users create a single page, it is represented by a logic node under the "User Page" category in the library browser. Users can connect one page node to another within the logic node editor. It has its own property and defined output.

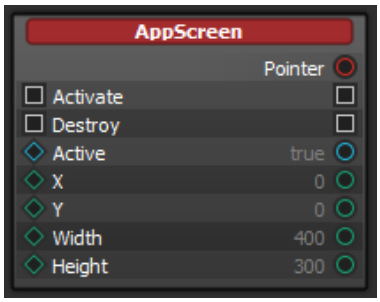


Figure 48 - Composite Logic Node - Page Node

Widget Node

The *Widget Node* is one type of logic node which has visual appearance and is able to handle user input. It has the built-in *Start*, *Update*, *Idle*, *Render* and *End slots*.

For more information on built-in slots, please refer to Built-in Slot.

Within the logic node editor, the widget name is highlighted with the green colour title as shown in the sample picture below -

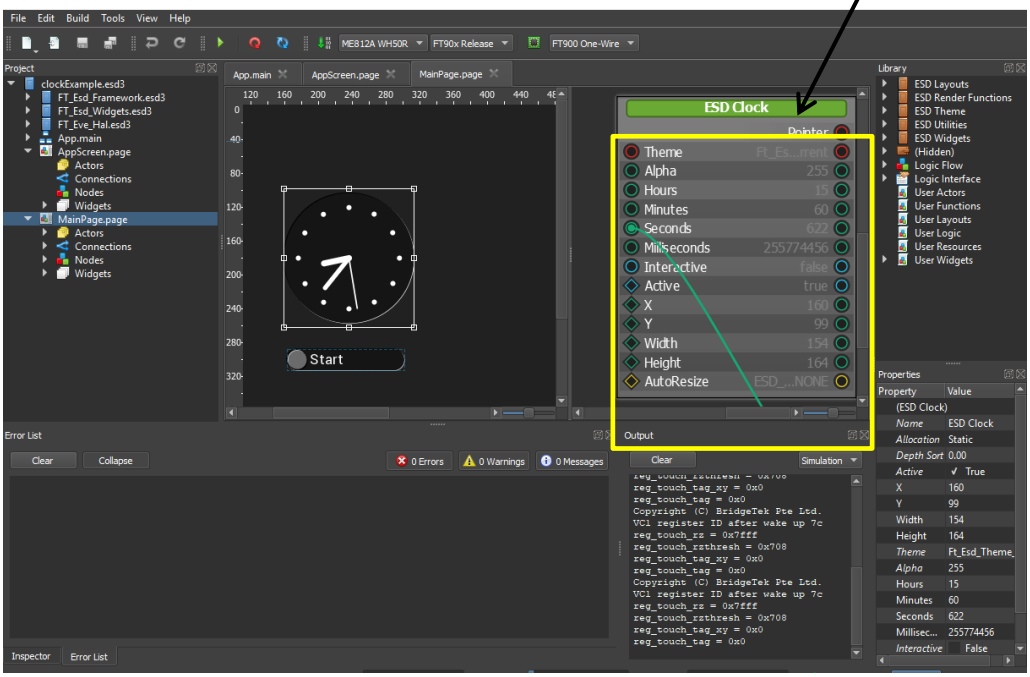


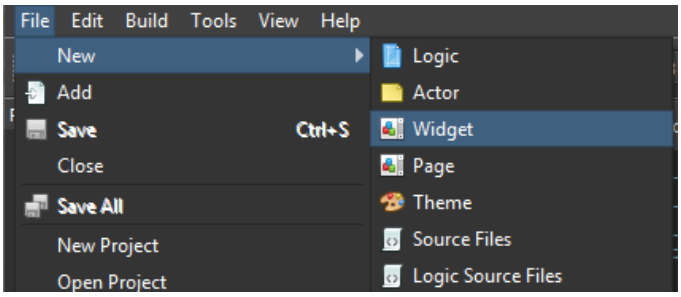
Figure 49 - Logic Node Editor – Widget Node

The highlighted sections provide information on adding user widgets; the position and size properties; rendering the widget and widget theme.

Copyright © Bridgetek Pte Ltd

Adding User Widgets

A *User Widget*, also called as *custom widget* is a reusable user interface element created by a user. To create a widget, from the menu, click and select **"File → New → Widget"**.



In order to make use of these slots, users need to drag the slots from the library browser and drop them into the logic node editor. By renaming it to the slot defined above, users can connect widget control flow to application control flow.

Position & Size Properties

As a user interface element, widgets have at least position and size properties so that users can control them using a visible handle. Position property is defined by an "Input" node with "ft_int16_t" type named as "X" or "Y". Size property is defined by an "Input" node with "ft_int16_t" type named as "Width" or "Height". Figure 50 provides a sample of the "X" property.

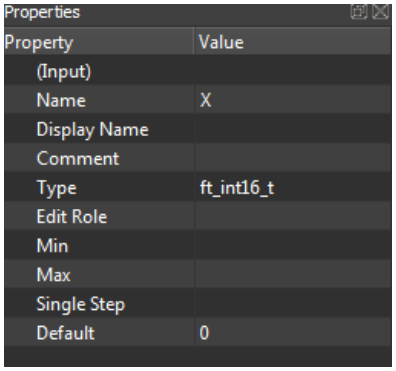


Figure 50 - User Widget - Position & Size Properties

Rendering a widget

Rendering of a widget may be implemented fully in the logic editor, or directly in the code using the logic editor to glue together the code with the generated structures. To create the render function, simply add a *Slot* from the Logic Interface category of the library browser, name it "Render" and double click to create the user method for handling the rendering. The widget is expected to restore any graphics state it changes, with the exception of the state values specified below.

Preferably use the Ft_Esd_Dl_ utility functions for display list drawing when available. Several state parameters which may be expected to be changed often to the same value implement de-duplication here to shorten the display list. The de-duplicated values are: *Palette source, Color RGB, Color A, Handle, Cell,* and *Vertex Format*. These graphic state values are not required to be restored to their original values when the Ft_Esd_Dl_ functions are used. These values should be expected to be any undefined value at the start of the Rendering function. Access the value of the X input property as given below:

```
int x = context->X(context->Owner);
```


Copyright © Bridgetek Pte Ltd

This calls the `get()` function assigned by the parent to the X property, passing the owner's context to the function to handle it there.

Theme

Widgets should provide property input to override the default application theme. An input property with type *Ft_Esd_Theme* * should be added by dragging one input node from the "Logic Interface" category of the library browser. Colours from the theme may be accessed from the logic editor using the utility functions under the ESD Theme category, or from the user code as given below:

```
Ft_Esd_Theme *theme = context->Theme(context->Owner);
if (theme == NULL) theme = Ft_Esd_Theme_GetCurrent();
ft_rgb32_t primaryColor = theme->PrimaryColor;
```

Touch Input

In order to handle the *touch input*, add a Touch Tag node from *"ESD Utilities"* category of the library browser to the logic editor, and either handle the signals or access its state through the Touch Tag interface. Users need to make use of the "Tag" value provided by the "Touch Tag" node during the Render function, and restore the "Tag" to value 0 after the relevant portion of the rendering.

To handle any of the signals from Touch Tag in user code, simply create an ESD_METHOD style function in the same format as the Render user code function, and it will be available from the User Functions category in the library to drag into the logic editor and connect to any of the signals from the Touch Tag node.

Layout Type Widget

Layout widget is a layout container for widgets. Unlike widget nodes which have visual appearance, a layout widget manages the layout specification for its child widgets. Layout widgets are introduced since ESD 4.0. Figure 51 & Figure 52 displays the various layout widgets supported in ESD 4.10.

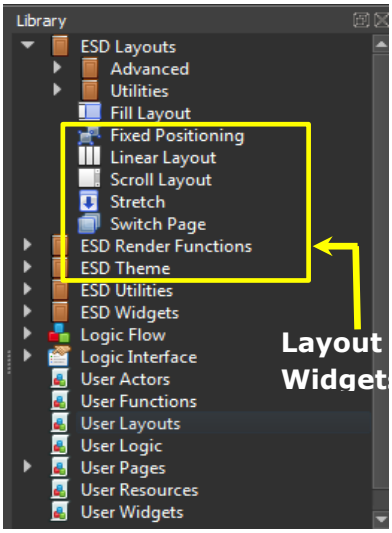


Figure 51 - Layout Widgets

Copyright © Bridgetek Pte Ltd

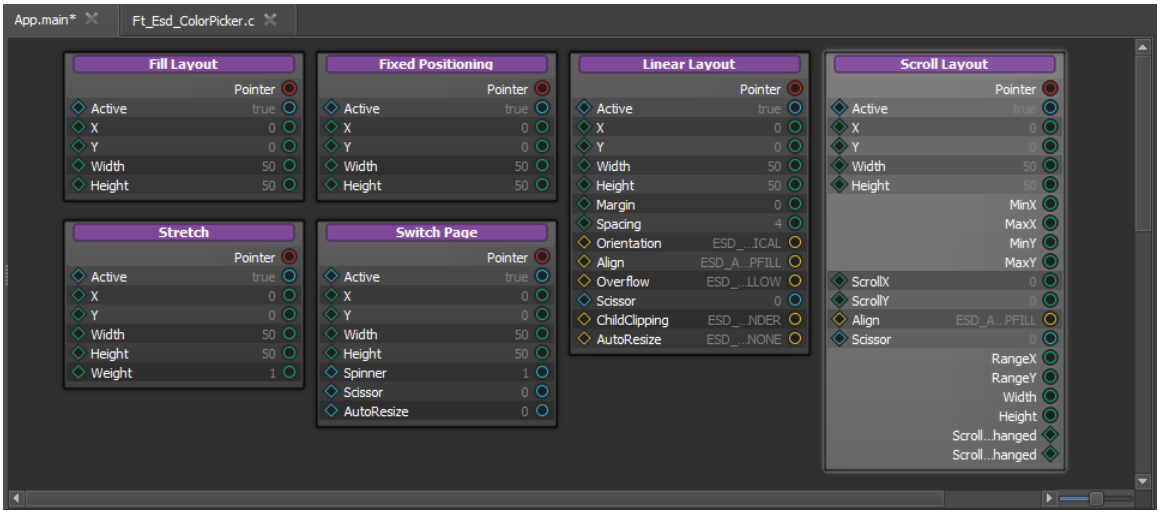


Figure 52 - Layout Widgets

Actor Node

Actor type is an extended type of logic node which functions similar to the widget node, but without the *Render* slot. Therefore, it has no visual appearance. When users create a new actor node, it does not show in the layout editor. It has the built-in *Start, Update, Idle* and *End* slots which are not available in a regular logic node. For more information about built-in slots, please refer to Built-in Slot.

An Actor node is useful for implementing both continuous and background running behaviours, such as animations, timers, and input data polling. The actor node name is highlighted with the yellow colour within the logic node editor.

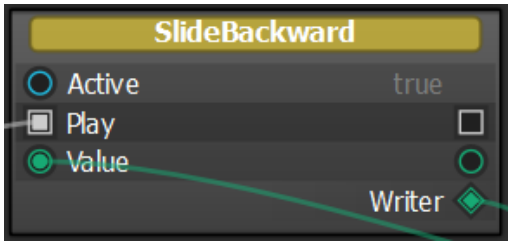
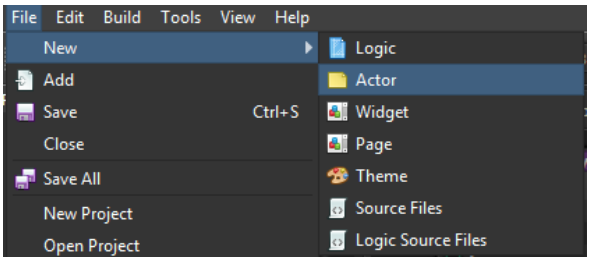


Figure 53 - Actor Node

To create an actor, from the menu, click and select **"File → New → Actor"**.



Refer to the example project **"ActorAnimator.esd"** under the **"EVE Screen Designer → Examples → Basic → ActorAnimators"** folder for adding and using actor node.

Copyright © Bridgetek Pte Ltd

Logic Object

A *Logic Object* is a basic logic block designed to express certain logic which has NO built-in slot. Hence, it is not possible to invoke it through the built-in render slot or update slot. However, it may have its own private or public property and output, signal and slot, depending on the users' implementation.

A logic object is generally used to simplify complex logic by splitting it into multiple smaller and simpler logic objects. It is defined and saved in a "*.logic" file.

Within the logic node editor, the logic object name is highlighted with the grey colour.

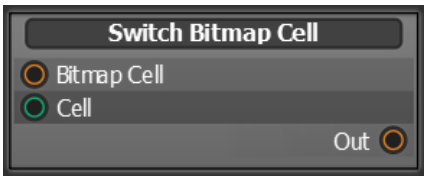


Figure 54 - Logic Object

To add a logic object, from the menu, click and select **"File → New → Logic"**.

Connections

Connections are lines between two nodes. The connection line's colour is determined by the ends. The following table provides the different types of ends and their description.

Connection Line End Type	Description
Event slots and Signals	Event slots and signals are implemented as functions which only take a pointer to the node context as a parameter. The node context contains the state of the node as well as a pointer to the parent node to signal back.
Data property bindings / /	Data property input and output bindings are implemented as get-functions which only take a pointer to the node context as a parameter and return a value. This means that these connections are completely evaluated every time the property is read, and consequently the property always reads as the latest value. A buffer variable can be used to cache values if necessary. The different colours denote the different data types.
Variable write	The write interface is translated into a function pointer during the code generation. Variables are plain variables, which can be written using the Set Variable node. They may also be set through the Writer interface which is implemented using a pointer to a set-function.

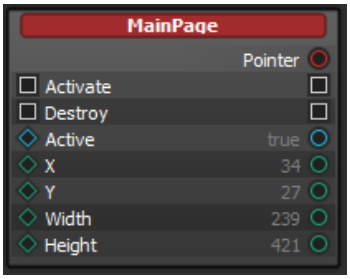
Table 9 - Connection End Types

In general, all of the input properties that can be connected can also be specified from the property editor when they are constant values; this is sometimes referred to as the "Default" value. In addition, it is possible to specify the Minimum and Maximum values for the input bindings and variables.

Rendering Order

Copyright © Bridgetek Pte Ltd

In the application logic file "App.main", the pages added on the logic node editor are to be represented with the symbols shown in the sample picture.



When there are more than two pages, the order in which the pages were created determines the rendering order.

The following implicit rule of the logic node editor is worthy to be noted, since this rule applies to all the logic nodes.

"Depth Sort" is used to determine the rendering order of the logic nodes. The greater the value of "Depth Sort", the later the logic node is rendered and higher the layer in which it is shown. When logic nodes have the same "Depth Sort" value, the logic node's "X"/"Y" coordinate (within layout widget) determines the rendering order. The page node with the lowest value of the "X"/"Y" coordinate is rendered first (as lower depth value).

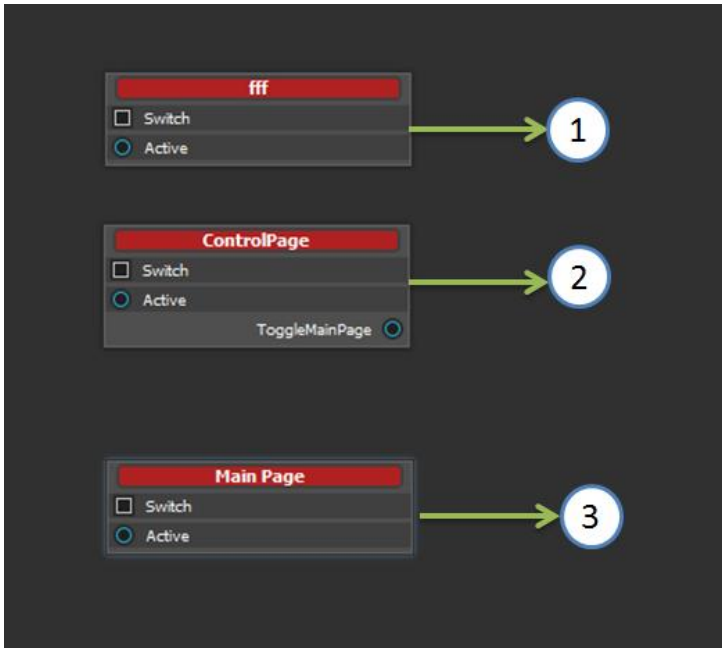


Figure 55 - Rendering Order Example

Logic Note Editor – User Interaction

The Logic Node Editor enables *Zoom-in* and *Zoom-out* functionality. Users can make use of the mouse wheel button to zoom in and zoom out.

"Panning" the logic node editor screen helps reveals the parts of the logic node editor which could not have been fit in a single screen. Panning can be done by 2 ways:

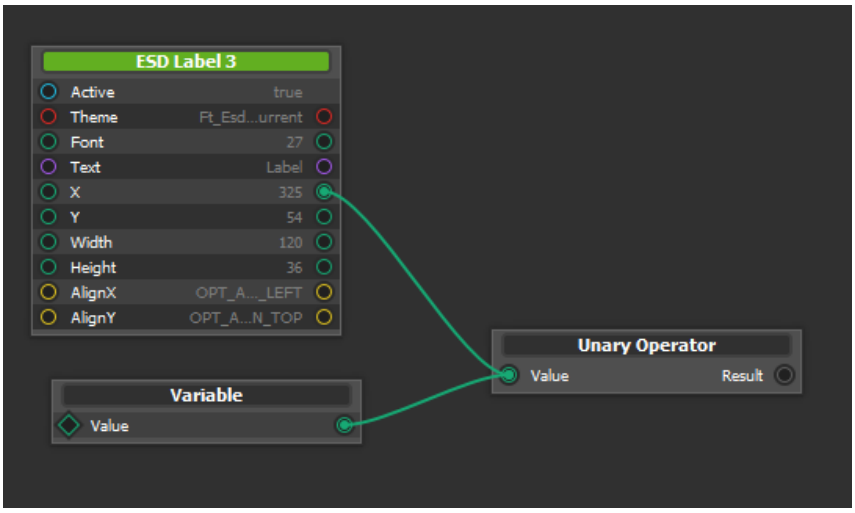
(i) Press *Alt Key + Left mouse button* and move the mouse
(ii) Press the *middle mouse* button and move the mouse.

Copyright © Bridgetek Pte Ltd

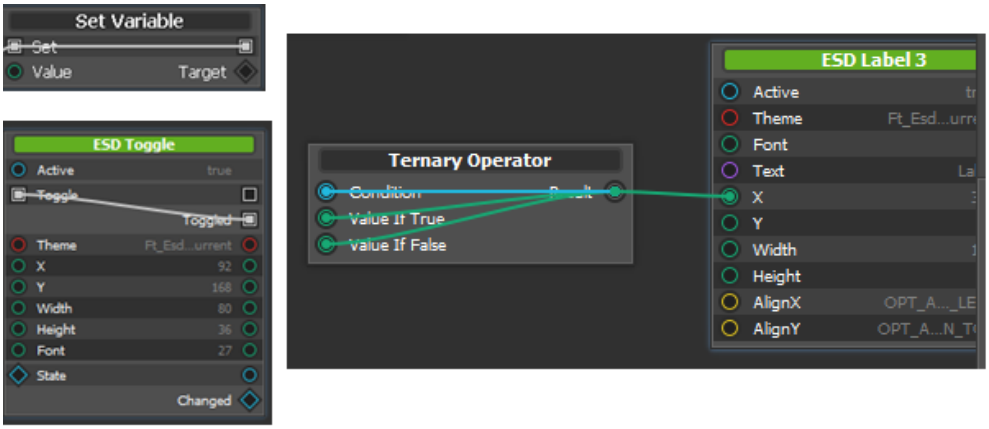
User can select multiple node items in the logic node editor by pressing Control key and selecting the desired objects by left mouse button click. Single/multiple objects can thus be moved if needed by moving the mouse while pressing the Left mouse button.

DON'TS

Avoid two connections connecting to the same port. Refer to the sample picture given below.



Avoid endless dead loops caused by connecting the input and output of the same object. It will cause an unexpected error. Refer to the sample pictures given below.



To effect the changes made to the logic note editor, ensure that all the changes are saved to the project, by clicking **"File → Save All"** from the menu or toolbar. In some cases, click the **"Recompile"** button to recompile the source code generated by ESD.

Copyright © Bridgetek Pte Ltd

Library Browser

The *Library Browser* consists of the basic components which are predefined and built in as part of ESD 4.10, as well as user defined components and/or resources. The components in the library include *Theme*, *Primitive, Widgets*, *Logic Flow* and *Logic Interface*. Users can select the appropriate components and drag them into the logic node editor or layout editor.

The library view depends on the currently selected node. The library browser will only show the contents which applies to the currently opened node.

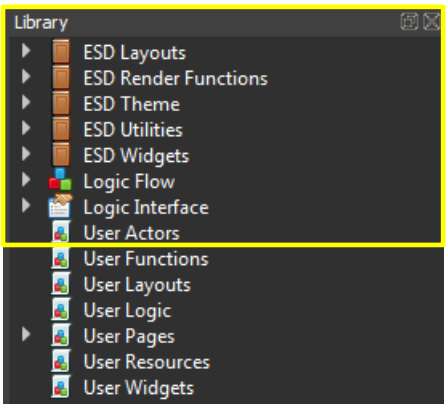


Figure 56 - Library Browser

In the Library Browser, items can be filtered by entering the search text in the textbox. Regular expression is supported. The matching text is highlighted as shown in the sample picture given below -

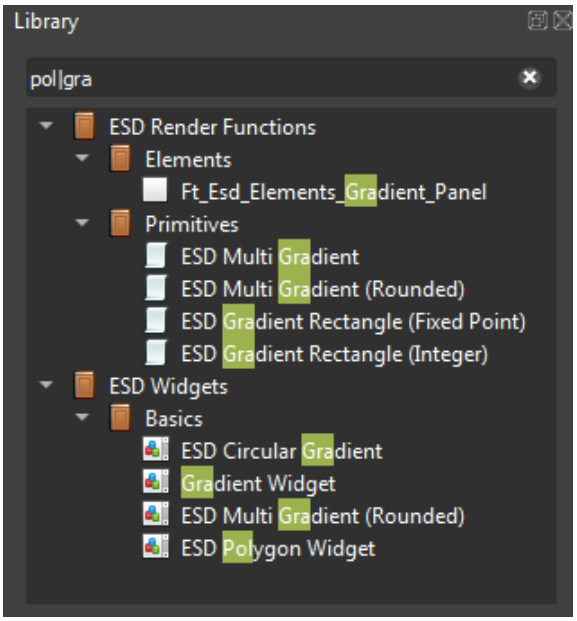


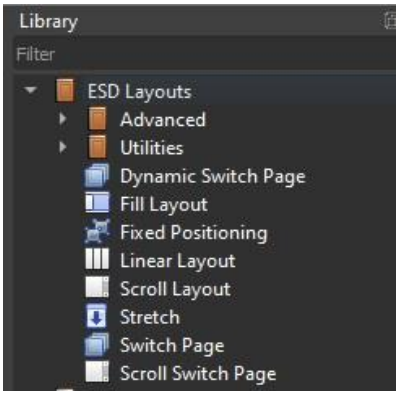
Figure 57 – Library Filter

The forthcoming sections provide information about the different components of the library browser.

Copyright © Bridgetek Pte Ltd

ESD Layouts

ESD 4.10 Layouts contains the collection of ESD 4.10 layout widgets and layout utilities functions.



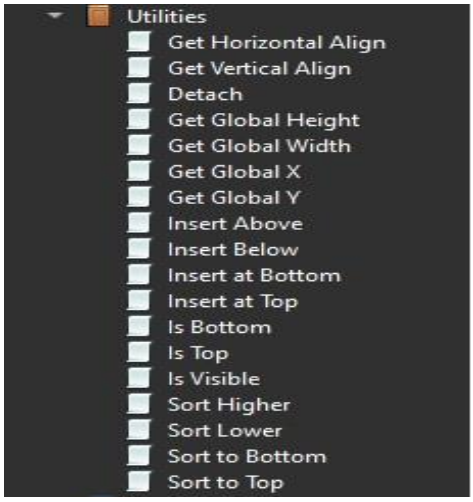
Layout	Description
Dynamic Switch Page	This Layout is used to achieve a fade-in/fade-out animation from one page to another page
Fill Layout	This layout is used to fill up the whole widget display region
Fixed Positioning	This layout is for fixed coordinate layouts
Linear Layout	This layout supports both horizontal and vertical layouts
Scroll Layout	Specific layout for scrollable panel
Stretch	This layout supports both horizontal and vertical layouts
Switch Page	To manage page switching logic
Scroll Switch Page	This layout supports scroll between pages according to the swipe direction.

There are also layout utilities and advanced layout features provided in ESD Layouts collection. These sections are for advanced user only.

Layout utilities are a set of helping functions to ease the development of layout. Users can build their own layout widget by making use of them. There are several items defined for sorting widgets purpose.

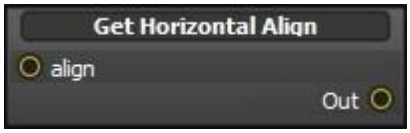
Utilities

ESD 4.10 provides various Layout utility functions while designing layouts.



Copyright © Bridgetek Pte Ltd

Get Horizontal Align



The *Get Horizontal Align* utility is used to align the widget horizontally.

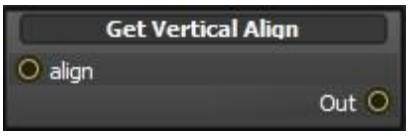
The function can be called as shown below:

```
ESD_FUNCTION(ESD_ALIGN_HORIZONTAL, Type = Esd_AlignHorizontal, DisplayName
= "Get Horizontal Align", Category = EsdLayoutUtilities, Macro)

ESD_PARAMETER(align, Type = Esd_Align)

#define ESD_ALIGN_HORIZONTAL(align) (align & 3)
```

Get Vertical Align



The *Get Vertical Align* utility is used to align the widget vertically.

The function can be called like below:

```
ESD_FUNCTION(ESD_ALIGN_VERTICAL, Type = Esd_AlignVertical, DisplayName =
"Get Vertical Align", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(align, Type = Esd_Align)
#define ESD_ALIGN_VERTICAL(align) (align & 12)
```

Detach

The *Detach* utility is used to detach the widget from the current parent layout

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_Detach, DisplayName = "Detach", Category =
EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_Detach(Ft_Esd_Widget *context);
```

Get Global Height

The *Get Global Height* utility is used to get the global X co-ordinate of widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalHeight, Type = ft_int16_t, DisplayName
= "Get Global Height", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define Ft_Esd_Widget_GetGlobalHeight(context) (((Ft_Esd_Widget *)context)-
>GlobalHeight)
```


Copyright © Bridgetek Pte Ltd

Get Global Width

The *Get Global Width* utility is used to get the width of the widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalWidth, Type = ft_int16_t, DisplayName =
"Get Global Width", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define Ft_Esd_Widget_GetGlobalWidth(context) (((Ft_Esd_Widget *)context)-
>GlobalWidth)
```

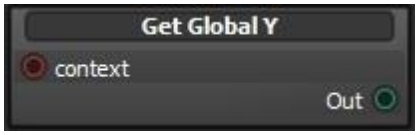
Get Global X

The *Get Global X* utility is used to get the global X co-ordinate of the widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalX, Type = ft_int16_t, DisplayName =
"Get Global X", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define Ft_Esd_Widget_GetGlobalX(context) (((Ft_Esd_Widget *)context)-
>GlobalX)
```

Get Global Y



The *Get Global Y* utility is used to get the global Y co-ordinate of the widget.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_GetGlobalY, Type = ft_int16_t, DisplayName =
"Get Global Y", Category = EsdLayoutUtilities, Macro)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
#define Ft_Esd_Widget_GetGlobalY(context) (((Ft_Esd_Widget *)context)-
>GlobalY)
```

Insert Above

The *Insert at Above* utility is used to insert a widget above the specified sibling widgets in the sibling widgets parent layout.

```
ESD_FUNCTION(Ft_Esd_Widget_InsertAbove, DisplayName = "Insert Above",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(sibling, DisplayName = "Sibling", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertAbove(Ft_Esd_Widget *context, Ft_Esd_Widget
*sibling);
```


Copyright © Bridgetek Pte Ltd

Insert Below

The *Insert at Below* utility is used to insert a widget below the specified sibling widgets in the sibling widgets parent layout.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_InsertBelow, DisplayName = "Insert Below",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(sibling, DisplayName = "Sibling", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertBelow(Ft_Esd_Widget *context, Ft_Esd_Widget
*sibling);
```

Insert at Bottom

The *Insert at Bottom* utility is used to insert a widget in the bottom position of the specified parent layout.

The function can be called like below:

```
ESD FUNCTION(Ft Esd Widget InsertBottom, DisplayName = "Insert at Bottom",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(parent, DisplayName = "Parent", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertBottom(Ft_Esd_Widget *context, Ft_Esd_Widget
*parent);
```

Insert at Top

The *Insert at Top* utility is used to insert a widget in the top position of the specified parent layout.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_InsertTop, DisplayName = "Insert at Top",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ESD_PARAMETER(parent, DisplayName = "Parent", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_InsertTop(Ft_Esd_Widget *context, Ft_Esd_Widget *parent);
```

Is Bottom

The *Is Bottom* utility returns FT_TRUE if the widget is the bottom widget within the current parent layout.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_IsBottom, Type = ft_bool_t, DisplayName = "Is
Bottom", Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ft_bool_t Ft_Esd_Widget_IsBottom(Ft_Esd_Widget *context);
```


Copyright © Bridgetek Pte Ltd

Is Top

This *Is Top* utility returns FT_TRUE if the widget is the topmost widget within the current parent layout

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_IsTop, Type = ft_bool_t, DisplayName = "Is Top",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
ft_bool_t Ft_Esd_Widget_IsTop(Ft_Esd_Widget *context);
```

Is Visible

The *Is Visible* utility is used to check if the widget is within the current screen scissor area.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_IsVisible, Type = ft_bool + t, DisplayName = "Is
Visible", Category = EsdLayoutUtilities)
ESD_PARAMETER(context, Type = Ft_Esd_Widget *)
ft_bool_t Ft_Esd_Widget_IsVisible(Ft_Esd_Widget *context);
```

Sort Higher

The *Sort Higher* utility is used to sort the widget above its current sibling within the current parent layout.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_SortHigher, DisplayName = "Sort Higher",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortHigher(Ft_Esd_Widget *context);
```

Sort Lower

The *Sort Lower* utility is used to sort the widget below its current sibling within the current parent layout.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_SortLower, DisplayName = "Sort Lower", Category
= EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortLower(Ft_Esd_Widget *context);
```


Copyright © Bridgetek Pte Ltd

Sort to Bottom

The *Sort to Bottom* utility is used to sort the widget to the bottom within the current parent layout.

The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_SortToBottom, DisplayName = "Sort to Bottom",
Category = EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortToBottom(Ft_Esd_Widget *context);
```

Sort to Top

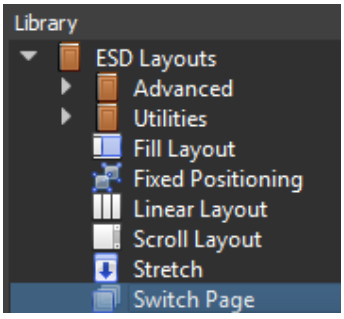
The *Sort to Top* utility is used to sort the widget to the top within the current parent layout.

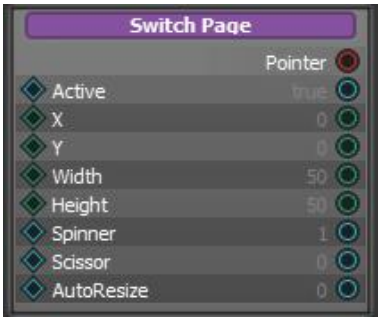
The function can be called like below:

```
ESD_FUNCTION(Ft_Esd_Widget_SortToTop, DisplayName = "Sort to Top", Category
= EsdLayoutUtilities)
ESD_PARAMETER(context, DisplayName = "Widget", Type = Ft_Esd_Widget *)
void Ft_Esd_Widget_SortToTop(Ft_Esd_Widget *context);
```

Switch Page

The *Switch Page* is a special layout which focuses on page/widget transitions. By default, there will be only one Switch Page layout in "AppScreen.Page" in every ESD project. However, this is optional and the users can remove or change the switch page layout in the project. Refer to the example project **"EVE Screen Designer → Examples → Basic → MenuPage → MenuPage.esd → AppScreen.page"** for a switch page demo.



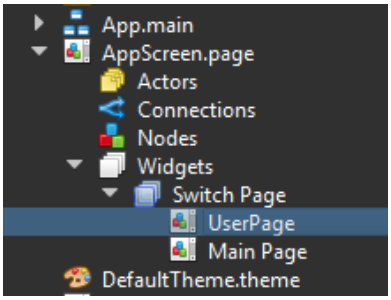


Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Spinner	Set true to show spinner/loading icon when switching page. This is useful user indication when a page has heavy resource.
Scissor	Set true, to trim off any content is drawn outside the layout region.
AutoResize	Set true, to enable auto resize for this layout.

Table 10 – Switch Page Layout Properties

Copyright © Bridgetek Pte Ltd

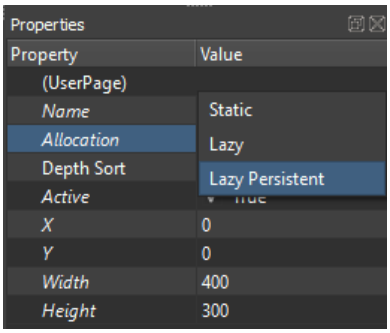
Switch Page Implementation



1. Set page/widget into Switch Page

Unlike widgets, Switch Page cannot be dragged on Screen Layout Editor or Logic Editor. In order to put a page under the control of the switch page layout, the user has to use Project Browser instead, drag the widget into the switch page layout.

2. Page Resource



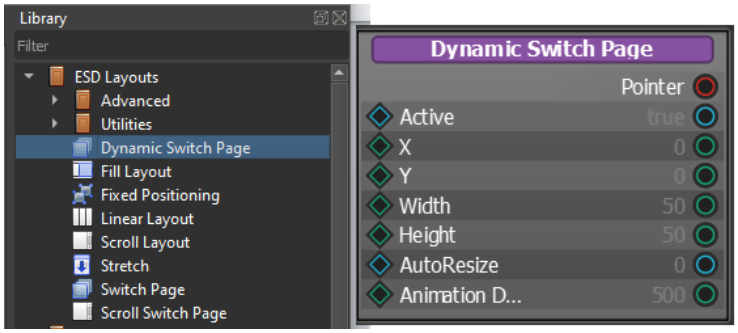
Since ESD 4.0, a page resource allocation can be:

- Static – always load at the start
- Lazy – allocate when it is active, and free it when it is inactive.
- Lazy persistent – allocate when it is first time active, and it will remain in system until reset.

When a new page is activated, its parent object, the Switch Page will also mark the current page as inactive and free it from memory when it has "Lazy" mode of allocation.

Dynamic Switch Page

The *Dynamic Switch Page* is a special layout which focuses on page/widget transitions. The primary purpose of this widget is to achieve a fade-in/fade-out animation while transiting from one page to another page. Like Switch Page widget, only one of the children of Switch Page would be visible at any point of time. But unlike Switch Page widget which abruptly changes the visibility between the transiting pages, Dynamic Switch Page widget will fade-in and fade-out the new to-be-visible page and old not-to-visible page.

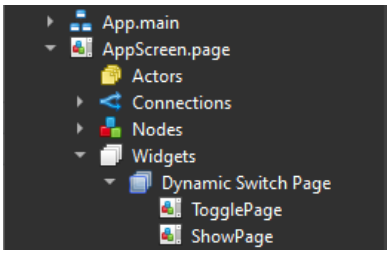


Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
AutoResize	Set true, to enable auto resize for this layout.
Animation Duration	Sets the time within which the animation needs to be finished.

Table 11 – Dynamic Switch Page Layout Properties

Copyright © Bridgetek Pte Ltd

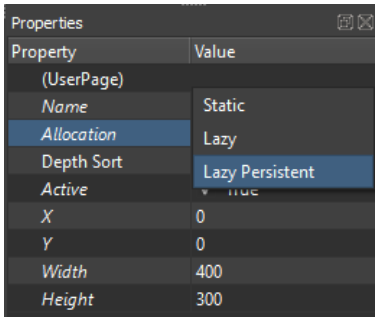
Dynamic Switch Page Implementation



1. Set page/widget into Dynamic Switch Page

Unlike widgets, Dynamic Switch Page cannot be dragged on Screen Layout Editor or Logic Editor. In order to put a page under the control of the switch page layout, the user has to use Project Browser instead, drag the widget into the Dynamic switch page layout.

2. Page Resource

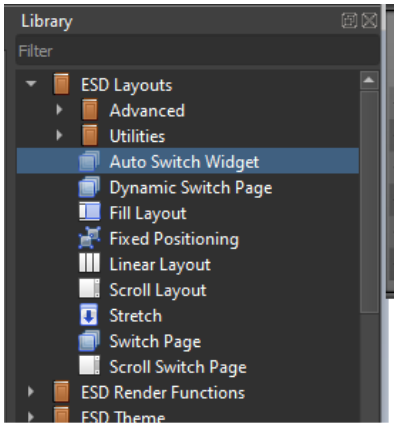


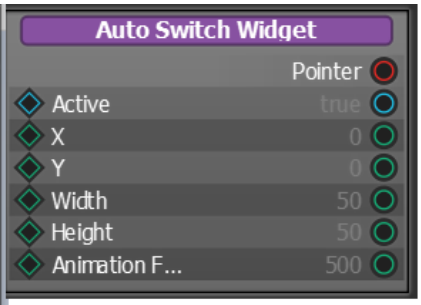
Since ESD 4.0, a page resource allocation can be:

- Static – always load at the start
- Lazy – allocate when it is active, and free it when it is inactive.
- Lazy persistent – allocate when it is first time active, and it will remain in system until reset.

When a new page is activated, its parent object, the Dynamic Switch Page will also mark the current page as inactive and free it from memory when it has "Lazy" mode of allocation.

Auto Switch Widget





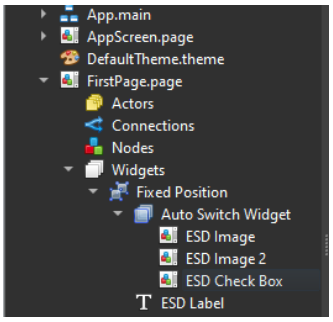
The *Auto Switch Widget* is a layout which focuses on rapid transitions between its child widgets. The primary purpose of this widget is to rapidly activate its child widget one by one in the order of its depth. The rate of change is set by the user. This can be used to generate a gif like animation effect. Like Switch Page widget and Dynamic Switch Page widget, only one of its child is active at any given point of time. But unlike Switch Page and Dynamic Switch Page, the transition between child widget is automatic and not triggered. Use case of the widget can be found in Examples/Basic/AutoSwitchLayout.

Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Animation Frequency	Sets the rate of change of child widget activations

Table 12 – Auto Switch Page Layout Properties

Copyright © Bridgetek Pte Ltd

Auto Switch Page Implementation



1. Set widgets into Dynamic Switch Page

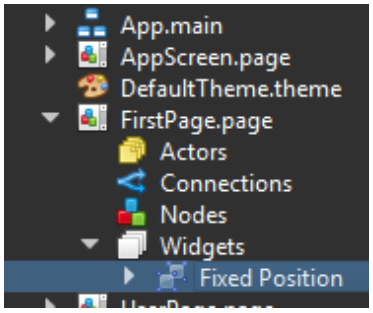
Auto Switch widget can be dragged on Logic Editor. In order to put a widget under the control of the switch page layout, the user has to use Project Browser instead, drag the widget into the Auto Switch Widget layout.

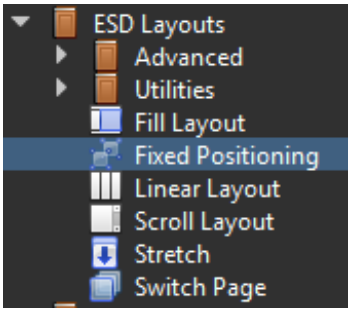
2. Widget Rendering order

All child widgets are activated one by one and rendered in a sequence according to their depth orders. Thus to rearrange the order of activation, user needs to update the depth order of the children widgets.

Fixed Positioning

The *Fixed Positioning* is a layout widget which allows its child widgets to be fixed at design time. By default, every page starts with one fixed positioning layout, such that all widgets under the page are placed at design time. Fixed Positioning layout does not update its child widget's dimensions on runtime. Users should get what they see on design time, on runtime as well.

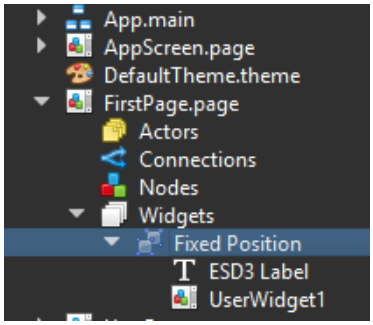




Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height

Table 13 – Fixed Positioning Layout Properties

Fixed Positioning Implementation



1. Set page/widget into Fixed Positioning Layout

Drag and drop a new widget into the Screen Layout Editor will be the highest depth sort widget in the layout by default. However, the depth sort values are configurable by the users. If a widget is not in fixed positioning layout, the Screen layout editor will not able to change it is position as it desired. Always check on project browser as picture on the right.

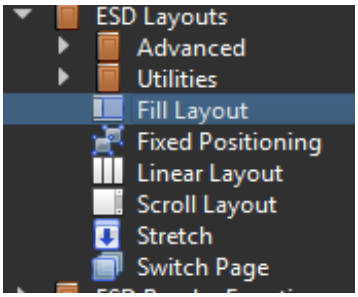
Copyright © Bridgetek Pte Ltd

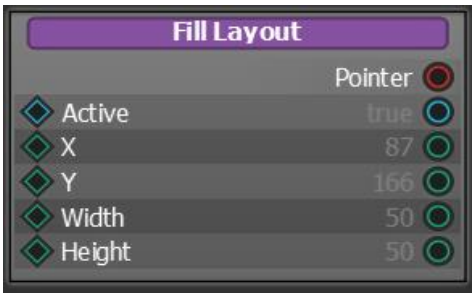
2. Change Child Widgets' depth sort

Changing the depth sort value of child widgets will affect their rendering sequence. This functionality is same as in ESD 3.0.

Fill Layout

The *Fill Layout* is the base layout for every widget/page since in ESD 4.10. A widget always comes with a Fill Layout in it. However, a fill layout in a page has a fixed size due to the screen size defined from the hardware. All widgets in the Fill Layout will auto resize to fit the whole layout area.





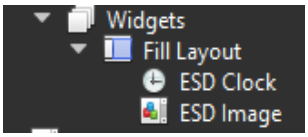
Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height

Table 14 – Fill Layout Properties

Fill Layout Implementation

1. Change Widget sequence

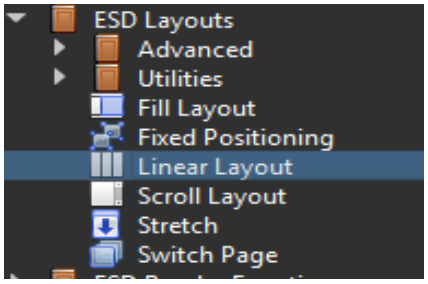
The fill layout sequence is ordered in the sequence of widgets' depth sort. Changing a particular widget to a higher depth sort value, will increase the priority widget's rendering. The example below shows a clock widget and an image widget under the same Fill Layout. Both widgets are filling up the full "Fill Layout" area. However, by setting the depth value of clock widget bigger than the image widget's depth sort, the clock widget will be rendered on top of the image widget.

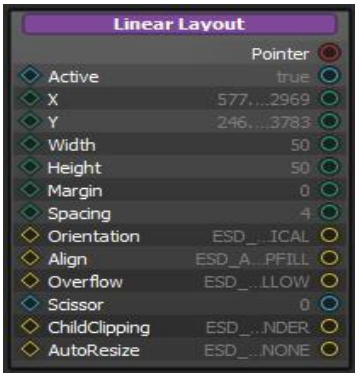


Linear Layout

The *Linear Layout* supports both horizontal and vertical alignment in ESD 4.10. All child widgets will be distributed in the selected orientation. Users cannot change the sequence of widgets in the linear layout from Screen Layout Designer in ESD 4.10. However, changing their depth sort value from the property browser will change the sequence.

Copyright © Bridgetek Pte Ltd





Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Spacing	Set the spacing gap between adjacent child widgets.
Orientation	Set Linear Layout orientation. It supports: • ESD_ORIENTATION_HORIZONTAL • ESD_ORIENTATION_VERTICAL
Alignment	Set the alignment setting for the child widgets in the layout
Overflow	Set child widget overflow mode. It supports: • ESD_OVERFLOW_ALLOW • ESD_OVERFLOW_CLIP • ESD_OVERFLOW_FILL
Scissor	Set true to enable scissor to clip off overflow
ChildClipping	Set the flags when child clipping should check. It supports: • ESD_CLIP_RENDER • ESD_CLIP_UPDATE • ESD_CLIP_IDLE
AutoResize	Set auto resize mode for this layout.

Table 15 – Linear Layout Properties

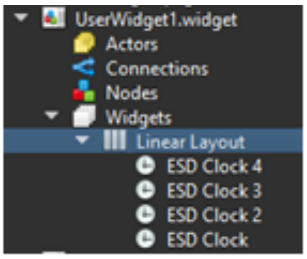
Linear Layout Implementation

1. Horizontal Layout

All child widgets of a linear layout with its orientation are set to "ESD_ORIENTATION_HORIZONTAL". They will be distributed horizontally with some spacing in between adjacent widgets. The layout used as in the example here has parameters:

- Orientation = ESD_ORIENTATION_HORIZONTAL
- Spacing = 4
- Align = ESD_ALIGN_TOPFILL
- Overflow = ESD_OVERFLOW_ALLOW

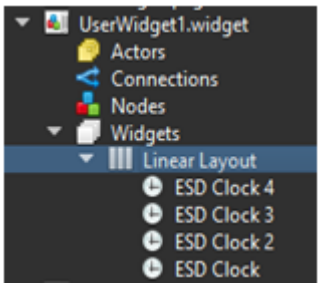
Copyright © Bridgetek Pte Ltd



2. Vertical Layout

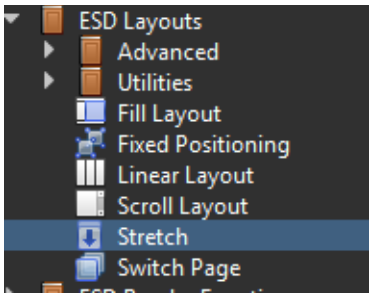
All child widgets of a linear layout with its orientation set to "ESD_ORIENTATION_VERTICAL". They will be distributed vertically with some spacing in between adjacent widgets. The layout used as in the example here has parameters:

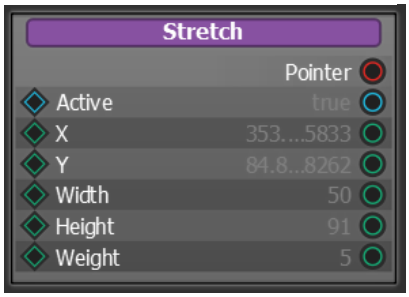
- Orientation = ESD_ORIENTATION_VERTICAL
- Spacing = 4
- Align = ESD_ALIGN_TOPFILL
- Overflow = ESD_OVERFLOW_ALLOW



Stretch

The *Stretch Layout* is designed to work together with the linear layout. In the event some widgets need to have higher weightage in the layout, Stretch Layouts are used.





Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Weight	The weight factor when determine the stretch scale in the selected orientation

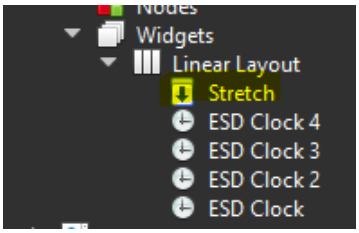
Table 16 – Stretch Layout Properties

Copyright © Bridgetek Pte Ltd

Stretch Layout Implementation

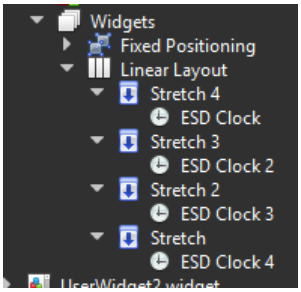
1. Space Stretching

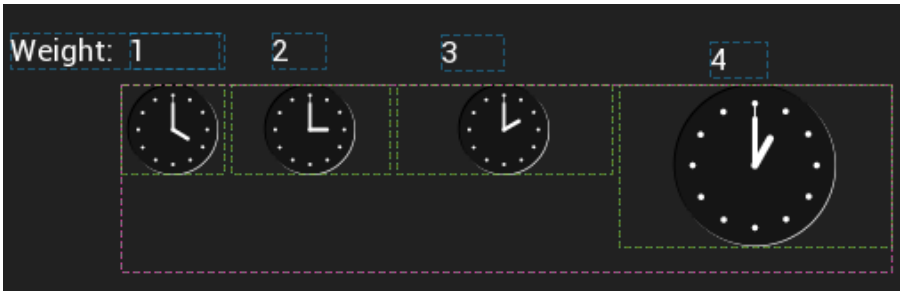
The example here demonstrates how to stretch unused space horizontally. A stretch layout has been added into the Linear Layout. The weight factor used here is 1.



2. Widget Weight Stretching

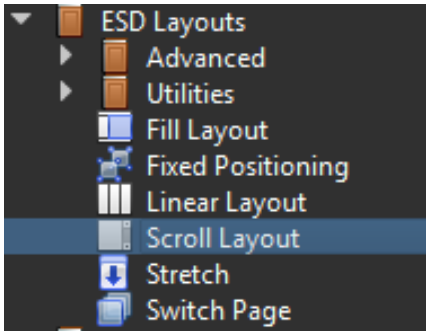
The example here demonstrates widget stretching. In this example, every clock widget has different stretch weight values. The result every widget gains their proportion of width as shown below -





Scroll Layout

The *Scroll Layout* is a support runtime resizable and scrollable layout in ESD 4.10.



Copyright © Bridgetek Pte Ltd

Property Name	Description
Pointer	The pointer reference of the widget object
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
ScrollX	Getter and Setter for scroll value along X-axis
ScrollY	Getter and Setter for scroll value along Y-axis
Align	Set alignment for its child widgets
Scrollbars	Set when scrollbar should be visible
TouchScroll	Set true to enable touch scroll
Scroll Bar Size	To set scroll bar size when it is visible

Table 17 – Scroll Layout Properties

Scroll Layout Implementation

For the implementation of scroll layout, refer to the example project - **"EVE Screen Designer → Examples → Intermediate → ScrollPanel.**

Scroll Switch Page Layout

The *Scroll Switch Page Layout* consists of a switch page manager and a scroll panel for runtime scrollable support.

Scroll Switch Page layout requires 3 lazy page instances and a current index to start with, namely - the previous page instance, the current page instance and the next page instance.

Upon user touch swipe event, the page will switch according to the swipe direction. The behaviour of this scroll switch page layout are as below:

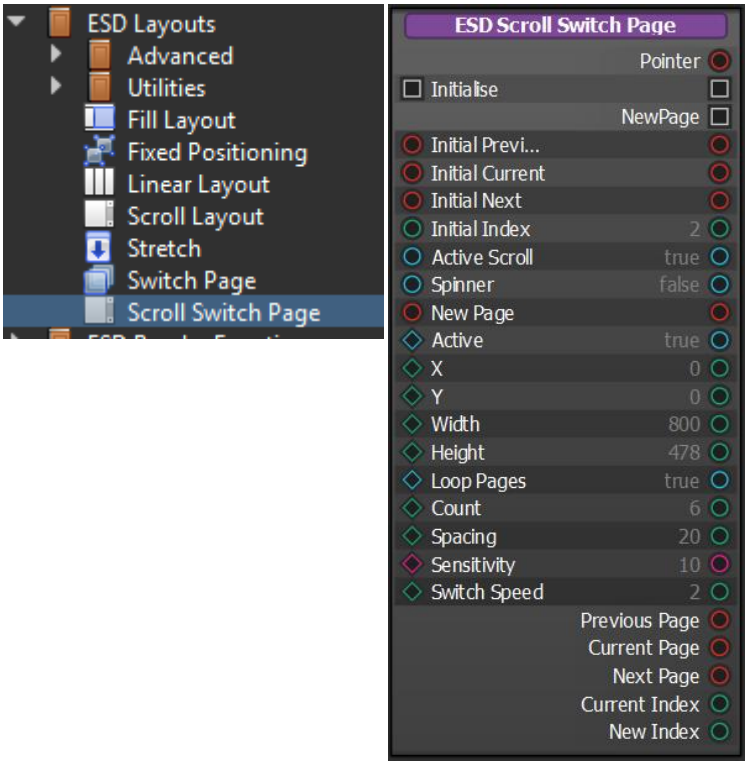


Figure 58 - Scroll Switch Page Layout

- On non-swiping runtime, it only renders the current page, but runs the update logic of all the 3 pages.
- On swiping event, it displays both the current page and the swiping to page (can be previous and next page instance).
- On touch up event, it will either recover to the original position if the swiped range has not reached the minimum threshold or switch to the new page. The touch scroll will be temporally disabled until it has recovered or switched to the target page.
- Upon switching page complete, the scroll switch page will fire "NewPage" event and requests a new page instance to be allocated and assigns it as the "New Page" input.

Copyright © Bridgetek Pte Ltd

Property Name	Description
Pointer	The pointer reference of the widget object
Initialise	The "initialise" slot for initialling the scroll switch page layout
Initial Previous	Initial Previous page instance at start
Initial Current	Initial Current page instance at start
Initial Next	Initial Next page instance at start
Initial Index	Initial Current page index at start
Active Scroll	Set true to override child widgets touch tag event where tag is default(255)
Spinner	Set true to show spinner upon loading the new page instance.
New Page	New page instance input
Active	Enable or disable displaying this widget
X	x coordinate of the top-left of the widget, in pixels
Y	Y coordinate of the top-left of the widget, in pixels
Width	Widget width
Height	Widget Height
Loop Pages	Set true to loop the pages
Count	Set pages count/limit
Spacing	Defines the spacing between pages
Sensitivity	Define the scroll switch sensitivity, range from: 0.05 to 20.00
Switch Speed	Define the switch speed during page transition
Previous Page	The output of previous page instance
Current Page	The output of current page instance
Next Page	The output of next page instance
Current Index	The output of current index
New Index	The output of new index

Table 18 – Scroll Switch Page Layout Properties

Scroll Switch Page layout Implementation

In typical configuration for a scroll switch page layout, the figure below demonstrates a sample setup. It should have the following rules:

- Connect start slot to the "Initialise" slot of scroll switch page layout instance.
- All switching pages should be configured as child widgets of Scroll Switch Page.
- All child page widgets should be lazy allocation and has same width as the Scroll Switch Page.
- Initial Previous Page, Initial Current Page and the Initial Next Page should be active by default while the rest of child pages should be inactive.
- Initial index should be the index of initial current page.
- Recommend using the "Switch" node to trigger the specific "Activate" signal of the target page.
- Recommend using the "Switch Value" node to get the specific target page instances from all the pages.
- Recommend adding header and footer pages which are Sibling widgets of the Scroll Switch Page Layout instance.

Copyright © Bridgetek Pte Ltd

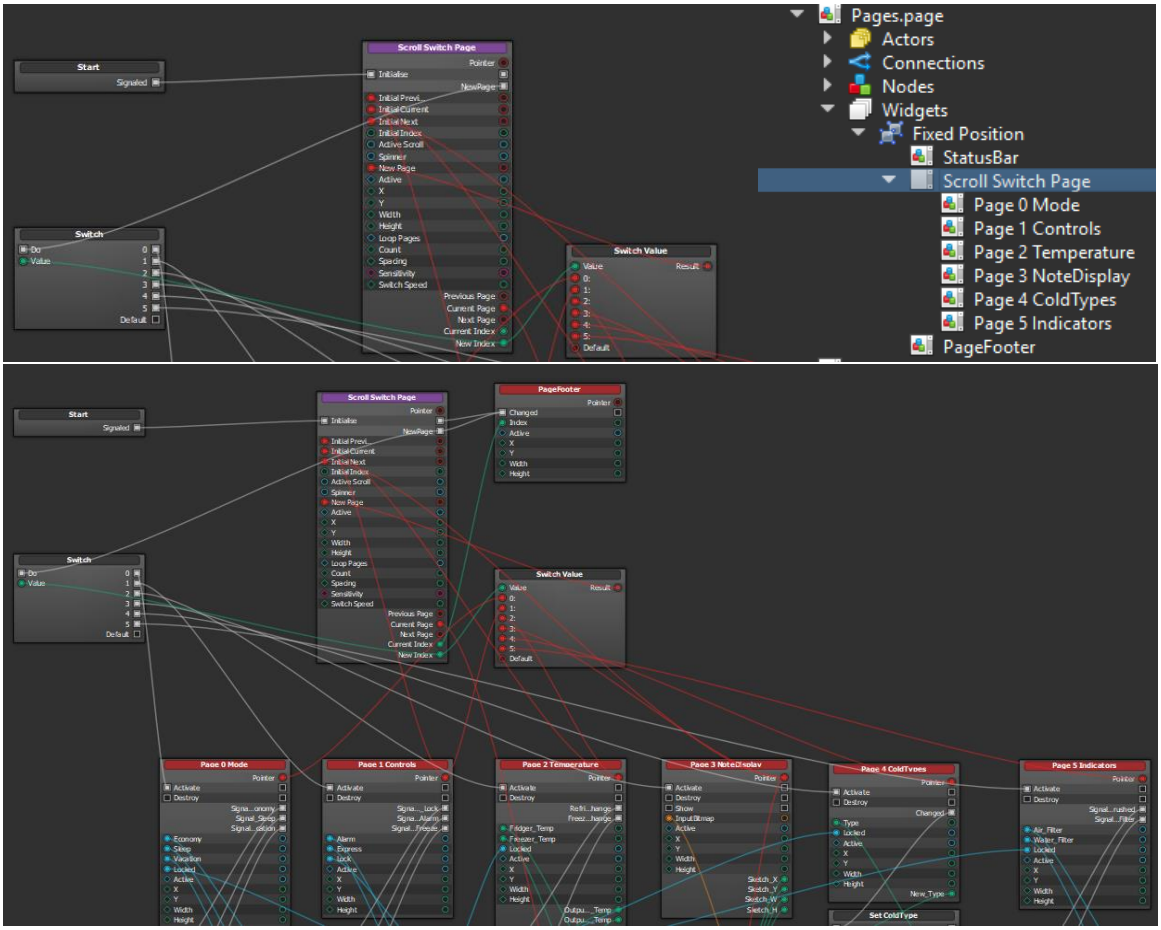
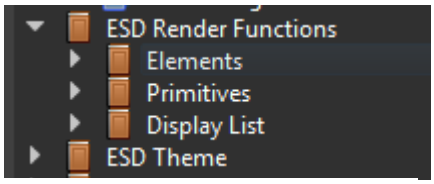


Figure 59 - Sample Scroll Switch Page Implementation

ESD Render Functions

All render functions are non-widget. They are not managed by layout in ESD 4.10. Non-widget functions are to be used only within a widget. These functions are provided as utilities for user to make their own custom widgets.



Note: In order to preview the render functions result in design time, render slot in the custom widget is required.

Elements

The *Elements* are both simple and basic widgets, usually used to construct higher level and more complex widgets. Normally, they do not handle any touch input.

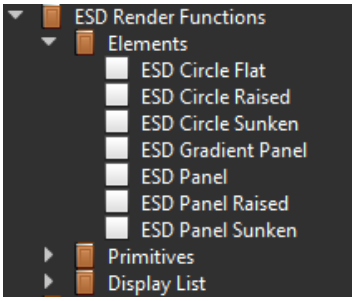


Figure 60 - ESD Elements

Copyright © Bridgetek Pte Ltd

ESD Circle

The *Circle* element allows the user to draw circles on the screen. The following styles are available – *ESD Circle Flat; ESD Circle Raised* and *ESD Circle Sunken*.

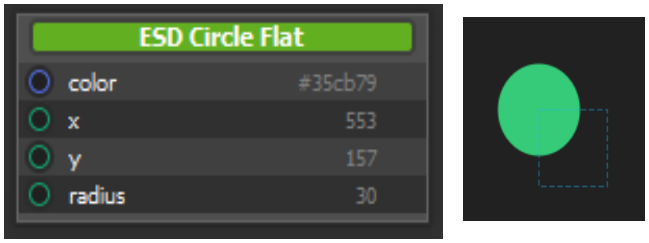


Figure 61 - ESD Circle Element

Property Name	Description
Color	RGB value to be rendered inside the circle
X	x coordinate of central point, in pixels
Y	Y coordinate of central point, in pixels
radius	Radius of the point

Table 19 - ESD Circle Element Properties

ESD Panel

The *ESD Panel* is an adjustable rectangular widget that defines a coloured area. The colour can be specified by a theme or user's selection. Internally, it is constructed by EVE primitives RECTS and SCISSORS. Two styles of ESD Panel are available namely – *ESD Panel Raised* and *ESD Panel Sunken*.

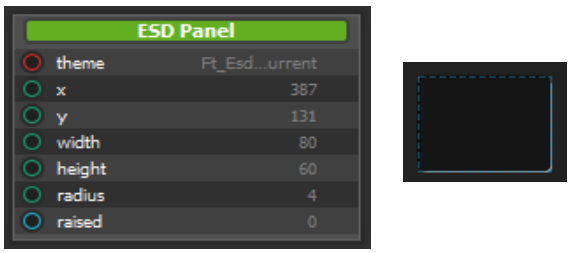


Figure 62 - ESD Panel Element

Property Name	Description
Theme	Theme to be applied on the panel
x	x coordinate of the top-left point, in pixels
y	Y coordinate of the top-left point, in pixels
width	Panel width
height	Panel Height
radius	radius of the corners
raised	Set to true to make panel in raised style

Table 20 - ESD Panel Element Properties

ESD Gradient Panel

The *Gradient Panel* element is an adjustable rectangular widget that defines a gradient coloured area. The gradient colour can be specified by two colours (one at start and one at the end). The gradient direction is adjustable by changing the direction.

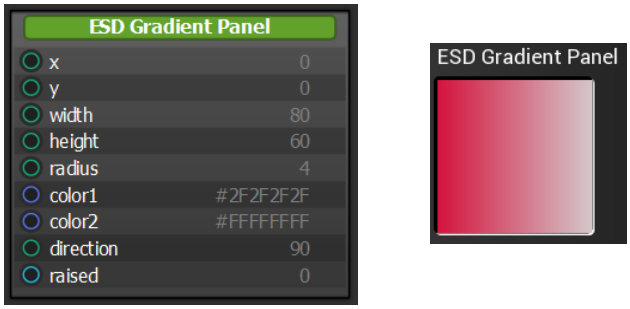


Figure 63 - ESD Gradient Panel Element

Copyright © Bridgetek Pte Ltd

Property Name	Description
x	x coordinate of the top-left point, in pixels
y	Y coordinate of the top-left point, in pixels
width	Panel width
height	Panel Height
radius	radius of the corners
color1	Start colour for gradient effect
color2	End colour for gradient effect
direction	Gradient effect direction
raised	Set to true to make panel in raised style

Table 21 - ESD Gradient Panel Element Properties

Primitives

A *Primitive* is a special type of logic node which is a wrapper for a rendering function called by Render slot. It has no return value and its properties cannot be output to other logic nodes. Similar to widgets, it appears on the screen, but is unable to handle user input.

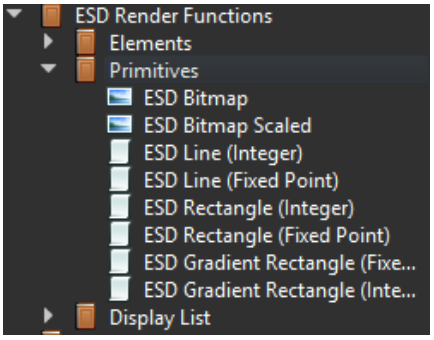


Figure 64 - ESD 4.10 Primitives

ESD Bitmap

The *ESD Bitmap* object allows users to display a bitmap resource. A bitmap resource is generated from an image file by adding it into a project. A bitmap resource is treated as a single bitmap cell by default with the name "(Bitmap Resource)_0".

For example, if the bitmap resource is named "photo.jpg", the default bitmap resource will be named as "photo_0". Users can define how many bitmap cells they want by specifying the *"CellHeight"* property.

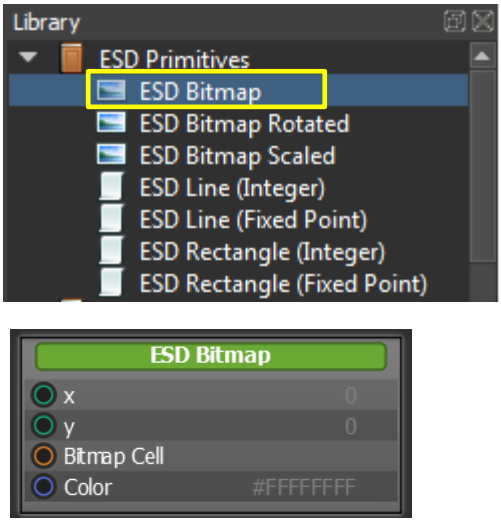


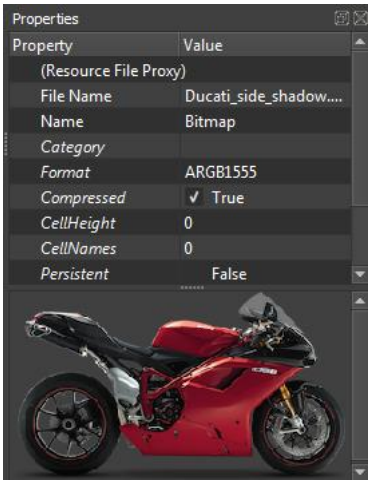
Figure 65 - ESD Bitmap Primitive

Property Name	Description
x	x coordinate of Bitmap, top-left, in pixels
y	Y coordinate of Bitmap, top-left, in pixels
Bitmap Cell	Bitmap cell to display
Color	Color to be applied to bitmap

Table 22 - ESD Bitmap Properties

Copyright © Bridgetek Pte Ltd

To add a bitmap

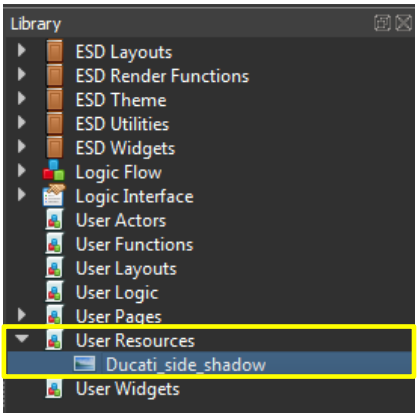


Click and select **"File → Add"** or click from the toolbar. A bitmap cell is then assigned. ESD 4.10 supports the following image file formats as input – *".png", ".jpg"* and *".jpeg"*. Bitmap resources can contain a single or multiple cells based on the *CellHeight* value.

If the value of the *"CellHeight"* is zero, then the number of cells is one.

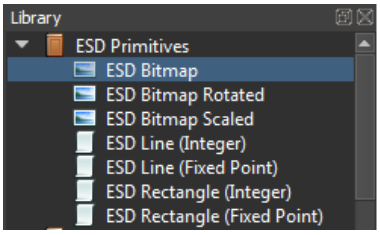
If the value of the *"CellHeight"* is non-zero, then the number of cells value is calculated as shown below –

`Number of Cell = Bitmap Height / CellHeight`

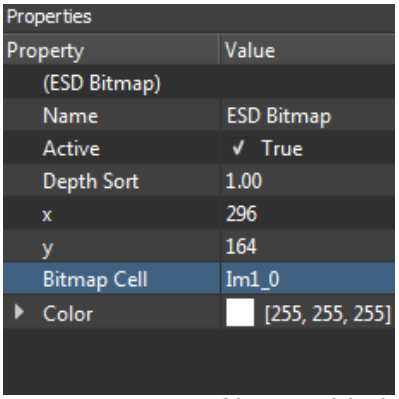


Upon adding a Bitmap, the new bitmap resource will be added into the Library under the category – *"User Resources"*.

To use an *ESD Bitmap*, drag and drop the ESD Bitmap Node into the *Layout Editor* or *Logic Node Editor*.



Choose a Bitmap Cell to display from the dropdown list.



The list of Bitmap Cell values is updated automatically after a new image file is added into a project.

Copyright © Bridgetek Pte Ltd

Use ESD Image to display bitmap as ESD bitmap is non-widget, and used only as a rendering function in ESD 4.10. Users are required to upgrade ESD Bitmap to ESD Image from ESD 3.X.

ESD Line

The *ESD Line* allows users to draw lines on the screen.

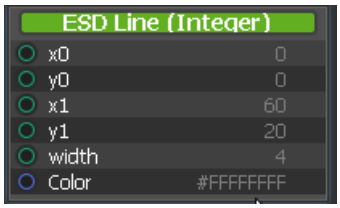


Figure 66 - ESD Line

Use ESD Widgets – Line Widget to display line as ESD line is non-widget, and used only as a rendering function in ESD 4.10. Users are required to upgrade ESD Line to Line Widget.

Property Name	Description
x0	x – coordinate of the start point, in pixels
y0	y – coordinate of the start point, in pixels
x1	x – coordinate of the end point, in pixels
y1	y – coordinate of the end point, in pixels
Width	Line width, in pixel
Color	Line color, in RGBA format

Table 23 - ESD Line Properties

Fixed Point Variant

ESD Line (Fixed point) expresses the properties with pixel attribute in fixed point format. It provides more control to lines.

ESD Rectangle

The *ESD Rectangle* allows users to draw rectangle on the screen.

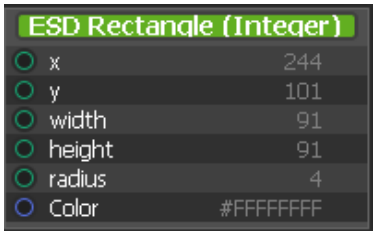


Figure 67 - ESD Rectangle

Copyright © Bridgetek Pte Ltd

Property Name	Description
X	x coordinate of Bitmap, top-left, in pixels
Y	Y coordinate of Bitmap, top-left, in pixels
width	width, in pixels
height	height, in pixels
radius	radius of the round corner, in pixels
Color	color of the rectangle

Table 24 - ESD Rectangle Properties

Fixed Point Variant

ESD Rectangle (Fixed point) expresses the properties with pixel attribute in fixed point format. It provides more control to rectangles.

Display List

The Display List provides a collection of utilities functions related to EVE Display List.

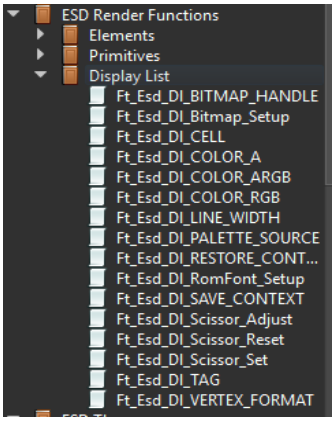


Figure 68 - ESD Display List

ESD Theme

A *Theme* is a collection of colours which can be used across the application in order to maintain a consistent style. The library browser provides the necessary logic nodes to make most use of the theme function.

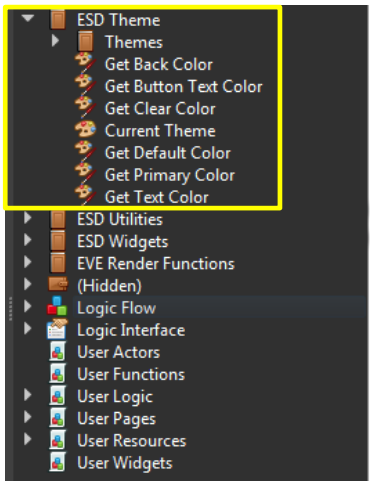


Figure 69 - Library Browser - ESD Theme

Copyright © Bridgetek Pte Ltd

Built In Themes

ESD 4.10 provides two *Built-in Themes* for users to configure the colour scheme of project. They are: *Ft_Esd_Theme_LightBlue* theme and *Ft_Esd_Theme_DarkOrange* theme.

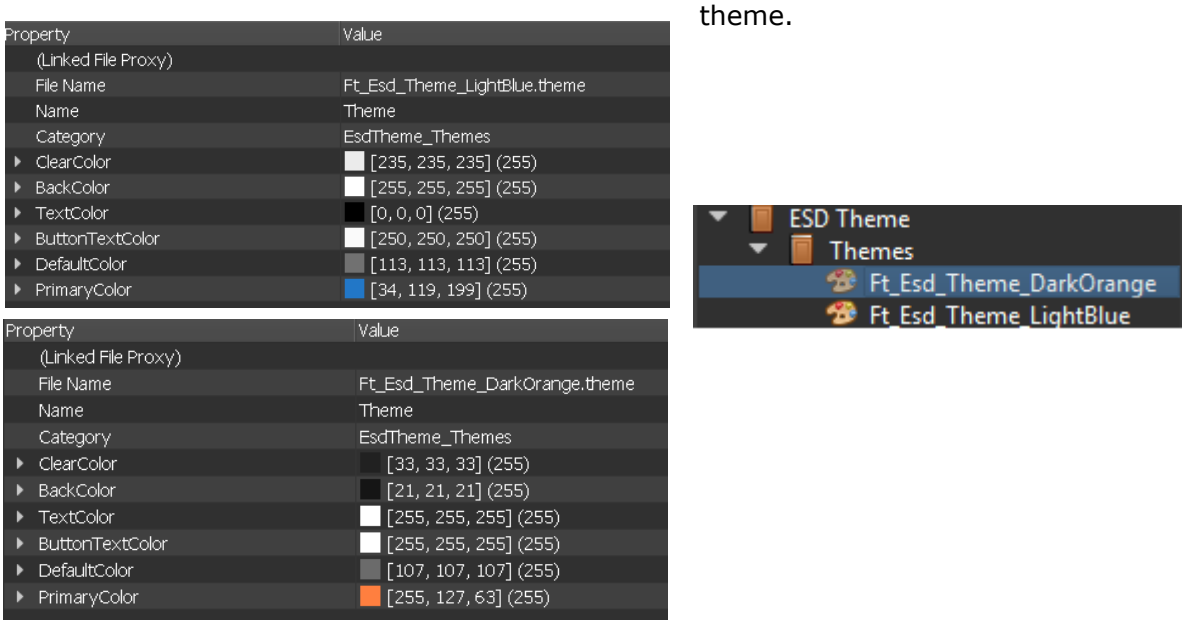


Figure 70 - ESD 4.10 Built-in Themes

Table 25 and Table 26 provides information about the built-in themes property -

Property Name	Type	Value (RGBA)	Description
Name	String	Theme	Ft_Esd_Theme_LightBlue
ClearColor	RGBA	235,235,235(255)	Clear Color
BackColor	RGBA	255,255,255(255)	Background Color
TextColor	RGBA	0,0,0(255)	Text Color
ButtonTextColor	RGBA	250,250,250(255)	Button Text Color
DefaultColor	RGBA	113,113,113(255)	Default Color
PrimaryColor	RGBA	34,119,199(255)	Primary Color

Table 25 - Ft_Esd_Theme_LightBlue Theme

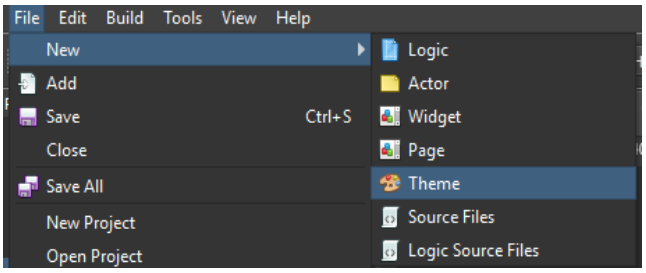
Property Name	Type	Value (Default)	Description
Name	String	Theme	Ft_Esd_Theme_DarkOrange
ClearColor	RGBA	33,33,33(255)	Clear Color
BackColor	RGBA	21,21,21(255)	Background Color
TextColor	RGBA	255,255,255(255)	Text Color
ButtonTextColor	RGBA	255,255,255(255)	Button Text Color
DefaultColor	RGBA	107,107,107(255)	Default Color
PrimaryColor	RGBA	255,127,63(255)	Primary Color

Table 26 - Ft_Esd_Theme_DarkOrange Theme

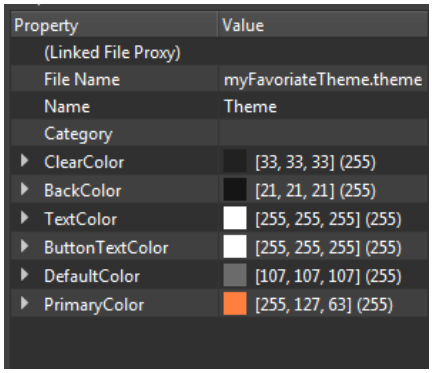
Copyright © Bridgetek Pte Ltd

To create a new theme:

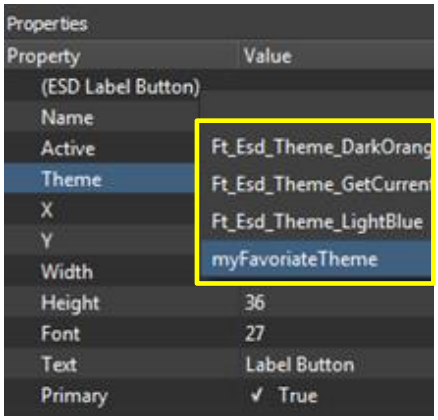
1. Click and select **"File → New → Theme"**.



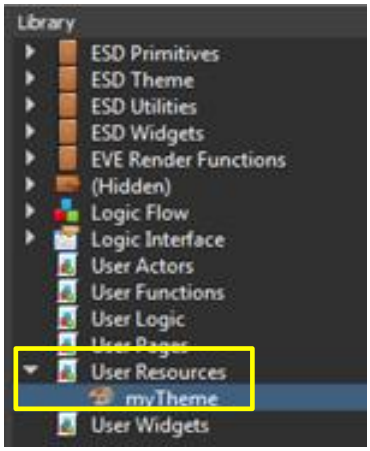
2. A new *"*.theme"* file is created within the Project folder. Using the Property Editor, users can configure it as per their requirement.



3. The newly-created theme will be available across the project and may be applied it to other widgets with the theme property.



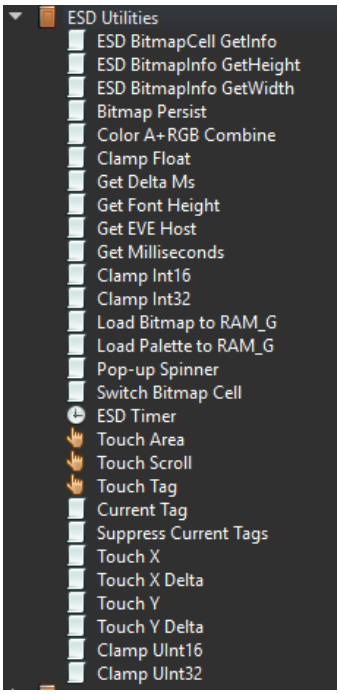
4. The newly created theme will also be available under **User Resources** in the *Library Browser*. Users can drag and drop the theme to the logic editor.



Copyright © Bridgetek Pte Ltd

ESD Utilities

ESD Utilities contains the collection ESD utilities functions. The following table provides the list of ESD Utilities and its functionality.



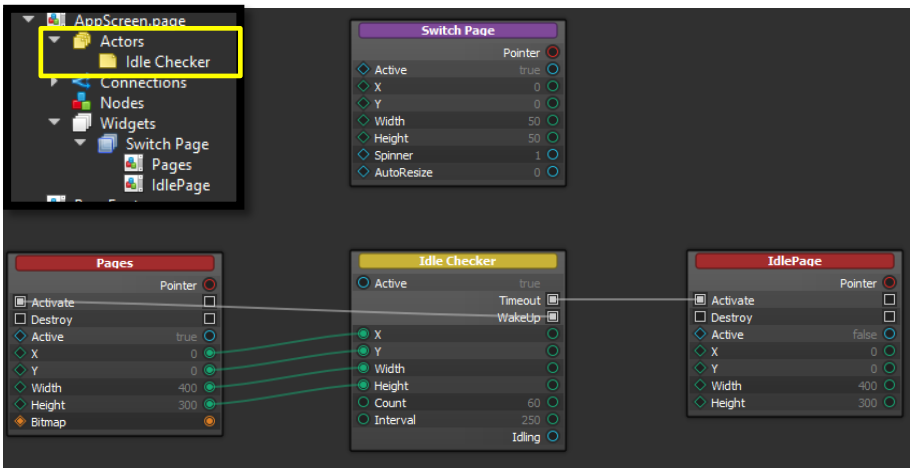
ESD Utility	Description
ESD BitmapCell GetInfo	Retrieves the BitmapCell information with given bitmap reference.
ESD BitmapInfo GetHeight	Returns the BitmapCell Height of given bitmap reference.
ESD BitmapInfo GetWidth	Retrieves the BitmapCell Width of given bitmap reference.
Bitmap Persist	To persist the given bitmap object
Color A+RGB Combine	Merge Alpha to RGB, to become RGBA colour
Clamp Float	Clamp float value with min and max limits
Get Delta Ms	Get Delta time difference in milliseconds since last frame update called
Get Font Height	Return Font Height
Get EVE Host	Get EVE Host
Get Milliseconds	Get time in milli-seconds for current frame
ESD Idle Checker	Check when to sleep and wake up based on the touch events.
Clamp Int16/UInt16	Clamp Int16/UInt16 value with min and max limits
Clamp Int32/UInt32	Clamp Int32/UInt32 value with min and max limits
Load Bitmap to RAM_G	Load Bitmap to RAM_G
Load Palette to RAM_G	Load Palette to RAM_G
Pop-up Spinner	Pop-up Spinner when the frame is rendered
Switch Bitmap Cell	Switch Bitmap Cell
ESD Timer	ESD Timer Actor
Touch Area	Touch Area Actor
Touch Scroll	Touch Scroll Actor
Touch Tag	Touch Tag Actor
Current Tag	Current Touch Tag
Suppress Current Tags	Suppress Current Tags
Touch X/Y	Touch X/Y coordinate
Touch X/Y Delta	Touch X/Y Delta

Table 27 – ESD Utilities & Description

Copyright © Bridgetek Pte Ltd

ESD Idle Checker

ESD Idle Checker Actor provides an actor to check when the application should go to sleep mode or wake up from sleep mode. One of the main use cases of the idle checker actor is to check for idling and wake up event for displaying the screen saver page. The picture on the right demonstrated in "AppScreen.page" uses Idle Checker to switch between application pages and the idle page. The idle checker has an interval of 250 ms and count of 60 intervals. In other words, it may timeout when there is no touch event for 15 secs (60 * 250 = 15000 ms).



Logic Flow

A *Logic Flow* is an ESD built-in logic node. It is used to define the logic or control flow and contains the following nodes:

- *Condition*
- *Binary Condition*
- *Local Method*
- *Binary Operator*
- *Ternary Operator*
- *Unary Operator*
- *Sequence*
- *Switch*
- *Switch Value*
- *Set Variable*
- *Watch Variables*

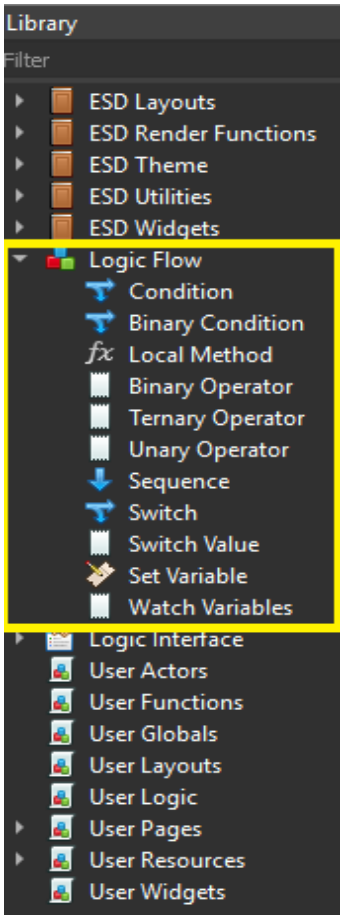


Figure 71 - Logic Flow

Copyright © Bridgetek Pte Ltd

Condition

Condition node provides a set of rules to be performed if a certain condition is met. In other words, it allows applying decision points.

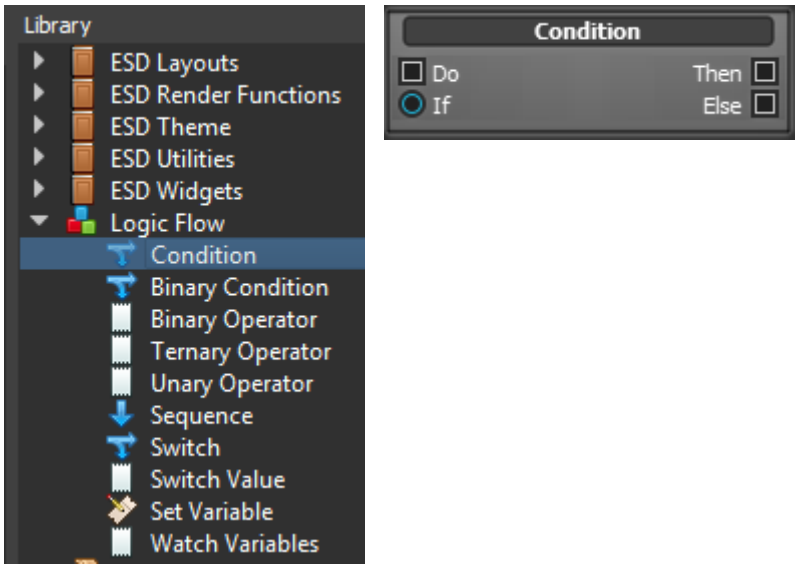


Figure 72 - Logic Flow – Condition Node

Whenever a condition node is called, a *"Then"* condition will be called when the *"If"* expression is True, otherwise an *"Else"* condition will be called.

Ensure that the input for the "If" connection is a Boolean variable (i.e. *True* or *False*).

Binary Condition

Binary Condition node provides a set of rules to be performed based on the result of a binary operator on two input values. In other words, it allows applying decision points.

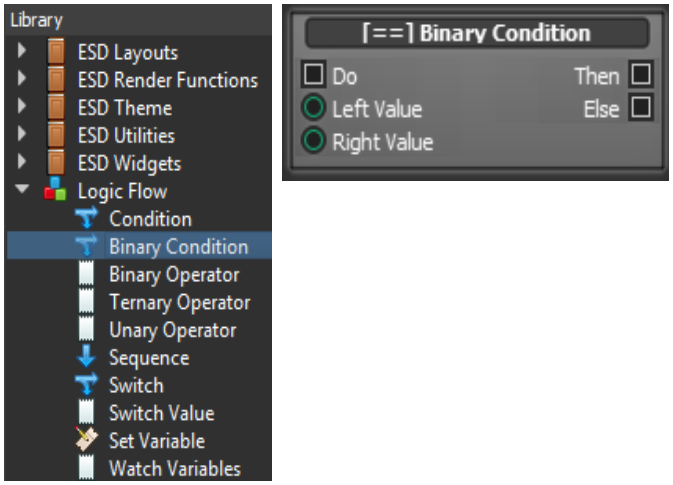


Figure 73 - Logic Flow – Binary Condition Node

Whenever a binary condition node is called, a *"Then"* condition will be called when the result of binary operator on left and right value is True, otherwise an *"Else"* condition will be called.

The binary operator is specified by user and it will be evaluated when the node is called through "*Do*" input call.

Copyright © Bridgetek Pte Ltd

Local Method

Local Method node is used to add a visual function node. It works as a place holder where user will provide input using parameters and put the function body in logic editor. It gives the option to specify the no. of parameters, parameter type and parameter name to the function as input. User can select the parameter count using scroll bar and add parameters (type and name) from the properties tab.

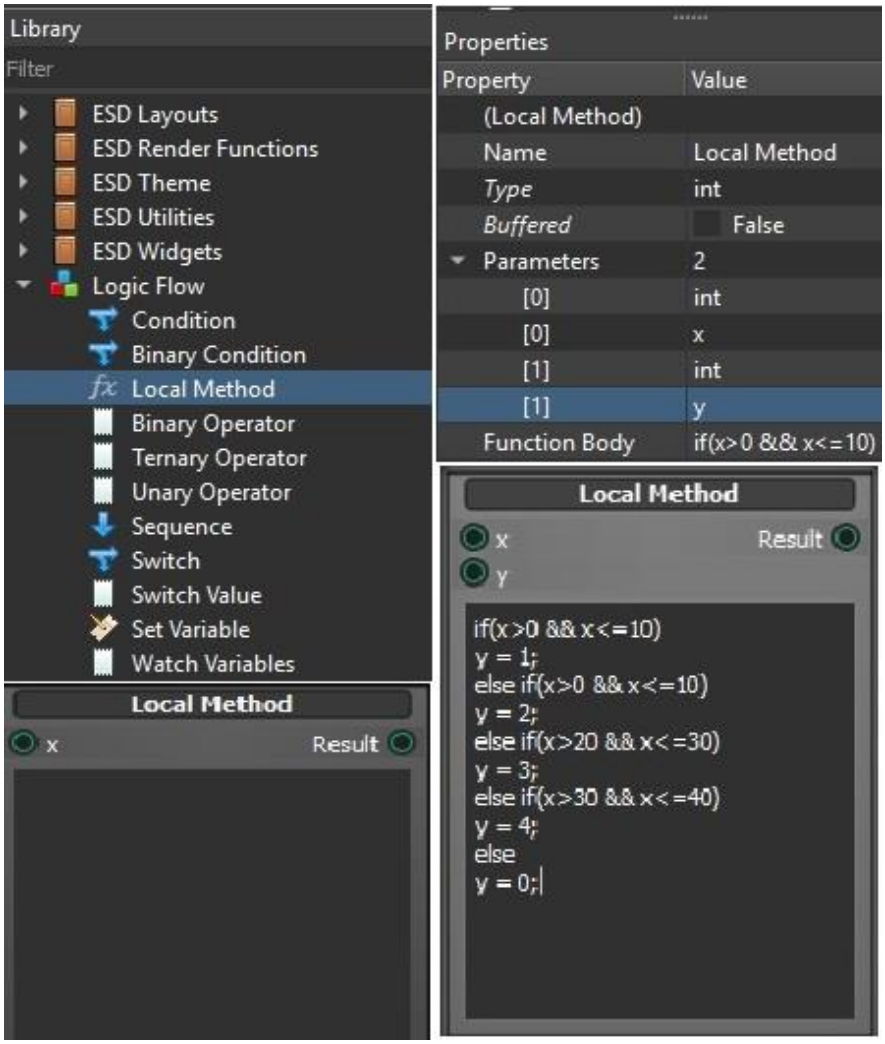


Figure 74 - Logic Flow – Local Method

- "Parameter Type" – Type of input parameter to be selected from drop down menu i.e. int, float, char* or Ft_Esd_Theme* etc.
- "Parameter Name" – Name of the variable should be according to ANSI C coding standard.
- "Result" – Return value from this function which will be assigned to a variable or input to new widget or assigned to output directly.

Copyright © Bridgetek Pte Ltd

Binary Operator

This node contains *Binary Operator* that operates on two operands (inputs) and manipulates them to return an output. For example: *Result=Left Value * Right Value*.

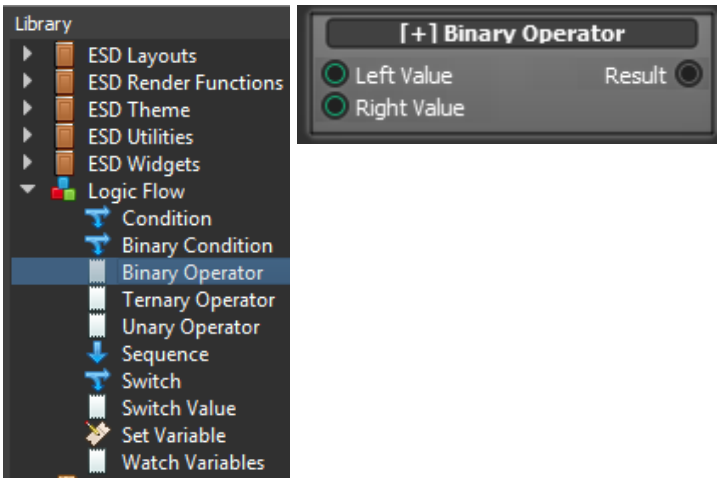


Figure 75 - Logic Flow - Binary Operator Node

The following table provides a list of binary operators supported by ESD 4.10.

Binary Operator	Description
==	Equal to comparison operator
!=	Not Equal to comparison operator
&&	Logical AND operator
&	Binary AND operator
||	Logical OR operator
|	Logical OR operator
*	Multiplication operator
/	Division operator
%	Modulus operator
^	Binary XOR operator
+	Addition operator
-	Subtraction operator
,	Comma operator
>	Greater than operator
<	Less than operator
>=	Greater than or equal to
<=	Less than or equal to
<<	Binary Left Shift Operator
>>	Binary Right Shift Operator

Table 28 - Binary Operators Supported by ESD 4.10

Copyright © Bridgetek Pte Ltd

Ternary Operator

This node contains *Ternary Operator* that operates on three arguments. First is the comparison argument; second is the result upon a true comparison and the third is the result upon a false comparison.

For example: *Result= condition ? "Value if true": "Value if false".*

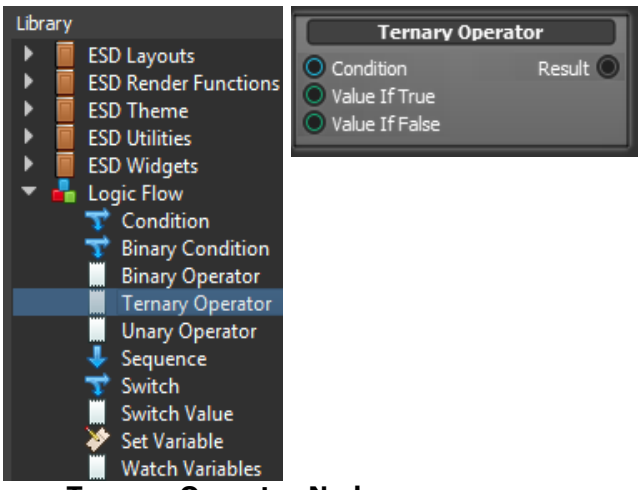


Figure 76 - Logic Flow - Ternary Operator Node

Unary Operator

This node contains *Unary Operator* that operates on a single operand (input). For example: *~.*

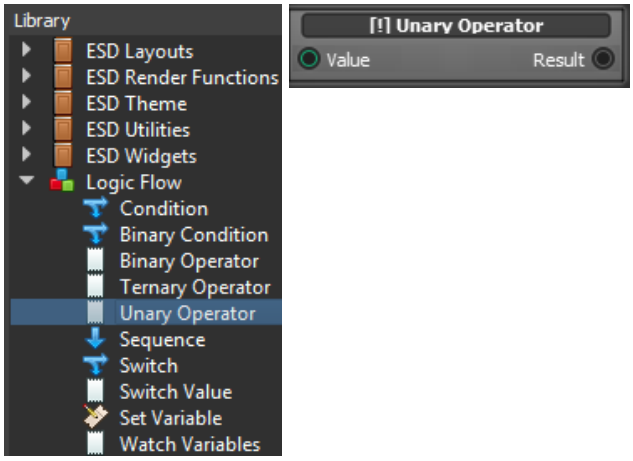


Figure 77 - Logic Flow - Unary Operator Node

The following table provides a list of unary operators supported by ESD 4.10.

Unary Operator	Description
!	Logic Negation
&	Address of
*	De-reference
+	Positive operator
-	Negative operator
~	One's complement
!!	Non-zero value converts to 1and other values converts to 0.

Table 29 – Unary Operators Supported by ESD 4.10

Copyright © Bridgetek Pte Ltd

Sequence

"*Sequence*" node allows users to define a series of calling actions. When the "Do" input call is invoked, the calls will be triggered sequentially. The number of calls is specified by users in property "Count".

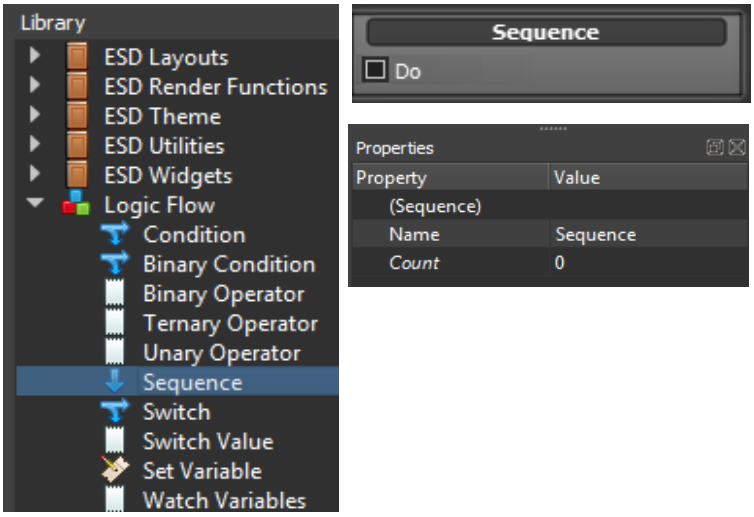


Figure 78 - Logic Flow – Sequence Node

Switch Node

"*Switch*" node will call different branches based on the input value, like a "Switch-Case" statement in C language. Users can define any number of branches and the values to be compared with input value through "Cases" property. Figure 101 shows an example of 3 branches defined and called respectively if input value is equal to 1 or 2 or 3. Since, the input value is pre-set to 4; the default branch is called if no connection is made on it.

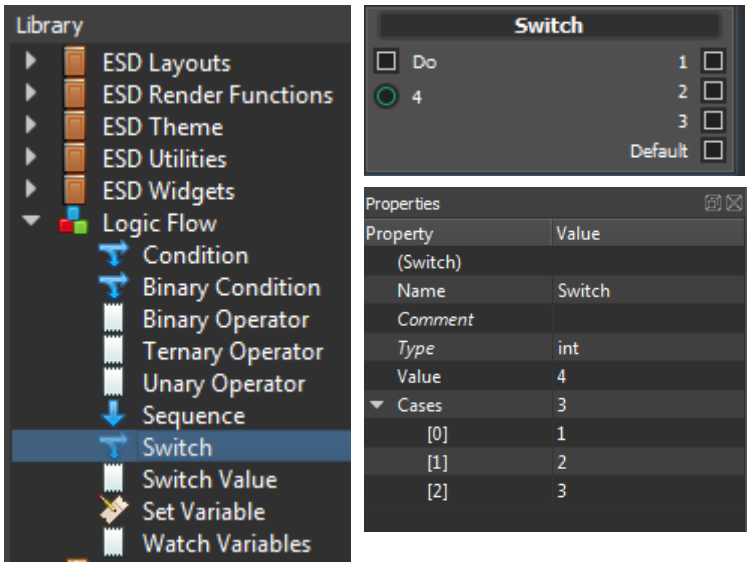
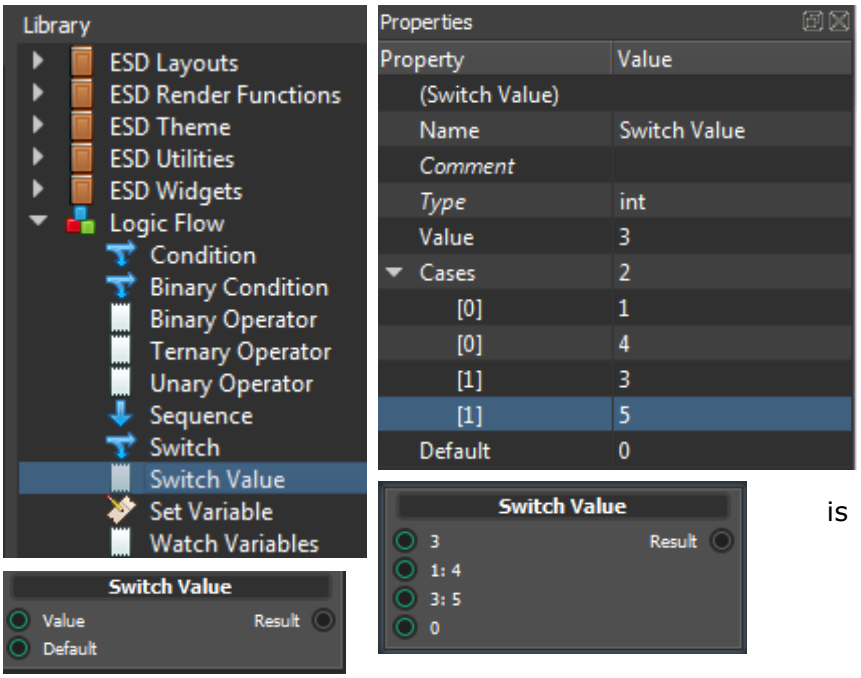


Figure 79 - Logic Flow – Switch Node

Copyright © Bridgetek Pte Ltd

Switch Value

"*Switch Value*" is a node to select one output value from multiple predefined values in "Cases" property based on input value. The example in Figure 80 shows 2 "*Cases*" branch and property "*Cases*" 0 defines such logic: The "*Result*" value is 4 if input value is 1. Therefore, the result value is 5 when the input value 3.



is

Figure 80 - Logic Flow – Switch Value

Set Variable

The *Set Variable* node is used to set the values of a variable. It links an event call with a data binding to a Writer (output a functional call through "writer") or a *Variable* (sets a variable to the value).

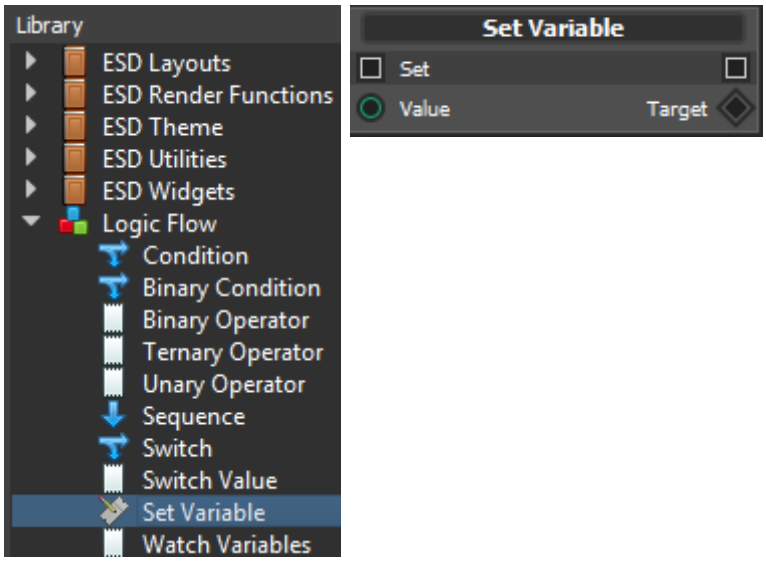
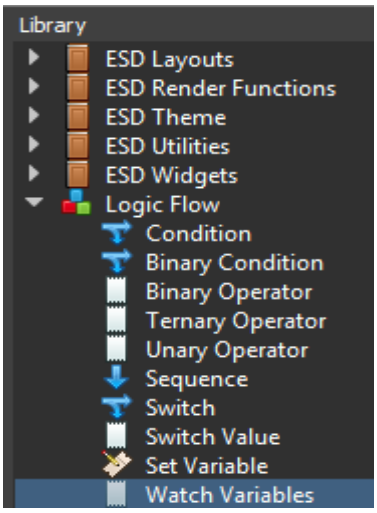


Figure 81 - Logic Flow - Set Variable Node

Copyright © Bridgetek Pte Ltd

Watch Variables

The Watch variables node is used to monitor a set of variable updates. A single watch variable node can monitor 'n' number of variables which can be different (primitive) types. However, the more the input variables have connected, the more frequent the output "Changed" will be triggered. User can also connect input slots:



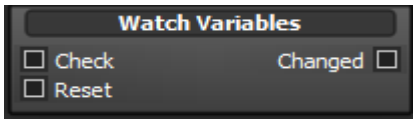


Figure 82 - Logic Flow - Watch Variable Node

- "Check" – used to trigger an immediate check for the input variables against previously checked values.
- "Reset" – set base values as current input values, and monitors changes from there.

This watch variable could be very useful during the debugging, or multiple input trigger events. It simplifies nested-if conditional flows.

Logic Interface

A *Logic Interface* is an ESD built-in logic node. It is used to define an interface and contains the following nodes:

- *Input*
- *Output*
- *Signal*
- *Slot*
- *Variable*
- *Writer*
- *Widget Interface*

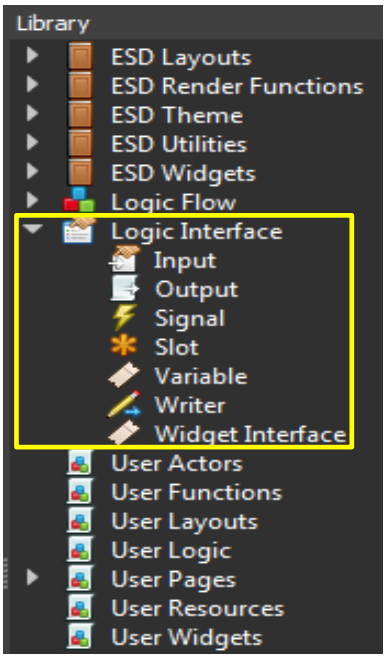


Figure 83 - Logic Interface

Copyright © Bridgetek Pte Ltd

Input

The *Input* node is used to create an input interface for its parent logic node to bind incoming data. For instance, while creating a widget, an input node will define a property for the widget whose value depends on the connection. Users can configure its property through the Property Editor.

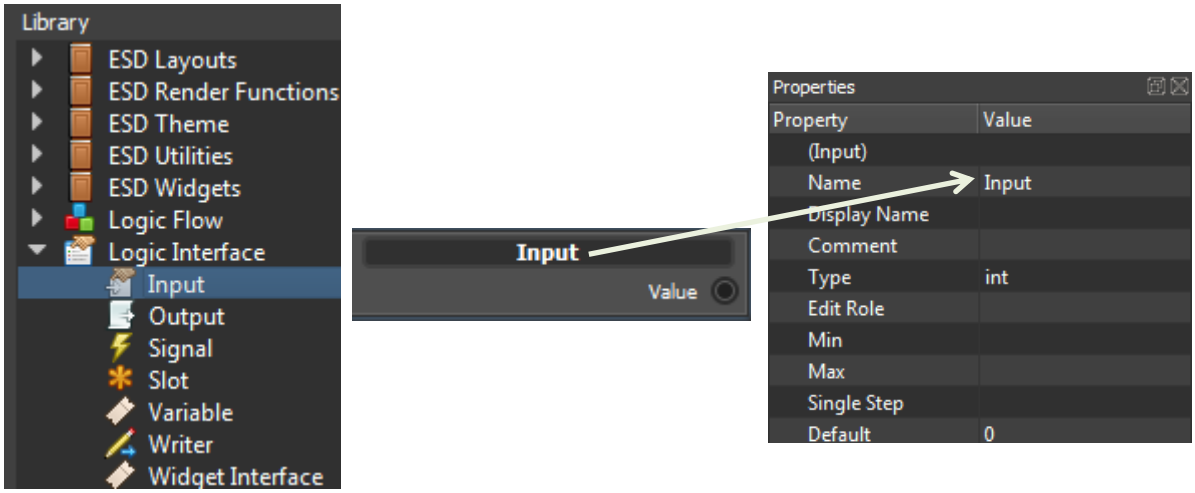


Figure 84 - Logic Interface - Input Node

Output

The *Output* node is used to create an output interface for its parent logic to bind the outgoing data. For instance, while creating a widget, an output node will define an interface for the widget to display values to the users. Users can configure its property through the Property Editor.

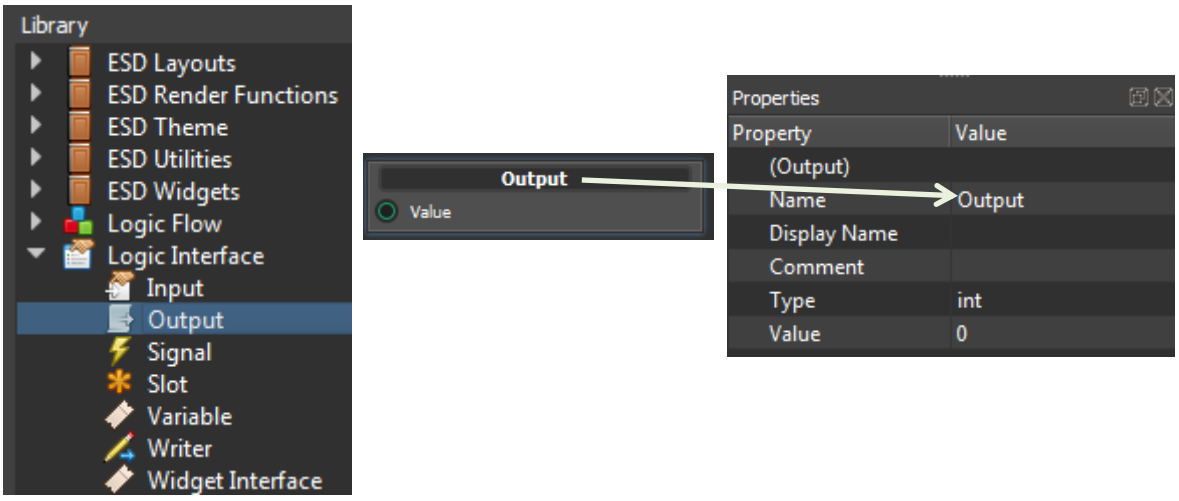


Figure 85 - Logic Interface - Output Node

Copyright © Bridgetek Pte Ltd

Signal

The *Signal* node is used to create an outgoing event function call. Users can configure its property through the Property Editor.

Figure 86 - Logic Interface - Signal Node

Slot

The *Slot* node is used to create an incoming event function call.

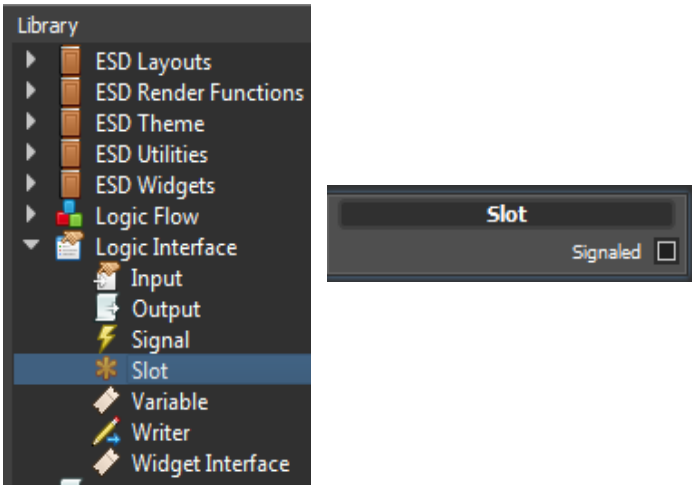


Figure 87 - Logic Interface - Slot Node

Built-in Slot

There are a number of *built-in slots* available for Widgets, Pages and Applications which are automatically called through the node hierarchy. The following table provides the list of built-in slots.

Copyright © Bridgetek Pte Ltd

Built-in Slot	Description
Start	Called when the node is created in memory
Render	Writing rendering calls to the co-processor command buffer. It is repeatedly called.
Update	Called once before the frame render. It is called frequently.
Idle	Called repeatedly while waiting for the frame to flip
End	Called when the node is removed from the memory

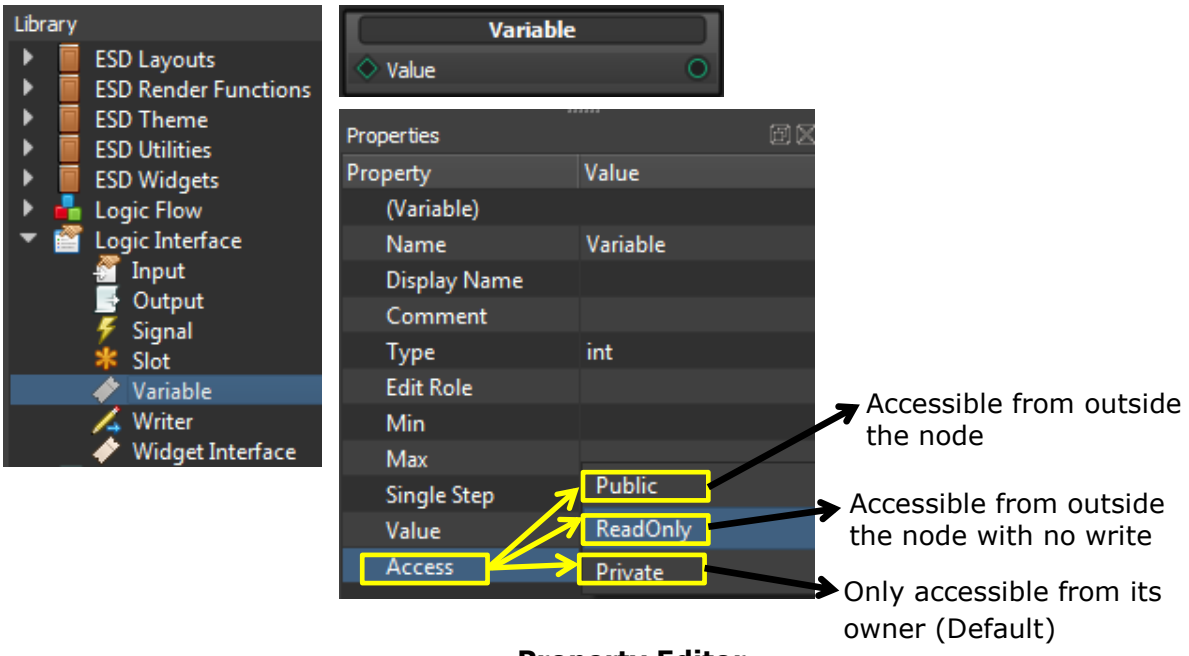
Table 30 - Built-in Slots

The following pseudo-code may help you understand how these slots are invoked:

```
//Application is booting up and started
Start()
//Now UI is launched
while(UI is running)
{
    Update();
    Render();
    Idle();
}
//Application is ending
End()
```

Variable

The *Variable* node is used to define a plain C variable. Users can drag and drop the Variable from the Library browser to the logic node editor and configure its property through the Property Editor.



Property Editor

Figure 88 - Logic Interface - Variable Node

Copyright © Bridgetek Pte Ltd

Writer

The *Writer* node is used to define a value writing interface for its parent node. Generally, it is used to setup the value for the external variable. For instance, while creating the ESD Spinbox widget, the "Changed" writer is defined to update the connected external variables of Spinbox widget once Spinbox value is changed.

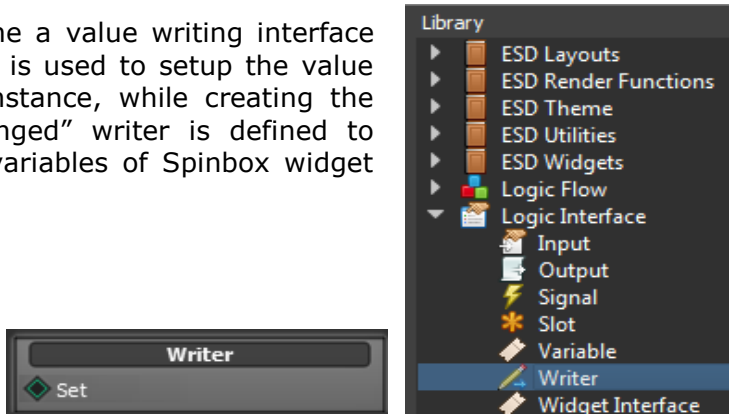


Figure 89 - Logic Interface - Writer Node

Widget Interface

The *Widget Interface* node shows the interface of base widget interface. Since ESD 4.0, all layouts, pages and widgets have been implemented from this widget interface. For advanced users, this widget interface node could be useful for designing user customized widget in order to use its base interface for accessing information from ESD widget framework.

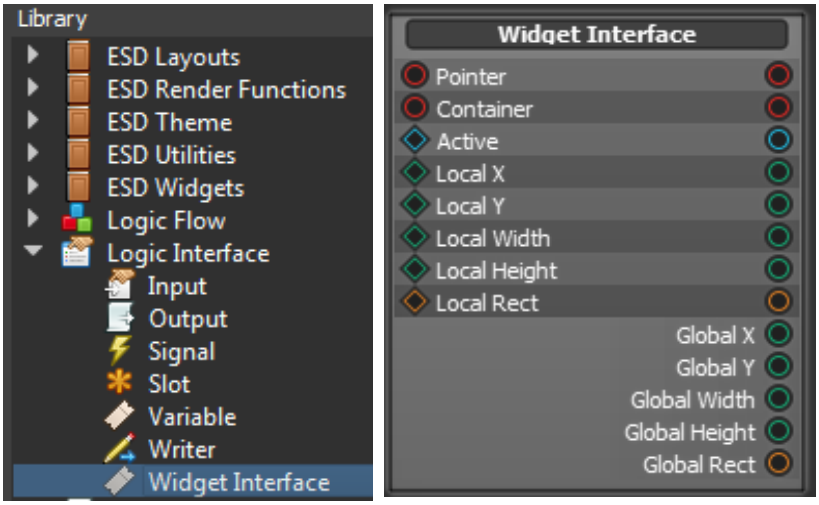


Figure 90 : Widget Interface

Property Name	Description
Pointer	Provides the pointer of this widget, allow other widget/layout/page to refer
Container	Provides the pointer of this widget's immediate container. Normally, a widget's immediate container is a layout
Active	Set and get the current widget active
Local X	Set and get the current widget local X. Local X refers to the local x-coordinate with reference to its container
Local Y	Set and get the current widget local Y. Local Y refers to the local y-coordinate with reference to its container
Local Width	Set and get the current widget local width. Local width refers to the designed width with reference to its container
Local Height	Set and get the current widget local height. Local height refers to the designed height with reference to its container
Local Rect	Set and get the current widget local dimension with reference to its container
Global X	Set and get the current widget global X. Global X refers to the global x-coordinate with reference to screen
Global Y	Set and get the current widget global Y. Global Y refers to the global y-coordinate with reference to screen

Copyright © Bridgetek Pte Ltd

Global Width	Set and get the current widget global width. Global width refers to the designed width with reference to screen
Global Height	Set and get the current widget global height. Global height refers to the designed height with reference to screen
Global Rect	Set and get the current widget global dimension with reference to its screen

Table 31 – Widget Interface Properties

Advanced users are encouraged to refer the example project **"WindowLayout"** for details.

- All Local properties are subjected to difference from the respective Global properties. As the global values are determined by the layers of the layouts as containers.
- Global properties are the values determined at the runtime. While Local properties are values determined at design time.
- Global properties are eventually stable in runtime, as layers of layouts may require multiple rendering cycle to stabilize.

Copyright © Bridgetek Pte Ltd

Property Editor

The *Property Editor* provides user a unified interface to edit properties. Each logic node in ESD 4.10, including a widget, has its own predefined properties. Users can edit the property values using the Property Editor to customize logic nodes.

The sample picture given below shows an example of the ESD 4.10 widget "ESD Radio Button".

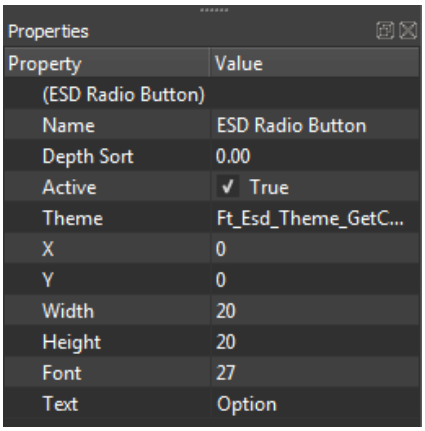


Figure 91 - Property Editor

Common Properties

While each widget has its own properties defined for its own specific features, the following are some of the properties that are commonly found in all the widgets of ESD 4.10 application -

- *Name*
- *Depth Sort*
- *Active*

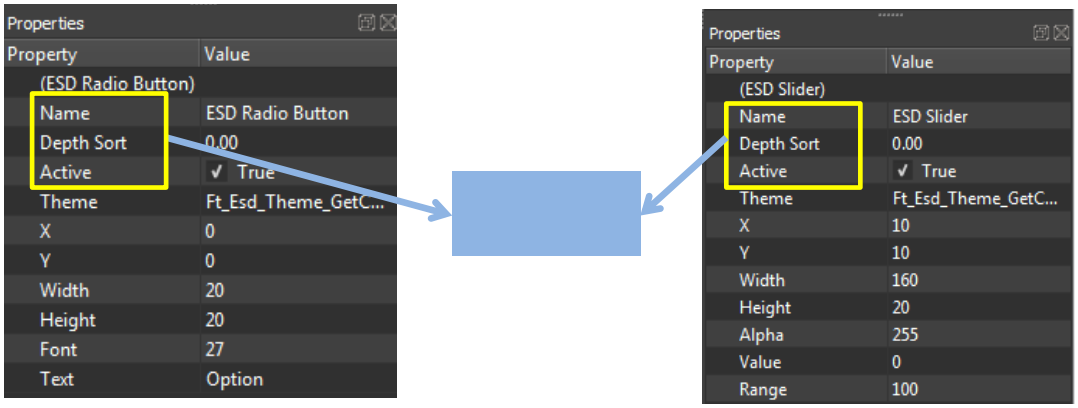


Figure 92 - ESD Common Properties

Copyright © Bridgetek Pte Ltd

Name

Each node must have one unique *name* as an identifier. Users are required to define a name that is valid C identifier, since these names will be used by ESD for generating the source code.

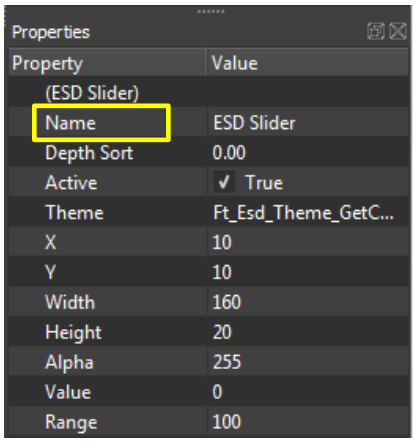


Figure 93 – Name Property

Depth Sort

Each widget and page has one common property called *Depth Sort*. It is of floating point type, and defines the sequence of rendering. The default value is "0.00" and users may change it accordingly to adjust the sequence of the rendering process.

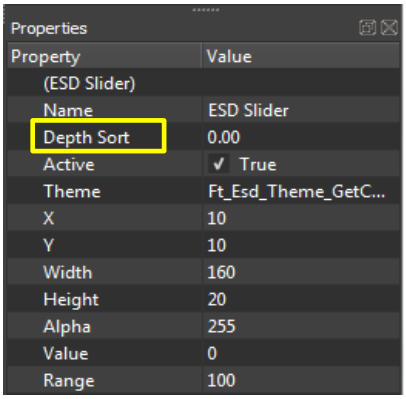


Figure 94 – Depth Sort Property

The context menu helps users adjust the rendering sequence without dealing with the values and can be accessed within the layout editor by selecting the widget or page and right-clicking it.

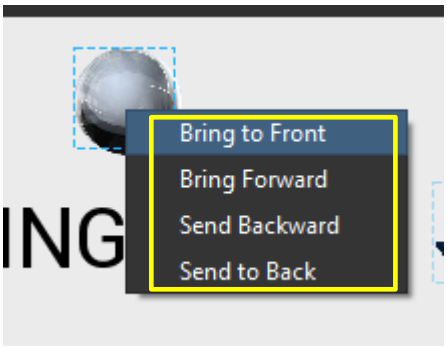


Figure 95 – Right Click Menu

The widget or page with the **greatest** positive "Depth Sort" value will be rendered **last** and appear on the **topmost** layer.

Copyright © Bridgetek Pte Ltd

Active

Each widget has one *Active* property of Boolean type which controls whether or not it is rendered on the screen. Active property in false state does not mean that the widget is released from the memory. Instead, it means that the widget is not rendered on the screen, but still is part of the memory.

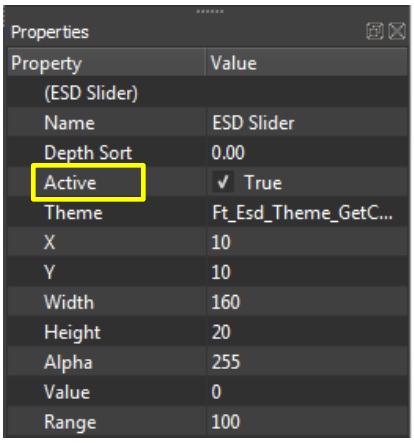


Figure 96 – Active Property

Users must ensure that the NAME property follows the standard **C identifier** syntax in order to make the generated source code more readable.

Programming Features

This section explains about the programming features. ESD 4.10 enables users to write C code and make this code available in the form of logic nodes so that the code can be re-used across multiple projects.

Macros

In order to provide the development environment with necessary information about the available functionality in a user's code, ESD 4.10 provides a set of macros. These macros are necessary and useful when users add their own code that needs to be used from the *logic node editor* or *layout designer*.

These macros are generally used together with the source code. It is recommended that the user defines all the definitions associated with macros immediately following the macros declarations where ever needed and in the same file. Not doing so, may result in unexpected behaviour. For e.g., ESD_FUNCTION macro immediately needs to be followed with function definition. Comments are acceptable between the macro and the definitions.

ESD_TYPE

ESD_TYPE provides information about a primitive type. All types are assumed to have a default value of 0.

Syntax Example:

```
ESD_TYPE(ft_uint32_t, Native = UInt32, Edit = Int)
ESD_TYPE(char *, Native = Latin1, Edit = String)
ESD_TYPE(Esd_BitmapCell *, Native = Pointer, Edit = Library)
```


Copyright © Bridgetek Pte Ltd

Parameter Type	Parameters	Description
Native		Type used internally for formatting C literals and binary representations
	Void	Not a type (default)
	Int64, Int32, Int16, Int 8,IntPtr	Signed integer
	UInt64, UInt32, UInt16, UInt8, UIntPtr	Unsigned integer
	Double	64bit floating point
	Float	32bit floating point
	Char	An 8bit text character
	Bool	C99 Bool
	Utf8	String with UTF-8 encoding
	Latin1	String with Latin 1 (ISO8859-1) encoding
	Pointer	Pointer type
	Struct	Struct type (not fully supported yet)
Edit		Used by the Properties Editor to specify the control to edit the value and by code generation when using the fixed point formats
	None	No editor control (default)
	Int	Integer, spin-box integer control
	Real	Floating point, spin-box floating point control
	Boolean	One or Zero, checkbox
	String	Text, single-line entry
	Fixed1, Fixed2, ..., Fixed32	Fixed point, spin-box floating point control
	ColorARGB	Color, rgb with alpha
	ColorRGB	Color, rgb without alpha
	Library	Select a function specified with ESD_FUNCTION to provide the value (useful for pointer values)

Table 32 - Macros – ESD_TYPE - Parameters

ESD_ENUM

Enumeration types are usable as types from the logic editor. It will use the specified Type in the generated code. Internally, only the enumeration identifiers are parsed; values are ignored. Enumerations used in the properties are serialized as their identifier string. Enumerations are edited using a drop-down control in the *Properties Editor*.

Syntax Example:

Using pre-processor definitions, the symbol right after each *#define* statement is used.

```
ESD_ENUM(Ft_BitmapFormat, Type = ft_uint32_t)
#define PALETTED              8UL
#define PALETTED4444          15UL
#define PALETTED565           14UL
#define PALETTED8             16UL
ESD_END()
```

Parsing from an enum requires the identifiers to be on separate lines, the first identifier of each line is used.

Copyright © Bridgetek Pte Ltd

```
ESD_ENUM(Ft_BitmapFormat)
enum Ft_BitmapFormat
{
    PALETTED,
    PALETTED565,
    PALETTED8
};
ESD_END()
```

In case the enumeration is defined in another file, a macro is available to manually specify the identifiers.

```
ESD_ENUM(Ft_BitmapFormat, Type = ft_uint32_t)
ESD_IDENTIFIER(PALETTED)
ESD_IDENTIFIER(PALETTED565)
ESD_END()
```

When the Flag value is specified, the enumeration can combine multiple values, and will show as a series of check-boxes in the Properties Editor instead of a drop-down menu.

```
ESD_ENUM(Ft_CoPro_Opt, Type = ft_uint16_t, Flags)
#define OPT_CENTERX          512UL
ESD_END()
```

To allow users to select 0 as a valid value, the Zero flag can be set. This flag has no effect when the Flag value is specified.

ESD_FUNCTION

ESD_FUNCTION is a macro to register a user defined function to ESD 4.10, so that the application can add that function into the *User Functions* category node which is located in the Library browser. *ESD_FUNCTION* is not applicable for static functions even though ESD permits to make any function with *ESD_FUNCTION* macro.

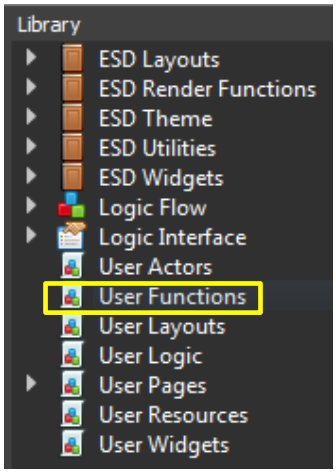


Figure 97 – User Functions Example

Syntax Example:

The following syntax example is to register a function called *"hsvToRgb"* to ESD 4.10.

```
ESD_FUNCTION(hsvToRgb, Type = ft_argb32_t )
ESD_PARAMETER(h, Type = double)
ESD_PARAMETER(s, Type = double)
ESD_PARAMETER(v, Type = double)
ft_argb32_t hsvToRgb(double h, double s, double v)
```

Upon registering the function, it is added to the *User Functions* category.

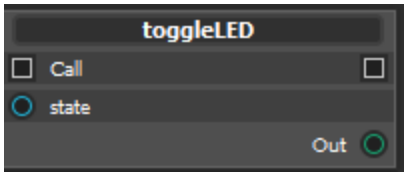
Copyright © Bridgetek Pte Ltd

Parameters	Description
Type	The function's return value type

Table 33 - Macros – ESD_FUNCTION - Parameters

NOTE

- Each ESD_PARAMETER defines only one function parameter. It must be kept in a single line for each parameter.
- "Buffered" flag can be attached after "Type". It will create the output of the function stored in a member variable, whenever the function is called. Without the Buffered flag, you'll have a function with just one output value which is called whenever the output value is used. Here is an example:




```
ESD_FUNCTION(toggleLED, Type=int,Buffered)
ESD_PARAMETER(state,Type=ft_bool_t)
```

ESD_METHOD

ESD_METHOD works similar to ESD_FUNCTION with an enhanced feature. It takes a logic/actor/widget/page/app context pointer as first parameter. It can be used only within the logic editor of that logic/actor/widget/page/app.

Syntax Example:

The following syntax example defines one method to handle "Pushed" event for an ESD Push Button widget contained in "MainPage.page".

```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context = MainPage)
void MainPage_ESD_Push_Button_Pushed(MainPage *context)
{
        // ...
}
```

The logic node connection and the corresponding output is shown in the sample picture below –

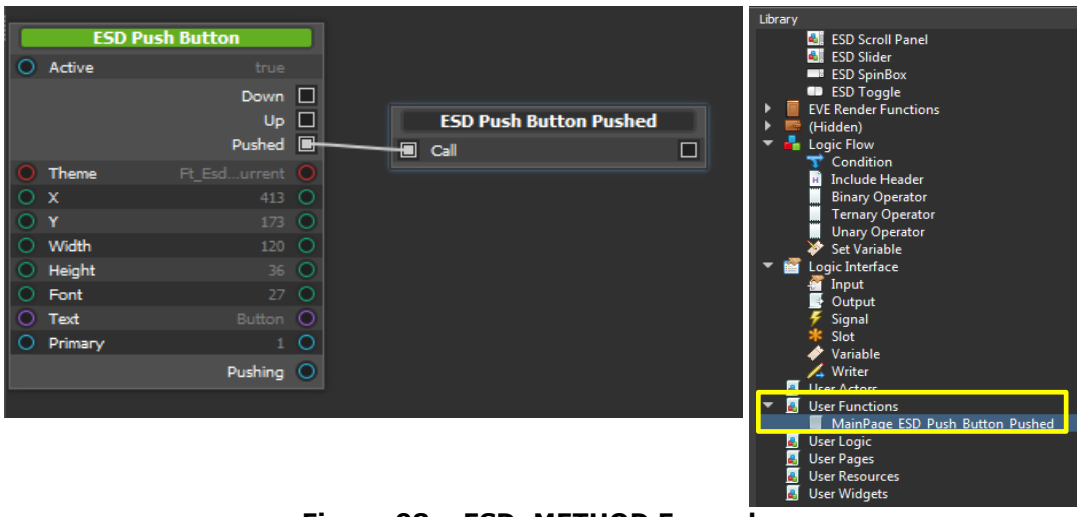


Figure 98 – ESD_METHOD Example

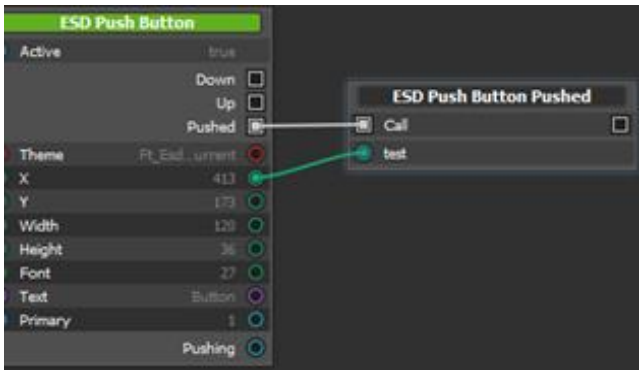
Copyright © Bridgetek Pte Ltd

Parameters	Description
Context	Pointing to the caller's context

Table 34 - Meta Macros – ESD_METHOD - Parameters

NOTE

- Additional parameters can be defined similar to ESD_FUNCTION. But these parameters must be added only after the Context parameter. Here is an example:




```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context =
MainPage)
ESD_PARAMETER(test,Type = ft_bool_t)
void MainPage_ESD_Push_Button_Pushed(MainPage
*context,ft_bool_t test)
{
     // ...
}
```

NOTE

- If the return value of method function is to be defined, "Buffered" keyword needs to be added after the "Type" definition. Refer to the syntax example -

```
ESD_METHOD(MainPage_ESD_Push_Button_Pushed, Context =
MainPage,Type=int,Buffered)
ESD_PARAMETER(test,Type = ft_bool_t)
int MainPage_ESD_Push_Button_Pushed(MainPage
*context,ft_bool_t test)
{
     // ...
}
```

ESD_VARIABLE

ESD_VARIABLE is a built-in macro used to define a local variable of a logic node.

Syntax Example:

```
ESD_VARIABLE(Margin, Type = ft_int16_t, Default = 0, Public)
ft_int16_t Margin;
```


Copyright © Bridgetek Pte Ltd

ESD_GLOBAL

ESD_GLOBAL is a built-in macro used to define a global variable accessed across different pages of a logic node. Global variable declaration should present in the header file and definition should be in source file.

Syntax Example:

```
ESD_GLOBAL(Var, Type = int)
extern int Var;
```

ESD_INPUT

ESD_INPUT is a built-in macro used to define an input variable of a logic node.

Syntax Example:

```
ESD_INPUT(Margin, Type = int)
int(* Margin)(void *context);
```

ESD_OUTPUT

ESD_OUTPUT is a built-in macro used to define an output variable of a logic node.

Syntax Example:

```
ESD_OUTPUT(FirstPos, Type = int)
int LayoutSplit_FirstPos(LayoutSplit *context);
```

ESD_UPDATE

ESD_UPDATE macro is used to register a function that is called repeatedly by built-in "Update" slot. This function shall not have any return value.

Syntax Example:

```
ESD_UPDATE(Ft_Esd_BitmapPersist)
ESD_PARAMETER(bitmapCell, Type = Ft_Esd_BitmapCell *)
void Ft_Esd_BitmapPersist(Ft_Esd_BitmapCell *bitmapCell);
```

Parameters:

Users can define their own parameter used by the registered function by using the ESD_PARAMETER.

Pre-compiler options

Within the ESD C code simulation engine, the following macros are predefined in generated source code:

- *ESD_SIMULATION*
- *FT900_PLATFORM*

Copyright © Bridgetek Pte Ltd

ESD_SIMULATION

ESD_SIMULATION macro is defined when the code is running through the ESD simulation engine. If some part of the user's C source code is not supposed to be run under the ESD simulation engine, for example, accessing some hardware specific registers, users may choose to skip the code by using this macro.

FT900_PLATFORM

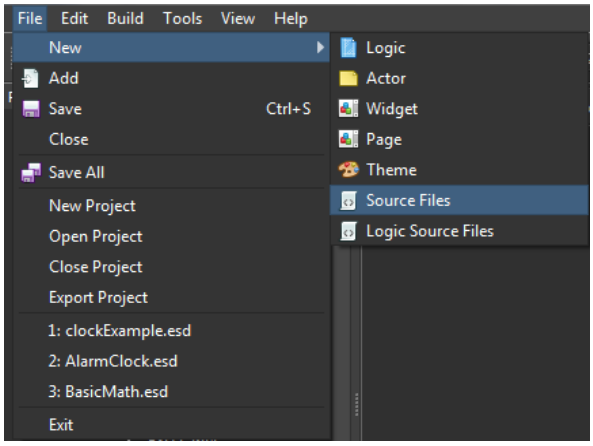
FT900_PLATFORM macro is defined when the target platform is FT90X based. It can be set when the ESD_SIMULATION macro is defined.

Add User Functions

Users can define their own functions by writing the C source code directly. If these functions are defined by using the predefined "ESD_FUNCTION", they will be shown under the "User Functions" category in the library browser. Users can drag and drop these nodes into the logic node editor to use them.

Creating Source File

To add user functions, users need to create a C source file by clicking **File → New → Source Files**.



Upon adding the source file, include the following header **(mandatory)** in the source file.

```
#include "Ft_Esd.h"
```

Editing the Source File

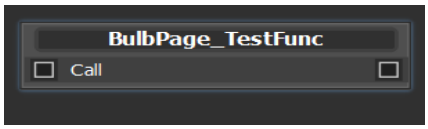
To make ESD responsive of the user functions, the ESD specific macro *"ESD_FUNCTION or ESD_METHOD"* need to be added before the function definition. Refer to the examples given below –

Syntax Example:

- Without parameter input and return value

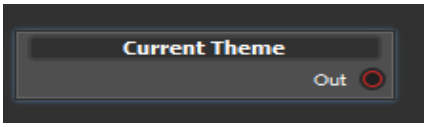
```
ESD_FUNCTION(BulbPage_TestFunc)
void BulbPage_TestFunc(){}
```


Copyright © Bridgetek Pte Ltd



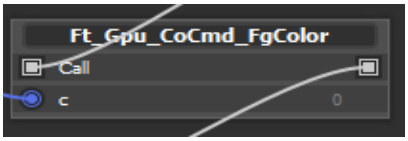
- Without parameter input, but with the return value

```
ESD_FUNCTION(Ft_Esd_Theme_GetCurrent, Type = Ft_Esd_Theme *)
Ft_Esd_Theme *Ft_Esd_Theme_GetCurrent();
```



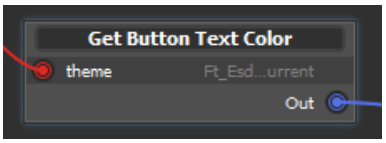
- Without return value defined, but with the parameter input

```
ESD_FUNCTION(Ft_Esd_Theme_GetButtonTextColor, Type = ft_rgb32_t)
ESD_PARAMETER(theme, Type = Ft_Esd_Theme *)
ft_rgb32_t Ft_Esd_Theme_GetButtonTextColor(Ft_Esd_Theme *theme) { //.. }
```



- Without return value and parameter input

```
ESD_FUNCTION(Ft_Esd_Theme_GetButtonTextColor, Type = ft_rgb32_t)
ESD_PARAMETER(theme, Type = Ft_Esd_Theme *)
ft_rgb32_t Ft_Esd_Theme_GetButtonTextColor(Ft_Esd_Theme *theme)
{
    //..
}
```



Copyright © Bridgetek Pte Ltd

F. Appendix A – List of Figures

Figure 1 - Screen Designer - System Error 10
Figure 2 - ASTC applied for images and fonts 18
Figure 3 - Load resources from flash 18
Figure 4 - Use Unicode font with pre-defined character set 19
Figure 5 - Chinese font used in application 19
Figure 6 - ASTC Format 19
Figure 7 - Font size and Character set 20
Figure 8 - Select Unicode Font Resource 20
Figure 9 - Unicode font applied for Label and Button 20
Figure 10 - Mark resource as storage in Flash 21
Figure 11 - Upload Flash file to device 21
Figure 12 - Making Value Specific to Current Target 23
Figure 13 - Removing Target Specific Value 23
Figure 14 - EVE Screen Designer 4.10 User Interface Components 24
Figure 15 - Menu bar 25
Figure 16 - Toolbar 1 26
Figure 17 – Toolbar 2 27
Figure 18 - MCU Type is not applicable to selected platform 28
Figure 19 - Project Explorer window 29
Figure 20 - Project Explorer Filter 29
Figure 21 - Screen Layout Editor – App.main 30
Figure 22 - Screen Layout Editor - MainPage.c (C Source Code) 30
Figure 23 - Context menu on Tab bar 31
Figure 24 - Logic Node Editor 31
Figure 25 - Library Browser 32
Figure 26 - Error List Window 33
Figure 27 - Output Window 33
Figure 28 - Property Editor (ESD Push Button) 34
Figure 29 - Status bar 34
Figure 30 - Application Project Structure 35
Figure 31 - Application Workflow 36
Figure 32 - Property Editor 38
Figure 33 : New Target button 41
Figure 34 : Configure custom target 41
Figure 35 : Select custom build target 42
Figure 36 - Build & Upload 42
Figure 37 – Switch Page 46
Figure 38 - Sample Switch Page UI 47
Figure 39 - Properties Editor – Page Allocation 47
Figure 40 - Sample Source File 48
Figure 41 - Zoom In & Out 48
Figure 42 - Main File 49
Figure 43 – Sample Logic File 49
Figure 44 - C File / C Editor 50
Figure 45 – Logic Node Editor - Basic Logic Node 50
Figure 46 - Library Browser - Basic Logic Nodes 51
Figure 47 - Composite Logic Node 51
Figure 48 - Composite Logic Node - Page Node 52
Figure 49 - Logic Node Editor – Widget Node 52
Figure 50 - User Widget - Position & Size Properties 53

Copyright © Bridgetek Pte Ltd

Figure 51 - Layout Widgets 54
Figure 52 - Layout Widgets 55
Figure 53 - Actor Node 55
Figure 54 - Logic Object 56
Figure 55 - Rendering Order Example 57
Figure 56 - Library Browser 59
Figure 57 – Library Filter 59
Figure 58 - Scroll Switch Page Layout 73
Figure 59 - Sample Scroll Switch Page Implementation 75
Figure 60 - ESD Elements 75
Figure 61 - ESD Circle Element 76
Figure 62 - ESD Panel Element 76
Figure 63 - ESD Gradient Panel Element 76
Figure 64 - ESD 4.10 Primitives 77
Figure 65 - ESD Bitmap Primitive 77
Figure 66 - ESD Line 79
Figure 67 - ESD Rectangle 79
Figure 68 - ESD Display List 80
Figure 69 - Library Browser - ESD Theme 80
Figure 70 - ESD 4.10 Built-in Themes 81
Figure 71 - Logic Flow 84
Figure 72 - Logic Flow – Condition Node 85
Figure 73 - Logic Flow – Binary Condition Node 85
Figure 74 - Logic Flow – Local Method 86
Figure 75 - Logic Flow - Binary Operator Node 87
Figure 76 - Logic Flow - Ternary Operator Node 88
Figure 77 - Logic Flow - Unary Operator Node 88
Figure 78 - Logic Flow – Sequence Node 89
Figure 79 - Logic Flow – Switch Node 89
Figure 80 - Logic Flow – Switch Value 90
Figure 81 - Logic Flow - Set Variable Node 90
Figure 82 - Logic Flow - Watch Variable Node 91
Figure 83 - Logic Interface 91
Figure 84 - Logic Interface - Input Node 92
Figure 85 - Logic Interface - Output Node 92
Figure 86 - Logic Interface - Signal Node 93
Figure 87 - Logic Interface - Slot Node 93
Figure 88 - Logic Interface - Variable Node 94
Figure 89 - Logic Interface - Writer Node 95
Figure 90 : Widget Interface 95
Figure 91 - Property Editor 97
Figure 92 - ESD Common Properties 97
Figure 93 – Name Property 98
Figure 94 – Depth Sort Property 98
Figure 95 – Right Click Menu 98
Figure 96 – Active Property 99
Figure 97 – User Functions Example 101
Figure 98 – ESD_METHOD Example 102

Copyright © Bridgetek Pte Ltd

G. Appendix B – List of Tables

Table 1 - Installation Folder 17
Table 2 - Menu & Description 26
Table 3 - Toolbar & Description 27
Table 4 - ESD 4.10 Libraries 32
Table 5 - Bitmap Resource Properties 39
Table 6 - Output files for different Build Configuration & Host Platforms 43
Table 7 - Build Targets, Host platform and the corresponding export projects 44
Table 8 – List of Folder & File created upon exporting project 44
Table 9 - Connection End Types 56
Table 10 – Switch Page Layout Properties 65
Table 11 – Dynamic Switch Page Layout Properties 66
Table 12 – Auto Switch Page Layout Properties 67
Table 13 – Fixed Positioning Layout Properties 68
Table 14 – Fill Layout Properties 69
Table 15 – Linear Layout Properties 70
Table 16 – Stretch Layout Properties 71
Table 17 – Scroll Layout Properties 73
Table 18 – Scroll Switch Page Layout Properties 74
Table 19 - ESD Circle Element Properties 76
Table 20 - ESD Panel Element Properties 76
Table 21 - ESD Gradient Panel Element Properties 77
Table 22 - ESD Bitmap Properties 77
Table 23 - ESD Line Properties 79
Table 24 - ESD Rectangle Properties 80
Table 25 - Ft_Esd_Theme_LightBlue Theme 81
Table 26 - Ft_Esd_Theme_DarkOrange Theme 81
Table 27 – ESD Utilities & Description 83
Table 28 - Binary Operators Supported by ESD 4.10 87
Table 29 – Unary Operators Supported by ESD 4.10 88
Table 30 - Built-in Slots 94
Table 31 – Widget Interface Properties 96
Table 32 - Macros – ESD_TYPE - Parameters 100
Table 33 - Macros – ESD_FUNCTION - Parameters 102
Table 34 - Meta Macros – ESD_METHOD - Parameters 103

Copyright © Bridgetek Pte Ltd

H. Appendix C – Revision History

Document Title	:	BRT_AN_029 EVE Screen Designer 4.10 User Guide
Document Reference No.	:	BRT_000218
Clearance No.	:	BRT#159
Product Page	:	https://brtchip.com/eve-toolchains
Document Feedback	:	Send Feedback

Revision	Changes	Date
Version Draft 0.3	Initial user guide release for ESD 4.5 version features & enhancements	22-06-2018
Version Draft 0.6	Updated user guide as per ESD 4.8 version features and enhancements	27-09-2019
Version 1.0	User Guide updated as per the following changes for ÈSD 4.10 – II. Overview/What's new in ESD 4.10 V.ESD Workflow/TargetSelection V.ESD Workflow/Build&Upload V. ESD Workflow/Export	03-12-2020

Copyright © Bridgetek Pte Ltd