Security	机密

EUI 用户使用手册

文档版本 2.0

发布日期 2020-05-28

更新记录

版本	更新日期	说明
V1.0.0	2019 年 08 月 05 日	初始版本
V2.0.0	2019 年 12 月 18 日	完善控件属性和 api 接口功能
V2.1.1	2020 年 04 月 14 日	● 修复进度条（bar）设置对称属性无效 ● 修复图表（chart）控件设置系列点卡死 ● 修复图表（chart）控件设置点数会改变点的顺序 ● 修复下拉列表（ddlist）设置自适应高度无效 ● 修复仪表（gauge）设置针数跑飞与设置范围存在负数时不对 ● 修复图像（img）第二次显示时不正常和删除图片无效 ● 修复图像按键（imgbtn）设置文本 ID 无效 ● 修复标签（label）是长模式滚动时跑飞 ● 修复 LED 灯设置亮度跑飞 ● 增加图像按键边框风格效果 ● 增加初始化配置风格 ● 增加视图打开和关闭功能 ● 增加键盘中文输入法相关风格配置 ● 增加边框为四个直角风格配置
V2.2.0	2020 年 05 月 28 日	● 添加 EUI 配置文件（UI 旋转角度，偏移，触摸分辨率以及触摸节点等参数） ● 支持图片软解（目前支持的格式有 jpg, png, bmp, gif, psd, tga, hdr, pic 和 pnm） ● 添加视图按键消息回调函数 ● 修复打开链接视图有可能失败 ● 修复标签控件更新字符串后没有重新对齐 ● 修复 API 接口使用错误控件 ID 崩溃 ● 集成 blocklinker 使用方法到 EUI ● 修复低概率异步操作崩溃 ● 增加视图显示时自动置顶 ● 增加鼠标和触摸事件记录和回放功能 ● 修复动态列表删除选项错误 ● 修复 slider 控件响应函数参数不对 ● 增加 API 同异步处理机制(API get 参数接口只能在 EUI 线程中（即响应函数）使用有效)
V2.3.0	2021 年 04 月 21 日	● 在 linux 上适配屏保、触摸、按键、blocklinker，

		声音等功能 ● 修复按键音在点击无效区域时无效 ● 修复 EUI 去初始化时重复释放内存 ● 修复 roller 控件滑动无效 ● 修复光标位置不随触摸位置改变而改变 ● 修复 ddlist 创建时设置选中项无效 ● 优化语言切换方式，所有带有字符串控件都能切换与刷新 ● 增加控件标签自动对齐功能 ● 增加动态列表首尾切换以及跳转刷新功能 ● 增加压缩 gif 图片显示 ● 增加 EUI 配置文件中鼠标、按键设备节点配置 ● 增加控件动态创建、删除以及相关操作接口

EUI 用户使用手册

目录

EUI 用户使用手册

1. 概述

1.1. 简介

EUI 图形库提供了嵌入式 GUI 所需的一切，具有美观的视觉效果和低内存，低 CUP 占用等特点。

EUI 依赖 JSON 文件设计，主要是通过属性值描述控件。然后将控件组合为一个页面来设计 UI。

由软件提供对 JSON 文件解析、组织控件、形成界面，并实现基本的视图界面切换。并且可以通过 EUI API 重新设置属性值来更改控件的显示状态。

EUI 核心提供 30 个控件对象。

1.2. EUI 框图

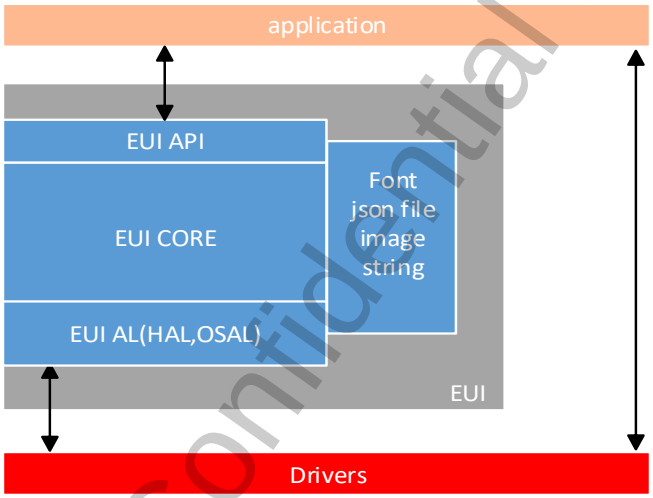

图 1.1

1.3. EUI 流程图

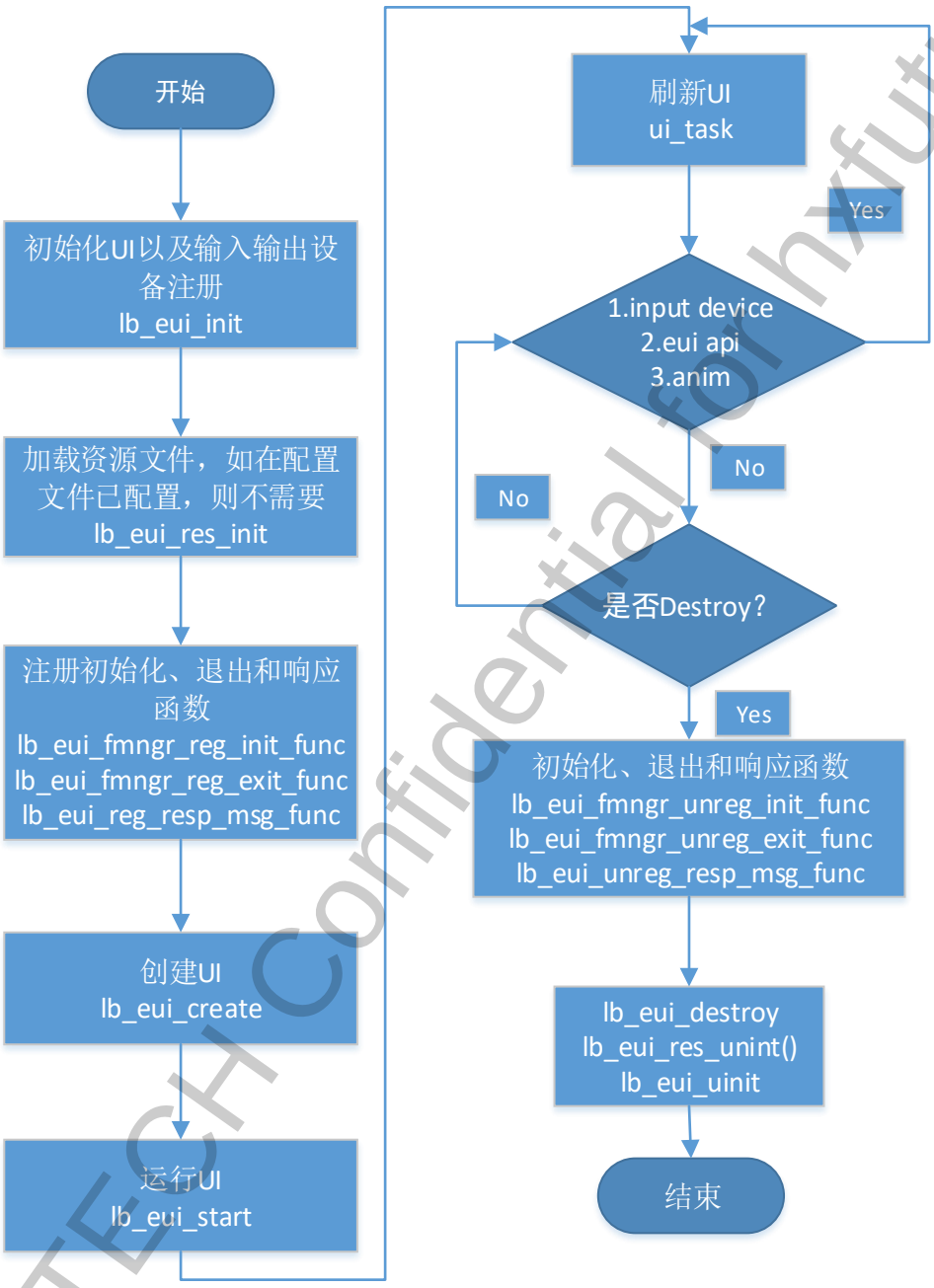

图 1.2

1. 开始

   设置 EUI 库的基本行为，禁止未使用的模块和功能，配置屏幕分辨率，色位等。

2. 初始化 EUI 库

   1）初始化 EUI 核心库，内存，创建任务线程；

   2）硬件输入和输出设备注册；

   3）创建屏幕控件；

3. 加载资源文件

   初始化资源文件包 res(包括图片，字库，字符串等)， 如在配置文件中已配置，则不需要加载;

4. 注册初始化、退出和响应函数

   调用 lb_eui_fmngr_reg_init_func(func_name, func)

   调用 lb_eui_fmngr_reg_exit_func (func_name, func)

   调用 lb_eui_reg_resp_msg_func (resp_msg, resp_func)

   注册控件的初始化、退出以及事件响应函数

5. 创建 UI

   1）通过解析 UI 入口 json 文件，加载以下资源文件：

      1）样式资源（style）；

      2）字库资源；

      3）视图 json 文件。

   2） 创建消息队列（EUI 通过消息通知运行）；

   3）解析视图 json 文件，设置控件属性，创建视图，组织视图与视图，视图与控件，控件与控件之间的关系；

   4）显示主视图。

6. 运行 EUI

   运行 EUI，等待输入，或者更新消息，当没有输入，或者更新消息时，EUI 挂起。

7. 触发 EUI 刷新

   当输入设备，eui api 或者动画触发时，启动 EUI 刷新任务；

8. 注销初始化、退出和响应函数

   调用 lb_eui_fmngr_unreg_init_func(func)

   调用 lb_eui_fmngr_unreg_exit_func (func)

   调用 lb_eui_unreg_resp_msg_func (resp_func)

9. 销毁当前 UI

   销毁消息队列，销毁 UI 资源（样式，字库，控件，控件属性）；

10. 去初始化 EUI 库

2. EUI 资源文件

EUI 资源文件包括通过 EUIEditor 工具生成并打包成 res.iso 的压缩文件和描述 UI 的 json 文件。

2.1. 压缩资源文件

压缩资源文件（res.iso） 打包了字库，图片，多国语言字符串。

1）点阵字库文件

名称	修改日期	类型	大小
font_25.edf	2019/7/16 16:44	EDF 文件	207 KB
font_30.edf	2019/7/16 16:44	EDF 文件	241 KB
font_60.edf	2019/7/16 16:44	EDF 文件	59 KB

图 2.1

字库文件由 EUIEditor 生成，生成步骤见 EUIEditor 使用手册，或者 EUIEditor 工具中点击帮助，会自动打开帮助文档。

2）图片文件

3）多国语言字符串

	Varchar	English	Simplified	Traditional
8	STR_ERR_PLAY	play error	播放文件出错	播放文件出錯
9	STR_MAIN_INFORMATION	Information	信息	信息
10	STR_FILE_ALL	All files	所有文件	所有文件
11	STR_FILE_VIDEO	Video	视频	视频
12	STR_FILE_IMAGE	Image	图片	圖片
13	STR_CARD_INFO	Card info	卡信息	卡信息
14	STR_WARN	Are you sure	确定吗	確定嗎
15	STR_WARN_OK	Yes	确定	確定
16	STR_WARN_CLOSE	No	取消	取消
17	STR_CARD_FORMAT_WORKING	Formating	格式化中	格式化中

图 2.2

2.2. 矢量字体

目前主要使用的是 ttf 字库文件。

名称	修改日期	类型	大小
font	2020/4/10 17:15	文件夹	
chinese.ttf	2020/1/10 17:32	TrueType 字体文件	9,390 KB
res.iso	2020/4/10 17:20	光盘映像文件	71 KB

2.3. 音频文件

各种提示音的音频文件，例如按键音。

tone.wav	2019/7/16 16:44	波形声音	29 KB

图 2.3

2.4. JSON 文件

1）入口 JSON 文件

入口 JSON 文件不属于界面 JSON 文件，入口文件指明 JSON UI 的需要的资源（比如字体文件、样式设计文件、静态界面文件、独立界面文件、风格数量）的信息。入口 JSON 包含：

- 版本信息
- 样式文件
- 字体信息
- 多国语言选择
- 按键声音频文件
- 静态界面入口文件
- 独立界面列表
- 风格数量

2）样式 JSON 文件

样式用于设置对象的外观，样式是一个结构变量，具有颜色，填充，不透明度，字体等属性。

每种对象类型都有一个通用的样式类型结构变量，通过设置这个结构变量可以改变对象的外观。

EUI 自带有十三种自带的基本样式结构变量，见 style.json。

样式参数说明：

编号	参数名称	描述
1	id	风格 ID 号，每一种风格都有一样独特的 id，方便区分使用，这个 id 在 home 应用中默认范围是 0-99，二级应用中是从 100 开始；若不够用，可通过在 root.json 中配置“styles_num”参数，同时二级应用自定义风格 id 必须从 styles_num 参数值开始
2	glass	继承风格标志，0-可继承这种风格，1-不要继承这种风格，默认为 0；如果控件的风格为 NULL，则其风格将从其父控件的风格继承。这样可以更轻松地创建一致的设计。同时设置 glass 风格属性将防止继承该样式（即导致子控件从其祖父母控件那里继承其样式）。如果样式是透明的，则应使用它，以便子控件使用祖父母的颜色和特征；否则，子控件也将是透明的
3	copy_from	风格 id 号，表示从当前风格列表中的某一个风格拷贝。指定 copy_from 属性后，只需要填写差异属性即可
body		
4	opa	不透明度。255 表示完全不透明，0 表示透明，整数值
5	main_color	主颜色(RGB)，十六进制数字符串
6	grad_color	变化的最后颜色(RGB)，十六进制数字符串。（如果定义纯色即不含颜色渐变，则将 grad_color 设置为 main_color 的值）
7	radius	定义了 body 四角是否有圆角，如果有设置为圆角的半径。如果为 0，则表示为直角，十六进制数字符串
8	empty	背景透明，只会画边框，为 1 时有效，整数值
padding		
9	ver	外部垂直方向上的边距，整数值
10	hor	外部水平方向上的边距，整数值

11	inner	内部边距，整数值
border		
12	color	边界的颜色(RGB)，十六进制数字符串
13	opa	边界的不透明度，255 表示完全不透明，0 表示透明，整数值
14	width	边界的宽度，整数值
15	part	边界的位置，（底部(0x1)、顶部(0x2)、左边(0x4)、右边(0x8)、四周(0xf)或者内部(0x10)）、四个直角(0x20)。
shadow		
16	color	阴影的颜色(RGB)， 十六进制数字符串
17	type	阴影类型，0-底部， 1-四周，整数值
18	width	阴影宽度，整数值
text		
19	opa	不透明度，255 表示完全不透明，0 表示透明，整数值
20	color	字符颜色(RGB)， 十六进制数字符串
21	letter_space	字符之间的间距，整数值
22	line_space	字符行距，整数值
23	font	字体大小(矢量字体可任意填写大小，点阵字体还能根据 root 文件配置的大小，否则为默认字体大小)，整数值
image		
24	opa	不透明度，255 表示完全不透明，0 表示透明，整数值
25	color	颜色（背景）(RGB)， 十六进制数字符串
26	intense	颜色混合比例，范围是 0-255，整数值
line		
27	opa	不透明度，255 表示完全不透明，0 表示透明，整数值
28	color	线的颜色(RGB)， 十六进制数字符串
29	width	线的宽度，整数值
30	rounded	是否在线端画圆角

3）静态视图 JSON 文件

每个界面的 JSON 文件包含背景和控件列表两个部分。

4）独立视图

独立视图 json 描述与静态视图一致， 独立视图是在入口 json 中定义，参考入口 json 文件中的"isolates"参数配置。

2.5. 配置文件

为了更加灵活的使用 EUI，增加配置文件来适配响应的需求，配置文件（文件内容为 json 格式）主要是配置相关输入输出设备，具体参数如下：

{

```
    "rotate": 0,                /* UI 旋转角度  0 - 不旋转  1 - 旋转 90 度 2 - 旋转 180 度 3 - 旋转 270 度 */
    "ui_offset_x": 0,           /* UI X 方向的坐标偏移 */
    "ui_offset_y": 0,           /* UI Y 方向的坐标偏移 */
    "tp_device": "",            /* 触摸屏的设备节点名称（在 Linux 系统上用到触摸屏时需要填写） */
    "tp_width": 0,          /* 触摸屏的实际的宽度（触摸大小与屏大小不一致需要填写） */
    "tp_height": 0,             /* 触摸屏的实际的高度（触摸大小与屏大小不一致需要填写） */
    "use_tp": 1,                /* 是否使用触摸 */
    "use_external_button": 0,   /* 是否使用外部按键 */
    "use_mouse": 1,             /* 是否使用鼠标 */
    "mouse_dev": "",            /* 鼠标的设备节点（在 Linux 系统上用到鼠标时需要填写）*/
    "use_key": 1,               /* 是否使用按键板*/
    "key_dev": "",              /* 按键板的设备节点（在 Linux 系统上用到按键板时需要填写）*/
    "use_file_input": 0,        /* 是否使用鼠标和触摸事件记录和回放 */
    "res_file": "/res/res.iso",     /* 资源文件路径 */
    "use_screen_standby": 1     /* 是否使用屏保功能*/
}
```

3. EUI 控件对象

3.1. 控件对象集

编号	名称	类别标识	说明
1	弧	arc	绘制弧形图形
2	进度条	bar	进度条
3	按键	btn	普通按钮
4	按键矩阵	btnm	由多个普通按钮组成的按钮矩阵
5	日历	calendar	日历
6	复选框	checkbox	复选框
7	图表	chart	图表控件，可以显示线形图、点图、柱形图或三种图形的组合显示
8	容器	container	容器控件
9	下拉列表	ddlist	下拉列表框
10	仪表显示	gauge	仪表显示，带刻度及指针的仪表，如汽车油量显示等
11	图像	img	图片显示框
12	图像按键	imgbtn	带按钮功能的图片显示框，如需文本，则需要给该控件添加一个标签控件
13	键盘	keyboard	字符输入控件，有数字键盘与字符键盘区分
14	标签	label	标签
15	LED	led	LED 灯
16	线	line	绘制一条或多条直线

17	列表	lmeter	菜单列表或自定义列表
18	线路表	list	显示带刻度的弧形（另类仪表，不带指针）
19	消息框	msgbox	消息提示框，可带多个点击按钮
20	页面	page	页面，与容器功能类似
21	滚动列表	roller	鼠标拖动选择列表中的项
22	滑块	slider	滑块，如电脑音量调节器
23	开关	switch	开关
24	文本输入框	textarea	接收输入的文本框
25	选项卡	tabview	选项卡
26	窗体	window	自定义窗体
27	预装载	preload	加载等待提示
28	画布	canvas	画布
29	微调框	spinbox	微调框
30	平铺视图	tileview	平铺视图

3.2. 参数信息

3.2.1. 公共参数

编号	参数名称	描述
1	type	控件类型，取默认值即可，无需更改
2	id	控件编号
3	x	控件左上角 x 位置
4	y	控件左上角 y 位置
5	w	控件宽度
6	h	控件高度
7	in_type	控件包含在容器类控件中的类型，0 表示没有在任何容器类控件中；1 表示在容器控件中；2 表示在页控件中；3 表示在选项卡控件中；4 表示在窗口控件中；5 表示在列表控件中；其他值为非法值
8	in_which_obj	指明容器类控件的 ID，仅当控件位于容器类控件内时使用。如果控件没有在容器类，不写出该字段或者将该字段设置为 255，整数值
9	in_which_tab	指明控件在选项卡控件中的哪个选项卡中，仅当控件位于选项卡控件内时使用。如果控件没有在选项卡内，不写出该字段或者将该字段设置为 255，整数值
10	align_y_offset	对齐时 y 方向偏移量
11	align_x_offset	对齐时 x 方向偏移量
12	align_mode	控件对齐方式

13	align_with	控件对齐目标控件 ID
14	hidden	是否隐藏。0 表示显示，1 表示隐藏，2 表示透明（目前只有在 container 控件有效）
15	in_which_btn	指明控件在列表控件中的哪个按钮中，仅当控件位于列表控件内时使用。如果控件没有在列表内，不写出该字段或者将该字段设置为 255，整数值
16	in_which_win	指明窗口类空间的 ID，仅当控件位于窗口类控件内时使用。如果控件没有在窗口内，不写出该字段或者将该字段设置为 255，整数值
17	init_func	控件初始化函数
18	exit_func	控件退出函数，必须有初始化函数，才能执行退出函数
19	click	是否能够点击，1 Enable；0 Disable;默认为 1

3.2.2. 弧

编号	参数名称	描述
1	start_angle	整数值，弧的起始角度
2	end_angle	整数值，弧度的结束角度
5	style_main_idx	弧的风格

3.2.3. 进度条

编号	参数名称	描述
1	min	为进度条表示的最小值，整数值
2	max	为进度条表示的最大值，整数值
3	cur	为进度条的当前值，整数值
4	en_anim	是否支持动画显示，整数值
5	anim_time	动画时间(单位毫秒)，整数值
6	en_sym	是否支持对称(以零为中心点)显示，整数值
7	style_bg_idx	进度条背景风格
8	style_indic_idx	进度条指示器风格

3.2.4. 按键

编号	参数名称	描述
1	name	控件对应的文本，字符串类型
2	ink_in_time	泼墨效果时间，以毫秒为单位，整数值
3	ink_wait_time	泼墨效果完成到开始消失的等待时间，以毫秒为单位，整数值
4	ink_out_time	泼墨效果消失的时间，以毫秒为单位，整数值
5	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
6	pr_resp_msg	按住时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
7	long_pr_resp_msg	长按时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
8	long_pr_repeat_msg	长按时重复发送的消息，十六进制整数值，范围是 0xE200-0xEFFF

9	is_ret	是否返回上一级按键,自动调用系统内部函数返回上一级=1；其它=0
10	link	按键链接到下一视图的 json 文件的路径，如果文件设置了链接，则 is_ret 设置为 0
11	style_rel_idx	按键释放时风格，整数值
12	style_pr_idx	按键按下时风格，整数值
13	style_tgl_rel_idx	按键切换释放风格，整数值
14	style_tgl_pr_idx	按键切换按下风格，整数值
15	style_ina_idx	按键非活动状态风格，整数值
16	has_dialog	是否有对话框，整数值。设置为 1 时，有对话框；设置为 0 时，无对话框，整数值
17	iso_dialog	独立对话框索引值
18	txt_layout	文本相对于按键内的对齐方式，整数值。与对齐方式一致，范围是 0-8
19	txt_offset_x	字体控件 x 方向的偏移，整数值
20	txt_ofset_y	字体控件 y 方向的偏移，整数值
21	layout	按键内的布局方式，范围是 0-9，整数值
22	en_hor	按键是否水平方向自动适应，1.自动适应，0.不自动适应，整数值
23	en_ver	按键是否垂直方向自动适应，1.自动适应，0.不自动适应，整数值
24	state	按键的状态值，范围是 0-4，0-释放状态，1-按下状态，2-toggle 释放状态，3-toggle 按下状态，4-禁止状态
25	toggle	按键切换使能, 1-enable, 0-disable
26	txt_align	字符内部对齐方式，0 表示向左对齐，1 表示向中间对齐，2 表示向右边对齐
27	txt_long_mode	标签文本过长模式。0-扩大标签大小与文本一致，1 保持标签的宽度，将文本过长的换行，增加标签的高度，2 扩大标签大小并且文本在标签上滚动，3 保持标签大小，如果文本过长则在结尾写"..."，4 保持标签大小并且无限的滚动文本，5 标签大小并且裁剪文本

3.2.5. 按键矩阵

编号	参数名称	描述
1	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
2	btns	按钮集合
4	style_bg_idx	按键背景风格
5	style_rel_idx	按键释放时风格
6	style_pr_idx	按键按下时风格
7	style_tgl_rel_idx	按键切换释放风格
8	style_tgl_pr_idx	按键切换按下风格

9	style_ina_idx	按键非活动状态风格
10	en_toggle	按键切换使能, 1-enable, 0-disable
11	toggle_id	切换使能按键索引，整数值
12	recolor	是否重新着色, 1-enable, 0-disable
button		
	active	按键是否为激活状态，整数值。0 表示不激活，1 表示激活
	repeat	按键是否接受重复消息，整数值。1 表示重复模式，0 表示非重复模式
	hidden	按键是否隐藏，整数值。1 表示隐藏，0 表示显示
	line_end	是否换行，整数值。1 表示换行，0 表示不换行。换行的意思是按键本身不显示，下一个按键另外一行显示
	width_unit	按键占用几个宽度单位，整数值。宽度单位的取值范围为 0~7
	name	按键对应的文本，字符串值。文本处理时根据该文本查找对应语言的字符串

3.2.6. 日历

编号	参数名称	描述
1	today	今天的日期，对象类型
2	showed	显示的月份。日被忽略
3	highlight	高亮显示的日期数组
4	style_bg_idx	日历背景风格
5	style_header_idx	日历标题风格
6	style_header_pr_idx	日历按下标题风格
7	style_day_names_idx	日期名称风格
8	style_highlighted_days_idx	高亮天数风格
9	style_inactive_days_idx	可见日期风格
10	style_week_box_idx	日历周框风格
11	style_today_box_idx	日历今天框风格
12	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
13	pr_resp_msg	按住时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
14	long_pr_resp_msg	长按时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
15	long_pr_repeat_msg	长按时重复发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
16	day_names	星期名字字符串数组
17	month_names	月份名字字符串数组

3.2.7. 复选框

编号	参数名称	描述
1	text	文本，字符串

2	state	状态，1 表示选中，0 表示未选中
3	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
4	style_bg_idx	checkbox 背景风格
5	style_box_rel_idx	释放框风格
6	style_box_pr_idx	按下框风格
7	style_box_tgl_rel_idx	复选框风格
8	style_box_tgl_pr_idx	复选框风格
9	style_box_ina_idx	checkbox 非活动状态风格

3.2.8. 图表

编号	参数名称	描述
1	ymin	y 方向上的最小值，整数值
2	ymax	y 方向上的最大值，整数值
3	hdivs	水平方向的等分数，整数值
4	vdivs	垂直方向的等分数，整数值
5	chart_type	图表类型，范围是 0~15，整数值。1 表示线图，2 表示柱状图，4 表示点图。其它是组合图，十六进制数字符串
6	series_width	线的宽度或者原点的半径，整数值
7	series_opa	线的不透明度，整数值，取值 0~255
8	series_dark	点或者柱状图的透明度，取值 0~255
9	point_count	点或者柱状图的数量
series		
	points	点数组，表示在图表上显示的一组数据。支持多组数据。整数值
	color	点或者柱状图的颜色
10	series_num	点或者柱状图的组数
11	style_idx	图表风格

3.2.9. 容器

编号	参数名称	描述
1	hor_fit	是否水平方向自适应，整数值。1 表示允许，0 表示禁止
2	ver_fit	是否垂直方式自适应，整数值。1 表示允许，0 表示禁止
3	layout	布局方式。0 表示关闭布局；1 表示中心对齐；2 表示列左对齐；3 表示列中间对齐；4 表示列右对齐；5 表示行上对齐；6 表示行中对齐；7 表示行底对齐；8 表示一行显示尽可能多的内容；9 表示网格对齐
4	style_idx	容器风格
5	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
6	link	按键链接到下一视图的 json 文件的路径，字符串

7	event_resp_msg	滑动时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF

3.2.10. 下拉列表

编号	参数名称	描述
1	anim_time	动画时间，整数值
2	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
3	options	为选项的文本数组，将所有的选项的文本写入到数组中，文本处理时查到到对应的语言的文本
4	style_bg_idx	下拉列表背景风格
5	style_sel_idx	下拉列表选中项风格
6	style_sb_idx	下拉列表滚动条模式风格
7	fix_height	适应高度，只有在下拉列表打开时有效，整数值
8	sel_opt	选中项，整数值
9	hor_fit	水平适应，1 表示水平适应，0 表示不设置或者不水平适应
10	draw_arrow	在下拉列表上画箭头，1.enable，0.disable
11	txt_align	字符内部对齐方式，0 表示向左对齐，1 表示向中间对齐，2 表示向右边对齐
12	sb_mode	滚动条模式，0 表示关闭滚动条；1 表示总是开启滚动条；2 表示拖动时才显示滚动条；3 表示只有当选项足够多时才显示滚动条；4 表示隐藏滚动条；5 表示不隐藏滚动条。
13	open_flag	是否打开下拉列表，1 表示打开，0 表示不设置或者不打开
14	en_anim	是否开启动画，1 表示开启，0 表示不设置或者不开启
15	style_scrl_idx	下拉列表滚动部分风格

3.2.11. 仪表显示

编号	参数名称	描述
1	min	仪表的最小值，整数值
2	max	仪表的最大值，整数值
3	angle	仪表整个刻度的弧度，整数值
4	line_cnt	仪表中刻度线的数目，整数值。刻度线数目=（最大值-最小值）/单位刻度值+1
5	label_cnt	仪表中表示刻度值的文本数目，整数值。刻度文本数目=（最大值-最小值）/单位刻度值+1
6	critical_value	大于该值和小于该值，刻度线的颜色不一样，整数值
7	needle_color	指针的颜色数组，24 位的十六进制数表示 RGB 颜色值
8	needle_value	指针对应的值数组，整数数组
9	style_idx	表盘风格

3.2.12. 图像

编号	参数名称	描述
1	src_index	当前图像索引值
2	auto_size	是否自动适应图像大小消失，整数值。1 表示自动适应，0 表示非自动适应
3	color_format	颜色格式，整数值。取值范围 0~14
4	style_idx	图像风格
5	src_list	图像对应的变量、文件或者符号的描述。字符串数组类型

3.2.13. 图像按键

编号	参数名称	描述
1	rel_img_index	释放状态下的按键的图像文件索引值，整数值
2	rel_img_list	释放状态时按键的图像文件数组，字符串指针类型
3	pr_img_index	按下状态时按键的图像文件索引值，整数值
4	pr_img_list	按下状态时按键的图像文件数组，字符串指针类型
5	tgl_rel_img_index	toggle 释放时按键的图像文件索引值，整数值
6	tgl_rel_img_list	toggle 释放状态对应的图像文件数组，字符串指针类型
7	tgl_pr_img_index	toggle 按下时按键的图像文件索引值，整数值
8	tgl_pr_img_list	toggle 按下状态对应的图像文件数组，字符串指针类型
9	ina_img_index	禁止状态时按键的图像文件索引值，整数值
10	ina_img_list	禁止状态对应的图像文件数组，字符串指针类型
11	name	按键对应的文本，字符串类型。文本处理时查找对应的语言的字符串
12	link	按键链接到下一视图的 json 文件的路径，如果文件设置了链接，则 is_ret 设置为 0
13	is_ret	是否返回上一级按键，整数值。自动调用系统内部函数返回上一级=1,自定义处理=0
14	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
15	pr_resp_msg	按住时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
16	long_pr_resp_msg	长按时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
17	long_pr_repeat_msg	长按时重复发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
18	txt_layout	文本相对于按键内的对齐方式，整数值。与对齐方式一致，范围是 0-8
19	txt_offset_x	文本 x 轴偏移量
20	txt_offset_y	文本 y 轴偏移量
21	style_rel_idx	图片释放时风格值
22	style_pr_idx	图片按下时风格值
23	style_tgl_rel_idx	按键切换释放风格值
24	style_tgl_pr_idx	图片切换按 xia 风格值

25	style_ina_idx	图片非活动状态风格
26	has_dialog	是否有对话框，整数值。设置为 1 时，有对话框；设置为 0 时，无对话框
27	iso_dialog	独立对话框索引值
28	txt_long_mode	标签文本过长模式。0-扩大标签大小与文本一致，1 保持标签的宽度，将文本过长的换行，增加标签的高度，2 扩大标签大小并且文本在标签上滚动，3 保持标签大小，如果文本过长则在结尾写”...”，4 保持标签大小并且无限的滚动文本，5 标签大小并且裁剪文本
29	txt_align	字体对齐方式，0-向左对齐，1-居中，2-向右对齐
30	auto_size	是否自动适应图像大小消失，整数值。1 表示自动适应，0 表示非自动适应
31	img_layout	图片相对于按键内的对齐方式，整数值。与对齐方式一致，范围是 0-8
32	img_offset_x	图片 x 轴偏移量
33	img_offset_y	图片 y 轴偏移量

3.2.14. 键盘

编号	参数名称	描述
1	ok_msg	点击 OK 时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
2	hide_msg	点击隐藏时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
3	mode	键盘模式，整数值。0=小写字符键盘，1=大写字符键盘，2=特殊符号键盘，3=中文输入法键盘，4=数字键盘
4	style_bg_idx	按键背景风格
5	style_rel_idx	按键释放时风格
6	style_pr_idx	按键按下时风格
7	style_tgl_rel_idx	按键切换释放风格
8	style_tgl_pr_idx	按键切换按下风格
9	style_ina_idx	按键非活动状态风格
10	style_par_idx	输入法背景风格
11	style_result_rel_idx	输入法检索按键释放风格
12	style_result_pr_idx	输入法检索按键按下风格
13	style_result_bg_idx	输入法检索按键背景风格
14	style_input_bg_idx	输入法拼音行背景风格
15	toggle	键盘切换使能，1=启用（enable），0=禁用（disable）

3.2.15. 标签

编号	参数名称	描述
1	name	文本，字符串。文本处理时查找对应的不同语言的字符串
2	style_idx	标签风格
3	long_mode	标签文本过长模式。0-扩大标签大小与文本一致，1 保持标签的宽度，将文

		本过长的换行，增加标签的高度，2 扩大标签大小并且文本在标签上滚动，3 保持标签大小，如果文本过长则在结尾写"..."，4 保持标签大小并且无限的滚动文本，5 标签大小并且裁剪文本
4	txt_align	字符内部对齐方式，0 表示向左对齐，1 表示向中间对齐，2 表示向右边对齐
5	recolor	是否重新着色, 1-enable, 0-disable
6	body_draw	是否画背景, 1-enable, 0-disable
7	anim_speed	long_mode 为 2 或 4 时有效，动画显示的速度，整数值

3.2.16. LED

编号	参数名称	描述
1	bright	LED 的最大亮度，整数值
2	on	LED 的缺省状态，整数值。1 表示 on，0 表示 off
3	style_idx	LED 风格

3.2.17. 线

编号	参数名称	描述
1	points	点数组，表示线经过的点的位置坐标，整数数组
2	auto_size	是否自动调整大小，整数值。1 表示自动调整大小，0 表示不自动调整大小
3	y_inv	是否 y 方向反转，整数值。1 表示反转，0 表示不反转。反转的概念是指，通常屏幕的左上角为坐标系原点。Y 反转后，左下角为坐标系原点
4	style_idx	线风格

3.2.18. 列表

编号	参数名称	描述
1	anim_time	动画时间，整数值
2	sb_mode	滚动条模式，整数值。0 表示关闭滚动条；1 表示总是开启滚动条；2 表示拖动时才显示滚动条；3 表示只有当选项足够多时才显示滚动条；4 表示隐藏滚动条；5 表示不隐藏滚动条
3	options	列表选项对应的文本，字符串类型。文本处理时查找对应的不同语言的字符串
4	style_bg_idx	列表背景风格
5	style_scrl_idx	列表可滚动部分风格
6	style_sb_idx	列表滚动条模式风格
7	style_rel_idx	列表释放时风格
8	style_pr_idx	列表按下时风格
9	style_tgl_rel_idx	列表切换释放风格
10	style_tgl_pr_idx	列表切换按下风格

11	style_ina_idx	列表非活动状态风格
12	has_link	是否有链接到下一视图的 json 文件的路径，整数值。设置为 1 时且 link 数据为 NULL 时，使用默认链接路径；设置为 0 时，则根据 link 数据判断是否有连接路径
13	options_links	按键链接到下一视图的 json 文件的路径
14	options_num	按键的个数，整数值
15	options_w	按键的宽度，整数值
16	options_h	按键的高度，整数值
17	click_resp_msg	单击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
18	en_anim	是否开启动画，1 表示开启，0 表示不设置或者不开启
19	single_mode	单个按键选中模式，1 表示开启，0 表示不设置或者不开启
20	sb_propagation	滚动传播特性，开启时如果列表的滚动条不能在移动，将会移动列表的父控件，1 表示开启，0 表示不设置或者不开启
21	edge_flash	边缘闪光效果，1 表示开启，0 表示不设置或者不开启
22	dynamic_load	动态加载，1 表示开启，0 表示不设置或者不开启
23	page_elem_num	一页的元素（行）个数，整数值
24	col_num	一行的列数，整数值
25	col_percent	每一列所占百分比，总和为 100，整数数组
26	sel_btn	列表选中项的索引值
27	style_edge_falsh_idx	列表边缘闪烁风格
28	en_sel	触摸是否需要选中，1 表示开启，0 表示不设置或者不开启

3.2.19. 线路表

编号	参数名称	描述
1	angle	线表的弧度，整数值，取值范围为 0~360
2	line_cnt	线表刻度线数目，整数值。计算方法同 gauge
3	min	线表最小值，整数值
4	max	线表最大值，整数值
5	cur	线表的当前值，整数值。小于 cur 值和大于 cur 使用不同的颜色
6	style_idx	线表风格

3.2.20. 消息框

编号	参数名称	描述
1	text	消息框的内容，字符串类型
2	btnm	消息框中的按键文本，字符串类型。文本处理时查找对应的不同语言的字符串
3	close_ms	关闭时的消息，十六进制整数值，范围是 0xE200-0xEFFF

4	style_bg_idx	消息框背景风格
5	style_btn_idx	消息框按键风格
6	style_rel_idx	消息框按键释放时风格
7	style_pr_idx	消息框按键按下时风格
8	style_tgl_rel_idx	消息框按键切换释放风格
9	style_tgl_pr_idx	消息框按键切换按下风格
10	style_ina_idx	消息框列表非活动状态风格
11	delay	消息框自动关闭的时间，单位为秒，为 0 时不消失，整数值
12	offset	按键个数为 0 时，是消息框的内容与消息框顶部的距离；否则，是消息框的内容与按键文本的距离
13	recolor	是否重新着色, 1-enable, 0-disable
14	anim_time	动画时间(单位毫秒)
15	close_flag	消息框关闭标志，1 表示关闭当前视图，2 表示关闭消息框控件
16	toggle_id	切换状态的按键索引值

3.2.21. 页面

编号	参数名称	描述
1	sb_mode	滚动条模式，整数值
2	en_arrow	是否允许箭头，整数值。1 表示允许，0 表禁止
3	en_hor_fit	是否水平方向自适应，整数值。1 表示允许，0 表示禁止
4	en_ver_fit	是否垂直方式自适应，整数值。1 表示允许，0 表示禁止
5	scrl_width	滚动条宽度，整数值。当允许水平方向自适应时，无效
6	scrl_height	滚动条高度，整数值。当允许垂直方向自适应时，无效
7	layout	布局方式。0 表示关闭布局；1 表示中心对齐；2 表示列左对齐；3 表示列中间对齐；4 表示列右对齐；5 表示行上对齐；6 表示行中对齐；7 表示行底对齐；8 表示一行显示尽可能多的内容；9 表示网格对齐
8	rel_msg	释放时的消息，十六进制整数值，范围是 0xE200-0xEFFF
9	pr_msg	按下时的消息，十六进制整数值，范围是 0xE200-0xEFFF
10	style_bg_idx	页背景风格
11	style_scrl_idx	页可滚动部分风格
12	style_sb_idx	页滚动条模式风格
13	en_edge_flash	边缘闪光效果，1 表示开启，0 表示不设置或者不开启
14	en_scrl_pro	滚动传播特性，开启时如果列表的滚动条不能在移动，将会移动列表的父控件，1 表示开启，0 表示不设置或者不开启
15	style_edge_flash_idx	页边缘闪烁风格

3.2.22. 预装载

编号	参数名称	描述
1	arc_len	表示预装载中运动的弧的角度，整数值。取值范围 0~360
2	spin_time	表示运动的弧运动一圈所需要的时间，整数值
3	style_main_idx	预装载主风格
4	anim_type	动画类型，0 表示弧旋转效果，1 表示填补旋转效果

3.2.23. 滚动列表

编号	参数名称	描述
1	anim_time	滚动动画时间，整数值
2	visible_cnt	roller 中可见选项的数目，整数值
3	en_hor_fit	是否允许水平方向自适应，整数值。1 表示允许，0 表示禁止
4	sel_msg	选中时的消息，十六进制整数值，范围是 0xE200-0xEFFF
5	options	roller 中的选项对应的文本数组，字符串数组。处理时查找不同语言对应的字符串
6	style_bg_idx	roller 背景风格
7	style_sel_idx	roller 选中风格
8	selected	选中项，整数值
9	anim_en	是否开启动画，1 表示开启，0 表示不设置或者不开启
10	style_scrl_idx	Roller 滚动部分风格

3.2.24. 滑块

编号	参数名称	描述
1	min	滑块的最小值，整数值
2	max	滑块的最大值，整数值
3	cur	滑块的当前位置，整数值
4	ch_msg	滑块位置改变时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
5	style_bg_idx	滑块背景风格
6	style_indic_idx	滑块指示器风格
7	style_knob_idx	滑块旋钮风格
8	knob_in	滑块旋钮位置，1 一直在滑块上；0 能够超出滑块的边缘
9	anim_time	动画时间

3.2.25. 开关

编号	参数名称	描述
1	sw_msg	开关状态改变时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
2	state	缺省状态，整数值。0 表示关，1 表示开

3	style_bg_idx	开关背景风格
4	style_indic_idx	开关指示器风格
5	style_knob_off_idx	开关关闭时旋钮风格
6	style_knob_on_idx	开关打开时旋钮风格
7	anim_time	动画时间
8	en_anim	是否开启动画，1 表示开启，0 表示不设置或者不开启

3.2.26. 文本输入框

编号	参数名称	描述
1	max_length	可显示文本的最大长度，整数值
2	pwd_mode	是否以密码的方式显示文本，整数值。1 表示以密码方式显示，即全部显示为“*”，0 表示正常显示
3	one_line_mode	单行模式，整数值。1 表示单行模式，0 表示多行模式
4	cursor_type	光标类型，整数值。0 表示没有光标，1 表示光标是竖线，2 表示光标是块状，3 表示光标是方框，4 表示光标是下划线，8 表示隐藏光标
5	click_msg	点击时发送的消息，16 进制整数值
6	style_bg_idx	文本背景风格
7	style_sb_idx	文本滚动条模式风格
8	style_cursor_idx	文本光标风格
9	cursor_pos	光标的位置（以字符数来计算），整数值
10	sb_mode	滚动条的模式，0 表示关闭滚动条；1 表示总是开启滚动条；2 表示拖动时才显示滚动条；3 表示只有当选项足够多时才显示滚动条；4 表示隐藏滚动条；5 表示不隐藏滚动条
11	scr_pro	滚动传播特性，开启时如果列表的滚动条不能在移动，将会移动列表的父控件，1 表示开启，0 表示不设置或者不开启
12	edge_flash	边缘闪光效果，1 表示开启，0 表示不设置或者不开启
13	txt_align	字符内部对齐方式，0 表示向左对齐，1 表示向中间对齐，2 表示向右边对齐
14	style_scrl_idx	文本滚动部分风格

3.2.27. 选项卡

编号	参数名称	描述
1	tab_cur	当前激活选项卡，整数值
2	anim_time	选项卡切换动画时间，整数值
3	en_sliding	是否允许选项卡之间滑动，整数值。1 表示允许，0 表示禁止
4	draging	是否允许被拖动，整数值。1 表示允许，0 表示禁止
5	drag_hor	是否允许水平方向拖动，整数值。1 表示允许，0 表示禁止

6	btn_pos	选项卡头的位置，整数值。1 表示在底部，0 表示在顶部
7	load_msg	点击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
8	tabs	Tab 页标题
9	style_bg_idx	选项卡背景风格
10	style_indic_idx	选项卡指示器风格
11	style_btn_bg_idx	按键背景风格
12	style_btn_rel_idx	按键释放风格
13	style_btn_pr_idx	按键按下风格
14	style_btn_tgl_rel_idx	按键切换释放风格
15	style_btn_tgl_pr_idx	按键切换按下风格
16	style_tab_bg_idx	选项背景风格
17	style_tab_scrl_idx	选项滚动部分风格
18	btn_hidden	是否隐藏选项卡按键，1 表示隐藏，0 表示不设置或者不隐藏
19	btns_height	选项卡按键高度，为 0 则默认为字体高度加上两倍按键释放风格的垂直间距，再加上两倍按键背景风格的垂直距离

3.2.28. 窗体

编号	参数名称	描述
1	sb_mode	设置窗口的滚动条模式，整数值
2	layout	设置窗口的布局方式，整数值
3	title	设置窗口的主题，字符串。字符串处理时查找不同语言对应的字符串
4	style_bg_idx	窗口背景风格
5	style_content_bg_idx	内容页背景风格
6	style_content_scrl_idx	内容页可滚动部分风格
7	style_sb_idx	按键释放风格
8	style_header_idx	按键按下风格
9	style_btn_rel_idx	滚动条模式风格
10	style_btn_pr_idx	标题风格
11	drag	是否允许被拖动，整数值。1 表示允许，0 表示禁止
12	btn_size	控制按钮的大小，整数值
13	anim_time	动画时间（单位毫秒），整数值

3.2.29. 画布

编号	参数名称	描述
1	style_main_idx	画布主风格
2	color_format	颜色格式，整数值。默认为 4，范围是 6-10，为 6 时支持 ARGB 色，但画

		图片不支持透明色
3	bg_color	背景颜色（ARGB），十六进制数字符串
4	polygon_list	多边形（包含线）参数列表
5	circle_list	圆参数列表
6	image_list	图片参数列表
polygon_list		
	points	点数组，表示多边形经过的点的位置坐标，整数数组
	line_color	线的颜色（ARGB），十六进制数字符串
	fill_color	多边形区域填充颜色（ARGB），十六进制数字符串
	en_fill	整数值。1 表示填充，0 或者不设置表示不填充
circle_list		
	coordinate	点数组，圆点的位置坐标，整数数组
	border_color	线的颜色（ARGB），十六进制数字符串
	fill_color	圆区域填充颜色（ARGB），十六进制数字符串
	en_fill	整数值。1 表示填充，0 或者不设置表示不填充
image_list		
	coordinate	点数组，起始的位置坐标，整数数组
	image_src	图片路径或者图片数据
	type	图片类型，1-表示 image_src 是图片路径，2-表示 image_src 是图片数据
	width	图片的宽度(当是图片数据时需要填充)
	height	图片的高度(当是图片数据时需要填充)
	multiply	图片颜色是否与背景颜色混合，1 表示混合，0 表示不设置或者不混合

3.2.30. 微调框

编号	参数名称	描述
1	value	当前值，整数值
2	min	最小值，整数值
3	max	最大值，整数值
4	step	步进值，正数表述增加，负数表示减少，整数值
5	padding	数字计数中的符号与数字的距离（位数），整数值
6	digit_count	输入值的最大位数（小数点和符号除外），整数值
7	separator_pos	小数点的位置，整数值
8	style_bg_idx	背景风格
9	style_sb_idx	滚动条模式风格
10	style_cursor_idx	光标风格
11	style_scrl_idx	滚动部分风格

3.2.31. 平铺视图

编号	参数名称	描述
1	act_tile	当前平铺视图坐标
3	en_anim	是否开启动画，1 表示开启，0 表示不设置或者不开启
4	en_edge_flash	边缘闪光效果，1 表示开启，0 表示不设置或者不开启
5	valid_pos	平铺视图有效位置的坐标数组
6	load_msg	点击时发送的消息，十六进制整数值，范围是 0xE200-0xEFFF
7	style_bg_idx	平铺视图背景风格

4. EUI API

4.1. EUI 初始化

UI 初始化包括 EUI 核心库初始化与激活，硬件输入和输出设备注册。接口定义如下：

lb_eui_res_init

功能	UI 资源文件加载。
函数	lb_int32 lb_eui_res_init(char *path_iso)
参数	path_iso：iso 资源文件路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_res_uinit

功能	UI 资源文件卸载。
函数	lb_int32 lb_eui_res_uinit(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_init

功能	UI 初始化，输入输出设备注册。
函数	lb_int32 lb_eui_init(const char *cfg)
参数	cfg：配置文件路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_uinit

功能	UI 去初始化。
函数	lb_int32 lb_eui_uinit(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_language_init

功能	初始化语言配置。
函数	lb_int32 lb_eui_language_init(lb_uint32 index)
参数	index： 语言索引。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_customize_init

功能	UI 快速初始化（只初始化 EUI 定制部分的功能）。
函数	lb_int32 lb_eui_customize_init(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_customize_uinit

功能	UI 快速去初始化。
函数	lb_int32 lb_eui_customize_uinit(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_start

功能	激活 UI 库。
函数	lb_int32 lb_eui_start(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_set_trans_area

功能	UI 设置免读区域，仅在 linux 平台上使用。
函数	lb_int32 lb_eui_set_trans_area(char *area_str)
参数	area_str：区域坐标字符串，例如“50,50,100,100”。
返回值	0 表示成功，其他负值表示错误码。

4.2. UI 资源与视图

UI 资源包括传入 root json 文件时，创建的静态视图树、独立视图链表、字体库资源、风格资源等。接口定义如下：

lb_eui_create

功能	UI 资源创建, UI 资源创建包括创建静态视图树、独立视图链表、创建字库、设置风格等操作。
函数	lb_int32 lb_eui_create(char *root_file)
参数	root_file：root JSON 文件名，包含完整路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_destroy

功能	UI 资源销毁, 销毁当前应用创建的静态视图树、独立视图链表、字库、风格等内存资源。
函数	lb_int32 lb_eui_destroy(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_static_new

功能	新建静态视图树链表。在多应用项目中进入非主应用时，打开新的应用，为新的应用创建新的 EUI 静

	态视图树链表。 注意：此接口只能调用一次，再次调用需调用 lb_eui_static_delete ()，以删除当前应用对应的 UI 资源。
函数	lb_int32 lb_eui_static_new (char *file)
参数	file：JSON 文件名，包含完整路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_static_delete

功能	删除 lb_eui_static_new()新增的静态视图树链表。
函数	lb_int32 lb_eui_static_delete (void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_static_add

功能	在 lb_eui_static_new 创建的静态视图下添加一个静态视图。
函数	lb_int32 lb_eui_static_add(char *file)
参数	file：JSON 文件名，包含完整路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_style_load

功能	加载风格资源，在多应用项目中进入非主应用时，打开新的应用，为新的应用创建新的 EUI 风格属性。 注意：非主应用风格 ID 在配置的时候默认从 100 开始
函数	lb_int32 lb_eui_style_load(char *filename)
参数	filename：JSON 文件名，包含完整路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_style_unload

功能	卸载 lb_eui_style_load 加载的风格资源。
函数	lb_int32 lb_eui_style_unload(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_isolate_new

功能	打开独立视图。
函数	lb_int32 lb_eui_isolate_new(lb_uint32 idx)
参数	idx：视图索引值，在 EUIEditor 工具中会将独立视图索引生成在 eguieditor.h 中。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_isolate_delete

功能	关闭独立视图。
函数	lb_int32 lb_eui_isolate_delete(lb_uint16 idx)
参数	idx：视图索引值，在 EUIEditor 工具中会将独立视图索引生成在 eguieditor.h 中。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_enter

功能	当控件不需要主动进入下一视图时，打开控件链接的视图，通常与容器搭配使用。
函数	lb_int32 lb_eui_enter(lb_uint32 id)
参数	Id： 控件 ID。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_return

功能	返回上一视图， 当前视图位于当前应用的最底层视图时，则不会返回。
函数	lb_int32 lb_eui_return(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_get_current_view_id

功能	获取当前视图的背景 ID。
函数	lb_int32 lb_eui_get_current_view_id(lb_uint32 *id)
参数	Id： 背景 ID。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_get_last_view_id

功能	获取上一层视图的背景 ID。
函数	lb_int32 lb_eui_get_last_view_id(lb_uint32 *id)
参数	Id： 背景 ID。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_get_current_view_id

功能	获取当前视图的背景 ID。
函数	lb_int32 lb_eui_get_current_view_id(lb_uint32 *id)
参数	Id： 背景 ID。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_send_msg

功能	发送 UI 消息，不是控件响应函数的消息。
函数	lb_int32 lb_eui_send_msg(lb_int32 type, void *msg_data, lb_int32 msg_len)
参数	type： 消息值。范围是 0xE200-0xEFFF, msg_data:消息数据，msg_len：消息数据长度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_send_key

功能	发送键值消息。
函数	lb_int32 lb_eui_send_key(lb_uint32 keycode, lb_int32 status)
参数	keycode：按键键值；status：按键状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_get_disp_info

功能	获取屏幕信息。
函数	lb_int32 lb_eui_get_disp_info(lb_eui_disp_info_t *disp_info)
参数	disp_info：屏幕信息，包含分辨率和旋转角度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_view_open

功能	打开视图。
函数	lb_int32 lb_eui_view_open(lb_uint32 view_id)
参数	view_id： 视图 ID 值，在 EUIEditor 工具中会将视图 ID 生成在 eguieditor.h 中。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_view_close

功能	关闭视图。
函数	lb_int32 lb_eui_view_close(lb_uint32 view_id, lb_uint8 flag)
参数	view_id： 视图 ID 值，在 EUIEditor 工具中会将视图 ID 生成在 eguieditor.h 中； flag：关闭标志，为 1 时表示需要执行相应动作，配合 lb_eui_set_ok_exit_func 使用。
返回值	0 表示成功，其他负值表示错误码。

4.3. 函数管理接口

函数模型：

```
typedef lb_int32(*pobj_init)(void *param);
typedef lb_int32(*pobj_resp_func)(void *msg_data);
typedef lb_int32(*psys_resp_func)(lb_uint32 msgtype, void *msg_data);
```

函数参数说明：

pobj_init 的参数为控件 ID

pobj_resp_func 分为以下几种：

btnm：点击按钮所对应的字符串；

calendar：lb_eui_date_t 数据；

container：在事件响应函数中为 lb_eui_cont_event_info_t 数据；

点击响应事件时为控件 ID 值；

list：lb_eui_list_click_t 数据；

roller：lb_eui_roller_msg_data_t 数据；

slider：lb_eui_slider_move_t 数据；

其余都为控件 ID 值；

psys_resp_func：自定义。

4.3.1. 初始化函数注册

lb_eui_fmngr_reg_init_func

功能	初始化函数注册
函数	lb_int32 lb_eui_fmngr_reg_init_func(char *p_name, pobj_init pfunc)
参数	p_name：初始化函数注册名称；pfunc：初始化函数。
返回值	0 表示成功，其他负值表示错误码。

4.3.2. Exit 函数注册

lb_eui_fmngr_reg_exit_func

功能	exit 函数注册
函数	lb_int32 lb_eui_fmngr_reg_exit_func(char *p_name, pobj_init pfunc)
参数	p_name：exit 函数注册名称；pfunc：exit 函数。
返回值	0 表示成功，其他负值表示错误码。

4.3.3. 消息响应函数注册

lb_eui_reg_resp_msg_func

功能	消息响应函数注册，msg 范围是 0xE200-0xEFFF
函数	lb_int32 lb_eui_reg_resp_msg_func(lb_uint32 msg, pobj_resp_func pfunc)
参数	msg：消息值；pfunc：消息响应函数。
返回值	0 表示成功，其他负值表示错误码。

4.3.4. 确定退出响应函数

lb_eui_set_ok_exit_func

功能	点击确定退出视图时的响应函数
函数	lb_int32 lb_eui_set_ok_exit_func(lb_uint32 view_id, void *pfunc, void *param)
参数	view_id：视图 ID；pfunc：响应函数；param：函数参数
返回值	0 表示成功，其他负值表示错误码。

4.3.5. 初始化函数注销

lb_eui_fmngr_unreg_init_func

功能	初始化函数注销
函数	lb_int32 lb_eui_fmngr_unreg_init_func(pobj_init pfunc)
参数	pfunc：初始化函数。
返回值	0 表示成功，其他负值表示错误码。

4.3.6. 退出函数注销

lb_eui_fmngr_unreg_exit_func

功能	exit 函数注销
函数	lb_int32 lb_eui_fmngr_unreg_exit_func(pobj_init pfunc)

参数	pfunc：exit 函数。
返回值	0 表示成功，其他负值表示错误码。

4.3.7. 消息函数注销

lb_eui_unreg_resp_msg_func

功能	消息响应函数注销
函数	lb_int32 lb_eui_unreg_resp_msg_func(pobj_resp_func pfunc)
参数	pfunc：消息响应函数。
返回值	0 表示成功，其他负值表示错误码。

4. 4. 通用 API 接口

目前 18-23 只在 eos 上有效。

多国语言定义（这个目前是有用户自定义的，此处仅供参考）：

```
Typedef enum {
        LB_EUI_ENGLISH,                 /*英文*/
        LB_EUI_SIMPLE_CHINESE,          /*简体中文*/
        LB_EUI_TRADITION_CHEINESE,      /*繁体中文*/
        LB_EUI_LANGUAGE_MAX
}lb_eui_language_e;
```

4.4.1. 设置多国语言

lb_eui_common_set_language

功能	设置多国语言
函数	lb_int32 lb_eui_common_set_language(lb_uint32 index)
参数	index：多国语言索引。
返回值	0 表示成功，其他负值表示错误码。

4.4.2. 获取当前语言索引

lb_eui_common_get_language

功能	获取当前多国语言索引
函数	lb_int32 lb_eui_common_get_language(lb_uint32 *index)
参数	index：返回当前多国语言索引。
返回值	0 表示成功，其他负值表示错误码。

4.4.3. 创建图片数据

lb_eui_eimage_create_img_buf

功能	根据 res.iso 中的图片路径创建图片资源
函数	lb_eui_img_dsc_t *lb_eui_eimage_create_img_buf(const char *path)
参数	path：打包在 res.iso 中图片文件。
返回值	图片资源数据指针。

4.4.4. 删除图片数据

lb_eui_eimage_destory_img_buf

功能	删除图片资源
函数	lb_int32 lb_eui_eimage_destory_img_buf(lb_eui_img_dsc_t *img_buf)
参数	img_buf：图片资源数据指针。
返回值	0 表示成功，其他负值表示错误码。

4.4.5. 创建图片资源

lb_eui_eimage_create_img_buf_thumb

功能	根据有头信息 ARGB8888 格式的图片数据创建图片资源
函数	lb_eui_img_dsc_t *lb_eui_eimage_create_img_buf_thumb(void *thumb_data_buf)
参数	thumb_data_buf：无头信息 ARGB888 格式的图片数据指针。
返回值	图片资源数据指针。

4.4.6. 删除图片资源

lb_eui_eimage_destory_img_buf_thumb

功能	删除图片资源
函数	lb_int32 lb_eui_eimage_destory_img_buf_thumb(lb_eui_img_dsc_t *thumb_img_buf)
参数	thumb_data_buf：图片资源数据指针。
返回值	0 表示成功，其他负值表示错误码。

4.4.7. 获取资源字符串

lb_eui_common_lang_get_string_json

功能	根据索引字符取资源文件中字符串
函数	const char *lb_eui_common_lang_get_string_json(const char *json_str)
参数	json_str：索引字符串。
返回值	字符串。

4.4.8. 获取资源字符串的索引值

lb_eui_common_lang_get_string_id_json

功能	索引字符取资源文件中字符串索引值
函数	lb_uint32 lb_eui_common_lang_get_string_id_json(const char *json_str)
参数	json_str：索引字符串。
返回值	字符串索引值。

4.4.9. 设置按键键值表

lb_eui_external_button_set_keymap

功能	设置外部按键的键值表。
函数	lb_int32 lb_eui_external_button_set_keymap(const lb_eui_key_map_t *keymap, lb_uint8 num)
参数	Keymap：键值表； num：按键数量。
返回值	0 表示成功，其他负值表示错误码。

4.4.10. 设置按键坐标

lb_eui_external_button_set_points

功能	设置外部按键的所对应的坐标。
函数	lb_int32 lb_eui_external_button_set_points(const lb_eui_point_t *point)
参数	point：坐标数组，一个按键对应一个坐标。
返回值	0 表示成功，其他负值表示错误码。

4.4.11. 设置按键进程

lb_eui_key_proc

功能	将键值传递给 EUI 使用。
函数	lb_int32 lb_eui_key_proc(lb_uint8 key_code)
参数	key_code：按键值。
返回值	0 表示成功，其他负值表示错误码。

4.4.12. 设置延时

lb_eui_common_sleep

功能	延时进程，单位毫秒。
函数	lb_uint32 lb_eui_common_sleep(lb_int32 ms)
参数	ms：时间，毫秒。
返回值	0 表示成功，其他负值表示错误码。

4.4.13. 获取图片类型

lb_eui_image_get_type

功能	获取图片类型。
函数	lb_int32 lb_eui_image_get_type(char *img_path, lb_uint8 *type)
参数	img_path：图片路径；type：图片类型。
返回值	0 表示成功，其他负值表示错误码。

4.4.14. 加载图片数据

lb_eui_image_load

功能	根据图片路径或者 lb_eui_img_dsc_t 数据输出指定大小的 lb_eui_img_dsc_t 数据。
函数	lb_int32 lb_eui_image_load(void *img_src, lb_uint8 type, lb_int32 *layers, lb_int32 **delay, void **output_buff, lb_int32 output_w, lb_int32 output_h)
参数	img_src：图片路径或者 lb_eui_img_dsc_t 数据；type：图片类型；layers：图片层数；delay：图片显示时间；output_buff：lb_eui_gif_t 数据或者 lb_eui_img_dsc_t 数据；output_w：输出图片的宽度；output_h：输出图片的高低。
返回值	0 表示成功，其他负值表示错误码。

4.4.15. 卸载图片数据

lb_eui_image_unload

功能	释放图片资源。
函数	lb_int32 lb_eui_image_unload(lb_uint8 type, void *buff , void *delay)
参数	type：图片类型；buff：lb_eui_gif_t 数据或者 lb_eui_img_dsc_t 数据，delay：图片显示时间；。
返回值	0 表示成功，其他负值表示错误码。

4.4.16. 初始化 blocklinker

lb_eui_bl_draw_init

功能	初始化 blocklinker
函数	lb_int32 lb_eui_bl_draw_init(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

4.4.17. 结束 blocklinker

lb_eui_bl_draw_exit

功能	结束 blocklinker
函数	lb_int32 lb_eui_bl_draw_exit(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

4.4.18. 显示 blocklinker

lb_eui_bl_draw_show

功能	显示 blocklinker

函数	lb_int32 lb_eui_bl_draw_show(void)
参数	json_str：索引字符串。
返回值	0 表示成功，其他负值表示错误码。

4.4.19. 隐藏 blocklinker

lb_eui_bl_draw_hide

功能	隐藏 blocklinker
函数	lb_int32 lb_eui_bl_draw_hide(void)
参数	无。
返回值	0 表示成功，其他负值表示错误码。

4.4.20. 申请 blocklinker

lb_eui_bl_draw_request

功能	申请 blocklinker
函数	lb_int32 lb_eui_bl_draw_request(lb_int32 idx)
参数	idx：blocklinker 索引值。
返回值	0 表示成功，其他负值表示错误码。

4.4.21. 释放 blocklinker

lb_eui_bl_draw_release

功能	释放 blocklinker
函数	lb_int32 lb_eui_bl_draw_release(lb_int32 idx)
参数	idx：blocklinker 索引值。
返回值	0 表示成功，其他负值表示错误码。

4.4.22. 添加 blocklinker

lb_eui_bl_draw_add

功能	添加 blocklinker 到显示列表
函数	lb_int32 lb_eui_bl_draw_add(lb_int32 idx)
参数	idx：blocklinker 索引值。
返回值	0 表示成功，其他负值表示错误码。

4.4.23. 移除 blocklinker

lb_eui_bl_draw_remove

功能	将 blocklinker 从显示列表移除
函数	lb_int32 lb_eui_bl_draw_remove(lb_int32 idx)
参数	idx：blocklinker 索引值。
返回值	0 表示成功，其他负值表示错误码。

4.4.24. 填充 blocklinker

lb_eui_bl_draw_buff

功能	将显示 buff 填充到 blocklinker
函数	lb_int32 lb_eui_bl_draw_buff(lb_int32 idx, lb_eui_rect_t rect, lb_uint8 *buff)
参数	idx：blocklinker 索引值； rect：buff 显示区域；buff：缓存区数据。
返回值	0 表示成功，其他负值表示错误码。

4.4.25. 申请 blocklinker 中的 framebuff

lb_eui_bl_draw_alloc_fb

功能	申请特定的 fb 供 block 使用
函数	lb_int32 lb_eui_bl_draw_alloc_fb(lb_eui_ion_buf_t *buf)
参数	buf：ion 申请 buff 信息，其中 share_id 与 buf 宽高是需要填写的。
返回值	0 表示成功，其他负值表示错误码。

4.4.26. 释放 blocklinker 中的 framebuff

lb_eui_bl_draw_free_fb

功能	将显示 buff 填充到 blocklinker
函数	lb_int32 lb_eui_bl_draw_free_fb(lb_eui_ion_buf_t *buf)
参数	buf：ion 申请 buff 信息。
返回值	0 表示成功，其他负值表示错误码。

4.4.27. 设置屏幕休眠使能

lb_eui_screen_standby_enable

功能	设置屏幕休眠使能
函数	lb_int32 lb_eui_screen_standby_enable (bool en)
参数	en：屏幕休眠属性,true 表示启动屏幕休眠功能，false 表示关闭屏幕休眠功能。
返回值	0 表示成功，其他负值表示错误码。

4.4.28. 获取屏幕休眠使能状态

lb_eui_screen_standby_get_enable

功能	获取屏幕休眠使能状态
函数	lb_eui_screen_standby_get_enable (bool *en)
参数	en：返回屏幕休眠使能状态,true 表示启动屏幕休眠功能，false 表示关闭屏幕休眠功能。
返回值	0 表示成功，其他负值表示错误码。

4.4.29. 获取屏幕当前状态

lb_eui_screen_standby_get_status

功能	获取屏幕休眠使能状态
函数	lb_uint8 lb_eui_screen_standby_get_status (void)
参数	无。
返回值	en：返回屏幕休眠使能状态,true 表示屏幕休眠，false 表示屏幕未休眠。

4.4.30. 设置屏幕自动休眠时间

lb_eui_screen_standby_set_time

功能	设置屏幕自动休眠时间
函数	lb_int32 lb_eui_screen_standby_set_time (lb_uint32 time_sec)
参数	en：屏幕休眠自动进入休眠时间,单位为秒，0 表示关闭自动休眠功能。
返回值	0 表示成功，其他负值表示错误码。

4.4.31. 获取屏幕自动休眠时间

lb_eui_screen_standby_get_time

功能	获取屏幕自动休眠时间
函数	lb_int32 lb_eui_screen_standby_get_time (lb_uint32 *time_sec)
参数	time_sec：返回屏幕自动休眠时间,单位为秒。
返回值	0 表示成功，其他负值表示错误码。

4.4.32. 切换屏幕状态

lb_eui_screen_standby_switch

功能	切换屏幕状态,若当前屏幕为关闭状态，则打开
函数	lb_int32 lb_eui_screen_standby_switch (void)
参数	无
返回值	0 表示成功，其他负值表示错误码。

4.4.33. 设置按键音状态

lb_eui_tone_set_flag

功能	设置按键音播放状态
函数	lb_int32 lb_eui_tone_set_flag (lb_uint8 tone_flag)
参数	tone_flag：0，关闭按键音，1，打开按键音
返回值	0 表示成功，其他负值表示错误码。

4.4.34. 播放按键音

lb_eui_tone_play_default

功能	播放默认提示音
函数	lb_int32 lb_eui_tone_play_default (void)
参数	无
返回值	0 表示成功，其他负值表示错误码。

4.4.35. 播放音频

lb_eui_tone_play

功能	根据音频文件播放提示音，音频文件格式为“.wav”。
函数	llb_int32 lb_eui_tone_play (char *filepath)
参数	filepath：提示音文件路径。
返回值	0 表示成功，其他负值表示错误码。

4.4.36. 设置音频回调

lb_eui_tone_set_cb

功能	设置音频播放回调函数，用户自己填写播放接口。
函数	lb_int32 lb_eui_tone_set_cb(tone_play_cb cb)
参数	filepath：提示音文件路径。
返回值	0 表示成功，其他负值表示错误码。

4.5. 控件 API 接口

每个控件分别提供 get/set API 接口。用于获取/设置控件的属性。以下针对目前使用的控件分别说明如下：

4.5.1. 控件公共属性

```
typedef lb_int32(*tone_play_cb)(void *);
typedef lb_uint8 (*lb_eui_action_t)(void *obj);
typedef lb_uint8 (*lb_eui_cont_action_t)(void *cont, void *param);
typedef lb_uint8 (*lb_eui_btnm_action_t) (void *btnm, const char *txt);
typedef void (*lb_eui_anim_cb_t)(void *);

t typedef enum {
    LB_EUI_ACTION_CLICK,
    LB_EUI_ACTION_PR,
```

```
    LB_EUI_ACTION_LONG_PR,
    LB_EUI_ACTION_LONG_PR_REPEAT,
    LB_EUI_ACTION_NUM,
} lb_eui_action_type;
typedef enum {
    LB_EUI_KB_OK_ACTION,
    LB_EUI_KB_HIDE_ACTION,
} lb_eui_kb_action_type;

/* scrollbar modes */
typedef enum {
    LB_EUI_SB_MODE_OFF    = 0x0,/*Never show scrollbars*/
    LB_EUI_SB_MODE_ON     = 0x1,/*Always show scrollbars*/
    LB_EUI_SB_MODE_DRAG = 0x2,/*Show scrollbars when page is being dragged*/
    LB_EUI_SB_MODE_AUTO = 0x3,/*Show scrollbars when the scrollable
                *container is large enough to be scrolled*/
    LB_EUI_SB_MODE_HIDE = 0x4,/*Hide the scroll bar temporally*/
    LB_EUI_SB_MODE_UNHIDE = 0x5,
    /*Unhide the previously hidden scrollbar. Recover it's type too*/
} lb_eui_sb_mode_e;

typedef enum {
    LB_EUI_CURSOR_NONE,
    LB_EUI_CURSOR_LINE,
    LB_EUI_CURSOR_BLOCK,
    LB_EUI_CURSOR_OUTLINE,
    LB_EUI_CURSOR_UNDERLINE,
    /*Or it to any value to hide the cursor temporally*/
    LB_EUI_CURSOR_HIDDEN = 0x08,
} lb_eui_cursor_type_e;

typedef enum {
    LB_EUI_ALIGN_CENTER = 0,
    LB_EUI_ALIGN_IN_TOP_LEFT,
    LB_EUI_ALIGN_IN_TOP_MID,
    LB_EUI_ALIGN_IN_TOP_RIGHT,
    LB_EUI_ALIGN_IN_BOTTOM_LEFT,
```

```
    LB_EUI_ALIGN_IN_BOTTOM_MID,
    LB_EUI_ALIGN_IN_BOTTOM_RIGHT,
    LB_EUI_ALIGN_IN_LEFT_MID,
    LB_EUI_ALIGN_IN_RIGHT_MID,
    LB_EUI_ALIGN_OUT_TOP_LEFT,
    LB_EUI_ALIGN_OUT_TOP_MID,
    LB_EUI_ALIGN_OUT_TOP_RIGHT,
    LB_EUI_ALIGN_OUT_BOTTOM_LEFT,
    LB_EUI_ALIGN_OUT_BOTTOM_MID,
    LB_EUI_ALIGN_OUT_BOTTOM_RIGHT,
    LB_EUI_ALIGN_OUT_LEFT_TOP,
    LB_EUI_ALIGN_OUT_LEFT_MID,
    LB_EUI_ALIGN_OUT_LEFT_BOTTOM,
    LB_EUI_ALIGN_OUT_RIGHT_TOP,
    LB_EUI_ALIGN_OUT_RIGHT_MID,
    LB_EUI_ALIGN_OUT_RIGHT_BOTTOM,
} lb_eui_align_e;

typedef enum {
    LB_EUI_LAYOUT_OFF = 0,
    LB_EUI_LAYOUT_CENTER,
    LB_EUI_LAYOUT_COL_L, /* Column left align */
    LB_EUI_LAYOUT_COL_M,     /* Column middle align */
    LB_EUI_LAYOUT_COL_R, /* Column right align */
    LB_EUI_LAYOUT_ROW_T,     /* Row top align */
    LB_EUI_LAYOUT_ROW_M,     /* Row middle align */
    LB_EUI_LAYOUT_ROW_B,     /* Row bottom align */
    LB_EUI_LAYOUT_PRETTY,    /* Put as many object as possible
                              * in row and begin a new row */
    LB_EUI_LAYOUT_GRID,   /* Align same-sized object into a grid */
} lb_eui_layout_e;

typedef struct tag_lb_eui_key_msg_data {
    lb_uint32 code;
    lb_int32  value;
} lb_eui_key_msg_data_t;
```

```
/* eui object style struct*/
typedef struct tag_lb_eui_style {
    lb_uint8 glass:1;

    struct {
        lb_uint32 main_color;
        lb_uint32 grad_color;
        lb_int32 radius;
        lb_uint8 opa;

        struct {
            lb_uint32 color;
            lb_int32 width;
            lb_uint8 part;
            lb_uint8 opa;
        } border;

        struct {
            lb_uint32 color;
            lb_int32 width;
            lb_uint8 type;
        } shadow;

        struct {
            lb_int32 ver;
            lb_int32 hor;
            lb_int32 inner;
        } padding;

        lb_uint8 empty:1;
    } body;

    struct {
        lb_uint32 color;
        const void *font;
        lb_int32 letter_space;
        lb_int32 line_space;
```

```
        lb_uint8 opa;
    } text;

    struct {
        lb_uint32 color;
        lb_uint8 intense;
        lb_uint8 opa;
    } image;

    struct {
        lb_uint32 color;
        lb_int32 width;
        lb_uint8 opa;
        lb_uint8 rounded:1;
    } line;
} lb_eui_style_t;

typedef struct tag_lb_eui_point {
    lb_int32        x;
    lb_int32        y;
} lb_eui_point_t;

typedef struct tag_lb_eui_size {
    lb_int32    w;
    lb_int32    h;
} lb_eui_size_t;

typedef struct tag_lb_eui_rect_t {
    lb_int32  x1;
    lb_int32  y1;
    lb_int32  x2;
    lb_int32  y2;
} lb_eui_rect_t;

typedef enum {
    LB_EUI_IMG_TYPE_VARIABLE,
    LB_EUI_IMG_TYPE_COMPRESS,
```

```
    LB_EUI_IMG_TYPE_JPEG,
    LB_EUI_IMG_TYPE_PNG,
    LB_EUI_IMG_TYPE_GIF,
    LB_EUI_IMG_TYPE_BMP,
    LB_EUI_IMG_TYPE_PSD,
    LB_EUI_IMG_TYPE_TGA,
    LB_EUI_IMG_TYPE_HDR,
    LB_EUI_IMG_TYPE_PIC,
    LB_EUI_IMG_TYPE_PNM,
    LB_EUI_IMG_TYPE_MAX
} lb_eui_img_type_e;

typedef struct tag_lb_eui_gif {
    lb_uint16       idx;
    void            *data;
    struct tag_lb_eui_gif     *next;
} lb_eui_gif_t;

typedef struct {
    /* The first 8 bit is very important to distinguish the different source types. */
    lb_uint32 cf:5;/* Color format */
    lb_uint32 always_zero:3;      /*It the upper bits of the first byte.
                    *Always zero to look like a non-printable character
                    */
    lb_uint32 reserved:2;     /*Reserved to be used later*/
    lb_uint32 w:11;      /*Width of the image map*/
    lb_uint32 h:11;      /*Height of the image map*/
} lb_eui_img_header_t;

/* Image header it is compatible with
  * the result image converter utility*/
typedef struct {
    lb_eui_img_header_t header;
    lb_uint32 data_size;
    lb_uint8 *data;
} lb_eui_img_dsc_t;
```

```
typedef struct tag_lb_eui_key_map {
    lb_uint8  key_code;
    lb_int16  btn_id;
} lb_eui_key_map_t;
typedef struct tag_lb_eui_polygon {
    lb_eui_point_t      points[8];
    lb_uint8        size;
    lb_int32        thick;
    lb_int32        meters;
    lb_uint32       color;
} lb_eui_polygon_t;

typedef struct tag_lb_eui_blk_draw_info {
    lb_uint8        type;
    lb_uint8        idx;
    lb_eui_rect_t       rect;
    lb_eui_polygon_t    polygon;
    lb_uint8        *buff;
    lb_int32        max_width;
    lb_int32        max_height;
#ifdef LINUX
    lb_uint32       fb_id;
#endif
} lb_eui_blk_draw_info_t;

typedef struct lb_eui_ion_buf {
    lb_int32 size;
    lb_uint32 width;
    lb_uint32 height;
    lb_int32 share_fd;
    lb_uint32 drm_hdl;
    lb_uint32 drm_fb_id;
} lb_eui_ion_buf_t;

typedef enum {
    LB_EUI_BLK_DRAW_NORMAL,
    LB_EUI_BLK_DRAW_POLYGON,
```

```
    LB_EUI_BLK_DRAW_METER,
} lb_eui_blk_draw_type_e;

typedef enum {
    OBJ_ARC = 0,
    OBJ_BAR = 1,
    OBJ_BUTTON = 2,
    OBJ_BTNM = 3,
    OBJ_CALENDAR = 4,
    OBJ_CHECKBOX = 5,
    OBJ_CHART = 6,
    OBJ_CONTAINER = 7,
    OBJ_DDLIST = 8,
    OBJ_GAUGE = 9,
    OBJ_IMG = 10,
    OBJ_IMGBTN = 11,
    OBJ_KB = 12,
    OBJ_LABEL = 13,
    OBJ_LED = 14,
    OBJ_LINE = 15,
    OBJ_LIST = 16,
    OBJ_LMETER = 17,
    OBJ_MSGBOX = 18,
    OBJ_PAGE = 19,
    OBJ_PRELOAD = 20,
    OBJ_ROLLER = 21,
    OBJ_SLIDER = 22,
    OBJ_SW = 23,
    OBJ_TA = 24,
    OBJ_TABVIEW = 25,
    OBJ_WINDOW = 26,
    OBJ_CANVAS = 27,
    OBJ_SPINBOX = 28,
    OBJ_TILEVIEW = 29,
    OBJ_MAX_TYPE
} lb_eui_obj_type_e;
```

其中 OBJ_LIST、OBJ_TABVIEW、OBJ_WINDOW、OBJ_CANVAS 和 OBJ_TILEVIEW 不支持动态操作接口

lb_eui_obj_set_pos

功能	设置控件的坐标
函数	lb_int32 lb_eui_obj_set_pos(lb_uint32 id, lb_int32 x, lb_int32 y)
参数	id：控件的 ID 号；x：x 方向的坐标；y：y 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_size

功能	设置控件的大小
函数	lb_int32 lb_eui_obj_set_size(lb_uint32 id, lb_int32 w, lb_int32 h)
参数	id：控件的 ID 号；w：控件的宽度；h：控件的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_hidden

功能	设置控件显示与隐藏
函数	lb_int32 lb_eui_obj_set_hidden(lb_uint32 id, bool en)
参数	id：控件的 ID 号；en：false 显示；true 隐藏。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_click

功能	设置控件的点击使能
函数	lb_int32 lb_eui_obj_set_click(lb_uint32 id, bool en)
参数	id：控件的 ID 号；en：false 不能点击；true 可点击。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_align

功能	设置控件的对齐
函数	lb_int32 lb_eui_obj_set_align(lb_uint32 id, lb_uint32 align_id,lb_eui_align_e align_mode, lb_int32 x_offset, lb_int32 y_offset)
参数	id：控件的 ID 号；align_id：对其控件的 ID 号；align_mode：对齐模式； x_offset：x 方向的偏移；y_offset：y 方向的偏移。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_in_which_btn

功能	设置控件的在列表控件的某一行
函数	lb_int32 lb_eui_obj_set_in_which_btn(lb_uint32 id, lb_uint32 list_id, lb_uint8 btn_index)
参数	id：控件的 ID 号；list_id：list 控件 ID；btn_index：列表控件行索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_x

功能	设置控件的 x 方向坐标
函数	lb_int32 lb_eui_obj_set_x(lb_uint32 id, lb_int32 x)
参数	id：控件的 ID 号；x：x 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_y

功能	设置控件的 y 方向坐标
函数	lb_int32 lb_eui_obj_set_y(lb_uint32 id, lb_int32 y)
参数	id：控件的 ID 号；y：y 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_w

功能	设置控件的宽度
函数	lb_int32 lb_eui_obj_set_w(lb_uint32 id, lb_int32 w)
参数	id：控件的 ID 号；w：控件的宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_set_h

功能	设置控件的高度
函数	lb_int32 lb_eui_obj_set_h(lb_uint32 id, lb_int32 h)
参数	id：控件的 ID 号；h：控件的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_get_x

功能	获取控件的 x 方向坐标
函数	lb_int32 lb_eui_obj_get_x(lb_uint32 id, lb_int32 *x)
参数	id：控件的 ID 号；x：x 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_get_y

功能	获取控件的 y 方向坐标
函数	lb_int32 lb_eui_obj_get_y(lb_uint32 id, lb_int32 *y)
参数	id：控件的 ID 号；y：y 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_get_width

功能	获取控件的宽度

函数	lb_int32 lb_eui_obj_get_width(lb_uint32 id, lb_int32 *w)
参数	id：控件的 ID 号；w：控件的宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_get_height

功能	获取控件的高度
函数	lb_int32 lb_eui_obj_get_height(lb_uint32 id, lb_int32 *h)
参数	id：控件的 ID 号；h：控件的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_get_hidden

功能	获取控件的隐藏属性
函数	lb_int32 lb_eui_obj_get_hidden(lb_uint32 id, bool *en)
参数	id：控件的 ID 号；start：en：false 显示；true 隐藏。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_get_click

功能	获取控件的点击属性
函数	lb_int32 lb_eui_obj_get_click(lb_uint32 id, bool *en)
参数	id：控件的 ID 号；en：false 不能点击；true 可点击。
返回值	0 表示成功，其他负值表示错误码。

动态接口

lb_eui_obj_create

功能	创建新的控件
函数	void *lb_eui_obj_create(lb_uint8 type, lb_uint32 parent_id, void *parent_obj)
参数	type：控件类型，参考 lb_eui_obj_type_e；parent_id：父控件 ID；parent_obj：父控件句柄，两个只需填写一个即可。
返回值	控件句柄。

lb_eui_obj_del

功能	删除控件
函数	lb_int32 lb_eui_obj_del(void *obj)
参数	obj：控件句柄。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_pos

功能	设置控件的坐标
函数	lb_int32 lb_eui_obj_ext_set_pos(void *obj, lb_int32 x, lb_int32 y)
参数	obj：控件句柄；x：x 方向的坐标；y：y 方向的坐标。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_size

功能	设置控件的大小
函数	lb_int32 lb_eui_ext_obj_set_size(void *obj, lb_int32 w, lb_int32 h)
参数	obj：控件句柄；w：控件的宽度；h：控件的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_hidden

功能	设置控件显示与隐藏
函数	lb_int32 lb_eui_obj_ext_set_hidden(void *obj, bool en)
参数	obj：控件句柄；en：false 显示；true 隐藏。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_click

功能	设置控件的点击使能
函数	lb_int32 lb_eui_obj_ext_set_click(void *obj, bool en)
参数	obj：控件句柄；en：false 不能点击；true 可点击。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_align

功能	设置控件的对齐
函数	lb_int32 lb_eui_obj_ext_set_align(void *obj, void *align, lb_uint32 align_id, lb_eui_align_e align_mode, lb_int32 x_offset, lb_int32 y_offset)
参数	obj：控件句柄；align：对齐控件句柄；align_id：对齐控件的 ID 号；align_mode：对齐模式； x_offset：x 方向的偏移；y_offset：y 方向的偏移， align 与 align_id 填写一个即可。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_x

功能	设置控件的 x 方向坐标
函数	lb_int32 lb_eui_obj_ext_set_x(void *obj, lb_int32 x)
参数	obj：控件句柄；x：x 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_y

功能	设置控件的 y 方向坐标
函数	lb_int32 lb_eui_obj_ext_set_y(void *obj, lb_int32 y)
参数	obj：控件句柄；y：y 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_w

功能	设置控件的宽度
函数	lb_int32 lb_eui_obj_ext_set_w(void *obj, lb_int32 w)
参数	obj：控件句柄；w：控件的宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_set_h

功能	设置控件的高度
函数	lb_int32 lb_eui_obj_ext_set_h(void *obj, lb_int32 h)
参数	obj：控件句柄；h：控件的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_get_x

功能	获取控件的 x 方向坐标
函数	lb_int32 lb_eui_obj_ext_get_x(void *obj, lb_int32 *x)
参数	obj：控件句柄；x：x 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_get_y

功能	获取控件的 y 方向坐标
函数	lb_int32 lb_eui_obj_ext_get_y(void *obj, lb_int32 *y)
参数	obj：控件句柄；y：y 方向的坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_get_width

功能	获取控件的宽度
函数	lb_int32 lb_eui_obj_ext_get_width(void *obj, lb_int32 *w)
参数	obj：控件句柄；w：控件的宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_get_height

功能	获取控件的高度
函数	lb_int32 lb_eui_obj_ext_get_height(void *obj, lb_int32 *h)
参数	obj：控件句柄；h：控件的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_get_hidden

功能	获取控件的隐藏属性

函数	lb_int32 lb_eui_obj_ext_get_hidden(void *obj, bool *en)
参数	obj：控件句柄；start：en：false 显示；true 隐藏。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_obj_ext_get_click

功能	获取控件的点击属性
函数	lb_int32 lb_eui_obj_ext_get_click(void *obj, bool *en)
参数	obj：控件句柄；en：false 不能点击；true 可点击。
返回值	0 表示成功，其他负值表示错误码。

4.5.2. Arc 控件

Arc 控件的 API 接口定义如下：

```
typedef enum {
    LB_EUI_ARC_STYLE_MAIN,
} lb_eui_arc_style_e;
```

lb_eui_arc_set_angles

功能	设置 arc 的弧度，范围是 0 – 360。
函数	lb_int32 lb_eui_arc_set_angles(lb_int32 id, lb_int16 start, lb_int16 end)
参数	id：arc 控件的 ID 号；start：arc 的起始角度；end：arc 的结束角度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_arc_set_style

功能	设置 arc 的风格。
函数	lb_int32 lb_eui_arc_set_style(lb_int32 id, lb_eui_arc_style_e type, lb_eui_style_t *lb_arc_style)
参数	id：arc 控件的 ID 号；type：风格类型，参考 lb_eui_arc_style_e；lb_arc_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_arc_get_angles

功能	获取 arc 的弧度
函数	lb_int32 lb_eui_arc_get_angles(lb_int32 id, lb_int16 *start, lb_int16 *end)
参数	id：arc 控件的 ID 号；start：返回 arc 的起始角度；end：返回 arc 的结束角度。
返回值	0 表示成功，其他负值表示错误码。

弧度以从 0 度到 360 度。垂直向下的方向为 0 度。

lb_eui_arc_get_style

功能	获取 arc 的风格。
函数	lb_int32 lb_eui_arc_get_style(lb_int32 id, lb_eui_arc_style_e type, lb_eui_style_t *lb_arc_style)
参数	id：arc 控件的 ID 号；type：风格类型，参考 lb_eui_arc_style_e；lb_arc_style：返回风格的指针。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_arc_ext_set_angles

功能	设置 arc 的弧度，范围是 0 – 360。
函数	lb_int32 lb_eui_arc_ext_set_angles(void *arc, lb_int16 start, lb_int16 end)
参数	acr：arc 控件句柄；start：arc 的起始角度；end：arc 的结束角度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_arc_ext_set_style

功能	设置 arc 的风格。
函数	lb_int32 lb_eui_arc_ext_set_style(void *arc, lb_eui_arc_style_e type, lb_eui_style_t *lb_arc_style)
参数	acr：arc 控件句柄；type：风格类型，参考 lb_eui_arc_style_e；lb_arc_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_arc_ext_get_angles

功能	获取 arc 的弧度
函数	lb_int32 lb_eui_arc_ext_get_angles(void *arc, lb_int16 *start, lb_int16 *end)
参数	acr：arc 控件句柄；start：返回 arc 的起始角度；end：返回 arc 的结束角度。
返回值	0 表示成功，其他负值表示错误码。

弧度以从 0 度到 360 度。垂直向下的方向为 0 度。

lb_eui_arc_ext_get_style

功能	获取 arc 的风格。
函数	lb_int32 lb_eui_arc_ext_get_style(void *arc, lb_eui_arc_style_e type, lb_eui_style_t *lb_arc_style)
参数	acr：arc 控件句柄；type：风格类型，参考 lb_eui_arc_style_e；lb_arc_style：返回风格的指针。
返回值	0 表示成功，其他负值表示错误码。

4.5.3. Bar 控件

```
typedef enum {
    LB_EUI_BAR_STYLE_BG,
    LB_EUI_BAR_STYLE_INDIC,
} lb_eui_bar_style_e;
```

/* setter function */

lb_eui_bar_set_range

功能	设置 bar 的范围
函数	lb_int32 lb_eui_bar_set_range(lb_int32 id, lb_int16 min, lb_int16 max)
参数	id：bar 控件的 ID 号；min：bar 的最小值；max：bar 的最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_set_value

功能	设置 bar 的值
函数	lb_int32 lb_eui_bar_set_value(lb_uint32 id, lb_int16 val, lb_uint8 en_anim)
参数	id：bar 控件的 ID 号；val：bar 的当前值；en_anim：是否开启动画。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_set_symmetric

功能	设置 bar 对称
函数	lb_int32 lb_eui_bar_set_symetric(lb_int32 id, bool en)
参数	id：bar 控件的 ID 号；en：true 为设置对称，即 0 点在 bar 的中点位置，false：正常模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_set_style

功能	设置 bar 风格
函数	lb_int32 lb_eui_bar_set_style(lb_int32 id, lb_eui_bar_style_e type, lb_eui_style_t *lb_bar_style)
参数	id：bar 控件的 ID 号；type：风格类型，参考 lb_eui_bar_style_e；lb_bar_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

/* getter function */

lb_eui_bar_get_range

功能	获取 bar 的范围
函数	lb_int32 lb_eui_bar_get_range(lb_int32 id, lb_int16 *min, lb_int16 *max)
参数	id：bar 控件的 ID 号；min：返回 bar 的最小值；max：返回 bar 的最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_get_value

功能	获取 bar 的值
函数	lb_int32 lb_eui_bar_get_value(lb_int32 id, lb_int16 *val)
参数	id：bar 控件的 ID 号；val：返回 bar 的当前值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_get_symetric

功能	获取 bar 对称

函数	lb_int32 lb_eui_bar_get_symetric(lb_int32 id, bool *en)
参数	id：bar 控件的 ID 号；*en：返回 bar 对称模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_get_style

功能	获取 bar 风格
函数	lb_int32 lb_eui_bar_get_style(lb_int32 id, lb_eui_bar_style_e type, lb_eui_style_t *lb_bar_style)
参数	id：bar 控件的 ID 号；type：风格类型，参考 lb_eui_bar_style_e；lb_bar_style：返回的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

/* setter function */

lb_eui_bar_ext_set_range

功能	设置 bar 的范围
函数	lb_int32 lb_eui_bar_ext_set_range(void *bar, lb_int16 min, lb_int16 max)
参数	bar：bar 控件句柄；min：bar 的最小值；max：bar 的最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_ext_set_value

功能	设置 bar 的值
函数	lb_int32 lb_eui_bar_ext_set_value(void *bar, lb_int16 val, lb_uint8 en_anim)
参数	bar：bar 控件句柄；val：bar 的当前值；en_anim：是否开启动画。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_ext_set_symmetric

功能	设置 bar 对称
函数	lb_int32 lb_eui_bar_ext_set_symetric(void *bar, bool en)
参数	bar：bar 控件句柄；en：true 为设置对称，即 0 点在 bar 的中点位置，false：正常模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_ext_set_style

功能	设置 bar 风格
函数	lb_int32 lb_eui_bar_ext_set_style(void *bar, lb_eui_bar_style_e type, lb_eui_style_t *lb_bar_style)
参数	bar：bar 控件句柄；type：风格类型，参考 lb_eui_bar_style_e；lb_bar_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

/* getter function */

lb_eui_bar_ext_get_range

功能	获取 bar 的范围
函数	lb_int32 lb_eui_bar_ext_get_range(void *bar, lb_int16 *min, lb_int16 *max)
参数	bar：bar 控件句柄；min：返回 bar 的最小值；max：返回 bar 的最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_ext_get_value

功能	获取 bar 的值
函数	lb_int32 lb_eui_bar_ext_get_value(void *bar, lb_int16 *val)
参数	bar：bar 控件句柄；val：返回 bar 的当前值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_ext_get_symetric

功能	获取 bar 对称
函数	lb_int32 lb_eui_bar_ext_get_symetric(void *bar, bool *en)
参数	bar：bar 控件句柄；*en：返回 bar 对称模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_bar_ext_get_style

功能	获取 bar 风格
函数	lb_int32 lb_eui_bar_ext_get_style(void *bar, lb_eui_bar_style_e type, lb_eui_style_t *lb_bar_style)
参数	bar：bar 控件句柄；type：风格类型，参考 lb_eui_bar_style_e；lb_bar_style：返回的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

4.5.4. Button 控件

```
typedef enum
{
    LB_EUI_BTN_STATE_REL, /* 释放状态 */
    LB_EUI_BTN_STATE_PR, /* 按下状态 */
    LB_EUI_BTN_STATE_TGL_REL, /* 释放保持状态 */
    LB_EUI_BTN_STATE_TGL_PR, /* 按下保持状态 */
    LB_EUI_BTN_STATE_INA, /* 无效状态 */
    LB_EUI_BTN_STATE_NUM,
}lb_eui_btn_state_e;

typedef enum {
    LB_EUI_BTN_STYLE_REL,
    LB_EUI_BTN_STYLE_PR,
```

```
    LB_EUI_BTN_STYLE_TGL_REL,
    LB_EUI_BTN_STYLE_TGL_PR,
    LB_EUI_BTN_STYLE_INA,
} lb_eui_btn_style_e;
```

Button 控件的 API 接口函数定义如下：

lb_eui_btn_set_toggle

功能	设置 button 控件是否支持 toggle 状态
函数	lb_int32 lb_eui_btn_set_toggle(lb_int32 id, bool en)
参数	id：button 控件的 ID 号；en：true 允许 toggle 状态，false 禁止 toggle 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_fit

功能	设置 button 控件内部控件水平和垂直方向自适应使能
函数	lb_int32 lb_eui_btn_set_fit(lb_uint32 id, bool en_hor, bool en_ver)
参数	id：button 控件的 ID 号；en_hor：true 水平方向自适应，false 水平方向默认排列； en_ver：true 垂直方向自适应，false 垂直方向默认排列。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_layout

功能	设置 button 控件内部布局方式
函数	lb_int32 lb_eui_btn_set_layout(lb_uint32 id, lb_eui_layout_e layout)
参数	id：button 控件的 ID 号；layout：布局方式，参考‘lb_eui_layout_e’。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_text

功能	设置 button 控件字符串
函数	lb_int32 lb_eui_btn_set_text(lb_uint32 id, char *txt)
参数	id：button 控件的 ID 号；txt：字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_text_id

功能	设置 button 控件字符串索引值
函数	lb_int32 lb_eui_btn_set_text_id(lb_uint32 id, lb_uint32 str_id)
参数	id：button 控件的 ID 号；str_id：字符串在资源文件中的索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_state

功能	设置 button 控件的状态。
函数	lb_int32 lb_eui_btn_set_state(lb_int32 id,lb_eui_btn_state_e state)
参数	id：button 控件的 ID 号；state：button 状态。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_text_longmode

功能	设置 button 控件字符串长模式
函数	lb_eui_btn_set_text_longmode (lb_uint32 id, lb_uint8 longmode)
参数	id：button 控件的 ID 号；longmode：长字符串的处理模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_text_align

功能	设置 button 控件文本自身对齐方式
函数	lb_int32 lb_eui_btn_set_text_align (lb_uint32 id, lb_uint8 align)
参数	id：button 控件的 ID 号；align：label 文本对齐方式，LABEL_ALIGN_LEFT 左对齐，LABEL_ALIGN_CENTER 居中，LABEL_ALIGN_RIGHT 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_text_alignmode

功能	设置 button 控件的文本相对于控件的对齐方式。
函数	lb_int32 lb_eui_btn_set_text_alignmode(lb_uint32 id, lb_uint8 alignmode,lb_int32 x_offset, lb_int32 y_offset)
参数	id：button 控件的 ID 号；align_mode：对齐模式；x_offset：x 方向的偏移；y_offset：y 方向的偏移。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_toggle

Button 切换： Release->toggle release； Press -> toggle press； Toggle release –> release； Toggle press -> press；	
功能	设置 button 控件的 toggle 状态切换。
函数	lb_int32 lb_eui_btn_toggle(lb_int32 id)
参数	id：button 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_set_style

功能	设置 button 控件的风格。
函数	lb_int32 lb_eui_btn_set_style(lb_int32 id, lb_eui_btn_style_e type, lb_eui_style_t *lb_btn_style);
参数	id：button 控件的 ID 号； type：风格类型，参考 lb_eui_btn_style_e；lb_btn_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

/* getter function */

lb_eui_btn_get_state

功能	获取 button 控件的状态。
函数	lb_int32 lb_eui_btn_get_state(lb_int32 id, lb_uint8 *state)
参数	id：button 控件的 ID 号；*state：返回 button 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_get_fit

功能	获取 button 控件水平和垂直方向自适应属性
函数	lb_int32 lb_eui_btn_get_fit(lb_uint32 id, bool *en_hor, bool *en_ver)
参数	id：button 控件的 ID 号；en_hor：true 水平方向自适应，false 水平方向默认排列； en_ver：true 垂直方向自适应，false 垂直方向默认排列。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_get_toggle

功能	获取 button 控件是否支持 toggle 状态
函数	lb_int32 lb_eui_btn_get_toggle(lb_int32 id, bool *en)
参数	id：button 控件的 ID 号；en：返回 button 的 toggle 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_get_style

功能	获取 button 控件的风格。
函数	lb_int32 lb_eui_btn_get_style(lb_int32 id, lb_eui_btn_style_e type, lb_eui_style_t *lb_btn_style)
参数	id：button 控件的 ID 号； type：风格类型，参考 lb_eui_btn_style_e；lb_btn_style：返回的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_toggle

功能	设置 button 控件是否支持 toggle 状态
函数	lb_int32 lb_eui_btn_ext_set_toggle(void *btn, bool en)
参数	btn：button 控件句柄；en：true 允许 toggle 状态，false 禁止 toggle 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_fit

功能	设置 button 控件内部控件水平和垂直方向自适应使能
函数	lb_int32 lb_eui_btn_ext_set_fit(void *btn, bool en_hor, bool en_ver)
参数	btn：button 控件句柄；en_hor：true 水平方向自适应，false 水平方向默认排列； en_ver：true 垂直方向自适应，false 垂直方向默认排列。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_layout

功能	设置 button 控件内部布局方式
函数	lb_int32 lb_eui_btn_ext_set_layout(void *, lb_eui_layout_e layout)

参数	btn：button 控件句柄；layout：布局方式，参考‘lb_eui_layout_e’。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_text

功能	设置 button 控件字符串
函数	lb_int32 lb_eui_btn_ext_set_text(void *btn, char *txt)
参数	btn：button 控件句柄；txt：字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_text_id

功能	设置 button 控件字符串索引值
函数	lb_int32 lb_eui_btn_ext_set_text_id(void *btn, lb_uint32 str_id)
参数	btn：button 控件句柄；str_id：字符串在资源文件中的索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_state

功能	设置 button 控件的状态。
函数	lb_int32 lb_eui_btn_ext_set_state(void *btn,lb_eui_btn_state_e state)
参数	btn：button 控件句柄；state：button 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_text_longmode

功能	设置 button 控件字符串长模式
函数	lb_eui_btn_ext_set_text_longmode (void *btn, lb_uint8 longmode)
参数	btn：button 控件句柄；longmode：长字符串的处理模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_text_align

功能	设置 button 控件文本自身对齐方式
函数	lb_int32 lb_eui_btn_ext_set_text_align (void *btn, lb_uint8 align)
参数	btn：button 控件句柄；align：label 文本对齐方式，LABEL_ALIGN_LEFT 左对齐，LABEL_ALIGN_CENTER 居中，LABEL_ALIGN_RIGHT 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_text_alignmode

功能	设置 button 控件的文本相对于控件的对齐方式。
函数	lb_int32 lb_eui_btn_ext_set_text_alignmode(void *btn, lb_uint8 alignmode,lb_int32 x_offset, lb_int32 y_offset)

参数	btn：button 控件句柄；align_mode：对齐模式；x_offset：x 方向的偏移；y_offset：y 方向的偏移。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_toggle

Button 切换： Release->toggle release； Press -> toggle press； Toggle release –> release； Toggle press -> press；	
功能	设置 button 控件的 toggle 状态切换。
函数	lb_int32 lb_eui_btn_ext_toggle(void *obj)
参数	btn：button 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_set_style

功能	设置 button 控件的风格。
函数	lb_int32 lb_eui_btn_ext_set_style(void *btn, lb_eui_btn_style_e type, lb_eui_style_t *lb_btn_style);
参数	btn：button 控件句柄； type：风格类型，参考 lb_eui_btn_style_e；lb_btn_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

/* getter function */

lb_eui_btn_ext_get_state

功能	获取 button 控件的状态。
函数	lb_int32 lb_eui_btn_ext_get_state(void *btn, lb_uint8 *state)
参数	btn：button 控件句柄；state：返回 button 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_get_fit

功能	获取 button 控件水平和垂直方向自适应属性
函数	lb_int32 lb_eui_btn_ext_get_fit(void *btn, bool *en_hor, bool *en_ver)
参数	btn：button 控件句柄；en_hor：true 水平方向自适应，false 水平方向默认排列； en_ver：true 垂直方向自适应，false 垂直方向默认排列。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_get_toggle

功能	获取 button 控件是否支持 toggle 状态
函数	lb_int32 lb_eui_btn_ext_get_toggle(void *btn, bool *en)
参数	btn：button 控件句柄；en：返回 button 的 toggle 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btn_ext_get_style

功能	获取 button 控件的风格。
函数	lb_int32 lb_eui_btn_ext_get_style(void *btn, lb_eui_btn_style_e type, lb_eui_style_t *lb_btn_style)
参数	btn：button 控件句柄； type：风格类型，参考 lb_eui_btn_style_e；lb_btn_style：返回的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

4.5.5. Button matrix 控件

```
typedef enum {
    LB_EUI_BTNM_STYLE_BG,
    LB_EUI_BTNM_STYLE_BTN_REL,
    LB_EUI_BTNM_STYLE_BTN_PR,
    LB_EUI_BTNM_STYLE_BTN_TGL_REL,
    LB_EUI_BTNM_STYLE_BTN_TGL_PR,
    LB_EUI_BTNM_STYLE_BTN_INA,
} lb_eui_btnm_style_e;
```

Btnm 控件支持的 API 接口如下：

lb_eui_btnm_set_map

Btnm map 中每个字符串表示 button matrix 中的一个 button 文本。Button 文本包含两个部分，首字节为控制字节，从第二个字节开始为 button 显示的文本。控制字节的定义如下： Bit 7…6：1； Bit 5：设置为 1 表示未激活，禁用状态； Bit 4：设置为 1 表示不响应长按； Bit 3：设置为 1 表示隐藏； Bit 2…0：button 相对宽度，即占用几个 button 宽度。	
功能	设置 btnm 控件 map。
函数	lb_int32 lb_eui_btnm_set_map(lb_int32 id, char **map)
参数	id：button 控件的 ID 号；map：字符串指针数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_set_toggle

功能	设置 btnm 中 button 的是否支持 toggle 状态。

函数	lb_int32 lb_eui_btnm_set_toggle(lb_int32 id, bool en, lb_int16 index)
参数	id：button 控件的 ID 号；en：true 表示支持 toggle 状态，false 表示不支持 toggle 状态；index：button 的索引号。
返回值	0 表示成功，其他负值表示错误码。

Recolor 是值解析按键文本中自带的颜色字符串。颜色字符串以#开头，以#结尾，使用 16 进制数表示，列如 char string[16] = {“#FF0000 red#”}，显示的是红色的 red 字符串。

lb_eui_btnm_set_recolor

功能	设置 btnm 中 button 的是否支持 recolor。
函数	lb_int32 lb_eui_btnm_set_recolor(lb_int32 id, bool en)
参数	id：button 控件的 ID 号；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_set_style

功能	设置 btnm 的风格。
函数	lb_int32 lb_eui_btnm_set_style(lb_int32 id, lb_eui_btnm_style_e type, lb_eui_style_t *lb_btnm_style)
参数	id：btnm 控件的 ID 号；type：风格类型，参考 lb_eui_btnm_style_e； lb_btnm_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_get_map

功能	获取 btnm 控件 map。
函数	char **lb_eui_btnm_get_map(lb_int32 id)
参数	id：button 控件的 ID 号。
返回值	按键矩阵的字符串指针数组。

lb_eui_btnm_get_pressed

功能	获取 btnm 当前被按下的按键索引。
函数	lb_int32 lb_eui_btnm_get_pressed(lb_int32 id， lb_uint16 *index)
参数	id：button 控件的 ID 号；index：返回按下状态的按键的索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_get_toggle

功能	获取 btnm 中处于 toggle 状态的 button 的索引值。
函数	lb_int32 lb_eui_btnm_get_toggle(lb_int32 id, lb_int16 *index)
参数	id：button 控件的 ID 号；index：保存当前处于 toggle 状态的 button 的索引号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_get_recolor

功能	获取 btnm 中 button 的是否支持 recolor。

函数	lb_int32 lb_eui_btnm_get_recolor(lb_int32 id, bool *en)
参数	id：button 控件的 ID 号；en：保存是否支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_get_style

功能	获取 btnm 的风格。
函数	lb_int32 lb_eui_btnm_get_style(lb_int32 id, lb_eui_btnm_style_e type, lb_eui_style_t *lb_btnm_style)
参数	id：btnm 控件的 ID 号；type：风格类型，参考 lb_eui_btnm_style_e； lb_btnm_style：需要获取的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_ext_set_map

Btnm map 中每个字符串表示 button matrix 中的一个 button 文本。Button 文本包含两个部分，首字节为控制字节，从第二个字节开始为 button 显示的文本。控制字节的定义如下： Bit 7…6：1； Bit 5：设置为 1 表示未激活，禁用状态； Bit 4：设置为 1 表示不响应长按； Bit 3：设置为 1 表示隐藏； Bit 2…0：button 相对宽度，即占用几个 button 宽度。

功能	设置 btnm 控件 map。
函数	lb_int32 lb_eui_btnm_ext_set_map(void *btnm, char **map)
参数	btnm：btnm 控件句柄；map：字符串指针数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_ext_set_action

功能	设置 btnm 中 button 的点击事件回调。
函数	lb_int32 lb_eui_btnm_ext_set_action (void *btnm, lb_eui_btnm_action_t action)
参数	btnm：btnm 控件句柄；action：点击事件回调。
返回值	0 表示成功，其他负值表示错误码。
Recolor 是值解析按键文本中自带的颜色字符串。颜色字符串以#开头，以#结尾，使用 16 进制数表示，列如 char string[16] = {“#FF0000 red#”}，显示的是红色的 red 字符串。	

lb_eui_btnm_ext_set_toggle

功能	设置 btnm 中 button 的是否支持 toggle 状态。
函数	lb_int32 lb_eui_btnm_ext_set_toggle(void *btnm, bool en, lb_int16 index)
参数	btnm：btnm 控件句柄；en：true 表示支持 toggle 状态，false 表示不支持 toggle 状态；index：button 的索引号。

返回值	0 表示成功，其他负值表示错误码。
Recolor 是值解析按键文本中自带的颜色字符串。颜色字符串以#开头，以#结尾，使用 16 进制数表示，列如 char string[16] = {“#FF0000 red#”}，显示的是红色的 red 字符串。	

lb_eui_btnm_ext_set_recolor

功能	设置 btnm 中 button 的是否支持 recolor。
函数	lb_int32 lb_eui_btnm_ext_set_recolor(void *btnm, bool en)
参数	btnm：btnm 控件句柄；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_ext_get_map

功能	获取 btnm 控件 map。
函数	char **lb_eui_btnm_ext_get_map(void *btnm)
参数	btnm：btnm 控件句柄。
返回值	按键矩阵的字符串指针数组。

lb_eui_btnm_ext_set_style

功能	设置 btnm 的风格。
函数	lb_int32 lb_eui_btnm_ext_set_style(void *btnm, lb_eui_btnm_style_e type, lb_eui_style_t *lb_btnm_style)
参数	btnm：btnm 控件句柄；type：风格类型，参考 lb_eui_btnm_style_e； lb_btnm_style：需要设置的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_ext_get_map

功能	获取 btnm 控件 map。
函数	char **lb_eui_btnm_ext_get_map(void *btnm)
参数	btnm：btnm 控件句柄。
返回值	按键矩阵的字符串指针数组。

lb_eui_btnm_ext_get_pressed

功能	获取 btnm 当前被按下的按键索引。
函数	lb_int32 lb_eui_btnm_ext_get_pressed(void *btnm，lb_uint16 *index)
参数	btnm：btnm 控件句柄；index：返回按下状态的按键的索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_ext_get_toggle

功能	获取 btnm 中处于 toggle 状态的 button 的索引值。
函数	lb_int32 lb_eui_btnm_ext_get_toggle(void *btnm, lb_int16 *index)
参数	btnm：btnm 控件句柄；index：保存当前处于 toggle 状态的 button 的索引号。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_ext_get_recolor

功能	获取 btnm 中 button 的是否支持 recolor。
函数	lb_int32 lb_eui_btnm_ext_get_recolor(void *btnm, bool *en)
参数	btnm：btnm 控件句柄；en：保存是否支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_btnm_ext_get_style

功能	获取 btnm 的风格。
函数	lb_int32 lb_eui_btnm_ext_get_style(void *btnm, lb_eui_btnm_style_e type, lb_eui_style_t *lb_btnm_style)
参数	btnm：btnm 控件句柄；type：风格类型，参考 lb_eui_btnm_style_e； lb_btnm_style：需要获取的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

4.5.6. Calendar 控件

```
typedef struct {
    lb_uint16 year;
    lb_int8 month;
    lb_int8 day;
} lb_eui_date_t;
typedef enum {
    LB_EUI_CALENDAR_STYLE_BG,          /*Also the style of the "normal" date numbers*/
    LB_EUI_CALENDAR_STYLE_HEADER,
    LB_EUI_CALENDAR_STYLE_HEADER_PR,
    LB_EUI_CALENDAR_STYLE_DAY_NAMES,
    LB_EUI_CALENDAR_STYLE_HIGHLIGHTED_DAYS,
    LB_EUI_CALENDAR_STYLE_INACTIVE_DAYS,
    LB_EUI_CALENDAR_STYLE_WEEK_BOX,
    LB_EUI_CALENDAR_STYLE_TODAY_BOX,
} lb_eui_calendar_style_e;
```

Calendar API 接口函数定义如下：

lb_eui_calendar_set_today

功能	设置 calendar 控件今天的日期。
函数	lb_int32 lb_eui_calendar_set_today(lb_uint32 id, lb_eui_date_t *today)
参数	id：calendar 控件的 ID 号；today：今天的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_set_showed

功能	设置 calendar 控件显示的日期。
函数	lb_int32 lb_eui_calendar_set_showed(lb_uint32 id, lb_eui_date_t *showed)
参数	id：calendar 控件的 ID 号；showed：显示的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_set_highlights

功能	设置 calendar 控件高亮的日期， 如果日期参数为 NULL 或者 num 为 0 则取消所有高亮日期。
函数	lb_int32 lb_eui_calendar_set_highlights(lb_uint32 id, lb_eui_date_t *highlighted, lb_uint16 date_num)
参数	id：calendar 控件的 ID 号；highlighted：高亮日期数组；num：高亮日期的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_set_day_names

功能	设置 calendar 一周七天的名称。
函数	lb_int32 lb_eui_calendar_set_day_names(lb_int32 id, char **day_names)
参数	id：calendar 控件的 ID 号；day_names：表示星期的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_set_month_names

功能	设置 calendar 月份的名称。
函数	lb_int32 lb_eui_calendar_set_month_names(lb_int32 id, char **month_names)
参数	id：calendar 控件的 ID 号；month_names：表示月份的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_set_style

功能	设置 calendar 的风格。
函数	lb_int32 lb_eui_calendar_set_style(lb_int32 id, lb_eui_calendar_style_e type, lb_eui_style_t *lb_calendar_style)
参数	id：calendar 控件的 ID 号；type：风格类型，参考 lb_eui_calendar_style_e； lb_calendar_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_get_today

功能	获取 calendar 控件今天的日期。
函数	lb_int32 lb_eui_calendar_get_today(lb_uint32 id, lb_eui_date_t **today)
参数	today：今天的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_get_showed

功能	获取 calendar 控件显示的日期。
函数	lb_int32 lb_eui_calendar_get_showed(lb_uint32 id, lb_eui_date_t **show_date)

参数	show_date：显示的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_get_highlights

功能	获取 calendar 控件高亮的日期。
函数	lb_int32 lb_eui_calendar_get_highlights(lb_uint32 id, lb_eui_date_t **highlighted , lb_uint16 *num)
参数	id：calendar 控件的 ID 号；highlighted：高亮日期数组；num：高亮日期个数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_get_day_names

功能	获取 calendar 一周七天的名称。
函数	lb_int32 lb_eui_calendar_get_day_names(lb_int32 id, char **day_names)
参数	id：calendar 控件的 ID 号；day_names：返回表示星期的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_get_month_names

功能	获取 calendar 月份的名称。
函数	lb_int32 lb_eui_calendar_get_month_names(lb_int32 id, char **month_names)
参数	id：calendar 控件的 ID 号；month_names：返回表示月份的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_get_style

功能	获取 calendar 的风格。
函数	lb_int32 lb_eui_calendar_get_style(lb_int32 id, lb_eui_calendar_style_e type, lb_eui_style_t *lb_calendar_style)
参数	id：calendar 控件的 ID 号；type：风格类型，参考 lb_eui_calendar_style_e； lb_calendar_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_set_action

功能	设置 calendar 控件事件响应函数。
函数	lb_int32 lb_eui_calendar_ext_set_action(void *calendar, lb_eui_action_type type, lb_eui_action_t action)
参数	calendar ：calendar 控件句柄；type：事件类型；action：事件回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_set_today

功能	设置 calendar 控件今天的日期。
函数	lb_int32 lb_eui_calendar_ext_set_today(void *calendar, lb_eui_date_t *today)

参数	calendar ： calendar 控件句柄；today：今天的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_set_showed

功能	设置 calendar 控件显示的日期。
函数	lb_int32 lb_eui_calendar_ext_set_showed(void *calendar, lb_eui_date_t *showed)
参数	calendar ： calendar 控件句柄；showed：显示的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_set_highlights

功能	设置 calendar 控件高亮的日期， 如果日期参数为 NULL 或者 num 为 0 则取消所有高亮日期。
函数	lb_int32 lb_eui_calendar_ext_set_highlights(void *calendar, lb_eui_date_t *highlighted, lb_uint16 date_num)
参数	calendar ： calendar 控件句柄；highlighted：高亮日期数组；num：高亮日期的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_set_day_names

功能	设置 calendar 一周七天的名称。
函数	lb_int32 lb_eui_calendar_ext_set_day_names(void *calendar, char **day_names)
参数	calendar ： calendar 控件句柄；day_names：表示星期的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_set_month_names

功能	设置 calendar 月份的名称。
函数	lb_int32 lb_eui_calendar_ext_set_month_names(void *calendar, char **month_names)
参数	calendar ： calendar 控件句柄；month_names：表示月份的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_set_style

功能	设置 calendar 的风格。
函数	lb_int32 lb_eui_calendar_ext_set_style(void *calendar, lb_eui_calendar_style_e type, lb_eui_style_t *lb_calendar_style)
参数	calendar ： calendar 控件句柄；type：风格类型，参考 lb_eui_calendar_style_e； lb_calendar_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_get_today

功能	获取 calendar 控件今天的日期。

函数	lb_int32 lb_eui_calendar_ext_get_today(void *calendar, lb_eui_date_t **today)
参数	calendar ： calendar 控件句柄；today：今天的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_get_showed

功能	获取 calendar 控件显示的日期。
函数	lb_int32 lb_eui_calendar_ext_get_showed(void *calendar, lb_eui_date_t **show_date)
参数	calendar ： calendar 控件句柄；show_date：显示的日期。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_get_highlights

功能	获取 calendar 控件高亮的日期。
函数	lb_int32 lb_eui_calendar_ext_get_highlights(void *calendar, lb_eui_date_t **highlighted , lb_uint16 *num)
参数	calendar ： calendar 控件句柄；highlighted：高亮日期数组；num：高亮日期个数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_get_day_names

功能	获取 calendar 一周七天的名称。
函数	lb_int32 lb_eui_calendar_ext_get_day_names(void *calendar, char **day_names)
参数	calendar ： calendar 控件句柄；day_names：返回表示星期的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_get_month_names

功能	获取 calendar 月份的名称。
函数	lb_int32 lb_eui_calendar_ext_get_month_names(void *calendar, char **month_names)
参数	calendar ： calendar 控件句柄；month_names：返回表示月份的名称字符串数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_calendar_ext_get_style

功能	获取 calendar 的风格。
函数	lb_int32 lb_eui_calendar_ext_get_style(void *calendar, lb_eui_calendar_style_e type, lb_eui_style_t *lb_calendar_style)
参数	calendar ： calendar 控件句柄；type：风格类型，参考 lb_eui_calendar_style_e； lb_calendar_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.7. Canvas 控件

Checkbox 的 API 接口函数说明如下：

```
enum {
    LB_EUI_IMG_CF_UNKOWN = 0,

    LB_EUI_IMG_CF_RAW,          /* Contains the file as it is.
                                 * Needs custom
                                 * decoder function */
    LB_EUI_IMG_CF_RAW_ALPHA,    /* Contains the file as it is.
                                 * The image has
                                 * alpha. Needs custom
                                 * decoder function */
    LB_EUI_IMG_CF_RAW_CHROMA_KEYED,  /* Contains the file as it is.
                                 * The image is
                                 * chroma keyed. Needs custom
                                 * decoder function */
    LB_EUI_IMG_CF_TRUE_COLOR,   /* Color format and depth
                                 * should match with
                                 * LB_COLOR settings */
    LB_EUI_IMG_CF_TRUE_COLOR_ALPHA,   /* Same as
                                 * `LB_EUI_IMG_CF_TRUE_COLOR`
                                 * but every pixel has an alpha byte */
    LB_EUI_IMG_CF_TRUE_COLOR_CHROMA_KEYED, /*Same as `
                                * LB_EUI_IMG_CF_TRUE_COLOR` but
                               *'0x00ff00' pixels will be transparent */
    LB_EUI_IMG_CF_INDEXED_1BIT,/* Can have 2 different colors in a palette
                                 * (always chroma keyed) */
    LB_EUI_IMG_CF_INDEXED_2BIT,/* Can have 4 different colors in a palette
                                 * (always chroma keyed) */
    LB_EUI_IMG_CF_INDEXED_4BIT,/* Can have 16 different colors in a palette
                                 * (always chroma keyed) */
    LB_EUI_IMG_CF_INDEXED_8BIT,/* Can have 256 different colors in a palette
                                 * (always chroma keyed) */
    LB_EUI_IMG_CF_ALPHA_1BIT,/* Can have one color and it can be
                                 * drawn or not */
    LB_EUI_IMG_CF_ALPHA_2BIT,/* Can have one color but 4 different
                                 * alpha value */
    LB_EUI_IMG_CF_ALPHA_4BIT,/* Can have one color but 16 different
                                 * alpha value */
```

```
    LB_EUI_IMG_CF_ALPHA_8BIT,/* Can have one color but 256 different
                              * alpha value */
};
typedef lb_uint8 lb_eui_img_cf_t;

typedef struct tag_lb_eui_canvas_polygon {
    lb_eui_point_t      points[8];/* points array, current max support
                                  * 8 points */
    lb_uint8        point_nums;   /* number of points*/
    lb_uint32       line_color; /* polygon line color(0-0xFFFFFF) */
    lb_uint8        en_fill; /* enable fill polygon area color,
                                  * 1. enable, 0. disable */
    lb_uint32       fill_color;/* fill color(0-0xFFFFFF) */
} lb_eui_canvas_polygon_t;

typedef struct tag_lb_eui_canvas_circle {
    lb_eui_point_t      coordinate;/* coordinate position */
    lb_int32        radius;    /* radius value */
    lb_uint32       border_color;/* border color */
    lb_uint8        en_fill;/* enable fill circle area color,
                             * 1. enable, 0. disable */
    lb_uint32       fill_color;/* fill color(0-0xFFFFFF) */
} lb_eui_canvas_circle_t;

typedef struct tag_lb_eui_canvas_image {
    /* LV_IMG_CF_TRUE_COLOR_ALPHA is not supported */
    lb_eui_point_t      coordinate;     /* start position */
    char            *image_src;    /* image source pointer */
    lb_uint16       src_len;
    lb_uint8        type;
    lb_int32        width;
    lb_int32        height;
    lb_uint8        multiply; /* */
} lb_eui_canvas_image_t;

typedef enum {
    LB_EUI_CANVAS_STYLE_MAIN,
```

} lb_eui_canvas_style_e;

lb_eui_canvas_set_px

功能	设置 canvas 某一点的颜色。
函数	lb_int32 lb_eui_canvas_set_px(lb_uint32 id, lb_int32 x, lb_int32 y, lb_uint32 color)
参数	id：checkbox 控件的 ID 号；x：点的 x 方向位置（相对于 canvas 的位置）；y：点的 y 方向位置（相对于 canvas 的位置）；color：ARGB 颜色值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_canvas_draw_background

功能	填充 canvas 背景颜色及颜色格式。
函数	lb_int32 lb_eui_canvas_draw_background(lb_uint32 id, lb_uint32 bg_color, lb_eui_img_cf_t color_format)
参数	id：checkbox 控件的 ID 号；bg_color：背景颜色值；color_format：颜色制式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_canvas_draw_image

功能	在 canvas 上填充图片，不支持透明色。
函数	lb_int32 lb_eui_canvas_draw_image(lb_uint32 id, lb_eui_canvas_image_t *image_data)
参数	id：checkbox 控件的 ID 号；image_data：图片数据信息。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_canvas_draw_circle

功能	在 canvas 上画圆。
函数	lb_int32 lb_eui_canvas_draw_circle(lb_uint32 id, lb_eui_canvas_circle_t *circle_data)
参数	id：checkbox 控件的 ID 号；circle_data：圆规格信息。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_canvas_draw_polygon

功能	在 canvas 上画多边形，包含线。
函数	lb_int32 lb_eui_canvas_draw_polygon(lb_uint32 id, lb_eui_canvas_polygon_t *polygon_data)
参数	id：checkbox 控件的 ID 号；polygon_data：多边形数据信息。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_canvas_set_style

功能	设置 canvas 的风格。
函数	lb_int32 lb_eui_canvas_set_style(lb_uint32 id, lb_eui_canvas_style_e type, lb_eui_style_t *lb_canvas_style)
参数	id：checkbox 控件的 ID 号；type：风格类型，参考 lb_eui_canvas_style_e；

	lb_canvas_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_canvas_get_px

功能	获取 canvas 某一点的颜色值。
函数	lb_int32 lb_eui_canvas_get_px(lb_uint32 id, lb_int32 x, lb_int32 y, lb_uint32 *color)
参数	id：checkbox 控件的 ID 号；x：x 方向的坐标；y：y 方向的坐标；color：颜色值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_canvas_get_style

功能	获取 canvas 的风格。
函数	lb_int32 lb_eui_canvas_get_style(lb_uint32 id, lb_eui_canvas_style_e type, lb_eui_style_t *lb_canvas_style)
参数	id：checkbox 控件的 ID 号；type：风格类型，参考 lb_eui_canvas_style_e； lb_canvas_style：需要的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.8. Checkbox 控件

```
typedef enum {
    LB_EUI_CB_STYLE_BG,
    LB_EUI_CB_STYLE_BOX_REL,
    LB_EUI_CB_STYLE_BOX_PR,
    LB_EUI_CB_STYLE_BOX_TGL_REL,
    LB_EUI_CB_STYLE_BOX_TGL_PR,
    LB_EUI_CB_STYLE_BOX_INA,
} lb_eui_cb_style_e;
```

Checkbox 的 API 接口函数说明如下：

lb_eui_cb_set_text

功能	设置 checkbox 的文本。
函数	lb_int32 lb_eui_cb_set_text(lb_int32 id, char *txt)
参数	id：checkbox 控件的 ID 号；txt：checkbox 的文本字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_set_checked

功能	设置 checkbox 被选中。
函数	lb_int32 lb_eui_cb_set_checked(lb_int32 id, bool checked)

参数	id：checkbox 控件的 ID 号；checked：true 表示选中，false 表示未被选中。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_set_inactive

功能	设置 checkbox 无效。
函数	lb_int32 lb_eui_cb_set_inactive(lb_int32 id)
参数	id：checkbox 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_set_style

功能	设置 checkbox 的风格。
函数	lb_int32 lb_eui_cb_set_style(lb_int32 id, lb_eui_cb_style_e type, lb_eui_style_t *lb_cb_style)
参数	id：checkbox 控件的 ID 号；type：风格类型，参考 lb_eui_cb_style_e； lb_cb_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_get_text

功能	获取 checkbox 的文本。
函数	lb_int32 lb_eui_cb_get_text(lb_int32 id, char *txt)
参数	id：checkbox 控件的 ID 号；txt：返回 checkbox 的文本字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_get_checked

功能	获取 checkbox 选中状态。
函数	lb_int32 lb_eui_cb_get_checked(lb_int32 id, bool *checked)
参数	id：checkbox 控件的 ID 号；checked：返回选中状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_get_style

功能	获取 checkbox 的风格。
函数	lb_int32 lb_eui_cb_get_style(lb_int32 id, lb_eui_cb_style_e type, lb_eui_style_t *lb_cb_style)
参数	id：checkbox 控件的 ID 号；type：风格类型，参考 lb_eui_cb_style_e； lb_cb_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_set_action

功能	设置 checkbox 的点击回调函数。
函数	lb_int32 lb_eui_cb_ext_set_action(void *checkbox, lb_eui_action_t action)

参数	checkbox：checkbox 控件句柄；action：事件回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_set_text

功能	设置 checkbox 的文本。
函数	lb_int32 lb_eui_cb_ext_set_text(void *checkbox, char *txt)
参数	checkbox：checkbox 控件句柄；txt：checkbox 的文本字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_set_checked

功能	设置 checkbox 被选中。
函数	lb_int32 lb_eui_cb_ext_set_checked(void *checkbox, bool checked)
参数	checkbox：checkbox 控件句柄；checked：true 表示选中，false 表示未被选中。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_set_inactive

功能	设置 checkbox 无效。
函数	lb_int32 lb_eui_cb_ext_set_inactive(void *checkbox)
参数	checkbox：checkbox 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_set_style

功能	设置 checkbox 的风格。
函数	lb_int32 lb_eui_cb_ext_set_style(void *checkbox, lb_eui_cb_style_e type, lb_eui_style_t *lb_cb_style)
参数	checkbox：checkbox 控件句柄；type：风格类型，参考 lb_eui_cb_style_e； lb_cb_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_get_text

功能	获取 checkbox 的文本。
函数	lb_int32 lb_eui_cb_ext_get_text(void *checkbox, char *txt)
参数	checkbox：checkbox 控件句柄；txt：返回 checkbox 的文本字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_get_checked

功能	获取 checkbox 选中状态。
函数	lb_int32 lb_eui_cb_ext_get_checked(void *checkbox, bool *checked)
参数	checkbox：checkbox 控件句柄；checked：返回选中状态。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_cb_ext_get_style

功能	获取 checkbox 的风格。
函数	lb_int32 lb_eui_cb_ext_get_style(void *checkbox, lb_eui_cb_style_e type, lb_eui_style_t *lb_cb_style)
参数	checkbox：checkbox 控件句柄；type：风格类型，参考 lb_eui_cb_style_e； lb_cb_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.9. Chart 控件

```
typedef enum
{
    LB_EUI_CHART_TYPE_LINE = 0x01, /* 线型图 */
    LB_EUI_CHART_TYPE_COLUMN = 0x02, /* 柱状图 */
    LB_EUI_CHART_TYPE_POINT = 0x04, /* 点图 */
    LB_EUI_CHART_TYPE_VERTICAL_LINE = 0x08, /* */
}lb_eui_chart_type_e;
```

Chart 控件的 API 接口定义如下：

lb_eui_chart_add_series

功能	设置 chart 的水平和垂直平分线数目。
函数	lb_int32 lb_eui_chart_add_series(lb_uint32 id, lb_uint32 color)
参数	id：chart 控件的 ID 号；color：系列颜色。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_div_line_count

功能	设置 chart 的水平和垂直平分线数目。
函数	lb_int32 lb_eui_chart_set_div_line_count(lb_int32 id, lb_uint8 hdiv, lb_uint8 vdiv)
参数	id：chart 控件的 ID 号；hdiv：水平方向平分线数目；vdiv：垂直方向平分线数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_range

功能	设置 chart 的垂直方向上的范围。
函数	lb_int32 lb_eui_chart_set_range(lb_int32 id, lb_int16 min, lb_int16 max)
参数	id：chart 控件的 ID 号；txt：checkbox 的文本字符串。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_type

功能	设置 chart 类型，上述可任意组合，范围 1-F。
函数	lb_int32 lb_eui_chart_set_type(lb_int32 id, lb_eui_chart_type_e type)
参数	id：chart 控件的 ID 号；type：chart 类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_point_count

功能	设置 chart 点的数目。
函数	lb_int32 lb_eui_chart_set_point_count(lb_int32 id, lb_uint16 point_num)
参数	id：chart 控件的 ID 号；point_num：点的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_series_opa

功能	设置 chart 的不透明度。
函数	lb_int32 lb_eui_chart_set_series_opa (lb_int32 id, lb_uint8 opa)
参数	id：chart 控件的 ID 号；opa：点的不透明度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_ series_width

功能	设置 chart 的宽度（点的半径）。
函数	lb_int32 lb_eui_chart_set_ series_width(lb_int32 id, lb_int32 width)
参数	id：chart 控件的 ID 号；width：宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_ series_darking

功能	设置 chart 的底部效果。
函数	lb_int32 lb_eui_chart_set_ series_darking(lb_int32 id, lb_uint8 dark_eff)
参数	id：chart 控件的 ID 号；dark_eff：底部的不透明效果。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_series_points

功能	设置 chart 的点序列。
函数	lb_int32 lb_eui_chart_set_series_points(lb_uint32 id, lb_uint8 index,lb_int32 *points, lb_uint16 point_nums)
参数	id：chart 控件的 ID 号；index：序列索引值；points：序列的点数组，为一组整数；point_nums：点个数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_set_style

功能	设置 chart 的风格。
函数	lb_int32 lb_eui_chart_set_style(lb_int32 id, lb_eui_style_t *lb_chart_style)
参数	id：chart 控件的 ID 号；lb_chart_style：需要设置的风格。

返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_chart_get_type

功能	获取 chart 类型。
函数	lb_int32 lb_eui_chart_get_type(lb_int32 id, lb_uint8 *type)
参数	id：chart 控件的 ID 号；type：返回 chart 类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_get_point_count

功能	获取 chart 点的数目。
函数	lb_int32 lb_eui_chart_get_point_count(lb_int32 id, lb_uint16 *point_num)
参数	id：chart 控件的 ID 号；point_num：返回点的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_get_series_opa

功能	获取 chart 的不透明度。
函数	lb_int32 lb_eui_chart_get_series_opa (lb_int32 id, lb_uint8 *opa)
参数	id：chart 控件的 ID 号；opa：返回点的不透明度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_get_ series_width

功能	获取 chart 的宽度（点的半径）。
函数	lb_int32 lb_eui_chart_get_ series_width(lb_int32 id, lb_int32 *width)
参数	id：chart 控件的 ID 号；width：返回宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_get_ series_darking

功能	获取 chart 的底部效果。
函数	lb_int32 lb_eui_chart_get_ series_darking(lb_int32 id, lb_uint8 *dark_eff)
参数	id：chart 控件的 ID 号；dark_eff：返回底部的不透明效果。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_get_style

功能	获取 chart 的风格。
函数	lb_int32 lb_eui_chart_get_style(lb_int32 id, lb_eui_style_t *lb_chart_style)
参数	id：chart 控件的 ID 号；lb_chart_style：需要获取的风格。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_add_series

功能	设置 chart 的水平和垂直平分线数目。
函数	lb_int32 lb_eui_chart_ext_add_series(void *chart, lb_uint32 color)
参数	chart：chart 控件句柄；color：系列颜色。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_div_line_count

功能	设置 chart 的水平和垂直平分线数目。
函数	lb_int32 lb_eui_chart_ext_set_div_line_count(void *chart, lb_uint8 hdiv, lb_uint8 vdiv)
参数	chart：chart 控件句柄；hdiv：水平方向平分线数目；vdiv：垂直方向平分线数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_range

功能	设置 chart 的垂直方向上的范围。
函数	lb_int32 lb_eui_chart_ext_set_range(void *chart, lb_int16 min, lb_int16 max)
参数	chart：chart 控件句柄；txt：checkbox 的文本字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_type

功能	设置 chart 类型，类型之间可任意组合，范围 1-F。
函数	lb_int32 lb_eui_chart_ext_set_type(chart：chart 控件句柄, lb_eui_chart_type_e type)
参数	chart：chart 控件句柄；type：chart 类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_point_count

功能	设置 chart 点的数目。
函数	lb_int32 lb_eui_chart_ext_set_point_count(void *chart, lb_uint16 point_num)
参数	chart：chart 控件句柄；point_num：点的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_series_opa

功能	设置 chart 的不透明度。
函数	lb_int32 lb_eui_chart_ext_set_series_opa (void *chart, lb_uint8 opa)
参数	chart：chart 控件句柄；opa：点的不透明度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_ series_width

功能	设置 chart 的宽度（点的半径）。
函数	lb_int32 lb_eui_chart_ext_set_series_width(void *chart, lb_int32 width)
参数	chart：chart 控件句柄；width：宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_ series_darking

功能	设置 chart 的底部效果。
函数	lb_int32 lb_eui_chart_ext_set_ series_darking(void *chart, lb_uint8 dark_eff)
参数	chart：chart 控件句柄；dark_eff：底部的不透明效果。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_series_points

功能	设置 chart 的点序列。
函数	lb_int32 lb_eui_chart_ext_set_series_points(void *chart, lb_uint8 index,lb_int32 *points, lb_uint16 point_nums)
参数	chart：chart 控件句柄；index：序列索引值；points：序列的点数组，为一组整数；point_nums：点个数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_set_style

功能	设置 chart 的风格。
函数	lb_int32 lb_eui_chart_ext_set_style(void *chart, lb_eui_style_t *lb_chart_style)
参数	chart：chart 控件句柄；lb_chart_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_chart_ext_get_type

功能	获取 chart 类型。
函数	lb_int32 lb_eui_chart_ext_get_type(void *chart, lb_uint8 *type)
参数	chart：chart 控件句柄；type：返回 chart 类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_get_point_count

功能	获取 chart 点的数目。
函数	lb_int32 lb_eui_chart_ext_get_point_count(void *chart, lb_uint16 *point_num)
参数	chart：chart 控件句柄；point_num：返回点的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_get_series_opa

功能	获取 chart 的不透明度。
函数	lb_int32 lb_eui_chart_ext_get_series_opa (void *chart, lb_uint8 *opa)
参数	chart：chart 控件句柄；opa：返回点的不透明度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_get_ series_width

功能	获取 chart 的宽度（点的半径）。
函数	lb_int32 lb_eui_chart_ext_get_series_width(void *chart, lb_int32 *width)
参数	chart：chart 控件句柄；width：返回宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_get_series_darking

功能	获取 chart 的底部效果。
函数	lb_int32 lb_eui_chart_ext_get_series_darking(void *chart, lb_uint8 *dark_eff)
参数	chart：chart 控件句柄；dark_eff：返回底部的不透明效果。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_chart_ext_get_style

功能	获取 chart 的风格。
函数	lb_int32 lb_eui_chart_ext_get_style(void *chart, lb_eui_style_t *lb_chart_style)
参数	chart：chart 控件句柄；lb_chart_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.10. Container 控件

Container 控件的 API 接口定义如下：

lb_eui_cont_set_layout

功能	设置 cont 布局方式。
函数	lb_int32 lb_eui_cont_set_layout(lb_int32 id, lb_eui_layout_e layout)
参数	id：cont 控件的 ID 号；layout：cont 布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_set_fit

功能	设置 cont 垂直方向，水平方向 fit 属性。
函数	lb_int32 lb_eui_cont_set_fit(lb_int32 id, bool hor_en, bool ver_en)
参数	id：cont 控件的 ID 号；hor_en：cont 水平方向 fit 属性；ver_en：cont 垂直方向 fit 属性;true 表示支持 fit，flase 表示不支持 fit。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_set_style

功能	设置 cont 的风格。
函数	lb_int32 lb_eui_cont_set_style(lb_int32 id, lb_eui_style_t *lb_cont_style)
参数	id：cont 控件的 ID 号；lb_cont_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_get_layout

功能	获取 cont 布局方式。
函数	lb_int32 lb_eui_cont_get_layout(lb_int32 id, lb_uint8 *layout)
参数	id：cont 控件的 ID 号；layout:返回 cont 布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_get_fit

功能	获取 cont 水平与垂直方向 fit 属性状态。
函数	lb_int32 lb_eui_cont_get_fit(lb_uint32 id, bool *hor_en, bool *ver_en)
参数	id：cont 控件的 ID 号；hor_en:返回 cont 水平方向 fit 属性状态，true 表示支持，flase 表示不支持；ver_en:返回 cont 垂直方向 fit 属性状态，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_get_fit_size

功能	获取 cont 有效宽度和高度，cont 控件宽度减去水平填充宽度，cont 控件高度减去垂直填充高度。
函数	lb_int32 lb_eui_cont_get_fit_size(lb_uint32 id, lb_int32 *width, lb_int32 *height)
参数	id：cont 控件的 ID 号；width:返回 cont 有效宽度；height:返回 cont 有效高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_get_style

功能	获取 cont 的风格。
函数	lb_int32 lb_eui_cont_get_style(lb_int32 id, lb_eui_style_t *lb_cont_style)
参数	id：cont 控件的 ID 号；lb_cont_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_set_action

功能	设置 cont 布局方式。
函数	lb_int32 lb_eui_cont_ext_set_action(void *cont, lb_eui_action_t action)
参数	cont：cont 控件句柄；action：事件回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_set_layout

功能	设置 cont 布局方式。
函数	lb_int32 lb_eui_cont_ext_set_layout(void *cont, lb_eui_layout_e layout)
参数	cont：cont 控件句柄；layout：cont 布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_set_fit

功能	设置 cont 垂直方向，水平方向 fit 属性。
函数	lb_int32 lb_eui_cont_ext_set_fit(void *cont, bool hor_en, bool ver_en)
参数	cont：cont 控件句柄；hor_en：cont 水平方向 fit 属性；ver_en：cont 垂直方向 fit 属性;true 表示支持 fit，

	flase 表示不支持 fit。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_set_style

功能	设置 cont 的风格。
函数	lb_int32 lb_eui_cont_ext_set_style(void *cont, lb_eui_style_t *lb_cont_style)
参数	cont：cont 控件句柄；lb_cont_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_get_layout

功能	获取 cont 布局方式。
函数	lb_int32 lb_eui_cont_ext_get_layout(void *cont, lb_uint8 *layout)
参数	cont：cont 控件句柄；layout:返回 cont 布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_get_fit

功能	获取 cont 水平与垂直方向 fit 属性状态。
函数	lb_int32 lb_eui_cont_ext_get_fit(void *cont, bool *hor_en, bool *ver_en)
参数	cont：cont 控件句柄；hor_en:返回 cont 水平方向 fit 属性状态，true 表示支持，flase 表示不支持；ver_en:返回 cont 垂直方向 fit 属性状态，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_get_fit_size

功能	获取 cont 有效宽度和高度，cont 控件宽度减去水平填充宽度，cont 控件高度减去垂直填充高度。
函数	lb_int32 lb_eui_cont_ext_get_fit_size(void *cont, lb_int32 *width, lb_int32 *height)
参数	cont：cont 控件句柄；width:返回 cont 有效宽度；height:返回 cont 有效高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_cont_ext_get_style

功能	获取 cont 的风格。
函数	lb_int32 lb_eui_cont_ext_get_style(void *cont, lb_eui_style_t *lb_cont_style)
参数	cont：cont 控件句柄；lb_cont_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.11. Drop-down list 控件

```
typedef enum {
```

```
    LB_EUI_LABEL_ALIGN_LEFT,
    LB_EUI_LABEL_ALIGN_CENTER,
    LB_EUI_LABEL_ALIGN_RIGHT,
} lb_eui_label_align_e;

typedef enum {
    LB_EUI_DDLIST_STYLE_BG,
    LB_EUI_DDLIST_STYLE_SCRL,
    LB_EUI_DDLIST_STYLE_SEL,
    LB_EUI_DDLIST_STYLE_SB,
} lb_eui_ddlist_style_e;
```

ddlist 控件的 API 接口定义如下：

lb_eui_ddlist_set_draw_arrow

功能	设置 ddlist 显示箭头属性。
函数	lb_int32 lb_eui_ddlist_set_draw_arrow(lb_int32 id, bool en)
参数	id：ddlist 控件的 ID 号；en：ddlit 箭头显示，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_options

功能	设置 ddlist 下拉选项字符串数组。
函数	lb_int32 lb_eui_ddlist_set_options(lb_int32 id, const char *options)
参数	id：ddlist 控件的 ID 号；options：ddlit 下拉选项字符串数组（e.g:"One\nTwo\nThree"）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_selected

功能	设置 ddlist 下拉选中项。
函数	lb_int32 lb_eui_ddlist_set_selected(lb_int32 id, lb_uint16 sel_opt)
参数	id：ddlist 控件的 ID 号；sel_opt：ddlit 下拉选中项索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_fix_height

功能	设置 ddlist 打开下拉时显示的高度。
函数	lb_int32 lb_eui_ddlist_set_fix_height(lb_int32 id, lb_int32 h)
参数	id：ddlist 控件的 ID 号；h：ddlit 打开下拉时显示的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_hor_fit

功能	设置 ddlist content 水平方向 fit 属性。
函数	lb_int32 lb_eui_ddlist_set_hor_fit(lb_int32 id, bool en)

参数	id：ddlist 控件的 ID 号；en：ddlit content 水平方向 fit 属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_sb_mode

功能	设置 ddlist 滚动条模式。
函数	lb_int32 lb_eui_ddlist_set_sb_mode(lb_int32 id, sb_mode_e mode)
参数	id：ddlist 控件的 ID 号；mode：ddlit scroll bar 模式,模式定义参考 sb_mode_e。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_anim_time

功能	设置 ddlist 打开或者关闭下拉时的动画时间。
函数	lb_int32 lb_eui_ddlist_set_anim_time(lb_int32 id, lb_int16 anim_time)
参数	id：ddlist 控件的 ID 号；anim_time：ddlit 打开关闭下拉时的动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_align

功能	设置 ddlist 字符的对齐方式。
函数	lb_int32 lb_eui_ddlist_set_align(lb_int32 id, lb_eui_label_align_e align)
参数	id：ddlist 控件的 ID 号；align：ddlit 字符对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_set_style

功能	设置 ddlist 的风格。
函数	lb_int32 lb_eui_ddlist_set_style(lb_int32 id, lb_eui_ddlist_style_e type, lb_eui_style_t *lb_ddlist_style)
参数	id：ddlist 控件的 ID 号；type：风格类型，参考 lb_eui_ddlist_style_e； lb_ddlist_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_open

功能	设置 ddlist 动画属性，调用此函数，下拉默认打开。
函数	lb_int32 lb_eui_ddlist_open(lb_int32 id, bool anim_en)
参数	id：ddlist 控件的 ID 号；anim：ddlit 动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_close

功能	设置 ddlist 动画属性，调用此函数，下拉默认关闭。
函数	lb_int32 lb_eui_ddlist_close(lb_int32 id, bool anim_en)
参数	id：ddlist 控件的 ID 号；anim：ddlit 动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_draw_arrow

功能	获取 ddlist 箭头显示属性状态。
函数	lb_int32 lb_eui_ddlist_get_draw_arrow(lb_int32 id, bool *en)

参数	id：ddlist 控件的 ID 号；en：返回 ddlit 显示属性状态，true 表示支持箭头显示，flase 表示不显示。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_options

功能	获取 ddlist 下拉选项字符串数组。
函数	lb_int32 lb_eui_ddlist_get_options(lb_int32 id, char **options)
参数	id：ddlist 控件的 ID 号；options：返回 ddlit 下拉选项字符串数组指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_selected

功能	获取 ddlist 下拉当前选中项。
函数	lb_int32 lb_eui_ddlist_get_selected(lb_int32 id, lb_uint16 *sel_opt)
参数	id：ddlist 控件的 ID 号；sel_opt：返回 ddlit 下拉当前选中项。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_selected_str

功能	获取 ddlist 下拉当前选中项字符。
函数	lb_int32 lb_eui_ddlist_get_selected_str(lb_int32 id, char **str)
参数	id：ddlist 控件的 ID 号；str：ddlit 返回下当前拉选中项字符。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_fix_height

功能	获取 ddlist 打开下拉时显示的高度。
函数	lb_int32 lb_eui_ddlist_get_fix_height(lb_int32 id, lb_int32 *h)
参数	id：ddlist 控件的 ID 号；h：返回 ddlit 打开下拉时显示的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_sb_mode

功能	获取 ddlist scroll bar 模式。
函数	lb_int32 lb_eui_ddlist_get_sb_mode(lb_int32 id, lb_uint8 *mode)
参数	id：ddlist 控件的 ID 号；mode：返回 ddlit scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_anim_time

功能	获取 ddlist 打开或者关闭下拉时动画时间。
函数	lb_int32 lb_eui_ddlist_get_anim_time(lb_int32 id, lb_int16 *anim_time)
参数	id：ddlist 控件的 ID 号；anim_time：返回 ddlit 打开关闭下拉时动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_align

功能	获取 ddlist 字符的对齐方式。
函数	lb_int32 lb_eui_ddlist_get_align(lb_int32 id, lb_uint8 *align)
参数	id：ddlist 控件的 ID 号；align：返回 ddlit 字符对齐方式（对齐方式定义参考

	ddlist_label_align_e）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_get_style

功能	获取 ddlist 的风格。
函数	lb_int32 lb_eui_ddlist_get_style(lb_int32 id, lb_eui_ddlist_style_e type, lb_eui_style_t *lb_ddlist_style)
参数	id：ddlist 控件的 ID 号；type：风格类型，参考 lb_eui_cb_style_e； lb_ddlist_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_action

功能	设置 ddlist 显示选中回调函数。
函数	lb_int32 lb_eui_ddlist_ext_set_action(void *ddlist, lb_eui_action_t action)
参数	ddlist：ddlist 控件句柄；action：选中回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_draw_arrow

功能	设置 ddlist 显示箭头属性。
函数	lb_int32 lb_eui_ddlist_ext_set_draw_arrow(void *ddlist, bool en)
参数	ddlist：ddlist 控件句柄；en：ddlit 箭头显示，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_options

功能	设置 ddlist 下拉选项字符串数组。
函数	lb_int32 lb_eui_ddlist_ext_set_options(void *ddlist, const char *options)
参数	ddlist：ddlist 控件句柄；options：ddlit 下拉选项字符串数组（e.g:"One\nTwo\nThree"）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_selected

功能	设置 ddlist 下拉选中项。
函数	lb_int32 lb_eui_ddlist_ext_set_selected(void *ddlist, lb_uint16 sel_opt)
参数	ddlist：ddlist 控件句柄；sel_opt：ddlit 下拉选中项索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_fix_height

功能	设置 ddlist 打开下拉时显示的高度。
函数	lb_int32 lb_eui_ddlist_ext_set_fix_height(void *ddlist, lb_int32 h)

参数	ddlist：ddlist 控件句柄；h：ddlit 打开下拉时显示的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_hor_fit

功能	设置 ddlist content 水平方向 fit 属性。
函数	lb_int32 lb_eui_ddlist_ext_set_hor_fit(void *ddlist, bool en)
参数	ddlist：ddlist 控件句柄；en：ddlit content 水平方向 fit 属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_sb_mode

功能	设置 ddlist 滚动条模式。
函数	lb_int32 lb_eui_ddlist_ext_set_sb_mode(void *ddlist, sb_mode_e mode)
参数	ddlist：ddlist 控件句柄；mode：ddlit scroll bar 模式,模式定义参考 sb_mode_e。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_anim_time

功能	设置 ddlist 打开或者关闭下拉时的动画时间。
函数	lb_int32 lb_eui_ddlist_ext_set_anim_time(void *ddlist, lb_int16 anim_time)
参数	ddlist：ddlist 控件句柄；anim_time：ddlit 打开关闭下拉时的动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_align

功能	设置 ddlist 字符的对齐方式。
函数	lb_int32 lb_eui_ddlist_ext_set_align(void *ddlist, lb_eui_label_align_e align)
参数	ddlist：ddlist 控件句柄；align：ddlit 字符对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_set_style

功能	设置 ddlist 的风格。
函数	lb_int32 lb_eui_ddlist_ext_set_style(void *ddlist, lb_eui_ddlist_style_e type, lb_eui_style_t *lb_ddlist_style)
参数	ddlist：ddlist 控件句柄；type：风格类型，参考 lb_eui_ddlist_style_e； lb_ddlist_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_open

功能	设置 ddlist 动画属性，调用此函数，下拉默认打开。
函数	lb_int32 lb_eui_ddlist_ext_open(void *ddlist, bool anim_en)
参数	ddlist：ddlist 控件句柄；anim_en：ddlit 动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_close

功能	设置 ddlist 动画属性，调用此函数，下拉默认关闭。
函数	lb_int32 lb_eui_ddlist_ext_close(void *ddlist, bool anim_en)

参数	ddlist：ddlist 控件句柄；anim_en：ddlit 动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_draw_arrow

功能	获取 ddlist 箭头显示属性状态。
函数	lb_int32 lb_eui_ddlist_ext_get_draw_arrow(void *ddlist, bool *en)
参数	ddlist：ddlist 控件句柄；en：返回 ddlit 显示属性状态，true 表示支持箭头显示，flase 表示不显示。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_options

功能	获取 ddlist 下拉选项字符串数组。
函数	lb_int32 lb_eui_ddlist_ext_get_options(lvoid *ddlist, char **options)
参数	ddlist：ddlist 控件句柄；options：返回 ddlit 下拉选项字符串数组指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_selected

功能	获取 ddlist 下拉当前选中项。
函数	lb_int32 lb_eui_ddlist_ext_get_selected(void *ddlist, lb_uint16 *sel_opt)
参数	ddlist：ddlist 控件句柄；sel_opt：返回 ddlit 下拉当前选中项。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_selected_str

功能	获取 ddlist 下拉当前选中项字符。
函数	lb_int32 lb_eui_ddlist_ext_get_selected_str(void *ddlist, char **str)
参数	ddlist：ddlist 控件句柄；str：ddlit 返回下当前拉选中项字符。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_fix_height

功能	获取 ddlist 打开下拉时显示的高度。
函数	lb_int32 lb_eui_ddlist_ext_get_fix_height(void *ddlist, lb_int32 *h)
参数	ddlist：ddlist 控件句柄；h：返回 ddlit 打开下拉时显示的高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_sb_mode

功能	获取 ddlist scroll bar 模式。
函数	lb_int32 lb_eui_ddlist_ext_get_sb_mode(void *ddlist, lb_uint8 *mode)
参数	ddlist：ddlist 控件句柄；mode：返回 ddlit scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_anim_time

功能	获取 ddlist 打开或者关闭下拉时动画时间。
函数	lb_int32 lb_eui_ddlist_ext_get_anim_time(void *ddlist, lb_int16 *anim_time)
参数	ddlist：ddlist 控件句柄；anim_time：返回 ddlit 打开关闭下拉时动画时间，单位 ms。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_align

功能	获取 ddlist 字符的对齐方式。
函数	lb_int32 lb_eui_ddlist_ext_get_align(void *ddlist, lb_uint8 *align)
参数	ddlist：ddlist 控件句柄；align：返回 ddlit 字符对齐方式（对齐方式定义参考 ddlist_label_align_e）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ddlist_ext_get_style

功能	获取 ddlist 的风格。
函数	lb_int32 lb_eui_ddlist_ext_get_style(void *ddlist, lb_eui_ddlist_style_e type, lb_eui_style_t *lb_ddlist_style)
参数	ddlist：ddlist 控件句柄；type：风格类型，参考 lb_eui_cb_style_e； lb_ddlist_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.12. Gauge 控件

```
typedef struct tag_lb_eui_gauge_needle {
    lb_int32        width;
    lb_int32        length;
    lb_uint32       color;
} lb_eui_gauge_needle_t;
```

gauge 控件的 API 接口定义如下：

lb_eui_gauge_set_needle_count

功能	设置 gauge 指针的数目。
函数	lb_int32 lb_eui_gauge_set_needle_count(lb_int32 id, lb_uint8 needle_cnt, lb_eui_gauge_needle_t *needles)
参数	id：gauge 控件的 ID 号；needle_cnt：gauge 指针数目，needles:指针信息（宽度、长度和颜色）数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_set_value

功能	设置 gauge 指针的数值。
函数	lb_int32 lb_eui_gauge_set_value(lb_int32 id, lb_uint8 needle_id, lb_int16 value)
参数	id：gauge 控件的 ID 号；needle_id：gauge 指针 id，value:指针数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_set_range

功能	设置 gauge 数值的范围。
函数	lb_int32 lb_eui_gauge_set_range(lb_int32 id, lb_int16 min, lb_int16 max)
参数	id：gauge 控件的 ID 号；min：gauge 最小范围数值，max:最大范围数值。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_set_critical_value

功能	设置 gauge 关键数值，此关键数值之后的刻度线将绘制不同颜色。
函数	lb_int32 lb_eui_gauge_set_critical_value(lb_int32 id, lb_int16 value)
参数	id：gauge 控件的 ID 号；value：gauge 关键值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_set_scale

功能	设置 gauge 刻度设置。
函数	lb_int32 lb_eui_gauge_set_scale(lb_int32 id, lb_int16 angle, lb_uint8 line_cnt, lb_uint8 label_cnt, lb_int16 *label_val)
参数	id：gauge 控件的 ID 号；angle：gauge 刻度的角度（0-360），line_cnt:刻度数目，label_cnt：标签数目，label_val：标签显示的数字数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_set_style

功能	设置 gauge 的指针。
函数	lb_int32 lb_eui_gauge_set_style(lb_int32 id, lb_eui_style_t *lb_gauge_style)
参数	id：gauge 控件的 ID 号；lb_gauge_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_get_needle_count

功能	获取 gauge 的指针的数目。
函数	lb_int32 lb_eui_gauge_get_needle_count(lb_uint32 id, lb_uint8 *needle_cnt)
参数	id：gauge 控件的 ID 号；needle_cnt：gauge 指针数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_get_value

功能	获取 gauge 某个指针的数值。
函数	lb_int32 lb_eui_gauge_get_value(lb_int32 id, lb_uint8 needle_id, lb_int16 *value)
参数	id：gauge 控件的 ID 号；needle_id：gauge 指针 id，value:返回指针数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_get_min_value

功能	获取 gauge 最小数值。
函数	lb_int32 lb_eui_gauge_get_min_value(lb_int32 id, lb_int16 *min)
参数	id：gauge 控件的 ID 号；min：返回 gauge 最小数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_get_max_value

功能	获取 gauge 最大数值。

函数	lb_int32 lb_eui_gauge_get_max_value(lb_int32 id, lb_int16 *max)
参数	id：gauge 控件的 ID 号；max：返回 gauge 最大数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_get_critical_value

功能	获取 gauge 关键数值。
函数	lb_int32 lb_eui_gauge_get_critical_value(lb_int32 id, lb_int16 *value)
参数	id：gauge 控件的 ID 号；value：返回 gauge 关键值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_get_style

功能	获取 gauge 的风格。
函数	lb_int32 lb_eui_gauge_get_style(lb_int32 id, lb_eui_style_t *lb_gauge_style)
参数	id：gauge 控件的 ID 号；lb_gauge_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_set_needle_count

功能	设置 gauge 指针的数目。
函数	lb_int32 lb_eui_gauge_ext_set_needle_count(void *gauge, lb_uint8 needle_cnt, lb_eui_gauge_needle_t *needles)
参数	gauge：gauge 控件句柄；needle_cnt：gauge 指针数目，needles:指针信息（宽度、长度和颜色）数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_set_value

功能	设置 gauge 指针的数值。
函数	lb_int32 lb_eui_gauge_ext_set_value(void *gauge, lb_uint8 needle_id, lb_int16 value)
参数	gauge：gauge 控件句柄；needle_id：gauge 指针 id，value:指针数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_set_range

功能	设置 gauge 数值的范围。
函数	lb_int32 lb_eui_gauge_ext_set_range(void *gauge, lb_int16 min, lb_int16 max)
参数	gauge：gauge 控件句柄；min：gauge 最小范围数值，max:最大范围数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_set_critical_value

功能	设置 gauge 关键数值，此关键数值之后的刻度线将绘制不同颜色。
函数	lb_int32 lb_eui_gauge_ext_set_critical_value(void *gauge, lb_int16 value)

参数	gauge：gauge 控件句柄；value：gauge 关键值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_set_scale

功能	设置 gauge 刻度设置。
函数	lb_int32 lb_eui_gauge_ext_set_scale(void *gauge, lb_int16 angle, lb_uint8 line_cnt, lb_uint8 label_cnt, lb_int16 *label_val)
参数	gauge：gauge 控件句柄；angle：gauge 刻度的角度（0-360），line_cnt:刻度数目，label_cnt：标签数目，label_val：标签显示的数字数组。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_set_style

功能	设置 gauge 的指针。
函数	lb_int32 lb_eui_gauge_ext_set_style(void *gauge, lb_eui_style_t *lb_gauge_style)
参数	gauge：gauge 控件句柄；lb_gauge_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_get_needle_count

功能	获取 gauge 的指针的数目。
函数	lb_int32 lb_eui_gauge_ext_get_needle_count(void *gauge, lb_uint8 *needle_cnt)
参数	gauge：gauge 控件句柄；needle_cnt：gauge 指针数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_get_value

功能	获取 gauge 某个指针的数值。
函数	lb_int32 lb_eui_gauge_ext_get_value(void *gauge, lb_uint8 needle_id, lb_int16 *value)
参数	gauge：gauge 控件句柄；needle_id：gauge 指针 id，value:返回指针数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_get_min_value

功能	获取 gauge 最小数值。
函数	lb_int32 lb_eui_gauge_ext_get_min_value(void *gauge, lb_int16 *min)
参数	gauge：gauge 控件句柄；min：返回 gauge 最小数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_get_max_value

功能	获取 gauge 最大数值。

函数	lb_int32 lb_eui_gauge_ext_get_max_value(void *gauge, lb_int16 *max)
参数	gauge：gauge 控件句柄；max：返回 gauge 最大数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_get_critical_value

功能	获取 gauge 关键数值。
函数	lb_int32 lb_eui_gauge_ext_get_critical_value(void *gauge, lb_int16 *value)
参数	gauge：gauge 控件句柄；value：返回 gauge 关键值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_gauge_ext_get_style

功能	获取 gauge 的风格。
函数	lb_int32 lb_eui_gauge_ext_get_style(void *gauge, lb_eui_style_t *lb_gauge_style)
参数	gauge：gauge 控件句柄；lb_gauge_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.13. Image 控件

image 控件的 API 接口定义如下：

Setter functions

lb_eui_img_set_src

功能	设置 image 资源数据。
函数	lb_int32 lb_eui_img_set_src(lb_uint32 id, void *src_img, lb_uint32 data_len)
参数	id：image 控件的 ID 号；src_img：资源数据，带头信息（4 bytes）的 ARGB8888 格式数据或者 lb_eui_img_dsc_t 数据。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_delete_src

功能	删除 image 资源数据。
函数	lb_int32 lb_eui_img_delete_src(lb_uint32 id)
参数	id：image 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_set_src_path

功能	设置 image 路径。
函数	lb_int32 lb_eui_img_set_src_path(lb_uint32 id, char *src_path)
参数	id：image 控件的 ID 号；src_path：图片路径。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_set_src_index

功能	设置 image 资源数据索引。
函数	lb_int32 lb_eui_img_set_src_index(lb_int32 id, lb_uint8 index)
参数	id：image 控件的 ID 号；index：资源数据索引，数据资源路径在 json 里描述。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_set_auto_size

功能	设置 image 控件大小是否支持自适应图片大小。
函数	lb_int32 lb_eui_img_set_auto_size(lb_int32 id, bool autosize_en)
参数	id：image 控件的 ID 号；autosize_en：true 支持适应图片大小，false 不支持适应图片大小。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_set_style

功能	设置 image 控件的风格。
函数	lb_int32 lb_eui_img_set_style(lb_int32 id, lb_eui_style_t *lb_img_style)
参数	id：image 控件的 ID 号；lb_img_style：要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_set_gif_speed

功能	设置 gif 图片播放速度。
函数	lb_int32 lb_eui_img_set_gif_speed(lb_uint32 id, lb_int32 gif_speed)
参数	id：image 控件的 ID 号；gif_speed：gif 图片播放速度，默认 100 是本身播放速度，小于 100 时播放越慢，反之，播放越快。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_img_get_auto_size

功能	获取 image 控件大小是否支持自适应图片大小。
函数	lb_int32 lb_eui_img_get_auto_size(lb_int32 id, bool *autosize_en)
参数	id：image 控件的 ID 号；autosize_en：返回控件大小是否支持自适应图片大小，true 支持，false 不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_get_style

功能	获取 image 控件的风格。
函数	lb_int32 lb_eui_img_get_style(lb_int32 id, lb_eui_style_t *lb_img_style)
参数	id：image 控件的 ID 号；lb_img_style：返回的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_ext_set_src

功能	设置 image 资源数据。
函数	lb_int32 lb_eui_img_ext_set_src(void *img, void *src_img, lb_uint32 data_len)
参数	img：image 控件句柄；src_img：资源数据，带头信息（4 bytes）的 ARGB8888 格式数据或者 lb_eui_img_dsc_t 数据。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_ext_delete_src

功能	删除 image 资源数据。
函数	lb_int32 lb_eui_img_ext_delete_src(void *img)
参数	img：image 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_ext_set_src_path

功能	设置 image 路径。
函数	lb_int32 lb_eui_img_ext_set_src_path(void *img, char *src_path)
参数	img：image 控件句柄；src_path：图片路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_ext_set_auto_size

功能	设置 image 控件大小是否支持自适应图片大小。
函数	lb_int32 lb_eui_img_ext_set_auto_size(void *img, bool autosize_en)
参数	img：image 控件句柄；autosize_en：true 支持适应图片大小，false 不支持适应图片大小。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_ext_set_style

功能	设置 image 控件的风格。
函数	lb_int32 lb_eui_img_ext_set_style(void *img, lb_eui_style_t *lb_img_style)
参数	img：image 控件句柄；lb_img_style：要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_ext_set_gif_speed

功能	设置 gif 图片播放速度。
函数	lb_int32 lb_eui_img_ext_set_gif_speed(void *img, lb_int32 gif_speed)
参数	img：image 控件句柄；gif_speed：gif 图片播放速度，默认 100 是本身播放速度，小于 100 时播放越慢，反之，播放越快。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_img_ext_get_auto_size

功能	获取 image 控件大小是否支持自适应图片大小。
函数	lb_int32 lb_eui_img_ext_get_auto_size(void *img, bool *autosize_en)
参数	img：image 控件句柄；autosize_en：返回控件大小是否支持自适应图片大小，true 支持，false 不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_img_ext_get_style

功能	获取 image 控件的风格。
函数	lb_int32 lb_eui_img_ext_get_style(void *img, lb_eui_style_t *lb_img_style)
参数	img：image 控件句柄；lb_img_style：返回的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.14. Image button 控件

Image button 控件的 API 接口定义见 lb_eui_btn_state_e：

```
enum {
    LB_EUI_IMGBTN_STATE_REL,
    LB_EUI_IMGBTN_STATE_PR,
    LB_EUI_IMGBTN_STATE_TGL_REL,
    LB_EUI_IMGBTN_STATE_TGL_PR,
    LB_EUI_IMGBTN_STATE_INA,
    LB_EUI_IMGBTN_STATE_MAX
};
typedef lb_uint8 lb_eui_imgbtn_state_t;

typedef enum {
    LB_EUI_IMGBTN_STYLE_REL,
    LB_EUI_IMGBTN_STYLE_PR,
    LB_EUI_IMGBTN_STYLE_TGL_REL,
    LB_EUI_IMGBTN_STYLE_TGL_PR,
    LB_EUI_IMGBTN_STYLE_INA,
} lb_eui_imgbtn_style_e;
```

Setter functions

lb_eui_imgbtn_set_src

功能	设置 imgbtn 指定状态下的图片数据。

函数	lb_int32 lb_eui_imgbtn_set_src(lb_uint32 id, lb_eui_imgbtn_state_t state,lb_uint8 type, void *src, lb_uint32 data_len)
参数	id：imgbtn 控件的 ID 号；state：imgbtn 状态；type：图片资源类型（此参数暂没有用到）；src：资源数据，带头信息（4bytes）的 ARGB8888 格式数据或者 lb_eui_img_dsc_t 数据；data_len：数据长度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_delete_src

功能	删除 imgbtn 指定状态下的图片数据。
函数	lb_int32 lb_eui_imgbtn_delete_src(lb_uint32 id, lb_eui_imgbtn_state_t state)
参数	id：imgbtn 控件的 ID 号；state：imgbtn 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_src_index

功能	设置 imgbtn 指定状态下的图片数据索引值。
函数	lb_int32 lb_eui_imgbtn_set_src_index(lb_uint32 id, lb_eui_imgbtn_state_t state, lb_uint8 index)
参数	id：imgbtn 控件的 ID 号；state：imgbtn 状态；index：资源数据索引，数据资源路径在 json 里描述。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_src_path

功能	设置 imgbtn 指定状态下的图片路径。
函数	lb_int32 lb_eui_imgbtn_set_src_path(lb_uint32 id, lb_eui_imgbtn_state_t state, char *src_path)
参数	id：imgbtn 控件的 ID 号；state：imgbtn 状态；src_path：图片路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_img_alignmode

功能	设置 imgbtn 图片在控件内的位置。
函数	lb_int32 lb_eui_imgbtn_set_img_alignmode(lb_uint32 id,lb_uint8 alignmode, lb_int32 x_offset, lb_int32 y_offset)
参数	id：imgbtn 控件的 ID 号；alignmode：对齐方式，参考 lb_eui_align_e；x_offset：x 方向的偏移；y_offset：y 方向的偏移。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_auto_size

功能	设置 imgbtn 自适应图片大小。
函数	lb_int32 lb_eui_imgbtn_set_auto_size(lb_uint32 id, bool auto_size)
参数	id：imgbtn 控件的 ID 号；auto_size：true 支持适应图片大小，false 不支持适应图片大小。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_text_longmode

功能	设置 imgbtn 的字符串长模式。

函数	lb_int32 lb_eui_imgbtn_set_text_longmode(lb_uint32 id, lb_uint8 longmode)
参数	id：imgbtn 控件的 ID 号；longmode：长字符串的处理模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_text_align

功能	设置 imgbtn 字符串本身对齐方式状态。
函数	lb_int32 lb_eui_imgbtn_set_text_align(lb_uint32 id, lb_uint8 align)
参数	id：imgbtn 控件的 ID 号；align：文本对齐方式，LABEL_ALIGN_LEFT 左对齐，LABEL_ALIGN_CENTER 居中，LABEL_ALIGN_RIGHT 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_toggle

功能	设置 imgbtn toggle 属性。
函数	lb_int32 lb_eui_imgbtn_set_toggle(lb_int32 id, bool tgl)
参数	id：imgbtn 控件的 ID 号；tgl：toggle 属性，true 表示支持 toggle，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_text

功能	设置 imgbtn 的字符串。
函数	lb_int32 lb_eui_imgbtn_set_text(lb_uint32 id, char *text)
参数	id：imgbtn 控件的 ID 号；text：字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_text_id

功能	设置 imgbtn 状态。
函数	lb_int32 lb_eui_imgbtn_set_text_id(lb_uint32 id, lb_uint32 str_id)
参数	id：imgbtn 控件的 ID 号；str_id：字符串在资源文件中的索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_state

功能	设置 imgbtn 状态。
函数	lb_int32 lb_eui_imgbtn_set_state(lb_int32 id, lb_eui_btn_state_e state)
参数	id：imgbtn 控件的 ID 号；state：新设置的状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_set_style

功能	设置 imgbtn 风格。
函数	lb_int32 lb_eui_imgbtn_set_style(lb_int32 id, lb_eui_imgbtn_style_e type,

	lb_eui_style_t *lb_imgbtn_style)
参数	id：imgbtn 控件的 ID 号；type：风格类型，参考 lb_eui_imgbtn_style_e；lb_imgbtn_style：新设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_imgbtn_get_toggle

功能	获取 imgbtn 是否支持 toggle。
函数	lb_int32 lb_eui_imgbtn_get_toggle(lb_int32 id, bool *tgl)
参数	id：imgbtn 控件的 ID 号；tgl：返回 toggle 状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_get_state

功能	获取 imgbtn 当前状态。
函数	lb_int32 lb_eui_imgbtn_get_state(lb_int32 id, lb_uint8 *state)
参数	id：imgbtn 控件的 ID 号；state：返回 imgbtn 当前状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_get_style

功能	获取 imgbtn 风格。
函数	lb_int32 lb_eui_imgbtn_get_style(lb_int32 id, lb_eui_imgbtn_style_e type, lb_eui_style_t *lb_imgbtn_style)
参数	id：imgbtn 控件的 ID 号；type ：风格类型，参考 lb_eui_btn_style_e；lb_imgbtn_style：获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_src

功能	设置 imgbtn 指定状态下的图片数据。
函数	lb_int32 lb_eui_imgbtn_ext_set_src(void *imgbtn, lb_eui_imgbtn_state_t state,lb_uint8 type, void *src, lb_uint32 data_len)
参数	imgbtn：imgbtn 控件句柄；state：imgbtn 状态；type：图片资源类型（此参数暂没有用到）；src：资源数据，带头信息（4bytes）的 ARGB8888 格式数据或者 lb_eui_img_dsc_t 数据；data_len：数据长度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_delete_src

功能	删除 imgbtn 指定状态下的图片数据。
函数	lb_int32 lb_eui_imgbtn_ext_delete_src(void *imgbtn, lb_eui_imgbtn_state_t state)
参数	imgbtn：imgbtn 控件句柄；state：imgbtn 状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_src_path

功能	设置 imgbtn 指定状态下的图片路径。
函数	lb_int32 lb_eui_imgbtn_ext_set_src_path(void *imgbtn, lb_eui_imgbtn_state_t state, char *src_path)
参数	imgbtn：imgbtn 控件句柄；state：imgbtn 状态；src_path：图片路径。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_img_alignmode

功能	设置 imgbtn 图片在控件内的位置。
函数	lb_int32 lb_eui_imgbtn_ext_set_img_alignmode(void *imgbtn,lb_uint8 alignmode, lb_int32 x_offset, lb_int32 y_offset)
参数	imgbtn：imgbtn 控件句柄；alignmode：对齐方式，参考 lb_eui_align_e；x_offset：x 方向的偏移；y_offset：y 方向的偏移。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_auto_size

功能	设置 imgbtn 自适应图片大小。
函数	lb_int32 lb_eui_imgbtn_ext_set_auto_size(void *imgbtn, bool auto_size)
参数	imgbtn：imgbtn 控件句柄；auto_size：true 支持适应图片大小，false 不支持适应图片大小。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_text_longmode

功能	设置 imgbtn 的字符串长模式。
函数	lb_int32 lb_eui_imgbtn_ext_set_text_longmode(void *imgbtn, lb_uint8 longmode)
参数	imgbtn：imgbtn 控件句柄；longmode：长字符串的处理模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_text_align

功能	设置 imgbtn 字符串本身对齐方式状态。
函数	lb_int32 lb_eui_imgbtn_ext_set_text_align(void *imgbtn, lb_uint8 align)
参数	imgbtn：imgbtn 控件句柄；align：文本对齐方式，LABEL_ALIGN_LEFT 左对齐，LABEL_ALIGN_CENTER 居中，LABEL_ALIGN_RIGHT 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_toggle

功能	设置 imgbtn toggle 属性。
函数	lb_int32 lb_eui_imgbtn_ext_set_toggle(void *imgbtn, bool tgl)
参数	imgbtn：imgbtn 控件句柄；tgl：toggle 属性，true 表示支持 toggle，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_text

功能	设置 imgbtn 的字符串。
函数	lb_int32 lb_eui_imgbtn_ext_set_text(void *imgbtn, char *text)
参数	imgbtn：imgbtn 控件句柄；text：字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_text_id

功能	设置 imgbtn 状态。
函数	lb_int32 lb_eui_imgbtn_ext_set_text_id(void *imgbtn, lb_uint32 str_id)
参数	imgbtn：imgbtn 控件句柄；str_id：字符串在资源文件中的索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_state

功能	设置 imgbtn 状态。
函数	lb_int32 lb_eui_imgbtn_ext_set_state(void *imgbtn, lb_eui_btn_state_e state)
参数	imgbtn：imgbtn 控件句柄；state：新设置的状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_set_style

功能	设置 imgbtn 风格。
函数	lb_int32 lb_eui_imgbtn_ext_set_style(void *imgbtn, lb_eui_imgbtn_style_e type,lb_eui_style_t *lb_imgbtn_style)
参数	imgbtn：imgbtn 控件句柄；type：风格类型，参考 lb_eui_imgbtn_style_e；lb_imgbtn_style：新设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_imgbtn_ext_get_toggle

功能	获取 imgbtn 是否支持 toggle。
函数	lb_int32 lb_eui_imgbtn_ext_get_toggle(void *imgbtn, bool *tgl)
参数	imgbtn：imgbtn 控件句柄；tgl：返回 toggle 状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_get_state

功能	获取 imgbtn 当前状态。
函数	lb_int32 lb_eui_imgbtn_ext_get_state(void *imgbtn, lb_uint8 *state)
参数	imgbtn：imgbtn 控件句柄；state：返回 imgbtn 当前状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_imgbtn_ext_get_style

功能	获取 imgbtn 风格。
函数	lb_int32 lb_eui_imgbtn_ext_get_style(void *imgbtn, lb_eui_imgbtn_style_e type, lb_eui_style_t *lb_imgbtn_style)
参数	imgbtn：imgbtn 控件句柄；type ：风格类型，参考 lb_eui_btn_style_e；lb_imgbtn_style：获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.15. Keyboard 控件

```
/* keyboard mode */
typedef enum {
    LB_EUI_KB_MODE_LOWERCASE,
    LB_EUI_KB_MODE_CAPITAL,
    LB_EUI_KB_MODE_SPECIAL,
    LB_EUI_KB_MODE_CHINESE,
    LB_EUI_KB_MODE_NUM,
} lb_eui_kb_mode_e;

typedef enum {
    LB_EUI_KB_KEY_UP,
    LB_EUI_KB_KEY_DOWN,
    LB_EUI_KB_KEY_RIGHT,
    LB_EUI_KB_KEY_LEFT,
    LB_EUI_KB_KEY_PREV,
    LB_EUI_KB_KEY_NEXT,
    LB_EUI_KB_KEY_ENTER,
    LB_EUI_KB_KEY_DEL,
    LB_EUI_KB_KEY_EXCHANGE,
} lb_eui_kb_key_e;

typedef enum {
    LB_EUI_KB_STYLE_BG,
    LB_EUI_KB_STYLE_BTN_REL,
    LB_EUI_KB_STYLE_BTN_PR,
    LB_EUI_KB_STYLE_BTN_TGL_REL,
    LB_EUI_KB_STYLE_BTN_TGL_PR,
    LB_EUI_KB_STYLE_BTN_INA,
```

```
    /* under style only support input method */
    LB_EUI_KB_STYLE_PAR,
    LB_EUI_KB_STYLE_INPUT_BG,
    LB_EUI_KB_STYLE_RESULT_BTN_REL,
    LB_EUI_KB_STYLE_RESULT_BTN_PR,
    LB_EUI_KB_STYLE_RESULT_BG
} lb_eui_kb_style_e;
```

keyboard 控件的 API 接口定义如下：

Setter functions

lb_eui_kb_set_ta

功能	设置 kb 的文本域，用于显示键盘按下的字符。
函数	lb_int32 lb_eui_kb_set_ta(lb_int32 id, lb_int32 ta_id)
参数	id：kb 控件的 ID 号；ta_id： 绑定到 kb 的 ta 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_set_mode

功能	设置 kb 模式。
函数	lb_int32 lb_eui_kb_set_mode(lb_int32 id, kb_mode_e mode)
参数	id：kb 控件的 ID 号；mode： kb 的模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_set_cursor_manage

功能	设置 kb 的光标是否在文本域显示。
函数	lb_int32 lb_eui_kb_set_cursor_manage(lb_int32 id, bool en)
参数	id：kb 控件的 ID 号；en：true 表示显示光标，false 表示隐藏光标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_set_toggle

功能	设置 kb 切换使能。
函数	lb_int32 lb_eui_kb_set_toggle(lb_uint32 id, bool en)
参数	id：kb 控件的 ID 号；en：true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_set_key

功能	设置 kb 操作按键值。
函数	lb_int32 lb_eui_kb_set_key(lb_uint32 id, lb_eui_kb_key_e key)
参数	id：kb 控件的 ID 号；key：按键类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_set_style

功能	设置 kb 的风格。
函数	lb_int32 lb_eui_kb_set_style(lb_int32 id, lb_eui_kb_style_e type, lb_eui_style_t *lb_kb_style)
参数	id：kb 控件的 ID 号；type：风格类型，参考 lb_eui_kb_style_e； lb_kb_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_kb_get_mode

功能	获取 kb 模式。
函数	lb_int32 lb_eui_kb_get_mode(lb_int32 id, lb_uint8 *mode)
参数	id：kb 控件的 ID 号；mode： 返回 kb 的模式（模式定义参考 kb_mode_e）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_get_cursor_manage

功能	获取 kb 的光标是否在文本域显示。
函数	lb_int32 lb_eui_kb_get_cursor_manage(lb_int32 id, bool *en)
参数	id：kb 控件的 ID 号；en：true 表示显示光标，false 表示隐藏光标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_get_style

功能	获取 kb 的风格。
函数	lb_int32 lb_eui_kb_get_style(lb_int32 id, lb_eui_kb_style_e type, lb_eui_style_t *lb_kb_style)
参数	id：kb 控件的 ID 号；type：风格类型，参考 lb_eui_cb_style_e； lb_kb_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_set_action

功能	设置 kb 事件回调函数。
函数	lb_int32 lb_eui_kb_ext_set_action(void *kb,lb_eui_kb_action_type type, lb_eui_action_t action)
参数	kb：kb 控件句柄；type：事件类型；action：事件回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_set_ta

功能	设置 kb 的文本域，用于显示键盘按下的字符。
函数	lb_int32 lb_eui_kb_ext_set_ta(void *kb, lb_int32 ta_id)
参数	kb：kb 控件句柄；ta_id： 绑定到 kb 的 ta 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_set_mode

功能	设置 kb 模式。
函数	lb_int32 lb_eui_kb_ext_set_mode(void *kb, kb_mode_e mode)
参数	kb：kb 控件句柄；mode： kb 的模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_set_cursor_manage

功能	设置 kb 的光标是否在文本域显示。
函数	lb_int32 lb_eui_kb_ext_set_cursor_manage(void *kb, bool en)
参数	kb：kb 控件句柄；en：true 表示显示光标，false 表示隐藏光标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_set_toggle

功能	设置 kb 切换使能。
函数	lb_int32 lb_eui_kb_ext_set_toggle(void *kb, bool en)
参数	kb：kb 控件句柄；en：true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_set_key

功能	设置 kb 操作按键值。
函数	lb_int32 lb_eui_kb_ext_set_key(void *kb, lb_eui_kb_key_e key)
参数	kb：kb 控件句柄；key：按键类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_set_style

功能	设置 kb 的风格。
函数	lb_int32 lb_eui_kb_ext_set_style(void *kb, lb_eui_kb_style_e type, lb_eui_style_t *lb_kb_style)
参数	kb：kb 控件句柄；type：风格类型，参考 lb_eui_kb_style_e；lb_kb_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_kb_ext_get_mode

功能	获取 kb 模式。
函数	lb_int32 lb_eui_kb_ext_get_mode(void *kb, lb_uint8 *mode)
参数	kb：kb 控件句柄；mode： 返回 kb 的模式（模式定义参考 kb_mode_e）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_get_cursor_manage

功能	获取 kb 的光标是否在文本域显示。
函数	lb_int32 lb_eui_kb_ext_get_cursor_manage(void *kb, bool *en)
参数	kb：kb 控件句柄；en：true 表示显示光标，false 表示隐藏光标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_kb_ext_get_style

功能	获取 kb 的风格。
函数	lb_int32 lb_eui_kb_ext_get_style(void *kb, lb_eui_kb_style_e type, lb_eui_style_t *lb_kb_style)
参数	kb：kb 控件句柄；type：风格类型，参考 lb_eui_cb_style_e；lb_kb_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.16. Label 控件

```
typedef enum {
    LB_EUI_LABEL_LONG_EXPAND,     /*Expand the object size to the text size*/
    LB_EUI_LABEL_LONG_BREAK,
    /*Keep the object width, break the too long lines and expand the object height*/
    LB_EUI_LABEL_LONG_SCROLL,
    /*Expand the object size and scroll the text on the parent
    (move the label object)*/
    LB_EUI_LABEL_LONG_DOT,
    /*Keep the size and write dots at the end if the text is too long*/
    LB_EUI_LABEL_LONG_ROLL,        /*Keep the size and roll the text infinitely*/
    LB_EUI_LABEL_LONG_CROP,        /*Keep the size and crop the text out of it*/
}lb_eui_label_long_mode_e;

typedef enum {
    LB_EUI_LABEL_ALIGN_LEFT,
    LB_EUI_LABEL_ALIGN_CENTER,
    LB_EUI_LABEL_ALIGN_RIGHT,
}lb_eui_label_align_e;
```

label 控件的 API 接口定义如下：

Setter functions

lb_eui_label_set_text_id

功能	设置 label 字符串索引。
函数	lb_int32 lb_eui_label_set_text_id(lb_uint32 id, lb_uint32 str_id)
参数	id：label 控件的 ID 号；str_id：字符串在资源文件中的索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_text

功能	设置 label 新的字符，label 内部自动申请内存保存字符内容。
函数	lb_int32 lb_eui_label_set_text(lb_int32 id, const char *text)
参数	id：label 控件的 ID 号；text：新字符指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_array_text

功能	设置 label 新的字符，字符来自字符数组，label 内部自动申请内存保存字符内容。
函数	lb_int32 lb_eui_label_set_array_text(lb_int32 id, const char *array, lb_uint16 size)
参数	id：label 控件的 ID 号；array：新字符数组指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_static_text

功能	设置静态 label 新的字符，label 不需要保存字符，即使 label 退出了，静态 label 的字符还存在。
函数	lb_int32 lb_eui_label_set_static_text(lb_int32 id, const char *text)
参数	id：label 控件的 ID 号；text：新字符指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_long_mode

功能	设置 label long 的模式， 当字符串长度大于 label 控件 size 时，设置不同的处理模式。
函数	lb_int32 lb_eui_label_set_long_mode(lb_int32 id, lb_eui_label_long_mode_e long_mode)
参数	id：label 控件的 ID 号；long_mode：label 对于长字符串的处理模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_align

功能	设置 label 文本对齐方式。
函数	lb_int32 lb_eui_label_set_align(lb_int32 id, lb_eui_label_align_e align)
参数	id：label 控件的 ID 号；align：label 文本对齐方式，LABEL_ALIGN_LEFT 左对齐，LABEL_ALIGN_CENTER 居中，LABEL_ALIGN_RIGHT 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_body_draw

功能	设置 label 是否显示背景，背景属性通过 style's body 设置。
函数	lb_int32 lb_eui_label_set_body_draw(lb_int32 id, bool en)
参数	id：label 控件的 ID 号；en：true 表示显示背景，false 表示不显示背景。
返回值	0 表示成功，其他负值表示错误码。

Recolor 是值解析文本中自带的颜色字符串。颜色字符串以#开头，以#结尾，使用 16 进制数表示，
列如 char string[16] = {“#FF0000 red#”}，显示的是红色的 red 字符串。

lb_eui_label_set_recolor

功能	设置 label 是否支持 recolor。
函数	lb_int32 lb_eui_label_set_recolor(lb_uint32 id, bool en)
参数	id：label 控件的 ID 号；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_anim_speed

功能	设置 label 滚动速度，当 long_mode 为 LB_EUI_LABEL_LONG_ROLL and SCROLL modes。
函数	lb_int32 lb_eui_label_set_anim_speed(lb_int32 id, lb_uint16 anim_speed)
参数	id：label 控件的 ID 号；anim_speed：文本滑动速度，单位（px/sec）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_set_style

功能	设置 label 的风格。
函数	lb_int32 lb_eui_label_set_style(lb_int32 id, lb_eui_style_t *lb_label_style)
参数	id：label 控件的 ID 号；lb_label_style：风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_label_get_text

功能	获取 label 的文本。
函数	lb_int32 lb_eui_label_get_text(lb_int32 id, char **text)
参数	id：label 控件的 ID 号；text：返回 label 的文本。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_long_mode

功能	获取 label long 的模式。
函数	lb_int32 lb_eui_label_get_long_mode(lb_int32 id, lb_uint8 *long_mode)
参数	id：label 控件的 ID 号；long_mode：返回 label long 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_align

功能	获取 label 文本对齐方式。
函数	lb_int32 lb_eui_label_get_align(lb_int32 id, lb_uint8 *align)
参数	id：label 控件的 ID 号；align：返回 label 文本对齐方式，0 左对齐，1 居中，2 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_recolor

功能	获取 label 是否支持 recolor。
函数	lb_int32 lb_eui_label_get_recolor(lb_uint32 id, bool *en)
参数	id：label 控件的 ID 号；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_body_draw

功能	获取 label 是否显示背景属性状态。
函数	lb_int32 lb_eui_label_get_body_draw(lb_int32 id, bool *en)
参数	id：label 控件的 ID 号；en：返回 label 显示背景属性状态，true 表示显示背景，false 表示不显示。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_anim_speed

功能	获取 label 文本滚动速度。
函数	lb_int32 lb_eui_label_get_anim_speed(lb_int32 id, lb_uint16 *anim_speed)
参数	id：label 控件的 ID 号；anim_speed：返回 label 文本滑动速度，单位（px/sec）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_letter_pos

功能	获取 label 指定索引 letter 的位置坐标。
函数	lb_int32 lb_eui_label_get_letter_pos(lb_int32 id, lb_uint16 index, lb_eui_point_t **pos)
参数	id：label 控件的 ID 号；index：label 的 letter 索引，范围（0 至字符串长度-1）；pos：返回 letter 的位置坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_letter_on

功能	获取 label 指定位置坐标所在的 letter 的索引值。
函数	lb_int32 lb_eui_label_get_letter_on(lb_int32 id, lb_point_t *pos, lb_uint16 *index)
参数	id：label 控件的 ID 号；pos：letter 的位置坐标;index：返回 label 的 letter 索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_get_style

功能	获取 label 的风格。
函数	lb_int32 lb_eui_label_get_style(lb_int32 id, lb_eui_style_t *lb_label_style)
参数	id：label 控件的 ID 号；lb_label_style：返回风格的指针。
返回值	0 表示成功，其他负值表示错误码。

Other functions

lb_eui_label_ins_text

功能	在 label 指定位置插入字符，label 不可以是 static 的。
函数	lb_int32 lb_eui_label_ins_text(lb_int32 id, lb_uint32 pos, const char *txt)
参数	id：label 控件的 ID 号；pos：label 插入位置索引值;txt：新插入字符。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_cut_text

功能	在 label 指定位置删除字符，label 不可以是 static 的。
函数	lb_int32 lb_eui_label_cut_text(lb_int32 id, lb_uint32 pos, lb_uint32 cnt)
参数	id：label 控件的 ID 号；pos：label 删除字符起始位置索引;cnt：删除字符的个数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_text_id

功能	设置 label 字符串索引。
函数	lb_int32 lb_eui_label_ext_set_text_id(void *label, lb_uint32 str_id)
参数	label：label 控件句柄；str_id：字符串在资源文件中的索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_text

功能	设置 label 新的字符，label 内部自动申请内存保存字符内容。
函数	lb_int32 lb_eui_label_ext_set_text(void *label, const char *text)
参数	label：label 控件句柄；text：新字符指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_array_text

功能	设置 label 新的字符，字符来自字符数组，label 内部自动申请内存保存字符内容。
函数	lb_int32 lb_eui_label_ext_set_array_text(void *label, const char *array, lb_uint16 size)
参数	label：label 控件句柄；array：新字符数组指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_static_text

功能	设置静态 label 新的字符，label 不需要保存字符，即使 label 退出了，静态 label 的字符还存在。

函数	lb_int32 lb_eui_label_ext_set_static_text(void *label, const char *text)
参数	label：label 控件句柄；text：新字符指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_long_mode

功能	设置 label long 的模式， 当字符串长度大于 label 控件 size 时，设置不同的处理模式。
函数	lb_int32 lb_eui_label_ext_set_long_mode(void *label, lb_eui_label_long_mode_e long_mode)
参数	label：label 控件句柄；long_mode：label 对于长字符串的处理模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_align

功能	设置 label 文本对齐方式。
函数	lb_int32 lb_eui_label_ext_set_align(void *label, lb_eui_label_align_e align)
参数	label：label 控件句柄；align：label 文本对齐方式，LABEL_ALIGN_LEFT 左对齐，LABEL_ALIGN_CENTER 居中，LABEL_ALIGN_RIGHT 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_body_draw

功能	设置 label 是否显示背景，背景属性通过 style's body 设置。
函数	lb_int32 lb_eui_label_ext_set_body_draw(void *label, bool en)
参数	label：label 控件句柄；en：true 表示显示背景，false 表示不显示背景。
返回值	0 表示成功，其他负值表示错误码。

Recolor 是值解析文本中自带的颜色字符串。颜色字符串以#开头，以#结尾，使用 16 进制数表示，列如 char string[16] = {“#FF0000 red#”}，显示的是红色的 red 字符串。

lb_eui_label_ext_set_recolor

功能	设置 label 是否支持 recolor。
函数	lb_int32 lb_eui_label_ext_set_recolor(void *label, bool en)
参数	label：label 控件句柄；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_anim_speed

功能	设置 label 滚动速度，当 long_mode 为 LB_EUI_LABEL_LONG_ROLL and SCROLL modes。
函数	lb_int32 lb_eui_label_ext_set_anim_speed(void *label, lb_uint16 anim_speed)
参数	label：label 控件句柄；anim_speed：文本滑动速度，单位（px/sec）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_set_style

功能	设置 label 的风格。
函数	lb_int32 lb_eui_label_ext_set_style(void *label, lb_eui_style_t *lb_label_style)
参数	label：label 控件句柄；lb_label_style：风格。

返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_label_ext_get_text

功能	获取 label 的文本。
函数	lb_int32 lb_eui_label_ext_get_text(void *label, char **text)
参数	label：label 控件句柄；text：返回 label 的文本。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_long_mode

功能	获取 label long 的模式。
函数	lb_int32 lb_eui_label_ext_get_long_mode(void *label, lb_uint8 *long_mode)
参数	label：label 控件句柄；long_mode：返回 label long 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_align

功能	获取 label 文本对齐方式。
函数	lb_int32 lb_eui_label_ext_get_align(void *label, lb_uint8 *align)
参数	label：label 控件句柄；align：返回 label 文本对齐方式，0 左对齐，1 居中，2 右对齐。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_recolor

功能	获取 label 是否支持 recolor。
函数	lb_int32 lb_eui_label_ext_get_recolor(void *label, bool *en)
参数	label：label 控件句柄；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_body_draw

功能	获取 label 是否显示背景属性状态。
函数	lb_int32 lb_eui_label_ext_get_body_draw(void *label, bool *en)
参数	label：label 控件句柄；en：返回 label 显示背景属性状态，true 表示显示背景，false 表示不显示。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_anim_speed

功能	获取 label 文本滚动速度。
函数	lb_int32 lb_eui_label_ext_get_anim_speed(void *label, lb_uint16 *anim_speed)
参数	label：label 控件句柄；anim_speed：返回 label 文本滑动速度，单位（px/sec）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_letter_pos

功能	获取 label 指定索引 letter 的位置坐标。
函数	lb_int32 lb_eui_label_ext_get_letter_pos(void *label, lb_uint16 index, lb_eui_point_t **pos)

参数	label：label 控件句柄；index：label 的 letter 索引，范围（0 至字符串长度-1）；pos：返回 letter 的位置坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_letter_on

功能	获取 label 指定位置坐标所在的 letter 的索引值。
函数	lb_int32 lb_eui_label_ext_get_letter_on(void *label, lb_point_t *pos, lb_uint16 *index)
参数	label：label 控件句柄；pos：letter 的位置坐标;index：返回 label 的 letter 索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_get_style

功能	获取 label 的风格。
函数	lb_int32 lb_eui_label_ext_get_style(void *label, lb_eui_style_t *lb_label_style)
参数	label：label 控件句柄；lb_label_style：返回风格的指针。
返回值	0 表示成功，其他负值表示错误码。

Other functions

lb_eui_label_ext_ins_text

功能	在 label 指定位置插入字符，label 不可以是 static 的。
函数	lb_int32 lb_eui_label_ext_ins_text(void *label, lb_uint32 pos, const char *txt)
参数	label：label 控件句柄；pos：label 插入位置索引值;txt：新插入字符。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_label_ext_cut_text

功能	在 label 指定位置删除字符，label 不可以是 static 的。
函数	lb_int32 lb_eui_label_ext_cut_text(void *label, lb_uint32 pos, lb_uint32 cnt)
参数	label：label 控件句柄；pos：label 删除字符起始位置索引;cnt：删除字符的个数。
返回值	0 表示成功，其他负值表示错误码。

4.5.17. Led 控件

Led 控件的 API 接口定义如下：

Setter functions

lb_eui_led_set_bright

功能	设置 led 的亮度。
函数	lb_int32 lb_eui_led_set_bright(lb_int32 id, lb_uint8 bright)
参数	id：led 控件的 ID 号；bright：led 亮度值，0 表示最暗，255 表示最亮。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_on

功能	设置 led 为点亮状态。

函数	lb_int32 lb_eui_led_on(lb_int32 id)
参数	id：led 控件的 ID 号；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_off

功能	设置 led 为熄灭状态 。
函数	lb_int32 lb_eui_led_off(lb_int32 id)
参数	id：led 控件的 ID 号；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_toggle

功能	切换 led 状态。
函数	lb_int32 lb_eui_led_toggle(lb_int32 id)
参数	id：led 控件的 ID 号；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_set_style

功能	设置 led 的风格。
函数	lb_int32 lb_eui_led_set_style(lb_int32 id, lb_eui_style_t *lb_led_style)
参数	id：led 控件的 ID 号；lb_led_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_led_get_bright

功能	获取 led 亮度。
函数	lb_int32 lb_eui_led_get_bright(lb_int32 id，lb_uint8 *bright)
参数	id：led 控件的 ID 号；bright：返回 led 亮度
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_get_style

功能	获取 led 的风格。
函数	lb_int32 lb_eui_led_get_style(lb_int32 id, lb_eui_style_t *lb_led_style)
参数	id：led 控件的 ID 号；lb_led_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_ext_set_bright

功能	设置 led 的亮度。
函数	lb_int32 lb_eui_led_ext_set_bright(void *led, lb_uint8 bright)
参数	led：led 控件句柄；bright：led 亮度值，0 表示最暗，255 表示最亮。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_ext_on

功能	设置 led 为点亮状态。
函数	lb_int32 lb_eui_led_ext_on(void *led)
参数	led：led 控件句柄；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_ext_off

功能	设置 led 为熄灭状态 。
函数	lb_int32 lb_eui_led_ext_off(void *led)
参数	i led：led 控件句柄；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_ext_toggle

功能	切换 led 状态。
函数	lb_int32 lb_eui_led_ext_toggle(void *led)
参数	led：led 控件句柄；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_ext_set_style

功能	设置 led 的风格。
函数	lb_int32 lb_eui_led_ext_set_style(void *led, lb_eui_style_t *lb_led_style)
参数	led：led 控件句柄；lb_led_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_led_ext_get_bright

功能	获取 led 亮度。
函数	lb_int32 lb_eui_led_ext_get_bright(void *led，lb_uint8 *bright)
参数	led：led 控件句柄；bright：返回 led 亮度
返回值	0 表示成功，其他负值表示错误码。

lb_eui_led_ext_get_style

功能	获取 led 的风格。
函数	lb_int32 lb_eui_led_ext_get_style(void *led, lb_eui_style_t *lb_led_style)
参数	led：led 控件句柄；lb_led_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.18. Line 控件

line 控件的 API 接口定义如下：

Setter functions

lb_eui_line_set_points

功能	设置 line 点的位置数组，line 会连接这些点。
函数	lb_int32 lb_eui_line_set_points(lb_int32 id, lb_eui_point_t *point_a, lb_uint16 point_num)
参数	id：line 控件的 ID 号；point_a：line 点的位置数组；point_num：点的位置数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_set_auto_size

功能	设置 line 自动尺寸属性。
函数	lb_int32 lb_eui_line_set_auto_size(lb_int32 id, bool en)
参数	id：line 控件的 ID 号；en：line 自动尺寸属性，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_set_y_invert

功能	设置 line y 坐标翻转属性。
函数	lb_int32 lb_eui_line_set_y_invert(lb_int32 id, bool en)
参数	id：line 控件的 ID 号；en：line y 坐标翻转属性，true 表示 line 支持 y 坐标翻转，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_set_style

功能	设置 line 的风格。
函数	lb_int32 lb_eui_line_set_style(lb_int32 id, lb_eui_style_t *lb_line_style)
参数	id：line 控件的 ID 号；lb_line_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_line_get_points

功能	获取 line 的点数组及个数。
函数	lb_int32 lb_eui_line_get_points(lb_uint32 id, void **point_a, lb_uint16 *point_num)
参数	id：line 控件的 ID 号；en：返回 line 自动尺寸属性状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_get_auto_size

功能	获取 line 自动尺寸属性。
函数	lb_int32 lb_eui_line_get_auto_size(lb_int32 id, bool *en)
参数	id：line 控件的 ID 号；en：返回 line 自动尺寸属性状态，true 表示支持，false 表示不支持。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_get_y_invert

功能	获取 line y 坐标翻转属性。
函数	lb_int32 lb_eui_line_get_y_invert(lb_int32 id, bool *en)
参数	id：line 控件的 ID 号；en：返回 line y 坐标翻转属性状态，true 表示 line 支持 y 坐标翻转，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_get_style

功能	获取 line 的风格。
函数	lb_int32 lb_eui_line_get_style(lb_int32 id, lb_eui_style_t *lb_line_style)
参数	id：line 控件的 ID 号；lb_line_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_ext_set_points

功能	设置 line 点的位置数组，line 会连接这些点。
函数	lb_int32 lb_eui_line_ext_set_points(void *line, lb_eui_point_t *point_a, lb_uint16 point_num)
参数	line：line 控件句柄；point_a：line 点的位置数组；point_num：点的位置数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_ext_set_auto_size

功能	设置 line 自动尺寸属性。
函数	lb_int32 lb_eui_line_ext_set_auto_size(void *line, bool en)
参数	line：line 控件句柄；en：line 自动尺寸属性，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_ext_set_y_invert

功能	设置 line y 坐标翻转属性。
函数	lb_int32 lb_eui_line_set_y_invert(void *line, bool en)
参数	line：line 控件句柄；en：line y 坐标翻转属性，true 表示 line 支持 y 坐标翻转，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_ext_set_style

功能	设置 line 的风格。
函数	lb_int32 lb_eui_line_ext_set_style(void *line, lb_eui_style_t *lb_line_style)
参数	line：line 控件句柄；lb_line_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_line_ext_get_points

功能	获取 line 的点数组及个数。
函数	lb_int32 lb_eui_ext_line_get_points(void *line, void **point_a, lb_uint16 *point_num)
参数	line：line 控件句柄；en：返回 line 自动尺寸属性状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_ext_get_auto_size

功能	获取 line 自动尺寸属性。
函数	lb_int32 lb_eui_line_ext_get_auto_size(void *line, bool *en)
参数	line：line 控件句柄；en：返回 line 自动尺寸属性状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_line_ext_get_y_invert

功能	获取 line y 坐标翻转属性。
函数	lb_int32 lb_eui_line_ext_get_y_invert(void *line, bool *en)
参数	line：line 控件句柄；en：返回 line y 坐标翻转属性状态，true 表示 line 支持 y 坐标翻转，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

4.5.19. List 控件

```
typedef enum {
    LB_EUI_LIST_STYLE_BG,
    LB_EUI_LIST_STYLE_SCRL,
    LB_EUI_LIST_STYLE_SB,
    LB_EUI_LIST_STYLE_EDGE_FLASH,
    LB_EUI_LIST_STYLE_BTN_REL,
    LB_EUI_LIST_STYLE_BTN_PR,
    LB_EUI_LIST_STYLE_BTN_TGL_REL,
    LB_EUI_LIST_STYLE_BTN_TGL_PR,
    LB_EUI_LIST_STYLE_BTN_INA,
} lb_eui_list_style_e;

typedef struct tag_eui_list_click {
```

```
    lb_uint32 id;
    lb_uint32 cur_index;
} lb_eui_list_click_t;
```

line 控件的 API 接口定义如下：

Add/remove functions

lb_eui_list_add

功能	添加 list 成员
函数	lb_int32 lb_eui_list_add(lb_int32 id, const void *img_src, const char *txt)
参数	id：list 控件的 ID 号；img_src：list 成员图片资源文件名称，NULL 时表示未使用图片；txt：list 成员文本字符串，NULL 时表示未使用 text。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_remove

功能	删除 list 成员
函数	lb_int32 lb_eui_list_remove(lb_int32 id, lb_uint32 index)
参数	id：list 控件的 ID 号；index：list 成员索引值（范围 0 - element num）。
返回值	0 表示成功，其他负值表示错误码。

Setter function

lb_eui_list_set_single_mode

功能	设置 list 单个按钮选中模式。
函数	lb_int32 lb_eui_list_set_single_mode(lb_int32 id, bool mode)
参数	id：list 控件的 ID 号；mode：单个按钮选中模式，true 表示只能选中一个按钮，false 表示可以选中不止一个按钮。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_set_btn_selected

功能	设置 list 按钮为选中状态。
函数	lb_int32 lb_eui_list_set_btn_selected(lb_int32 list_id, lb_uint32 btn_id)
参数	id：list 控件的 ID 号；btn_id：按钮控件 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_set_anim_time

功能	设置 list 滚动动画时间。
函数	lb_int32 lb_eui_list_set_anim_time(lb_int32 id, lb_uint16 anim_time)
参数	id：list 控件的 ID 号；anim_time：list 滚动动画时间。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_set_sb_mode

功能	设置 list scroll bar 模式。
函数	lb_int32 lb_eui_list_set_sb_mode(lb_int32 id, sb_mode_e mode)
参数	id：list 控件的 ID 号；mode：list scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_set_scroll_propagation

功能	设置 list 滚动传播属性，如果启用传播属性，当 list 没有可滚动空间时，将会移动 list 父控件。
函数	lb_int32 lb_eui_list_set_scroll_propagation(lb_int32 id, bool en)
参数	id：list 控件的 ID 号；en：list 滚动传播属性，true 表示启用 list 滚动传播，false 表示不启用。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_set_edge_flash

功能	设置 list 边沿闪光效果（当滑动到边沿时显示弧形）。
函数	lb_int32 lb_eui_list_set_edge_flash(lb_int32 id, bool en)
参数	id：list 控件的 ID 号；en：true 表示 list 支持边沿闪光效果，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_set_style

功能	设置 list 的风格。
函数	lb_int32 lb_eui_list_set_style(lb_int32 id, lb_eui_list_style_e type, lb_eui_style_t *lb_list_style)
参数	id：list 控件的 ID 号；type：风格类型，参考 lb_eui_list_style_e； lb_list_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_list_get_single_mode

功能	获取 list 单个按钮选中模式。
函数	lb_int32 lb_eui_list_get_single_mode(lb_int32 id, bool *mode)
参数	id：list 控件的 ID 号；mode：返回单个按钮选中模式状态，true 表示只能选中一个按钮，false 表示可以选中不止一个按钮。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_size

功能	获取 list 成员按钮的数目。
函数	lb_int32 lb_eui_list_get_size(lb_int32 id,lb_uint32 *num)
参数	id：list 控件的 ID 号；num：返回 list 成员按钮的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_btn_index

功能	获取 list 已选中按钮控件 ID 号。
函数	lb_int32 lb_eui_list_get_btn_index(void *btn, lb_uint32 *btn_index)
参数	id：list 控件的 ID 号；btn_index：返回 list 已选中按钮控件 ID 号或者索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_btn_selected

功能	获取 list 已选中按钮控件 ID 号。
函数	lb_int32 lb_eui_list_get_btn_selected(lb_int32 list_id, lb_int32 *btn_id)
参数	id：list 控件的 ID 号；btn_id：返回 list 已选中按钮控件 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_anim_time

功能	获取 list 滚动动画时间。
函数	lb_int32 lb_eui_list_get_anim_time(lb_int32 id, lb_uint16 *anim_time)
参数	id：list 控件的 ID 号；anim_time：返回 list 滚动动画时间。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_sb_mode

功能	获取 list scroll bar 模式。
函数	lb_int32 lb_eui_list_get_sb_mode(lb_int32 id, lb_uint8 *mode)
参数	id：list 控件的 ID 号；mode：返回 list scroll bar 模式，模式定义参考 sb_mode_e。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_scroll_propagation

功能	获取 list 滚动传播属性。
函数	lb_int32 lb_eui_list_get_scroll_propagation(lb_int32 id, bool *en)
参数	id：list 控件的 ID 号；en：返回 list 滚动传播属性，true 表示支持 list 滚动传播，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_edge_flash

功能	获取 list 边沿闪光效果（当滑动到边沿时显示弧形）。
函数	lb_int32 lb_eui_list_get_edge_flash(lb_int32 id, bool *en)
参数	id：list 控件的 ID 号；en：返回 list 边沿闪光效果属性状态，true 表示 list 支持边沿闪光效果，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_style

功能	获取 list 的风格。
函数	lb_int32 lb_eui_list_get_style(lb_int32 id, lb_eui_list_style_e type, lb_eui_style_t *lb_list_style)
参数	id：list 控件的 ID 号；type：风格类型，参考 lb_eui_list_style_e； lb_list_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

Dynamic function

```
typedef enum {
    LB_EUI_LIST_COL_TYPE_TEXT,
    LB_EUI_LIST_COL_TYPE_IMG,
    LB_EUI_LIST_COL_TYPE_IMGBTN,
    LB_EUI_LIST_COL_TYPE_NULL
} lb_eui_list_col_type_e;

typedef enum {
    LB_EUI_LIST_UPDATE_ONE_ROW,
    LB_EUI_LIST_UPDATE_ONE_GIRD,
    LB_EUI_LIST_UPDATE_ALL_ELEM
} lb_eui_list_update_type_e;

typedef enum {
    LB_EUI_LIST_REFRESH_START,
    LB_EUI_LIST_REFRESH_END,
    LB_EUI_LIST_REFRESH_OTHER
} lb_eui_list_refresh_type_e;

lb_eui_list_action_t 的参数为控件句柄，可与 lb_eui_list_get_btn_index 搭配使用
typedef void *(*lb_eui_list_load_t)(int32_t add_index, int32_t rm_index);
typedef lb_uint8(*lb_eui_list_action_t)(void *param);

typedef struct _tag_eui_list_imgbtn_t_ {
    void            *p_rel_img_src;
```

```
    void            *p_pr_img_src;
    lb_eui_list_action_taction;
    lb_uint8        reserved:4;
} lb_eui_list_imgbtn_t;

typedef struct _tag_eui_list_col_t {
    lb_uint8        type;
    lb_uint8        b_update;
    lb_uint8        reserved:2;
    lb_uint32       gird;
    void            *data;
    void            *obj;
} lb_eui_list_col_t;

typedef struct _tag_eui_list_elem_t {
    lb_int32        index;
    lb_uint8        b_created;
    lb_uint8        reserved:3;
    lb_eui_list_action_taction;
    lb_eui_list_col_t    *col;
    void            *liste;
} lb_eui_list_elem_t;

typedef struct _tag_eui_list_refresh_t {
    /* if type is LB_EUI_LIST_LOAD_FROM_START and LB_EUI_LIST_LOAD_FROM_OTHER,
     * it is start index; otherwise, it is end index.
     */
    lb_uint32       index;
    lb_uint8        type;
    lb_uint8        reserved:3;
} lb_eui_list_refresh_t;
```

lb_eui_list_set_dynamic_load_cb

功能	设置 list 动态加载回调函数。
函数	lb_int32 lb_eui_list_set_dynamic_load_cb(lb_uint32 id, lb_eui_list_load_t cb)
参数	id：list 控件的 ID 号；cb：回调函数，由用户填充。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_update_elem_info

功能	动态更新 list 控件选项信息。
函数	lb_int32 lb_eui_list_update_elem_info(lb_uint32 id, lb_uint8 type, void *elem_info, lb_uint32 len)
参数	id：list 控件的 ID 号；type：更新类型，参考 lb_eui_list_update_type_e； elem_info：更新数据，len：数据大小；。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_get_visible_items

功能	获取 list 当前加载项范围。
函数	lb_int32 lb_eui_list_get_visible_items(lb_uint32 id, lb_int32 *start, lb_int32 *end)
参数	id：list 控件的 ID 号；start：返回 list 可见成员起始序号值；end：返回 list 可见成员结束序号值。
返回值	0 表示成功，其他负值表示错误码。

Other functions

lb_eui_list_clean

功能	清除 list。
函数	lb_int32 lb_eui_list_up(lb_int32 id)
参数	id：list 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_up

功能	向上移动 list 成员（每次移动一个成员）。
函数	lb_int32 lb_eui_list_up(lb_int32 id)
参数	id：list 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_down

功能	向下移动 list 成员（每次移动一个成员）。
函数	lb_int32 lb_eui_list_down(lb_int32 id)
参数	id：list 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_focus

功能	设置 list 成员按钮焦点，确保按钮在 list 中可见。
函数	lb_int32 lb_eui_list_focus(lb_uint32 id，lb_int32 btn_id，bool anim_en)
参数	id：list 控件的 ID 号；btn_id：list 成员 btn 控件的 ID 号；anim_en：滚动动画属性，true 表示支持滚动动画，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_list_btn_checked

功能	检测当前选项是否存在。
函数	lb_int32 lb_eui_list_btn_checked(lb_uint32 id, lb_int32 btn_index)

参数	id：list 控件的 ID 号；btn_index：list 成员 btn 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

4.5.20. Line Meter 控件

Line meter 控件的 API 接口定义如下：

Setter functions

lb_eui_lmeter_set_value

功能	设置 lmeter 新的值。
函数	lb_int32 lb_eui_lmeter_set_value(lb_int32 id, lb_int16 value)
参数	id：lmeter 控件的 ID 号；value：lmeter 新的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_set_range

功能	设置 lmeter 值的范围。
函数	lb_int32 lb_eui_lmeter_set_range(lb_int32 id, lb_int16 min, lb_int16 max)
参数	id：lmeter 控件的 ID 号；min：lmeter 的最小值；max：lmeter 的最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_set_scale

功能	设置 lmeter 的刻度属性。
函数	lb_int32 lb_eui_lmeter_set_scale(lb_int32 id, lb_uint16 angle, lb_uint8 line_cnt)
参数	id：lmeter 控件的 ID 号；angle：lmeter 刻度线的角度（0-360）；line_cnt：刻度线的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_set_style

功能	设置 lmeter 的风格。
函数	lb_int32 lb_eui_lmeter_set_style(lb_int32 id, lb_eui_style_t *lb_lmeter_style)
参数	id：lmeter 控件的 ID 号；lb_lmeter_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_lmeter_get_value

功能	获取 lmeter 当前的值。
函数	lb_int32 lb_eui_lmeter_get_value(lb_int32 id, lb_int16 *value)
参数	id：lmeter 控件的 ID 号；value：返回 lmeter 当前的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_get_min_value

功能	获取 lmeter 值范围的最小值。
函数	lb_int32 lb_eui_lmeter_get_min_value(lb_int32 id, lb_int16 *min)

参数	id：lmeter 控件的 ID 号；min：返回 lmeter 的最小值；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_get_max_value

功能	获取 lmeter 值范围的最大值。
函数	lb_int32 lb_eui_lmeter_get_max_value(lb_int32 id, lb_int16 *max)
参数	id：lmeter 控件的 ID 号；max：返回 lmeter 的最大值；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_get_line_count

功能	获取 lmeter 刻度线的数目。
函数	lb_int32 lb_eui_lmeter_get_line_count(lb_int32 id, lb_uint8 *line_cnt)
参数	id：lmeter 控件的 ID 号；line_cnt：返回 lmeter 刻度线的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_get_scale_angle

功能	获取 lmeter 刻度线的角度。
函数	lb_int32 lb_eui_lmeter_get_scale_angle(lb_int32 id, lb_uint16 *angle)
参数	id：lmeter 控件的 ID 号；angle：返回 lmeter 刻度线的角度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_get_style

功能	获取 lmeter 的风格。
函数	lb_int32 lb_eui_lmeter_get_style(lb_int32 id, lb_eui_style_t *lb_lmeter_style)
参数	id：lmeter 控件的 ID 号；lb_lmeter_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_set_value

功能	设置 lmeter 新的值。
函数	lb_int32 lb_eui_lmeter_ext-set_value(void *lmeter, lb_int16 value)
参数	lmeter：lmeter 控件句柄；value：lmeter 新的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_set_range

功能	设置 lmeter 值的范围。
函数	lb_int32 lb_eui_lmeter_ext_set_range(void *lmeter, lb_int16 min, lb_int16 max)
参数	lmeter：lmeter 控件句柄；min：lmeter 的最小值；max：lmeter 的最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_set_scale

功能	设置 lmeter 的刻度属性。
函数	lb_int32 lb_eui_lmeter_ext_set_scale(void *lmeter, lb_uint16 angle, lb_uint8 line_cnt)
参数	lmeter：lmeter 控件句柄；angle：lmeter 刻度线的角度（0-360）；line_cnt：刻度线的数目。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_set_style

功能	设置 lmeter 的风格。
函数	lb_int32 lb_eui_lmeter_ext_set_style(void *lmeter, lb_eui_style_t *lb_lmeter_style)
参数	lmeter：lmeter 控件句柄；lb_lmeter_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_lmeter_ext_get_value

功能	获取 lmeter 当前的值。
函数	lb_int32 lb_eui_lmeter_ext_get_value(void *lmeter, lb_int16 *value)
参数	lmeter：lmeter 控件句柄；value：返回 lmeter 当前的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_get_min_value

功能	获取 lmeter 值范围的最小值。
函数	lb_int32 lb_eui_lmeter_ext_get_min_value(void *lmeter, lb_int16 *min)
参数	lmeter：lmeter 控件句柄；min：返回 lmeter 的最小值；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_get_max_value

功能	获取 lmeter 值范围的最大值。
函数	lb_int32 lb_eui_lmeter_ext_get_max_value(void *lmeter, lb_int16 *max)
参数	lmeter：lmeter 控件句柄；max：返回 lmeter 的最大值；
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_get_line_count

功能	获取 lmeter 刻度线的数目。
函数	lb_int32 lb_eui_lmeter_ext_get_line_count(void *lmeter, lb_uint8 *line_cnt)
参数	lmeter：lmeter 控件句柄；line_cnt：返回 lmeter 刻度线的数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_get_scale_angle

功能	获取 lmeter 刻度线的角度。
函数	lb_int32 lb_eui_lmeter_ext_get_scale_angle(void *lmeter, lb_uint16 *angle)
参数	lmeter：lmeter 控件句柄；angle：返回 lmeter 刻度线的角度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_lmeter_ext_get_style

功能	获取 lmeter 的风格。
函数	lb_int32 lb_eui_lmeter_ext_get_style(void *lmeter, lb_eui_style_t *lb_lmeter_style)
参数	lmeter：lmeter 控件句柄；lb_lmeter_style：需要获取的风格。

返回值	0 表示成功，其他负值表示错误码。

4.5.21. Message box 控件

```
typedef enum {
    LB_EUI_MBOX_STYLE_BG,
    LB_EUI_MBOX_STYLE_BTN_BG,
    LB_EUI_MBOX_STYLE_BTN_REL,
    LB_EUI_MBOX_STYLE_BTN_PR,
    LB_EUI_MBOX_STYLE_BTN_TGL_REL,
    LB_EUI_MBOX_STYLE_BTN_TGL_PR,
    LB_EUI_MBOX_STYLE_BTN_INA,
} lb_eui_mbox_style_e;
```

message 控件的 API 接口定义如下：

Setter functions

lb_eui_mbox_set_text

功能	设置 mbox 文本。
函数	lb_int32 lb_eui_mbox_set_text(lb_int32 id, const char *txt)
参数	id：mbox 控件的 ID 号；txt：mbox 新的文本。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_set_btns

功能	设置 mbox 选项按钮。
函数	lb_int32 lb_eui_mbox_set_btns(lb_uint32 id, const char **btn_map,lb_eui_btnm_action_t action)
参数	id：mbox 控件的 ID 号；btn_map：字符串数组，action：事件回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_set_anim_time

功能	设置 mbox 的动画时间。
函数	lb_int32 lb_eui_mbox_set_anim_time(lb_int32 id, lb_uint16 anim_time)
参数	id：mbox 控件的 ID 号；anim_time：mbox 的动画时间，单位为 ms,0 为无动画。
返回值	0 表示成功，其他负值表示错误码。

Recolor 是值解析文本中自带的颜色字符串。颜色字符串以#开头，以#结尾，使用 16 进制数表示，列如 char string[16] = {“#FF0000 red#”}，显示的是红色的 red 字符串。

lb_eui_mbox_set_recolor

功能	设置 mbox 是否支持 recolor。
函数	lb_int32 lb_eui_mbox_set_recolor(lb_uint32 id, bool en)
参数	id：mbox 控件的 ID 号；en：true 表示支持 recolor，false 表示不支持 recolor。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_btnm_toggle

功能	设置 mbox 按键切换状态。
函数	lb_int32 lb_eui_mbox_btnm_toggle(lb_uint32 id, lb_uint16 toggle_id)
参数	id：mbox 控件的 ID 号；toggle_id：切换状态的按键索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_auto_close

功能	设置 mbox 自动隐藏。
函数	lb_int32 lb_eui_mbox_auto_close(lb_uint32 id, lb_uint16 delay, lb_eui_anim_cb_t cb)
参数	id：mbox 控件的 ID 号；delay：延时时间（ms）；cb：隐藏回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_set_style

功能	设置 mbox 的风格。
函数	lb_int32 lb_eui_mbox_set_style(lb_int32 id, lb_eui_mbox_style_e type, lb_eui_style_t *lb_mbox_style)
参数	id：mbox 控件的 ID 号；type：风格类型，参考 lb_eui_mbox_style_e； lb_mbox_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_mbox_get_text

功能	获取 mbox 的文本内容。
函数	lb_int32 lb_eui_mbox_get_text(lb_int32 id, char **txt)
参数	id：mbox 控件的 ID 号；txt：返回 mbox 文本内容地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_get_anim_time

功能	获取 mbox 的动画时间。
函数	lb_int32 lb_eui_mbox_get_anim_time(lb_int32 id, lb_uint16 *anim_time)
参数	id：mbox 控件的 ID 号；anim_time：返回 mbox 的动画时间，单位为 ms,0 为无动画。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_get_recolor

功能	获取 mbox 是否支持 recolor。
函数	lb_int32 lb_eui_mbox_get_recolor(lb_uint32 id, bool *en)
参数	id：label 控件的 ID 号；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_get_style

功能	获取 mbox 的风格。

函数	lb_int32 lb_eui_mbox_get_style(lb_int32 id, lb_eui_mbox_style_e type, lb_eui_style_t *lb_mbox_style)
参数	id：mbox 控件的 ID 号；type：风格类型，参考 lb_eui_mbox_style_e； lb_mbox_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_set_text

功能	设置 mbox 文本。
函数	lb_int32 lb_eui_mbox_ext_set_text(void *mbox, const char *txt)
参数	mbox：mbox 控件句柄；txt：mbox 新的文本。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_set_btns

功能	设置 mbox 选项按钮。
函数	lb_int32 lb_eui_mbox_ext_set_btns(void *mbox, const char **btn_map,lb_eui_btnm_action_t action)
参数	mbox：mbox 控件句柄；btn_map：字符串数组，action：事件回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_set_anim_time

功能	设置 mbox 的动画时间。
函数	lb_int32 lb_eui_mbox_ext_set_anim_time(void *mbox, lb_uint16 anim_time)
参数	mbox：mbox 控件句柄；anim_time：mbox 的动画时间，单位为 ms,0 为无动画。
返回值	0 表示成功，其他负值表示错误码。

Recolor 是值解析文本中自带的颜色字符串。颜色字符串以#开头，以#结尾，使用 16 进制数表示， 列如 char string[16] = {“#FF0000 red#”}，显示的是红色的 red 字符串。

lb_eui_mbox_ext_set_recolor

功能	设置 mbox 是否支持 recolor。
函数	lb_int32 lb_eui_mbox_ext_set_recolor(void *mbox, bool en)
参数	mbox：mbox 控件句柄；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_btnm_toggle

功能	设置 mbox 按键切换状态。
函数	lb_int32 lb_eui_mbox_ext_btnm_toggle(void *mbox, lb_uint16 toggle_id)
参数	mbox：mbox 控件句柄；toggle_id：切换状态的按键索引值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_auto_close

功能	设置 mbox 自动隐藏。
函数	lb_int32 lb_eui_mbox_ext_auto_close(void *mbox, lb_uint16 delay, lb_eui_anim_cb_t cb)
参数	mbox：mbox 控件句柄；delay：延时时间（ms）；cb：隐藏回调函数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_set_style

功能	设置 mbox 的风格。
函数	lb_int32 lb_eui_mbox_set_style(void *mbox, lb_eui_mbox_style_e type, lb_eui_style_t *lb_mbox_style)
参数	mbox：mbox 控件句柄；type：风格类型，参考 lb_eui_mbox_style_e；lb_mbox_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_mbox_ext_get_text

功能	获取 mbox 的文本内容。
函数	lb_int32 lb_eui_mbox_ext_get_text(void *mbox, char **txt)
参数	mbox：mbox 控件句柄；txt：返回 mbox 文本内容地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_get_anim_time

功能	获取 mbox 的动画时间。
函数	lb_int32 lb_eui_mbox_ext_get_anim_time(void *mbox, lb_uint16 *anim_time)
参数	mbox：mbox 控件句柄；anim_time：返回 mbox 的动画时间，单位为 ms,0 为无动画。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_get_recolor

功能	获取 mbox 是否支持 recolor。
函数	lb_int32 lb_eui_mbox_ext_get_recolor(void *mbox, bool *en)
参数	mbox：mbox 控件句柄；en：true 表示支持 recolor，false 表示不支持 recolor。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_mbox_ext_get_style

功能	获取 mbox 的风格。
函数	lb_int32 lb_eui_mbox_ext_get_style(void *mbox, lb_eui_mbox_style_e type, lb_eui_style_t *lb_mbox_style)
参数	mbox：mbox 控件句柄；type：风格类型，参考 lb_eui_mbox_style_e； lb_mbox_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.22. Page 控件

page 控件的 API 接口定义如下：

scrollbar modes 定义：

```
typedef enum {
    LB_EUI_SB_MODE_OFF    = 0x0,        /*Never show scrollbars*/
    LB_EUI_SB_MODE_ON     = 0x1,        /*Always show scrollbars*/
    LB_EUI_SB_MODE_DRAG = 0x2,          /*Show scrollbars when page is being dragged*/
    LB_EUI_SB_MODE_AUTO = 0x3,
    /*Show scrollbars when the scrollable container is large enough to be scrolled*/
    LB_EUI_SB_MODE_HIDE = 0x4,          /*Hide the scroll bar temporally*/
    LB_EUI_SB_MODE_UNHIDE = 0x5,
    /*Unhide the previously hidden scrollbar. Recover it's type too*/
}lb_eui_sb_mode_e;

typedef enum {
    LB_EUI_PAGE_STYLE_BG,
    LB_EUI_PAGE_STYLE_SCRL,
    LB_EUI_PAGE_STYLE_SB,
    LB_EUI_PAGE_STYLE_EDGE_FLASH
} lb_eui_page_style_e;
```

Setter functions

lb_eui_page_set_sb_mode

功能	设置 page scroll bar 模式。
函数	lb_int32 lb_eui_page_set_sb_mode(lb_int32 id, lb_eui_sb_mode_e sb_mode)
参数	id：page 控件的 ID 号；sb_mode：page scrool bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_set_arrow_scroll

功能	设置 page scroll bar 滑动的时候是否支持显示箭头。
函数	lb_int32 lb_eui_page_set_arrow_scroll(lb_int32 id, bool en)
参数	id：page 控件的 ID 号；en：true 表示支持显示箭头，false 表示不支持显示箭头。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_set_scroll_propagation

功能	设置 page scroll 传播属性，如果启用传播属性，page 将会移动其父控件，当没控件滚动时。
函数	lb_int32 lb_eui_page_set_scroll_propagation(lb_int32 id, bool en)
参数	id：page 控件的 ID 号；en：true 表示启用滚动传播，false 表示不启用。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_set_edge_flash

功能	设置 page 是否支持闪现圆弧，当到达 edge 时。
函数	lb_int32 lb_eui_page_set_edge_flash(lb_int32 id, bool en)
参数	id：page 控件的 ID 号；en：true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_set_scrl_fit

功能	设置 page 可滚动部分的 fit 属性。
函数	lb_int32 lb_eui_page_set_scrl_fit(lb_int32 id, bool hor_en, bool ver_en)
参数	id：page 控件的 ID 号；hor_en：水平方向 fit 属性，true 表示支持，false 表示不支持；ver_en：垂直方向 fit 属性，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_set_scrl_size

功能	设置 page 可滚动部分的宽度与高度。
函数	lb_int32 lb_eui_page_set_scrl_size(lb_uint32 id, lb_int32 w, lb_int32 h)
参数	id：page 控件的 ID 号；w：page 可滚动部分宽度；h：page 可滚动部分高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_set_scrl_layout

Page 布局方式定义见 lb_eui_layout_e；

功能	设置 page 可滚动部分布局方式。
函数	lb_int32 lb_eui_page_set_scrl_layout(lb_int32 id, lb_eui_layout_e layout)
参数	id：page 控件的 ID 号；layout：page 可滚动部分的布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_set_style

功能	设置 page 的风格。
函数	lb_int32 lb_eui_page_set_style(lb_int32 id, lb_eui_page_style_e type, lb_eui_style_t *lb_page_style)
参数	id：page 控件的 ID 号；type：风格类型，参考 lb_eui_page_style_e； lb_page_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Setter functions

lb_eui_page_get_sb_mode

功能	获取 page scroll bar 模式。
函数	lb_int32 lb_eui_page_get_sb_mode(lb_int32 id, lb_uint8 *sb_mode)
参数	id：page 控件的 ID 号；sb_mode：返回 page scroll bar 模式状态。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_get_arrow_scroll

功能	获取 page scroll bar 滑动的时候是否支持显示箭头状态。
函数	lb_int32 lb_eui_page_get_arrow_scroll(lb_int32 id, bool *en)
参数	id：page 控件的 ID 号；en：返回 page scroll bar 滑动的时候是否支持显示箭头状态，true 表示支持显示箭头，false 表示不支持显示箭头。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_get_scroll_propagation

功能	获取 page scroll 传播属性状态。
函数	lb_int32 lb_eui_page_get_scroll_propagation(lb_int32 id, bool *en)
参数	id：page 控件的 ID 号；en：返回 page scroll 传播属性状态，true 表示启用传播，false 表示不启用。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_get_edge_flash

功能	获取 page 是否支持闪现圆弧状态。
函数	lb_int32 lb_eui_page_get_edge_flash(lb_int32 id, bool *en)
参数	id：page 控件的 ID 号；en：返回 page 是否支持闪现圆弧状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_get_scrl_size

功能	获取 page 可滚动部分的宽度与高度。
函数	lb_int32 lb_eui_page_get_scrl_size(lb_uint32 id, lb_int32 *w, lb_int32 *h)
参数	id：page 控件的 ID 号；w：返回 page 可滚动部分宽度；h：返回 page 可滚动部分高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_get_scrl_layout

功能	获取 page 可滚动部分布局方式。
函数	lb_int32 lb_eui_page_get_scrl_layout(lb_int32 id, lb_uint8 *layout)
参数	id：page 控件的 ID 号；layout：返回 page 可滚动部分的布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_get_scrl_fit

功能	获取 page 可滚动部分垂直方向 fit 属性状态。
函数	lb_int32 lb_eui_page_get_scrl_fit(lb_uint32 id, bool *en_hor, bool *en_ver)
参数	id：page 控件的 ID 号；en_hor：返回 page 可滚动部分水平方向 fit 属性状态，true 表示支持，false 表示不支持；en_ver：返回 page 可滚动部分垂直方向 fit 属性状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_get_style

功能	获取 page 的风格。

函数	lb_int32 lb_eui_page_get_style(lb_int32 id, lb_eui_page_style_e type, lb_eui_style_t *lb_page_style)
参数	id：page 控件的 ID 号；type：风格类型，参考 lb_eui_page_style_e； lb_page_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

Other functions

lb_eui_page_scroll_hor

功能	page 滚动条水平移动。
函数	lb_int32 lb_eui_page_scroll_hor(lb_uint32 id, lb_int32 dist)
参数	id：page 控件的 ID 号；dist：水平移动距离，dist<0，向右移动，dist>0，向左移动。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_scroll_ver

功能	page 滚动条垂直移动。
函数	lb_int32 lb_eui_page_scroll_ver(lb_uint32 id, lb_int32 dist)
参数	id：page 控件的 ID 号；dist：水平移动距离，dist<0，向下移动，dist>0，向上移动。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_sb_mode

功能	设置 page scroll bar 模式。
函数	lb_int32 lb_eui_page_ext_set_sb_mode(void *page, lb_eui_sb_mode_e sb_mode)
参数	page：page 控件句柄；sb_mode：page scrool bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_arrow_scroll

功能	设置 page scroll bar 滑动的时候是否支持显示箭头。
函数	lb_int32 lb_eui_page_ext_set_arrow_scroll(void *page, bool en)
参数	page：page 控件句柄；en：true 表示支持显示箭头，false 表示不支持显示箭头。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_scroll_propagation

功能	设置 page scroll 传播属性，如果启用传播属性，page 将会移动其父控件，当没控件滚动时。
函数	lb_int32 lb_eui_page_ext_set_scroll_propagation(void *page, bool en)
参数	page：page 控件句柄；en：true 表示启用滚动传播，false 表示不启用。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_edge_flash

功能	设置 page 是否支持闪现圆弧，当到达 edge 时。
函数	lb_int32 lb_eui_page_ext_set_edge_flash(void *page, bool en)

参数	page：page 控件句柄；en：true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_scrl_fit

功能	设置 page 可滚动部分的 fit 属性。
函数	lb_int32 lb_eui_page_ext_set_scrl_fit(void *page, bool hor_en, bool ver_en)
参数	page：page 控件句柄；hor_en：水平方向 fit 属性，true 表示支持，false 表示不支持；ver_en：垂直方向 fit 属性，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_scrl_size

功能	设置 page 可滚动部分的宽度与高度。
函数	lb_int32 lb_eui_page_ext_set_scrl_size(void *page, lb_int32 w, lb_int32 h)
参数	page：page 控件句柄；w：page 可滚动部分宽度；h：page 可滚动部分高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_scrl_layout

Page 布局方式定义见 lb_eui_layout_e；

功能	设置 page 可滚动部分布局方式。
函数	lb_int32 lb_eui_page_ext_set_scrl_layout(void *page, lb_eui_layout_e layout)
参数	page：page 控件句柄；layout：page 可滚动部分的布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_set_style

功能	设置 page 的风格。
函数	lb_int32 lb_eui_page_ext_set_style(void *page, lb_eui_page_style_e type, lb_eui_style_t *lb_page_style)
参数	page：page 控件句柄；type：风格类型，参考 lb_eui_page_style_e；lb_page_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Setter functions

lb_eui_page_ext_get_sb_mode

功能	获取 page scroll bar 模式。
函数	lb_int32 lb_eui_page_ext_get_sb_mode(void *page, lb_uint8 *sb_mode)
参数	page：page 控件句柄；sb_mode：返回 page scroll bar 模式状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_get_arrow_scroll

功能	获取 page scroll bar 滑动的时候是否支持显示箭头状态。
函数	lb_int32 lb_eui_page_ext_get_arrow_scroll(void *page, bool *en)
参数	page：page 控件句柄；en：返回 page scroll bar 滑动的时候是否支持显示箭头状态，true 表示支持显示箭头，false 表示不支持显示箭头。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_get_scroll_propagation

功能	获取 page scroll 传播属性状态。
函数	lb_int32 lb_eui_page_ext_get_scroll_propagation(void *page, bool *en)
参数	page：page 控件句柄；en：返回 page scroll 传播属性状态，true 表示启用传播，false 表示不启用。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_get_edge_flash

功能	获取 page 是否支持闪现圆弧状态。
函数	lb_int32 lb_eui_page_ext_get_edge_flash(void *page, bool *en)
参数	page：page 控件句柄；en：返回 page 是否支持闪现圆弧状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_get_scrl_size

功能	获取 page 可滚动部分的宽度与高度。
函数	lb_int32 lb_eui_page_ext_get_scrl_size(void *page, lb_int32 *w, lb_int32 *h)
参数	page：page 控件句柄；w：返回 page 可滚动部分宽度；h：返回 page 可滚动部分高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_get_scrl_layout

功能	获取 page 可滚动部分布局方式。
函数	lb_int32 lb_eui_page_ext_get_scrl_layout(void *page, lb_uint8 *layout)
参数	page：page 控件句柄；layout：返回 page 可滚动部分的布局方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_get_scrl_ fit

功能	获取 page 可滚动部分水平方向 fit 属性状态。
函数	lb_int32 lb_eui_page_ext_get_scrl_fit(void *page, bool *en_hor, bool *en_ver)
参数	page：page 控件句柄；en_hor：返回 page 可滚动部分水平方向 fit 属性状态，true 表示支持，false 表示不支持；en_ver：返回 page 可滚动部分垂直方向 fit 属性状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_get_style

功能	获取 page 的风格。
函数	lb_int32 lb_eui_page_ext_get_style(void *page, lb_eui_page_style_e type, lb_eui_style_t *lb_page_style)
参数	page：page 控件句柄；type：风格类型，参考 lb_eui_page_style_e； lb_page_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

Other functions

lb_eui_page_ext_scroll_hor

功能	page 滚动条水平移动。
函数	lb_int32 lb_eui_page_ext_scroll_hor(void *page, lb_int32 dist)
参数	page：page 控件句柄；dist：水平移动距离，dist<0，向右移动，dist>0，向左移动。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_page_ext_scroll_ver

功能	page 滚动条垂直移动。
函数	lb_int32 lb_eui_page_ext_scroll_ver(void *page, lb_int32 dist)
参数	page：page 控件句柄；dist：水平移动距离，dist<0，向下移动，dist>0，向上移动。
返回值	0 表示成功，其他负值表示错误码。

4.5.23. Preload 控件

```
typedef enum {
    LB_EUI_PRELOAD_TYPE_SPINNING_ARC,
    LB_EUI_PRELOAD_TYPE_FILLSPIN_ARC,
}lb_eui_preloader_type_e;

typedef enum {
    LB_EUI_PRELOAD_STYLE_MAIN,
} lb_eui_preload_style_e;
```

preload 控件的 API 接口定义如下：

Setter functions

lb_eui_preload_set_arc_length

功能	设置 preload 旋转弧的长度，单位为度（0-360）。
函数	lb_int32 lb_eui_preload_set_arc_length(lb_int32 id, lb_uint16 deg)
参数	id：preload 控件的 ID 号；deg：preload 旋转弧的长度，单位为度（0-360）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_set_spin_time

功能	设置 preload 旋转弧旋转一周时间，单位为 ms。
函数	lb_int32 lb_eui_preload_set_spin_time(lb_int32 id, lb_uint16 time)
参数	id：preload 控件的 ID 号；time：preload 旋转弧旋转一周时间，单位为 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_set_animation_type

功能	设置 preload 动画类型。

函数	lb_int32 lb_eui_preload_set_animation_type(lb_int32 id, lb_eui_preloader_type_e type)
参数	id：preload 控件的 ID 号；type：动画类型，参考 lb_eui_preloader_type_e。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_set_style

功能	设置 preload 的风格。
函数	lb_int32 lb_eui_preload_set_style(lb_int32 id, lb_eui_preload_style_e type, lb_eui_style_t *lb_preload_style)
参数	id：preload 控件的 ID 号；type：风格类型，参考 lb_eui_preload_style_e； lb_preload_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

/* Getter function */

lb_eui_preload_get_arc_length

功能	获取 preload 旋转弧的长度，单位为度（0-360）。
函数	lb_int32 lb_eui_preload_get_arc_length(lb_int32 id, lb_uint16 *deg)
参数	id：preload 控件的 ID 号；deg：返回 preload 旋转弧的长度，单位为度（0-360）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_get_spin_time

功能	获取 preload 旋转弧旋转一周时间，单位为 ms。
函数	lb_int32 lb_eui_preload_get_spin_time(lb_int32 id, lb_uint16 *time)
参数	id：preload 控件的 ID 号；time：preload 旋转弧旋转一周时间，单位为 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_get_animation_type

功能	获取 preload 动画类型。
函数	lb_int32 lb_eui_preload_get_animation_type(lb_int32 id, lb_uint8 *type)
参数	id：preload 控件的 ID 号；type：返回 preload 动画类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_get_style

功能	获取 preload 的风格。
函数	lb_int32 lb_eui_preload_get_style(lb_int32 id, lb_eui_preload_style_e type, lb_eui_style_t *lb_preload_style)
参数	id：preload 控件的 ID 号；type：风格类型，参考 lb_eui_preload_style_e； lb_preload_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_ext_set_arc_length

功能	设置 preload 旋转弧的长度，单位为度（0-360）。
函数	lb_int32 lb_eui_preload_ext_set_arc_length(void *preload, lb_uint16 deg)
参数	preload：preload 控件句柄；deg：preload 旋转弧的长度，单位为度（0-360）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_ext_set_spin_time

功能	设置 preload 旋转弧旋转一周时间，单位为 ms。
函数	lb_int32 lb_eui_preload_ext_set_spin_time(void *preload, lb_uint16 time)
参数	preload：preload 控件句柄；time：preload 旋转弧旋转一周时间，单位为 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_ext_set_animation_type

功能	设置 preload 动画类型。
函数	lb_int32 lb_eui_preload_ext_set_animation_type(void *preload, lb_eui_preloader_type_e type)
参数	preload：preload 控件句柄；type：动画类型，参考 lb_eui_preloader_type_e。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_ext_set_style

功能	设置 preload 的风格。
函数	lb_int32 lb_eui_preload_ext_set_style(void *preload, lb_eui_preload_style_e type, lb_eui_style_t *lb_preload_style)
参数	preload：preload 控件句柄；type：风格类型，参考 lb_eui_preload_style_e； lb_preload_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

/* Getter function */

lb_eui_preload_ext_get_arc_length

功能	获取 preload 旋转弧的长度，单位为度（0-360）。
函数	lb_int32 lb_eui_preload_ext_get_arc_length(void *preload, lb_uint16 *deg)
参数	preload：preload 控件句柄；deg：返回 preload 旋转弧的长度，单位为度（0-360）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_ext_get_spin_time

功能	获取 preload 旋转弧旋转一周时间，单位为 ms。
函数	lb_int32 lb_eui_preload_ext_get_spin_time(void *preload, lb_uint16 *time)
参数	preload：preload 控件句柄；time：preload 旋转弧旋转一周时间，单位为 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_ext_get_animation_type

功能	获取 preload 动画类型。
函数	lb_int32 lb_eui_preload_ext_get_animation_type(void *preload, lb_uint8 *type)
参数	preload：preload 控件句柄；type：返回 preload 动画类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_preload_ext_get_style

功能	获取 preload 的风格。
函数	lb_int32 lb_eui_preload_ext_get_style(void *preload, lb_eui_preload_style_e type, lb_eui_style_t *lb_preload_style)
参数	preload：preload 控件句柄；type：风格类型，参考 lb_eui_preload_style_e； lb_preload_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.24. Roller 控件

```
typedef enum {
    LB_EUI_ROLLER_STYLE_BG,
    LB_EUI_ROLLER_STYLE_SCRL,
    LB_EUI_ROLLER_STYLE_SEL,
} lb_eui_roller_style_e;

typedef struct tag_lb_eui_roller_msg_data {
    lb_uint32 roller_id;
    lb_uint32 sel_opt_id;
    lb_uint32 sel_opt_id_ori;
} lb_eui_roller_msg_data_t;
```

roller 控件的 API 接口定义如下：

Roller'options 对齐方式定义见 label 控件 lb_eui_label_align_e。

Setter functions

lb_eui_roller_set_align

功能	设置 Roller'options 对齐方式。
函数	lb_int32 lb_eui_roller_set_align(lb_int32 id, lb_eui_label_align_e align)
参数	id：roller 控件的 ID 号；align：Roller'options 对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_set_options

功能	设置 Roller options。

函数	lb_int32 lb_eui_roller_set_options(lb_int32 id, const char *options)
参数	id：roller 控件的 ID 号；options：options 字符，例如"One\nTwo\nThree"，用\n 分隔。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_set_options_id

功能	设置 Roller options id 数组。
函数	lb_int32 lb_eui_roller_set_options_id(lb_uint32 id, lb_uint32 *options, lb_uint16 num)
参数	id：roller 控件的 ID 号；options：options 字符串索引值数组；num：字符个数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_set_selected

功能	设置 Roller 选中项。
函数	lb_int32 lb_eui_roller_set_selected(lb_int32 id, lb_uint16 sel_opt, bool anim_en)
参数	id：roller 控件的 ID 号；sel_opt：roller 选中项的 id；anim_en：roller 是否支持动画，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_set_visible_row_count

功能	设置 Roller options 可显示高度（option 行数）。
函数	lb_int32 lb_eui_roller_set_visible_row_count(lb_int32 id, lb_uint8 row_cnt)
参数	id：roller 控件的 ID 号；row_cnt：options 可显示高度（行数）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_set_hor_fit

功能	设置 Roller 背景水平方向 fit 状态。
函数	lb_int32 lb_eui_roller_set_hor_fit(lb_int32 id, bool en)
参数	id：roller 控件的 ID 号；en：roller 背景水平方向 fit 状态，true 表示支持水平 fit，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_set_anim_time

功能	设置 Roller 打开/关闭动画时间。
函数	lb_int32 lb_eui_roller_set_anim_time(lb_int32 id, lb_uint16 anim_time)
参数	id：roller 控件的 ID 号；anim_time：roller 打开/关闭动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_set_style

功能	设置 Roller 的风格。
函数	lb_int32 lb_eui_roller_set_style(lb_int32 id, lb_eui_roller_style_e type,

	lb_eui_style_t *lb_roller_style)
参数	id：roller 控件的 ID 号；type：风格类型，参考 lb_eui_roller_style_e； lb_roller_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_roller_get_align

功能	获取 Roller'options 对齐方式。
函数	lb_int32 lb_eui_roller_get_align(lb_int32 id, lb_uint8 *align)
参数	id：roller 控件的 ID 号；align：返回 Roller'options 对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_options

功能	获取 Roller options。
函数	lb_int32 lb_eui_roller_get_options(lb_int32 id, char **options)
参数	id：roller 控件的 ID 号；options：返回 options 字符串地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_options_num

功能	获取 Roller options 的数量。
函数	lb_int32 lb_eui_roller_get_options_num(lb_uint32 id, lb_uint16 *opt_num)
参数	id：roller 控件的 ID 号；opt_num：options 的数量。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_selected

功能	获取 Roller 当前选中项 ID。
函数	lb_int32 lb_eui_roller_get_selected(lb_int32 id, lb_uint16 *sel_opt)
参数	id：roller 控件的 ID 号；sel_opt：返回 roller 选中项的 id。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_selected_str

功能	获取 Roller 当前选中项字符。
函数	lb_int32 lb_eui_roller_get_selected_str(lb_int32 id, char **buf)
参数	id：roller 控件的 ID 号；sel_opt：返回 roller 选中项的字符地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_hor_fit

功能	获取 Roller 背景水平方向 fit 状态。
函数	lb_int32 lb_eui_roller_get_hor_fit(lb_int32 id, bool *en)
参数	id：roller 控件的 ID 号；en：返回 roller 背景水平方向 fit 状态，true 表示支持，false 表示不支持。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_anim_time

功能	获取 Roller 打开/关闭动画时间。
函数	lb_int32 lb_eui_roller_get_anim_time(lb_int32 id, lb_uint16 *anim_time)
参数	id：roller 控件的 ID 号；anim_time：返回 roller 打开/关闭动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_style

功能	获取 Roller 的风格。
函数	lb_int32 lb_eui_roller_get_style(lb_int32 id, lb_eui_roller_style_e type,lb_eui_style_t *lb_roller_style)
参数	id：roller 控件的 ID 号；type：风格类型，参考 lb_eui_roller_style_e； lb_roller_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_align

功能	设置 Roller'options 对齐方式。
函数	lb_int32 lb_eui_roller_ext_set_align(void *roller, lb_eui_label_align_e align)
参数	roller：roller 控件句柄；align：Roller'options 对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_options

功能	设置 Roller options。
函数	lb_int32 lb_eui_roller_ext_set_options(void *roller, const char *options)
参数	roller：roller 控件句柄；options：options 字符，例如"One\nTwo\nThree"，用\n 分隔。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_options_id

功能	设置 Roller options id 数组。
函数	lb_int32 lb_eui_roller_ext_set_options_id(void *roller, lb_uint32 *options, lb_uint16 num)
参数	roller：roller 控件句柄；options：options 字符串索引值数组；num：字符个数。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_selected

功能	设置 Roller 选中项。
函数	lb_int32 lb_eui_roller_ext_set_selected(void *roller, lb_uint16 sel_opt, bool anim_en)
参数	roller：roller 控件句柄；sel_opt：roller 选中项的 id；anim_en：roller 是否支持动画，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_visible_row_count

功能	设置 Roller options 可显示高度（option 行数）。
函数	lb_int32 lb_eui_roller_ext_set_visible_row_count(void *roller, lb_uint8 row_cnt)

参数	roller：roller 控件句柄；row_cnt：options 可显示高度（行数）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_hor_fit

功能	设置 Roller 背景水平方向 fit 状态。
函数	lb_int32 lb_eui_roller_ext_set_hor_fit(void *roller, bool en)
参数	roller：roller 控件句柄；en：roller 背景水平方向 fit 状态，true 表示支持水平 fit，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_anim_time

功能	设置 Roller 打开/关闭动画时间。
函数	lb_int32 lb_eui_roller_ext_set_anim_time(void *roller, lb_uint16 anim_time)
参数	roller：roller 控件句柄；anim_time：roller 打开/关闭动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_set_style

功能	设置 Roller 的风格。
函数	lb_int32 lb_eui_roller_set_style(void *roller, lb_eui_roller_style_e type,lb_eui_style_t *lb_roller_style)
参数	roller：roller 控件句柄；type：风格类型，参考 lb_eui_roller_style_e； lb_roller_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_roller_ext_get_align

功能	获取 Roller'options 对齐方式。
函数	lb_int32 lb_eui_roller_ext_get_align(void *roller, lb_uint8 *align)
参数	roller：roller 控件句柄；align：返回 Roller'options 对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_get_options

功能	获取 Roller options。
函数	lb_int32 lb_eui_roller_ext_get_options(void *roller, char **options)
参数	roller：roller 控件句柄；options：返回 options 字符串地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_get_options_num

功能	获取 Roller options 的数量。
函数	lb_int32 lb_eui_roller_ext_get_options_num(void *roler, lb_uint16 *opt_num)
参数	roller：roller 控件句柄；opt_num：options 的数量。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_get_selected

功能	获取 Roller 当前选中项 ID。
函数	lb_int32 lb_eui_roller_ext_get_selected(void *roller, lb_uint16 *sel_opt)
参数	roller：roller 控件句柄；sel_opt：返回 roller 选中项的 id。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_get_selected_str

功能	获取 Roller 当前选中项字符。
函数	lb_int32 lb_eui_roller_ext_get_selected_str(void *roller, char **buf)
参数	roller：roller 控件句柄；sel_opt：返回 roller 选中项的字符地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_get_hor_fit

功能	获取 Roller 背景水平方向 fit 状态。
函数	lb_int32 lb_eui_roller_ext_get_hor_fit(void *roller, bool *en)
参数	roller：roller 控件句柄；en：返回 roller 背景水平方向 fit 状态，true 表示支持，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_get_anim_time

功能	获取 Roller 打开/关闭动画时间。
函数	lb_int32 lb_eui_roller_ext_get_anim_time(void *roller, lb_uint16 *anim_time)
参数	roller：roller 控件句柄；anim_time：返回 roller 打开/关闭动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_roller_ext_get_style

功能	获取 Roller 的风格。
函数	lb_int32 lb_eui_roller_ext_get_style(void *roller, lb_eui_roller_style_e type,lb_eui_style_t *lb_roller_style)
参数	roller：roller 控件句柄；type：风格类型，参考 lb_eui_roller_style_e； lb_roller_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.25. Slider 控件

```
typedef enum {
    LB_EUI_SLIDER_STYLE_BG,
    LB_EUI_SLIDER_STYLE_INDIC,
    LB_EUI_SLIDER_STYLE_KNOB,
} lb_eui_slider_style_e;
```

```
typedef struct tag_lb_eui_roller_msg_data {
    lb_uint32 roller_id;
    lb_uint32 sel_opt_id;
    lb_uint32 sel_opt_id_ori;
} lb_eui_roller_msg_data_t;
```

Slider 控件的 API 接口定义如下：

Setter functions

lb_eui_slider_set_value

功能	设置 slider 新数值（value）。
函数	lb_int32 lb_eui_slider_set_value(lb_int32 id, lb_int16 value)
参数	id：slider 控件的 ID 号；value：slider 新的数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_set_value_anim

功能	设置 slider 新数值（value）和动画时间。
函数	lb_int32 lb_eui_slider_set_value_anim(lb_int32 id, lb_int16 value, lv_uint16 anim_time)
参数	id：slider 控件的 ID 号；value：slider 新的数值；anim_time：动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_set_range

功能	设置 slider bar 数值范围。
函数	lb_int32 lb_eui_slider_set_range(lb_int32 id, lb_int16 min, lv_int16 max)
参数	id：slider 控件的 ID 号；min：slider bar 范围最小值；max：slider bar 范围最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_set_knob_in

功能	设置 slider knob 属性。
函数	lb_int32 lb_eui_slider_set_knob_in(lb_int32 id, bool in)
参数	id：slider 控件的 ID 号；in：true 表示 knob 显示在 slider 里，false 表示显示超出 slider 边沿。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_set_style

功能	设置 slider 风格属性。
函数	lb_int32 lb_eui_slider_set_style(lb_uint32 id, lb_eui_slider_style_e type, lb_eui_style_t *lb_slider_style)
参数	id：slider 控件的 ID 号；type：风格类型，参考 lb_eui_slider_style_e；lb_slider_style：要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_slider_get_value

功能	获取 slider 当前数值（value）。
函数	lb_int32 lb_eui_slider_get_value(lb_int32 id, lb_int16 *value)
参数	id：slider 控件的 ID 号；value：返回 slider 当前数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_get_min_value

功能	获取 slider bar 最小值。
函数	lb_int32 lb_eui_slider_get_min_value(lb_int32 id, lb_int16 *min)
参数	id：slider 控件的 ID 号；min：返回 slider bar 最小值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_get_max_value

功能	获取 slider bar 最大值。
函数	lb_int32 lb_eui_slider_get_max_value(lb_int32 id, lb_int16 *max)
参数	id：slider 控件的 ID 号；min：返回 slider bar 最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_get_knob_in

功能	获取 slider knob 属性。
函数	lb_int32 lb_eui_slider_get_knob_in(lb_int32 id, bool *in)
参数	id：slider 控件的 ID 号；in：返回 slider knob 属性状态，true 表示 knob 显示在 slider 里，false 表示显示超出 slider 边沿。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_is_dragged

功能	获取 slider knob 拖拽状态。
函数	lb_int32 lb_eui_slider_is_dragged(lb_int32 id, bool *drag)
参数	id：slider 控件的 ID 号；drag：返回 slider knob 状态，true 表示 knob 正在拖拽，false 表示未拖拽。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_get_style

功能	获取 slider 风格属性。
函数	lb_int32 lb_eui_slider_get_style(lb_uint32 id, lb_eui_slider_style_e type, lb_eui_style_t *lb_slider_style)
参数	id：slider 控件的 ID 号；type：风格类型，参考 lb_eui_slider_style_e；lb_slider_style：获取的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_set_value

功能	设置 slider 新数值（value）。
函数	lb_int32 lb_eui_slider_ext_set_value(void *slider, lb_int16 value)
参数	slider：slider 控件句柄；value：slider 新的数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_set_value_anim

功能	设置 slider 新数值（value）和动画时间。
函数	lb_int32 lb_eui_slider_ext_set_value_anim(void *slider, lb_int16 value, lb_uint16 anim_time)
参数	slider：slider 控件句柄；value：slider 新的数值；anim_time：动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_set_range

功能	设置 slider bar 数值范围。
函数	lb_int32 lb_eui_slider_ext_set_range(void *slider, lb_int16 min, lv_int16 max)
参数	slider：slider 控件句柄；min：slider bar 范围最小值；max：slider bar 范围最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_set_knob_in

功能	设置 slider knob 属性。
函数	lb_int32 lb_eui_slider_ext_set_knob_in(void *slider, bool in)
参数	slider：slider 控件句柄；in：true 表示 knob 显示在 slider 里，false 表示显示超出 slider 边沿。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_set_style

功能	设置 slider 风格属性。
函数	lb_int32 lb_eui_slider_ext_set_style(void *slider, lb_eui_slider_style_e type, lb_eui_style_t *lb_slider_style)
参数	slider：slider 控件句柄；type：风格类型，参考 lb_eui_slider_style_e；lb_slider_style：要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_slider_ext_get_value

功能	获取 slider 当前数值（value）。
函数	lb_int32 lb_eui_slider_ext_get_value(void *slider, lb_int16 *value)
参数	slider：slider 控件句柄；value：返回 slider 当前数值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_get_min_value

功能	获取 slider bar 最小值。
函数	lb_int32 lb_eui_slider_ext_get_min_value(void *slider, lb_int16 *min)
参数	slider：slider 控件句柄；min：返回 slider bar 最小值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_get_max_value

功能	获取 slider bar 最大值。
函数	lb_int32 lb_eui_slider_ext_get_max_value(void *slider, lb_int16 *max)
参数	slider：slider 控件句柄；min：返回 slider bar 最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_get_knob_in

功能	获取 slider knob 属性。
函数	lb_int32 lb_eui_slider_ext_get_knob_in(void *slider, bool *in)
参数	slider：slider 控件句柄；in：返回 slider knob 属性状态，true 表示 knob 显示在 slider 里，false 表示显示超出 slider 边沿。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_is_dragged

功能	获取 slider knob 拖拽状态。
函数	lb_int32 lb_eui_slider_ext_is_dragged(void *slider, bool *drag)
参数	slider：slider 控件句柄；drag：返回 slider knob 状态，true 表示 knob 正在拖拽，false 表示未拖拽。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_slider_ext_get_style

功能	获取 slider 风格属性。
函数	lb_int32 lb_eui_slider_ext_get_style(void *slider, lb_eui_slider_style_e type,lb_eui_style_t *lb_slider_style)
参数	slider：slider 控件句柄；type：风格类型，参考 lb_eui_slider_style_e；lb_slider_style：获取的风格的指针。
返回值	0 表示成功，其他负值表示错误码。

4.5.26. Spinbox 控件

```
typedef enum {
    LB_EUI_SPINBOX_STYLE_BG,
    LB_EUI_SPINBOX_STYLE_SCRL,
    LB_EUI_SPINBOX_STYLE_SB,
    LB_EUI_SPINBOX_STYLE_CURSOR,
```

} lb_eui_spinbox_style_e;

Spinbox 控件的 API 接口定义如下：

Setter functions

lb_eui_spinbox_set_value

功能	设置 spinbox 的值。
函数	lb_int32 lb_eui_spinbox_set_value(lb_uint32 id, lb_int32 value)
参数	id：spinbox 控件的 ID 号；value：被设置的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_set_digit_format

功能	设置 spinbox 最大的输入数。
函数	lb_int32 lb_eui_spinbox_set_digit_format(lb_uint32 id, lb_uint8 digit_count, lb_uint8 separator_pos)
参数	id：spinbox 控件的 ID 号；digit_count：输入值的最大位数（小数点和符号除外）；separator_pos：小数点的位置。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_set_step

功能	设置 spinbox 的步进值。
函数	lb_int32 lb_eui_spinbox_set_step(lb_uint32 id, lb_int32 step)
参数	id：spinbox 控件的 ID 号；step：步进值，正数为增，负数为减。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_set_range

功能	设置 spinbox 值的范围。
函数	lb_int32 lb_eui_spinbox_set_range(lb_uint32 id, lb_int32 min, lb_int32 max)
参数	id：spinbox 控件的 ID 号；min：最小值；max：最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_set_padding_left

功能	设置 spinbox 数字计数中的符号与数字的距离（位数）。
函数	lb_int32 lb_eui_spinbox_set_padding_left(lb_uint32 id, lb_uint8 padding)
参数	id：spinbox 控件的 ID 号；padding：数字计数中的符号与数字的距离（位数）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_set_style

功能	设置 spinbox 的风格。
函数	lb_int32 lb_eui_spinbox_set_style(lb_uint32 id, lb_eui_spinbox_style_e type, lb_eui_style_t *

	lb_spinbox_style)
参数	id：spinbox 控件的 ID 号；type：风格类型，参考 lb_eui_spinbox_style_e； lb_spinbox_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_get_value

功能	获取 spinbox 的值。
函数	lb_int32 lb_eui_spinbox_get_value(lb_uint32 id, lb_int32 *value)
参数	id：spinbox 控件的 ID 号；value：spinbox 的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_get_style

功能	设置 spinbox 的风格。
函数	lb_int32 lb_eui_spinbox_get_style(lb_uint32 id, lb_eui_spinbox_style_e type, lb_eui_style_t *lb_spinbox_style)
参数	id：spinbox 控件的 ID 号；type：风格类型，参考 lb_eui_spinbox_style_e； lb_spinbox_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_step_next

功能	设置 spinbox 的步进值为原来的十分之一。
函数	lb_int32 lb_eui_spinbox_step_next(lb_uint32 id)
参数	id：spinbox 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_step_previous

功能	设置 spinbox 的步进值为原来的十倍。
函数	lb_int32 lb_eui_spinbox_step_previous(lb_uint32 id)
参数	id：spinbox 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_increment

功能	增加 spinbox 的值，加一个步进值大小。
函数	lb_int32 lb_eui_spinbox_increment(lb_uint32 id)
参数	id：spinbox 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_decrement

功能	减少 spinbox 的值，减一个步进值大小。

函数	lb_int32 lb_eui_spinbox_decrement(lb_uint32 id)
参数	id：spinbox 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_set_value

功能	设置 spinbox 的值。
函数	lb_int32 lb_eui_spinbox_ext_set_value(void *spinbox, lb_int32 value)
参数	spinbox：spinbox 控件句柄；value：被设置的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_set_digit_format

功能	设置 spinbox 最大的输入数。
函数	lb_int32 lb_eui_spinbox_ext_set_digit_format(void *spinbox, lb_uint8 digit_count, lb_uint8 separator_pos)
参数	spinbox：spinbox 控件句柄；digit_count：输入值的最大位数（小数点和符号除外）；separator_pos：小数点的位置。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_set_step

功能	设置 spinbox 的步进值。
函数	lb_int32 lb_eui_spinbox_ext_set_step(void *spinbox, lb_int32 step)
参数	spinbox：spinbox 控件句柄；step：步进值，正数为增，负数为减。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_set_range

功能	设置 spinbox 值的范围。
函数	lb_int32 lb_eui_spinbox_ext_set_range(void *spinbox, lb_int32 min, lb_int32 max)
参数	spinbox：spinbox 控件句柄；min：最小值；max：最大值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_set_padding_left

功能	设置 spinbox 数字计数中的符号与数字的距离（位数）。
函数	lb_int32 lb_eui_spinbox_ext_set_padding_left(void *spinbox, lb_uint8 padding)
参数	spinbox：spinbox 控件句柄；padding：数字计数中的符号与数字的距离（位数）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_set_style

功能	设置 spinbox 的风格。
函数	lb_int32 lb_eui_spinbox_ext_set_style(void *spinbox, lb_eui_spinbox_style_e type, lb_eui_style_t *

	lb_spinbox_style)
参数	spinbox：spinbox 控件句柄；type：风格类型，参考 lb_eui_spinbox_style_e； lb_spinbox_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_get_value

功能	获取 spinbox 的值。
函数	lb_int32 lb_eui_spinbox_ext_get_value(void *spinbox, lb_int32 *value)
参数	spinbox：spinbox 控件句柄；value：spinbox 的值。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_get_style

功能	设置 spinbox 的风格。
函数	lb_int32 lb_eui_spinbox_ext_get_style(void *spinbox, lb_eui_spinbox_style_e type, lb_eui_style_t *lb_spinbox_style)
参数	spinbox：spinbox 控件句柄；type：风格类型，参考 lb_eui_spinbox_style_e； lb_spinbox_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_step_next

功能	设置 spinbox 的步进值为原来的十分之一。
函数	lb_int32 lb_eui_spinbox_ext_step_next(void *spinbox)
参数	spinbox：spinbox 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_step_previous

功能	设置 spinbox 的步进值为原来的十倍。
函数	lb_int32 lb_eui_spinbox_ext_step_previous(void *spinbox)
参数	spinbox：spinbox 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_increment

功能	增加 spinbox 的值，加一个步进值大小。
函数	lb_int32 lb_eui_spinbox_ext_increment(void *spinbox)
参数	spinbox：spinbox 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_spinbox_ext_decrement

功能	减少 spinbox 的值，减一个步进值大小。

函数	lb_int32 lb_eui_spinbox_ext_decrement(void *spinbox)
参数	spinbox：spinbox 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

4.5.27. Switch 控件

```
typedef enum {
    LB_EUI_SW_STYLE_BG,
    LB_EUI_SW_STYLE_INDIC,
    LB_EUI_SW_STYLE_KNOB_OFF,
    LB_EUI_SW_STYLE_KNOB_ON,
} lb_eui_sw_style_e;
```

Switch 控件的 API 接口定义如下：

Setter functions

lb_eui_sw_on

功能	打开 sw。
函数	lb_int32 lb_eui_sw_on(lb_uint32 id, bool en_anim)
参数	id：sw 控件的 ID 号；en_anim：动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_off

功能	关闭 sw。
函数	lb_int32 lb_eui_sw_off(lb_uint32 id, bool en_anim)
参数	id：sw 控件的 ID 号；en_anim：动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_toggle

功能	切换 sw 状态。
函数	lb_int32 lb_eui_sw_toggle(lb_uint32 id, bool en_anim)
参数	id：sw 控件的 ID 号；en_anim：动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_set_anim_time

功能	设置 sw 动画时间。
函数	lb_int32 lb_eui_sw_set_anim_time(lb_int32 id, lb_uint16 anim_time)
参数	id：sw 控件的 ID 号；anim_time：sw 动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_set_style

功能	设置 sw 的风格。

函数	lb_int32 lb_eui_sw_set_style(lb_int32 id, lb_eui_sw_style_e type, lb_eui_style_t *lb_sw_style)
参数	id：sw 控件的 ID 号； type：风格类型，参考 lb_eui_sw_style_e； lb_sw_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_sw_get_state

功能	获取 sw 当前状态。
函数	lb_int32 lb_eui_sw_get_state(lb_int32 id, bool *state)
参数	id：sw 控件的 ID 号； state：返回 sw 当前状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_get_anim_time

功能	获取 sw 动画时间。
函数	lb_int32 lb_eui_sw_get_anim_time(lb_int32 id, lb_uint16 *anim_time)
参数	id：sw 控件的 ID 号； anim_time：返回 sw 动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_get_style

功能	获取 sw 的风格。
函数	lb_int32 lb_eui_sw_get_style(lb_int32 id, lb_eui_sw_style_e type, lb_eui_style_t *lb_sw_style)
参数	id：sw 控件的 ID 号； type：风格类型，参考 lb_eui_sw_style_e； lb_sw_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_ext_on

功能	打开 sw。
函数	lb_int32 lb_eui_sw_ext_on(void *sw, bool en_anim)
参数	sw：sw 控件句柄；en_anim：动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_ext_off

功能	关闭 sw。
函数	lb_int32 lb_eui_sw_ext_off(void *sw, bool en_anim)
参数	sw：sw 控件句柄；en_anim：动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_ext_toggle

功能	切换 sw 状态。
函数	lb_int32 lb_eui_sw_ext_toggle(void *sw, bool en_anim)

参数	sw：sw 控件句柄；en_anim：动画属性，true 表示支持，flase 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_ext_set_anim_time

功能	设置 sw 动画时间。
函数	lb_int32 lb_eui_sw_ext_set_anim_time(void *sw, lb_uint16 anim_time)
参数	sw：sw 控件句柄；anim_time：sw 动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_ext_set_style

功能	设置 sw 的风格。
函数	lb_int32 lb_eui_sw_ext_set_style(void *sw, lb_eui_sw_style_e type, lb_eui_style_t *lb_sw_style)
参数	sw：sw 控件句柄； type：风格类型，参考 lb_eui_sw_style_e；lb_sw_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_sw_ext_get_state

功能	获取 sw 当前状态。
函数	lb_int32 lb_eui_sw_ext_get_state(void *sw, bool *state)
参数	sw：sw 控件句柄； state：返回 sw 当前状态。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_ext_get_anim_time

功能	获取 sw 动画时间。
函数	lb_int32 lb_eui_sw_ext_get_anim_time(void *sw, lb_uint16 *anim_time)
参数	sw：sw 控件句柄； anim_time：返回 sw 动画时间，单位 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_sw_ext_get_style

功能	获取 sw 的风格。
函数	lb_int32 lb_eui_sw_ext_get_style(void *sw, lb_eui_sw_style_e type, lb_eui_style_t *lb_sw_style)
参数	sw：sw 控件句柄； type：风格类型，参考 lb_eui_sw_style_e；lb_sw_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.28. Text Area 控件

```
typedef enum {
    LB_EUI_CURSOR_NONE,
    LB_EUI_CURSOR_LINE,
    LB_EUI_CURSOR_BLOCK,
    LB_EUI_CURSOR_OUTLINE,
```

```
    LB_EUI_CURSOR_UNDERLINE,
    /*Or it to any value to hide the cursor temporally*/
    LB_EUI_CURSOR_HIDDEN = 0x08,
}lb_eui_cursor_type_e;

typedef enum {
    LB_EUI_TA_STYLE_BG,
    LB_EUI_TA_STYLE_SCRL,
    LB_EUI_TA_STYLE_SB,
    LB_EUI_TA_STYLE_CURSOR,
} lb_eui_ta_style_e;

typedef enum {
    LB_EUI_TA_MOVE_CURSOR_LEFT,
    LB_EUI_TA_MOVE_CURSOR_RIGHT,
    LB_EUI_TA_MOVE_CURSOR_UP,
    LB_EUI_TA_MOVE_CURSOR_DOWN
} lb_eui_ta_cursor_direction_e;
```

Text area 控件的 API 接口定义如下：

Add/remove functions

lb_eui_ta_add_char

功能	ta 当前光标位置插入字符。
函数	lb_int32 lb_eui_ta_add_char(lb_int32 id, lb_uint32 c)
参数	id：ta 控件的 ID 号；c：新插入字符（例如：’a’ 或者 ASCII 码值）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_add_text

功能	ta 当前光标位置插入字符串。
函数	lb_int32 lb_eui_ta_add_text(lb_int32 id, const char *txt)
参数	id：ta 控件的 ID 号；txt：新插入以“\0”为终止符的字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_del_char

功能	删除 ta 当前光标位置左边的字符。
函数	lb_int32 lb_eui_ta_del_char(lb_int32 id)
参数	id：ta 控件的 ID 号。
返回值	0 表示成功，其他负值表示错误码。

Setter functions

lb_eui_ta_set_text

功能	设置 ta 的字符串。
函数	lb_int32 lb_eui_ta_set_text(lb_int32 id, const char *txt)
参数	id：ta 控件的 ID 号；txt：新字符串地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_cursor_pos

功能	设置 ta 光标位置。
函数	lb_int32 lb_eui_ta_set_cursor_pos(lb_int32 id, lb_int16 pos)
参数	id：ta 控件的 ID 号；pos：光标新的位置，范围 0 至 0x7FFF（最后一个字符后面）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_cursor_type

功能	设置 ta 光标类型。
函数	lb_int32 lb_eui_ta_set_cursor_type(lb_int32 id, lb_eui_cursor_type_e cur_type)
参数	id：ta 控件的 ID 号；cur_type：光标类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_pwd_mode

功能	设置 ta 是否为密码模式。
函数	lb_int32 lb_eui_ta_set_pwd_mode(lb_int32 id, bool en)
参数	id：ta 控件的 ID 号；en：true 表示为密码模式，false 表示为非密码模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_one_line

功能	设置 ta 为一行或者为恢复正常。
函数	lb_int32 lb_eui_ta_set_one_line(lb_int32 id, bool en)
参数	id：ta 控件的 ID 号；en：true 表示设置 ta 为一行，false 表示为设置 ta 为正常。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_text_align

功能	设置 ta 对齐方式，如果 ta 设置为一行模式，ta 可滚动，只能设置为 LB_EUI_LABEL_ALIGN_LEFT 对齐方式。
函数	lb_int32 lb_eui_ta_set_text_align(lb_int32 id, lb_eui_label_align_e align)
参数	id：ta 控件的 ID 号；align：对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_accepted_chars

功能	设置 ta 的字符限制列表，只有列表里的字符，才能被接受。

函数	lb_int32 lb_eui_ta_set_accepted_chars(lb_int32 id, const char *list)
参数	id：ta 控件的 ID 号；list：被接受的字符串地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_max_length

功能	设置 ta 的字符最大长度。
函数	lb_int32 lb_eui_ta_set_max_length(lb_int32 id,lb_uint16 num)
参数	id：ta 控件的 ID 号；num：ta 可以插入字符最大数目（此限制对 lb_eui_ta_set_text 无效）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_sb_mode

功能	设置 ta scroll bar 模式。
函数	lb_int32 lb_eui_ta_set_sb_mode(lb_int32 id,page_sb_mode_e mode)
参数	id：ta 控件的 ID 号；mode：ta scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_scroll_propagation

功能	设置 ta 滚动传播属性。
函数	lb_int32 lb_eui_ta_set_scroll_propagation(lb_int32 id,bool en)
参数	id：ta 控件的 ID 号；en：ta 滚动传播属性，true 表示支持滚动传播，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_edge_flash

功能	设置 ta 边沿闪光效果（到边沿时显示圆弧）。
函数	lb_int32 lb_eui_ta_set_edge_flash(lb_int32 id, bool en)
参数	id：ta 控件的 ID 号；en：ta 边沿闪光效果使能，true 表示支持闪光效果，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_set_style

功能	设置 ta 的风格。
函数	lb_int32 lb_eui_ta_set_style(lb_int32 id, lb_eui_ta_style_e type, lb_eui_style_t *lb_ta_style)
参数	id：ta 控件的 ID 号；type：风格类型，参考 lb_eui_ta_style_e； lb_ta_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_ta_get_text

功能	获取 ta 的字符串，如果 ta 我 passwd 模式，则返回的是实际的字符，不是'*'。
函数	lb_int32 lb_eui_ta_get_text(lb_int32 id,const char **txt)
参数	id：ta 控件的 ID 号；txt：返回字符串的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_cursor_pos

功能	获取 ta 当前光标的位置（返回字符索引）。
函数	lb_int32 lb_eui_ta_get_cursor_pos(lb_int32 id, lb_uint16 *pos)
参数	id：ta 控件的 ID 号；pos：返回当前光标的位置。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_cursor_type

功能	获取 ta 当前光标类型。
函数	lb_int32 lb_eui_ta_get_cursor_type(lb_int32 id, lb_uint8 *type)
参数	id：ta 控件的 ID 号；type：返回当前光标的类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_pwd_mode

功能	获取 ta 当前密码模式状态。
函数	lb_int32 lb_eui_ta_get_pwd_mode(lb_int32 id, bool *en)
参数	id：ta 控件的 ID 号；en：返回当前密码模式状态,true 表示当前 ta 为密码模式，false 表示非密码模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_one_line

功能	获取 ta 单行属性状态。
函数	lb_int32 lb_eui_ta_get_one_line(lb_int32 id, bool *en)
参数	id：ta 控件的 ID 号；en：返回当前单行属性状态,true 表示当前 ta 为单行，false 表示非单行。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_accepted_chars

功能	获取 ta 限定字符列表字符串。
函数	lb_int32 lb_eui_ta_get_accepted_chars(lb_int32 id, const char **list)
参数	id：ta 控件的 ID 号；list：返回 ta 限定字符列表字符串指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_max_length

功能	获取 ta 可插入字符最大长度。
函数	lb_int32 lb_eui_ta_get_max_length(lb_int32 id, lb_uint16 *num)
参数	id：ta 控件的 ID 号；num：返回 ta 可插入字符最大数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_sb_mode

功能	获取 ta scroll bar 模式。
函数	lb_int32 lb_eui_ta_get_sb_mode(lb_int32 id, lb_uint8 *mode)

参数	id：ta 控件的 ID 号；mode：返回 ta scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_scroll_propagation

功能	获取 ta 滚动传播属性状态。
函数	lb_int32 lb_eui_ta_get_scroll_propagation(lb_int32 id, bool *en)
参数	id：ta 控件的 ID 号；mode：返回 ta 滚动传播属性状态，true 表示支持传播，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_edge_flash

功能	获取 ta 边沿闪光属性状态。
函数	lb_int32 lb_eui_ta_get_edge_flash(lb_int32 id, bool *en)
参数	id：ta 控件的 ID 号；en：返回 ta 边沿闪光效果属性状态，true 表示支持传播，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_get_style

功能	获取 ta 的风格。
函数	lb_int32 lb_eui_ta_set_style(lb_int32 id, lb_eui_ta_style_e type, lb_eui_style_t *lb_ta_style)
参数	id：ta 控件的 ID 号；type：风格类型，参考 lb_eui_ta_style_e； lb_ta_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

Other functions

lb_eui_ta_move_cursor

功能	移动光标。
函数	lb_int32 lb_eui_ta_move_cursor(lb_uint32 id, lb_eui_ta_cursor_direction_e direction)
参数	id：ta 控件的 ID 号；direction：移动方向。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_add_char

功能	ta 当前光标位置插入字符。
函数	lb_int32 lb_eui_ta_ext_add_char(void *textarea, lb_uint32 c)
参数	textarea：ta 控件句柄；c：新插入字符（例如：’a’ 或者 ASCII 码值）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_add_text

功能	ta 当前光标位置插入字符串。
函数	lb_int32 lb_eui_ta_ext_add_text(void *textarea, const char *txt)
参数	textarea：ta 控件句柄；txt：新插入以“\0”为终止符的字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_del_char

功能	删除 ta 当前光标位置左边的字符。
函数	lb_int32 lb_eui_ta_ext_del_char(void *textarea)
参数	textarea：ta 控件句柄。
返回值	0 表示成功，其他负值表示错误码。

Setter functions

lb_eui_ta_ext_set_text

功能	设置 ta 的字符串。
函数	lb_int32 lb_eui_ta_ext_set_text(void *textarea, const char *txt)
参数	textarea：ta 控件句柄；txt：新字符串地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_cursor_pos

功能	设置 ta 光标位置。
函数	lb_int32 lb_eui_ta_ext_set_cursor_pos(void *textarea, lb_int16 pos)
参数	textarea：ta 控件句柄；pos：光标新的位置，范围 0 至 0x7FFF（最后一个字符后面）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_cursor_type

功能	设置 ta 光标类型。
函数	lb_int32 lb_eui_ta_ext_set_cursor_type(void *textarea, lb_eui_cursor_type_e cur_type)
参数	textarea：ta 控件句柄；cur_type：光标类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_pwd_mode

功能	设置 ta 是否为密码模式。
函数	lb_int32 lb_eui_ta_ext_set_pwd_mode(void *textarea, bool en)
参数	textarea：ta 控件句柄；en：true 表示为密码模式，false 表示为非密码模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_one_line

功能	设置 ta 为一行或者为恢复正常。
函数	lb_int32 lb_eui_ta_ext_set_one_line(void *textarea, bool en)
参数	textarea：ta 控件句柄；en：true 表示设置 ta 为一行，false 表示为设置 ta 为正常。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_text_align

功能	设置 ta 对齐方式，如果 ta 设置为一行模式，ta 可滚动，只能设置为 LB_EUI_LABEL_ALIGN_LEFT 对齐方式。

函数	lb_int32 lb_eui_ta_ext_set_text_align(void *textarea, lb_eui_label_align_e align)
参数	textarea：ta 控件句柄；align：对齐方式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_accepted_chars

功能	设置 ta 的字符限制列表，只有列表里的字符，才能被接受。
函数	lb_int32 lb_eui_ta_ext_set_accepted_chars(void *textarea, const char *list)
参数	textarea：ta 控件句柄；list：被接受的字符串地址。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_max_length

功能	设置 ta 的字符最大长度。
函数	lb_int32 lb_eui_ta_ext_set_max_length(void *textarea,lb_uint16 num)
参数	textarea：ta 控件句柄；num：ta 可以插入字符最大数目（此限制对 lb_eui_ta_set_text 无效）。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_sb_mode

功能	设置 ta scroll bar 模式。
函数	lb_int32 lb_eui_ta_ext_set_sb_mode(void *textarea,page_sb_mode_e mode)
参数	textarea：ta 控件句柄；mode：ta scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_scroll_propagation

功能	设置 ta 滚动传播属性。
函数	lb_int32 lb_eui_ta_ext_set_scroll_propagation(void *textarea,bool en)
参数	textarea：ta 控件句柄；en：ta 滚动传播属性，true 表示支持滚动传播，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_edge_flash

功能	设置 ta 边沿闪光效果（到边沿时显示圆弧）。
函数	lb_int32 lb_eui_ta_ext_set_edge_flash(void *textarea, bool en)
参数	textarea：ta 控件句柄；en：ta 边沿闪光效果使能，true 表示支持闪光效果，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_set_style

功能	设置 ta 的风格。
函数	lb_int32 lb_eui_ta_ext_set_style(void *textarea, lb_eui_ta_style_e type, lb_eui_style_t *lb_ta_style)
参数	textarea：ta 控件句柄；type：风格类型，参考 lb_eui_ta_style_e；lb_ta_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_ta_ext_get_text

功能	获取 ta 的字符串，如果 ta 我 passwd 模式，则返回的是实际的字符，不是'*'。
函数	lb_int32 lb_eui_ta_ext_get_text(void *textarea,const char **txt)
参数	textarea：ta 控件句柄；txt：返回字符串的指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_cursor_pos

功能	获取 ta 当前光标的位置（返回字符索引）。
函数	lb_int32 lb_eui_ta_ext_get_cursor_pos(void *textarea, lb_uint16 *pos)
参数	textarea：ta 控件句柄；pos：返回当前光标的位置。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_cursor_type

功能	获取 ta 当前光标类型。
函数	lb_int32 lb_eui_ta_ext_get_cursor_type(void *textarea, lb_uint8 *type)
参数	textarea：ta 控件句柄；type：返回当前光标的类型。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_pwd_mode

功能	获取 ta 当前密码模式状态。
函数	lb_int32 lb_eui_ta_ext_get_pwd_mode(void *textarea, bool *en)
参数	textarea：ta 控件句柄；en：返回当前密码模式状态,true 表示当前 ta 为密码模式，false 表示非密码模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_one_line

功能	获取 ta 单行属性状态。
函数	lb_int32 lb_eui_ta_ext_get_one_line(void *textartea, bool *en)
参数	textarea：ta 控件句柄；en：返回当前单行属性状态,true 表示当前 ta 为单行，false 表示非单行。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_accepted_chars

功能	获取 ta 限定字符列表字符串。
函数	lb_int32 lb_eui_ta_ext_get_accepted_chars(void *textarea, const char **list)
参数	textarea：ta 控件句柄；list：返回 ta 限定字符列表字符串指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_max_length

功能	获取 ta 可插入字符最大长度。

函数	lb_int32 lb_eui_ta_ext_get_max_length(void *textarea, lb_uint16 *num)
参数	textarea：ta 控件句柄；num：返回 ta 可插入字符最大数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_sb_mode

功能	获取 ta scroll bar 模式。
函数	lb_int32 lb_eui_ta_ext_get_sb_mode(void *textarea, lb_uint8 *mode)
参数	textarea：ta 控件句柄；mode：返回 ta scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_scroll_propagation

功能	获取 ta 滚动传播属性状态。
函数	lb_int32 lb_eui_ta_ext_get_scroll_propagation(void *textarea, bool *en)
参数	textarea：ta 控件句柄；mode：返回 ta 滚动传播属性状态，true 表示支持传播，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_edge_flash

功能	获取 ta 边沿闪光属性状态。
函数	lb_int32 lb_eui_ta_ext_get_edge_flash(void *textarea, bool *en)
参数	textarea：ta 控件句柄；en：返回 ta 边沿闪光效果属性状态，true 表示支持传播，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_ta_ext_get_style

功能	获取 ta 的风格。
函数	lb_int32 lb_eui_ta_ext_set_style(void *textarea, lb_eui_ta_style_e type, lb_eui_style_t *lb_ta_style)
参数	textarea：ta 控件句柄；type：风格类型，参考 lb_eui_ta_style_e；lb_ta_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

Other functions

lb_eui_ta_ext_move_cursor

功能	移动光标。
函数	lb_int32 lb_eui_ta_ext_move_cursor(void *textarea, lb_eui_ta_cursor_direction_e direction)
参数	textarea：ta 控件句柄；direction：移动方向。
返回值	0 表示成功，其他负值表示错误码。

4.5.29. Tabview 控件

```
typedef enum {
    LB_EUI_TABVIEW_BTNS_POS_TOP,
    LB_EUI_TABVIEW_BTNS_POS_BOTTOM,
```

```
}lb_eui_tabview_btns_pos_e;

typedef enum {
    LB_EUI_TABVIEW_STYLE_BG,
    LB_EUI_TABVIEW_STYLE_INDIC,
    LB_EUI_TABVIEW_STYLE_BTN_BG,
    LB_EUI_TABVIEW_STYLE_BTN_REL,
    LB_EUI_TABVIEW_STYLE_BTN_PR,
    LB_EUI_TABVIEW_STYLE_BTN_TGL_REL,
    LB_EUI_TABVIEW_STYLE_BTN_TGL_PR,
} lb_eui_tabview_style_e;
```

Tabview 控件的 API 接口定义如下：

Setter functions

lb_eui_tabview_set_tab_act

功能	设置 tabview 切换 tab。
函数	lb_int32 lb_eui_tabview_set_tab_act(lb_int32 id, lb_uint16 id, bool anim_en)
参数	id：tabview 控件的 ID 号； id：tabview 切换 tab 的索引； anim_en：切换 tab 动画使能，true 表示支持动画，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_set_sliding

功能	设置 tabview 是否支持触摸水平滑动切换 tab。
函数	lb_int32 lb_eui_tabview_set_sliding(lb_int32 id, bool en)
参数	id：tabview 控件的 ID 号； en：true 表示支持触摸水平滑动切换 tab，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_set_anim_time

功能	设置 tabview 切换 tab 动画时间。
函数	lb_int32 lb_eui_tabview_set_anim_time(lb_int32 id, lb_uint16 anim_time)
参数	id：tabview 控件的 ID 号； anim_time：tabview 切换 tab 的动画时间，单位为 ms。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_set_btns_pos

功能	设置 tabview 选项卡选择按钮的位置。
函数	lb_int32 lb_eui_tabview_set_btns_pos(lb_int32 id, lb_eui_tabview_btns_pos_e btns_pos)

参数	id：tabview 控件的 ID 号； btns_pos：tabview 选项卡选择按钮位置。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_set_btns_hidden

功能	设置 tabview 选项卡选择按钮是否影藏。
函数	lb_int32 lb_eui_tabview_set_btns_hidden(lb_int32 id, bool en)
参数	id：tabview 控件的 ID 号； en：true 表示隐藏，false 表示不隐藏。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_set_btns_height

功能	设置 tabview 选项卡选择按钮高度。
函数	lb_int32 lb_eui_tabview_set_btns_hidden(lb_int32 id, lb_int32 h)
参数	id：tabview 控件的 ID 号； h：选项卡选择按钮高度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_set_style

功能	设置 tabview 的风格。
函数	lb_int32 lb_eui_tabview_set_style(lb_int32 id, lb_eui_tabview_style_e type,lb_eui_style_t *lb_tabview_style)
参数	id：tabview 控件的 ID 号； type：风格类型，参考 lb_eui_tabview_style_e； lb_tabview_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_tabview_get_tab_act

功能	获取 tabview 当前活动选项卡的索引。
函数	lb_int32 lb_eui_tabview_get_tab_act(lb_int32 id, lb_uint16 *index)
参数	id：tabview 控件的 ID 号； index：返回当前活动选项卡的索引。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_get_tab_count

功能	获取 tabview 当前选项卡的数目。
函数	lb_int32 lb_eui_tabview_get_tab_count(lb_int32 id, lb_uint16 *num)
参数	id：tabview 控件的 ID 号； num：返回 tabview 当前选项卡数目。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_get_sliding

功能	获取 tabview 是否支持水平滑动切换选项卡。
函数	lb_int32 lb_eui_tabview_get_sliding(lb_int32 id, bool *en)
参数	id：tabview 控件的 ID 号； en：返回 tabview 是否支持水平滑动切换选项卡状态，true 表示支持，false

	表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_get_anim_time

功能	获取 tabview 切换选项卡动画时间。
函数	lb_int32 lb_eui_tabview_get_anim_time(lb_int32 id, lb_uint16 *time)
参数	id：tabview 控件的 ID 号； time：返回 tabview 切换选项卡动画时间。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_get_btns_pos

功能	获取 tabview 选项卡选择按钮位置。
函数	lb_int32 lb_eui_tabview_get_btns_pos(lb_int32 id, lb_uint8 *pos)
参数	id：tabview 控件的 ID 号； pos：返回 tabview 选项卡选择按钮位置,0 表示在顶部，1 表示在底部。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_get_btns_hidden

功能	获取 tabview 选项卡隐藏状态。
函数	lb_int32 lb_eui_tabview_get_btns_hidden(lb_int32 id, bool *en)
参数	id：tabview 控件的 ID 号； en：返回 tabview 选项卡隐藏状态，true 表示隐藏，false 表示不隐藏。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tabview_get_style

功能	获取 tabview 的风格。
函数	lb_int32 lb_eui_tabview_get_style(lb_int32 id, lb_eui_tabview_style_e type,lb_eui_style_t *lb_tabview_style)
参数	id：tabview 控件的 ID 号； type：风格类型，参考 lb_eui_tabview_style_e； lb_tabview_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

4.5.30. Tileview 控件

```
typedef enum {
    LB_EUI_TILEVIEW_STYLE_BG,
} lb_eui_tileview_style_e;
```

Tileview 控件的 API 接口定义如下：

Setter functions

lb_eui_tileview_set_act_tile

功能	设置 tileview 当前显示的区域坐标。

函数	lb_int32 lb_eui_tileview_set_act_tile(lb_uint32 id, lb_eui_point_t act_tile)
参数	id：tileview 控件的 ID 号； act_tile：区域坐标。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tileview_set_edge_flash

功能	设置 tileview 边沿闪光效果（到边沿时显示圆弧）。
函数	lb_int32 lb_eui_tileview_set_edge_flash(lb_uint32 id, bool en)
参数	id：tileview 控件的 ID 号； en：tileview 边沿闪光效果使能，true 表示支持闪光效果，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tileview_set_style

功能	设置 tileview 的风格。
函数	lb_int32 lb_eui_tileview_set_style(lb_uint32 id, lb_eui_tileview_style_e type, lb_eui_style_t *lb_tileview_style)
参数	id：tileview 控件的 ID 号； type：风格类型，参考 lb_eui_tileview_style_e； lb_tileview_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tileview_get_edge_flash

功能	获取 tabview 的风格。
函数	lb_int32 lb_eui_tileview_get_edge_flash(lb_uint32 id, bool *en)
参数	id：tileview 控件的 ID 号； en：tileview 边沿闪光效果使能，true 表示支持闪光效果，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_tileview_get_style

功能	获取 tileview 的风格。
函数	lb_int32 lb_eui_tileview_get_style(lb_uint32 id, lb_eui_tileview_style_e type,lb_eui_style_t *lb_tileview_style)
参数	id：tileview 控件的 ID 号； type：风格类型，参考 lb_eui_tileview_style_e； lb_tileview_style：需要设置的风格
返回值	0 表示成功，其他负值表示错误码。

4.5.31. Window 控件

```
typedef enum {
    LB_EUI_WIN_STYLE_BG,
    LB_EUI_WIN_STYLE_CONTENT_BG,
    LB_EUI_WIN_STYLE_CONTENT_SCRL,
    LB_EUI_WIN_STYLE_SB,
    LB_EUI_WIN_STYLE_HEADER,
    LB_EUI_WIN_STYLE_BTN_REL,
    LB_EUI_WIN_STYLE_BTN_PR,
} lb_eui_win_style_e;
```

Window 控件的 API 接口定义如下：

Setter functions

lb_eui_win_set_title

功能	设置 win 的标题。
函数	lb_int32 lb_eui_win_set_title(lb_int32 id, const char *title)
参数	id：win 控件的 ID 号； title：win 的标题字符串。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_btn_size

功能	设置 win 控件按钮大小。
函数	lb_int32 lb_eui_win_set_btn_size(lb_int32 id, lb_int32 size)
参数	id：win 控件的 ID 号； size：win 控件按钮大小。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_layout

功能	设置 win 布局模式。
函数	lb_int32 lb_eui_win_set_layout(lb_int32 id, layout_e layout)
参数	id：win 控件的 ID 号； layout：win 控件布局模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_sb_mode

scrollbar modes 定义见 page 控件下的 lb_eui_sb_mode_e;

功能	设置 win scroll bar 模式。
函数	lb_int32 lb_eui_win_set_sb_mode(lb_int32 id, lb_eui_sb_mode_e sb_mode)
参数	id：win 控件的 ID 号； sb_mode：win scroll bar 模式。

返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_drag

功能	设置 win 是否支持拖拽。
函数	lb_int32 lb_eui_win_set_drag(lb_int32 id, bool en)
参数	id：win 控件的 ID 号； en：win 拖拽状态，true 表示支持拖拽，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_child_focus

功能	设置 win 选中控件。
函数	lb_int32 lb_eui_win_set_child_focus(lb_uint32 id, lb_int32 child_id, lb_uint16 anim_time)
参数	id：win 控件的 ID 号； child_id：子控件 ID 号；anim_time：动画时间。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_scroll_hor

功能	设置 win 滚动条水平移动。
函数	lb_int32 lb_eui_win_set_scroll_hor(lb_uint32 id, lb_int32 offset)
参数	id：win 控件的 ID 号； offset：移动距离，offset<0，向右移动，offset>0，向左移动。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_scroll_ver

功能	设置 win 垂直移动。
函数	lb_int32 lb_eui_win_set_scroll_ver(lb_uint32 id, lb_int32 offset)
参数	id：win 控件的 ID 号；offset：移动距离，offset<0，向下移动，offset>0，向上移动。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_set_style

功能	设置 win 的风格。
函数	lb_int32 lb_eui_win_set_style(lb_int32 id, lb_eui_win_style_e type, lb_eui_style_t *lb_win_style)
参数	id：win 控件的 ID 号；type：风格类型，参考 lb_eui_win_style_e； lb_win_style：需要设置的风格。
返回值	0 表示成功，其他负值表示错误码。

Getter functions

lb_eui_win_get_title

功能	获取 win 标题 。
函数	lb_int32 lb_eui_win_get_title(lb_int32 id, const char **title)

参数	id：win 控件的 ID 号； title：返回标题字符指针。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_get_btn_size

功能	获取 win 控件按钮大小。
函数	lb_int32 lb_eui_win_get_btn_size(lb_int32 id, lb_int32 *size)
参数	id：win 控件的 ID 号； size：返回 win 控件按钮大小。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_get_from_btn

功能	通过 win 控件按钮 id 获取 win 控件 id。
函数	lb_int32 lb_eui_win_get_from_btn(lb_int32 btn_id, lb_int32 *win_id)
参数	btn_id：btn 控件的 ID 号； win_id：返回 win 控件 ID 号。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_get_layout

功能	获取 win 布局模式。
函数	lb_int32 lb_eui_win_get_layout(lb_int32 id, lb_uint8 *layout)
参数	id：win 控件的 ID 号； layout：返回 win 布局模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_get_sb_mode

功能	获取 win scroll bar 模式。
函数	lb_int32 lb_eui_win_get_sb_mode(lb_int32 id, lb_uint8 *sb_mode)
参数	id：win 控件的 ID 号； sb_mode：返回 win scroll bar 模式。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_get_width

功能	获取 win 内容区域宽度。
函数	lb_int32 lb_eui_win_get_width(lb_int32 id, lb_int32 *width)
参数	id：win 控件的 ID 号； width：返回 win 内容区域宽度。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_get_drag

功能	获取 win 是否支持拖拽状态。
函数	lb_int32 lb_eui_win_get_drag(lb_int32 id, bool *en)
参数	id：win 控件的 ID 号； en：返回 win 拖拽状态，true 表示支持拖拽，false 表示不支持。
返回值	0 表示成功，其他负值表示错误码。

lb_eui_win_get_style

功能	获取 win 的风格。
函数	lb_int32 lb_eui_win_set_style(lb_int32 id, lb_eui_win_style_e type, lb_eui_style_t *lb_win_style)
参数	id：win 控件的 ID 号；type：风格类型，参考 lb_eui_win_style_e； lb_win_style：需要获取的风格。
返回值	0 表示成功，其他负值表示错误码。

5. 使用示例

5.1. EUI 初始化

此处需要注意的是控件初始化、退出函数和响应函数注册是根据 EUIEditor 中控件的 init_func、exit_func 和 msg 值参数一一对应的，例如:

```
int main(void)
{
/* 激活 EUI lib，注册输入输出设备 */
        lb_eui_init("/etc/layout/eui.cfg");
        /* EUI 资源加载，字符、图片和字体库等 ， 如在配置文件中已配置，则不需要设置 */
        lb_eui_res_init("/out/res/res.iso");
        /* 控件初始化、退出函数和响应函数注册 */
        lb_eui_fmngr_reg_init_func("system_poweroff_init", system_poweroff_init);
        lb_eui_fmngr_reg_exit_func("system_poweroff_exit", system_poweroff_exit);
        /* 此处需注意，msg 值范围是 0xE200-0xEF00 */
        lb_eui_reg_resp_msg_func(LB_MSG_BACK, imgbtn_back);
        /* EUI 创建主界面 */
        lb_eui_create("/out/layout/common/root.json");
        /* EUI 启动 */
        lb_eui_start();
        while (1)
          delay(10);
```

```
    /* 控件初始化、退出函数和响应函数注销 */
    lb_eui_fmngr_unreg_init_func(system_poweroff_init);
    lb_eui_fmngr_unreg_exit_func(system_poweroff_exit);
    lb_eui_unreg_resp_msg_func(imgbtn_back);
    /* EUI 删除主界面 */
    lb_eui_destroy();
    /* EUI 资源卸载 */
    lb_eui_res_uinit();
    /* EUI lib freeze*/
    lb_eui_uinit();
}
```

其中 init_func、exit_func 函数原型可参考 lb_eui_ctrl_func.h 的类型定义

5.2. 创建和删除新的静态视图

此处需要注意的是 API 调用的先后顺序

创建新的静态视图：注册控件初始化、退出函数和响应函数函数－》风格加载－》界面加载

删除新的视图：注销控件响应函数－》界面卸载－》风格卸载－》注销控件初始化、退出函数

```
app_start(void )
{
    lb_int32 ret = 0;

    init_funcs(); /* 初始化、退出函数注册 */
    resp_funcs();/* 响应函数注册 */

    ret |= system_set_init();
    ret |= lb_eui_style_load("layout/system_setting/styles.json"); /* 加载风格资源 */
    ret |= lb_eui_static_new("layout/system_setting/main_view.json");/* 加载视图资源 */

    if (ret != 0) {
        APP_LOG_W("err lb ui init failed!\n");
        return ret;
    }

    return ret;
```

```
}

app_exit(void)
{
    lb_int32 ret = 0;

    ret = unresp_funcs(); /* 注销响应函数 */
    assert(ret == 0);

    lb_eui_static_delete();/* 删除当前应用视图资源 */
    lb_eui_style_unload();/* 卸载当前应用风格资源 */

    ret = uninit_funcs(); /* 注销初始化、退出函数 */
    assert (ret == 0);

    ret = system_set_save();
    assert(ret == 0);

    ret = system_set_exit();
    assert (ret == 0);

    return ret;
}
```

5.3. 动态列表使用

动态列表比较灵活，加载选项内容由用户自己填充，只是再 list 初始化的时候必须配置好动态加载回调函数，响应函数原型参考 lb_eui_list.h 的函数定义。

```
lb_eui_list_elem_t elem_info[30];
lb_eui_img_dsc_t *img_src;
lb_eui_img_dsc_t *imgbtn_src;

/**
 * eui_demo_list_click - Set the list's style.
 * @param: it is click object ext pointer.
 *
 * This function use to Set the list's style.
```

```
 *
 * Returns -0 if there is no error; otherwise, return the error code.
 */
lb_uint8 dynamic_list_click(void *param)
{
	lb_uint8	err = 0;

	//("in\n");

	if (param == NULL) {
		printf("Invalid param!\n");
		err = -1;
		goto exit;
	}

	printf("dynamic list click action!\n");

exit:
	//("out ret = 0x%X\n", err);
	return err;
}

/**
 * list_load_cb - Set an array of points
 * @add_index: the index of add option.
 * @rm_index: the index of remove option.
 *
 * This function use to list dynamic load.
 *
 * Returns -the option data pointer.
 */
static void *list_load_cb(int32_t add_index, int32_t rm_index)
{
	/* free remove elem data */
	if(rm_index >= 0 && rm_index < 30) {
		if (elem_info[rm_index].col) {
			for (int i = 0; i < LIST_DYNAMIC_ELEM_COL_NUM; i++) {
```

```
                lb_uint8 col_type =
                    elem_info[rm_index].col[i].type;
                switch (col_type) {
                case LB_EUI_LIST_COL_TYPE_TEXT:
                case LB_EUI_LIST_COL_TYPE_IMGBTN:
                    free(elem_info[rm_index].col[i].data);
                    break;
                default:
                    break;
                }
                elem_info[rm_index].col[i].data = NULL;
            }
            free(elem_info[rm_index].col);
            elem_info[rm_index].col = NULL;
        }
    }

    /* add new elem data */
    if (add_index < 0 || add_index >= 30)
        return NULL;

    elem_info[add_index].col = (lb_eui_list_col_t *)calloc
        (sizeof(lb_eui_list_col_t) * LIST_DYNAMIC_ELEM_COL_NUM, 1);
    elem_info[add_index].index = add_index;
    elem_info[add_index].col[0].type = LB_EUI_LIST_COL_TYPE_NULL;
    elem_info[add_index].col[1].type = LB_EUI_LIST_COL_TYPE_TEXT;
    char *str = (char *)calloc(16, 1);
    sprintf(str, "%02d", add_index);
    elem_info[add_index].col[1].data = str;
    elem_info[add_index].col[2].type = LB_EUI_LIST_COL_TYPE_IMG;
    elem_info[add_index].col[2].data = img_src;
    elem_info[add_index].col[3].type = LB_EUI_LIST_COL_TYPE_IMGBTN;
    lb_eui_list_imgbtn_t *imgbtn_info = (lb_eui_list_imgbtn_t *)calloc
                                (sizeof(lb_eui_list_imgbtn_t), 1);
    imgbtn_info->action = NULL;
    imgbtn_info->p_pr_img_src = imgbtn_src;
    imgbtn_info->p_rel_img_src = imgbtn_src;
```

```
    elem_info[add_index].col[3].data = imgbtn_info;
    elem_info[add_index].action = dynamic_list_click;

    return &elem_info[add_index];
}

lb_int32 eui_demo_list_dynamic_init(void *param)
{
    lb_int32 err = 0;

    memset(elem_info, 0, sizeof(elem_info));
    img_src = lb_eui_eimage_create_img_buf("V:/image/list_img.png.BGRA8888.ez");
    imgbtn_src = lb_eui_eimage_create_img_buf("V:/image/list_imgbtn.png.BGRA8888.ez");
    err = lb_eui_list_set_dynamic_load_cb(DEMO_LIST_LIST_DYNAMIC, list_load_cb);

    return err;
}

lb_int32 eui_demo_list_reg_init_exit_func(void)
{
    lb_int32 err;

    err = lb_eui_fmngr_reg_init_func("list_dynamic_init", eui_demo_list_dynamic_init);

    return err;
}
/* dynamic list must free resource before close view */
lb_int32 eui_demo_list_dynamic_free(void)
{
    lb_int32 start = 0, end = 0;

    /* free remove elem data */
    lb_eui_list_get_visible_items(DEMO_LIST_LIST_DYNAMIC, &start, &end);
    for (int j = start; j <= end; j++) {
        if (elem_info[j].col) {
            for (int i = 0; i < LIST_DYNAMIC_ELEM_COL_NUM; i++) {
                lb_uint8 col_type = elem_info[j].col [i].type;
```

```
                switch (col_type) {
                case LB_EUI_LIST_COL_TYPE_TEXT:
                case LB_EUI_LIST_COL_TYPE_IMGBTN:
                    free(elem_info[j].col[i].data);
                    break;
                default:
                    break;
                }
                elem_info[j].col[i].data = NULL;
            }
            free(elem_info[j].col);
            elem_info[j].col = NULL;
        }
    }

    if (img_src) {
        lb_eui_eimage_destory_img_buf(img_src);
        img_src = NULL;
    }
    if (imgbtn_src) {
        lb_eui_eimage_destory_img_buf(imgbtn_src);
        imgbtn_src = NULL;
    }
    return 0;
}
```

5.4. 图标闪烁

控制图片显示和隐藏即可实现。

```
while (1) {
        if (sem_trywait(&cm_rec.sem) == 0) {
            APP_LOG_W("exit\n");
            goto exit;
        }

        lb_recorder_ctrl(cm_rec.hdl, LB_REC_GET_TIME, (void *)&curr_time);
```

```
        if (curr_time != last_time) {
            memset(time_buf, 0, 32);
            sprintf(time_buf, "%02d:%02d", curr_time / 60, curr_time % 60);
            lb_eui_label_set_text(VCAM_TIME_ID, time_buf);;
            last_time = curr_time;
        }
        /* 控制图标显示和隐藏 */
        if (curr_time % 2 == 1)
            lb_eui_obj_set_hidden(VCAM_DOT_ID, 1);
        else
            lb_eui_obj_set_hidden(VCAM_DOT_ID, 0);

        rt_thread_delay(50);
    }
```

5.5. 提示框

提示框通常是作为独立视图配置的，显示的时候秩序根据索引或者根据视图 ID 打开即可，索引值在 eguieditor.h 中有定义

```
/* 根据独立视图加载的先后顺序定义索引值 */
#define DIALOG_FORMAT_SD 0
#define DIALOG_LOCK_RECORDER_FILE 1
#define DIALOG_LOWPOWER_SHUTDOWN 2
#define DIALOG_PARK_MONITOR 3
#define DIALOG_PLAY_ERROR 4
#define DIALOG_PLEASE_FORMAT_SDCARD 5
#define DIALOG_PLEASE_PLUGIN_AV 6
#define DIALOG_PLEASE_PLUGIN_SDCARD 7
#define DIALOG_PROGRESS_BAR 8
#define DIALOG_QUIT_RECORDER 9
#define DIALOG_SDCARD_FORMAT_FAILED 10
#define DIALOG_SDCARD_FORMAT_SUCCEED 11
#define DIALOG_SDCARD_FULL 12
/*视图 ID 是每个 ID 的第一个*/
/* led view */
#define DEMO_LED_VIEW 530
```

```
#define DEMO_LED_LED 531
#define DEMO_LED_PROMPT 532
#define DEMO_LED_PROMPT_EFFECT 533
#define DEMO_LED_PROMPT_PARAM 534
#define DEMO_LED_BRIGHT 535
#define DEMO_LED_SWITCH 536
#define DEMO_LED_TOGGLE 537
#define DEMO_LED_STYLES 538
/* line view */
#define DEMO_LINE_VIEW 500
#define DEMO_LINE_LINE 501
#define DEMO_LINE_PROMPT 502
#define DEMO_LINE_PROMPT_EFFECT 503
#define DEMO_LINE_PROMPT_PARAM 504
#define DEMO_LINE_POINT_SETTING 505
#define DEMO_LINE_AUTO_SIZE 506
#define DEMO_LINE_Y_INVERT 507
#define DEMO_LINE_STYLES 508

/* 显示的时候只需调用 API 接口即可 */
lb_eui_isolate_new(DIALOG_PLEASE_PLUGIN_SDCARD)
或者
lb_eui_view_open(DEMO_LED_VIEW);
```

5.6. Blocklinker

5.6.1. 初始化与申请 blocklinker

```
static int cdr_draw_request(void)
{
    int i = 0;
    int red_color = 0x66ff0000;
    int green_color = 0x6600ff00;
    int en_adas = 0;
    int en_bsd = 0;

    en_adas = adas_get_enable();
    en_bsd = bsd_get_enable();
```

```
if (en_adas == ENABLE_ON) {
    green_buff = (unsigned char *)rt_malloc(ADAS_BKL_RECT_SIZE_MAX);
    if (green_buff == NULL) {
        APP_LOG_E("malloc fail\n");
        return -1;
    }

    for (i = 0; i < ADAS_BKL_RECT_SIZE_MAX / 4; i++)
        ((int *)green_buff)[i] = green_color;

    red_buff = (unsigned char *)rt_malloc(ADAS_BKL_RECT_SIZE_MAX);
    if (red_buff == NULL) {
        APP_LOG_E("malloc fail\n");
        return -1;
    }

    for (i = 0; i < ADAS_BKL_RECT_SIZE_MAX / 4; i++)
        ((int *)red_buff)[i] = red_color;

    line_buff[0] = (unsigned char *)rt_malloc(ADAS_BKL_LINE_SIZE_MAX);
    if (line_buff[0] == NULL) {
        APP_LOG_E("malloc fail\n");
        return -1;
    }

    line_buff[1] = (unsigned char *)rt_malloc(ADAS_BKL_LINE_SIZE_MAX);
    if (line_buff[1] == NULL) {
        APP_LOG_E("malloc fail\n");
        return -1;
    }
}

if (en_bsd == ENABLE_ON) {
    int width, height;
    lb_eui_point_t point1, point2;
    int orange_color;
```

```
int alpha = 0xff;

width = BSD_BKL_LINE_WIDTH_MAX - 1;
height = BSD_BKL_LINE_HEIGHT_MAX - BSD_BKL_LINE_START_POS;

bsd_left_buff = (unsigned char *)rt_malloc(BSD_BKL_LINE_SIZE_MAX);
if (bsd_left_buff == NULL) {
    APP_LOG_E("malloc fail\n");
    return -1;
}
rt_memset(bsd_left_buff, 0, BSD_BKL_LINE_SIZE_MAX);

for (i = 0; i < BSD_BKL_LINE_WIDTH_MAX; i++) {
    point1.x = i;
    point1.y = 0;
    point2.x = width;
    point2.y = BSD_BKL_LINE_START_POS;
    orange_color = alpha << 24 | 0xfe571e;
    bsd_draw_line(bsd_left_buff, point1, point2, orange_color);
}

for (i = 0; i < height; i++) {
    point1.x = 0;
    point1.y = i;
    point2.x = width;
    point2.y = BSD_BKL_LINE_START_POS + i;
    orange_color = alpha << 24 | 0xfe571e;
    bsd_draw_line(bsd_left_buff, point1, point2, orange_color);
    if (alpha > 4)
        alpha -= 4;
}

bsd_right_buff = (unsigned char *)rt_malloc(BSD_BKL_LINE_SIZE_MAX);
if (bsd_right_buff == NULL) {
    APP_LOG_E("malloc fail\n");
    return -1;
}
```

```
        rt_memset(bsd_right_buff, 0, BSD_BKL_LINE_SIZE_MAX);

        alpha = 0xff;

        for (i = 0; i < BSD_BKL_LINE_WIDTH_MAX; i++) {
            point1.x = i;
            point1.y = BSD_BKL_LINE_HEIGHT_MAX - 1;
            point2.x = width;
            point2.y = height;
            orange_color = alpha << 24 | 0xfe571e;
            bsd_draw_line(bsd_right_buff, point1, point2, orange_color);
        }
        point1.y = BSD_BKL_LINE_HEIGHT_MAX;
        for (i = height - 1; i >= 0; i--) {
            point1.x = 0;
            point1.y--;
            point2.x = width;
            point2.y = i;
            orange_color = alpha << 24 | 0xfe571e;
            bsd_draw_line(bsd_right_buff, point1, point2, orange_color);

            if (alpha > 4)
                alpha -= 4;
        }
    }

    if (en_adas == ENABLE_ON && en_bsd != ENABLE_ON) {
        for (i = 0; i < ADAS_BKL_NUM_MAX; i++) {
            lb_eui_bl_draw_request(i);
            lb_eui_bl_draw_add(i);
        }
    } else if (en_adas != ENABLE_ON && en_bsd == ENABLE_ON) {
        for (i = ADAS_BKL_NUM_MAX; i < ADAS_BKL_NUM_MAX + BSD_BKL_NUM_MAX; i++) {
            lb_eui_bl_draw_request(i);
            lb_eui_bl_draw_add(i);
        }
```

```
    } else if (en_adas == ENABLE_ON && en_bsd == ENABLE_ON) {
        for (i = 0; i < ADAS_BKL_NUM_MAX + BSD_BKL_NUM_MAX; i++) {
            lb_eui_bl_draw_request(i);
            lb_eui_bl_draw_add(i);
        }
    }

    return 0;
}

int cdr_draw_init(void)
{
    int err = 0;

    err = lb_eui_bl_draw_init();

    err = cdr_draw_request();

    return err;
}
```

5.6.2. 填充 blocklinker

```
int cdr_bsd_draw_line(uint8_t idx, uint8_t hidden)
{
    lb_eui_blk_draw_info_t draw_info;

    draw_info.idx = idx;
    draw_info.type = LB_EUI_BLK_DRAW_NORMAL;



    if (idx == 9)
        draw_info.buff = bsd_left_buff;
    else if (idx == 10)
        draw_info.buff = bsd_right_buff;
    else {
        APP_LOG_D("It is not bsd draw!\n");
        return 0;
```

```
    }

    if (hidden == 1) {
        draw_info.rect.x1 = 0;
        draw_info.rect.x2 = 1;
        draw_info.rect.y1 = 0;
        draw_info.rect.y2 = 1;
    } else {
        if (idx == 9) {
            draw_info.rect.x1 = 0;
            draw_info.rect.x2 = BSD_BKL_LINE_WIDTH_MAX;
            draw_info.rect.y1 = 0;
            draw_info.rect.y2 = BSD_BKL_LINE_HEIGHT_MAX;
        } else if (idx == 10) {
            draw_info.rect.x1 = 0;
            draw_info.rect.x2 = BSD_BKL_LINE_WIDTH_MAX;
#ifdef SCREEN_ROT_90
            draw_info.rect.y1 = cdr_screen.width -
                        BSD_BKL_LINE_HEIGHT_MAX;
            draw_info.rect.y2 = cdr_screen.width;
#else
            draw_info.rect.y1 = cdr_screen.height -
                        BSD_BKL_LINE_HEIGHT_MAX;
            draw_info.rect.y2 = cdr_screen.height;
#endif
        }
        draw_info.max_height = BSD_BKL_LINE_HEIGHT_MAX;
        draw_info.max_width = BSD_BKL_LINE_WIDTH_MAX;
    }

    lb_eui_bl_draw_buff(draw_info);

    return 0;
}

int cdr_adas_draw_line(uint8_t idx, lb_eui_rect_t rect,
    lb_eui_point_t *points, uint8_t hidden)
```

```
{
    lb_eui_blk_draw_info_t draw_info;

    draw_info.idx = idx;
    draw_info.type = LB_EUI_BLK_DRAW_POLYGON;

    if (cur_line_buff_idx == 0) {
        rt_memset(line_buff[1], 0, ADAS_BKL_LINE_SIZE_MAX);
        draw_info.buff = line_buff[1];
        cur_line_buff_idx = 1;
    } else {
        rt_memset(line_buff[0], 0, ADAS_BKL_LINE_SIZE_MAX);
        draw_info.buff = line_buff[0];
        cur_line_buff_idx = 0;
    }

    if (hidden == 1) {
        draw_info.rect.x1 = 0;
        draw_info.rect.x2 = 1;
        draw_info.rect.y1 = 0;
        draw_info.rect.y2 = 1;
    } else {
        memcpy(&draw_info.rect, &rect, sizeof(lb_eui_rect_t));
        memcpy(draw_info.polygon.points, points, sizeof(lb_eui_point_t) * 4);
        draw_info.polygon.size = 4;
        draw_info.polygon.color = BL_DRAW_LINE_COLOR;

        draw_info.max_height = ADAS_BKL_LINE_HEIGHT_MAX;
        draw_info.max_width = ADAS_BKL_LINE_WIDTH_MAX;
    }

    lb_eui_bl_draw_buff(draw_info);

    return 0;
}

int cdr_adas_draw_rect(uint8_t idx, lb_eui_rect_t rect,
```

```
        uint8_t color_type, uint8_t hidden)
{
    lb_eui_blk_draw_info_t draw_info;

    draw_info.idx = idx;
    draw_info.type = LB_EUI_BLK_DRAW_NORMAL;

    if (color_type == 0)
        draw_info.buff = green_buff;
    else
        draw_info.buff = red_buff;

    if (hidden == 1) {
        draw_info.rect.x1 = 0;
        draw_info.rect.x2 = 1;
        draw_info.rect.y1 = 0;
        draw_info.rect.y2 = 1;
    } else {
        memcpy(&draw_info.rect, &rect, sizeof(lb_eui_rect_t));
        draw_info.max_height = ADAS_BKL_RECT_HEIGHT_MAX;
        draw_info.max_width = ADAS_BKL_RECT_WIDTH_MAX;
    }

    lb_eui_bl_draw_buff(draw_info);

    return 0;
}
```

5.6.3. 退出与释放 blocklinker

```
static int cdr_draw_release(void)
{
    int i = 0;
    int en_adas = 0;
    int en_bsd = 0;

    en_adas = adas_get_enable();
```

```
en_bsd = bsd_get_enable();

if (green_buff) {
    rt_free(green_buff);
    green_buff = NULL;
}

if (red_buff) {
    rt_free(red_buff);
    red_buff = NULL;
}

if (line_buff[0]) {
    rt_free(line_buff[0]);
    line_buff[0] = NULL;
}

if (line_buff[1]) {
    rt_free(line_buff[1]);
    line_buff[1] = NULL;
}

if (bsd_left_buff) {
    rt_free(bsd_left_buff);
    bsd_left_buff = NULL;
}

if (bsd_right_buff) {
    rt_free(bsd_right_buff);
    bsd_right_buff = NULL;
}

if (en_adas == ENABLE_ON && en_bsd != ENABLE_ON) {
    for (i = 0; i < ADAS_BKL_NUM_MAX; i++)
        lb_eui_bl_draw_release(i);
} else if (en_adas != ENABLE_ON && en_bsd == ENABLE_ON) {
    for (i = ADAS_BKL_NUM_MAX; i < ADAS_BKL_NUM_MAX + BSD_BKL_NUM_MAX; i++)
```

```
            lb_eui_bl_draw_release(i);
    } else if (en_adas == ENABLE_ON && en_bsd == ENABLE_ON) {
        for (i = 0; i < ADAS_BKL_NUM_MAX + BSD_BKL_NUM_MAX; i++)
            lb_eui_bl_draw_release(i);
    }

    return 0;
}

int cdr_draw_exit(void)
{
    int err = 0;

    err = cdr_draw_release();

    err = lb_eui_bl_draw_exit();

    return err;
}
```

5.7. 时钟

时钟是用 gauge（仪表）控件实现的简单示例，代码如下：

```
static void *clock_refr(void *param)
{
    static lb_int16 sec = 0;
    static lb_int16 min = 21;
    static lb_int16 hour = 2;

    lb_eui_gauge_set_value(D_G_GAUGE_322, CLOCK_HOUR, hour * 360 / 12 + min * 30 / 60);
    lb_eui_gauge_set_value(D_G_GAUGE_322, CLOCK_MIN, min * 360 / 60);
    lb_eui_gauge_set_value(D_G_GAUGE_322, CLOCK_SEC, sec);

    while (need_exit != 1) {
        if (sec >= 60) {
            sec = 0;
            min++;
```

```
            if (min >= 60) {
                min = 0;
                hour++;
                if (hour >= 12)
                    hour = 0;

            }
            lb_eui_gauge_set_value(D_G_GAUGE_322, CLOCK_MIN, min * 360 / 60);
            lb_eui_gauge_set_value(D_G_GAUGE_322, CLOCK_HOUR,
                hour * 360 / 12 + min * 30 / 60);
        }

        lb_eui_gauge_set_value(D_G_GAUGE_322, CLOCK_SEC, sec * 360 / 60);

        sec++;
        lb_eui_common_sleep(1000);
    }

    pthread_exit(0);
    return NULL;
}

static lb_int32 clock_init(void)
{
    pthread_attr_t tmp_attr;
    struct sched_param shed_param;
    lb_int32 ret = 0;

    if ((void *)clock_id) {
        printf("failed\n");
        ret = 0;
        goto exit;
    }

    need_exit = 0;

    ret = pthread_attr_init(&tmp_attr);
```

```
if (ret != 0) {
        printf("failed\n");
        ret = -1;
        goto exit;
}

ret = pthread_attr_setscope(&tmp_attr, PTHREAD_SCOPE_SYSTEM);
if (ret != 0) {
        printf("failed\n");
        ret = -1;
        goto exit;
}

ret = pthread_attr_getschedparam(&tmp_attr, &shed_param);
if (ret != 0) {
        printf("failed\n");
        ret = -1;
        goto exit;
}

shed_param.sched_priority = CLOCK_PRIO;
ret = pthread_attr_setschedparam(&tmp_attr, &shed_param);
if (ret != 0) {
        printf("failed\n");
        ret = -1;
        goto exit;
}

ret = pthread_attr_setstacksize(&tmp_attr, (size_t)CLOCK_SIZE);
if (ret != 0) {
        printf("failed\n");
        ret = -1;
        goto exit;
}

ret = pthread_create(&clock_id, &tmp_attr, &clock_refr, NULL);
if (ret != 0) {
```

```
            printf("failed\n");
            ret = -1;
            goto exit;
        }

        pthread_attr_destroy(&tmp_attr);

exit:
        return ret;
}

static lb_int32 clock_exit(void)
{
        lb_int32 ret = 0;

        if ((void *)clock_id) {
            need_exit = 1;
            pthread_join(clock_id, NULL);
            clock_id = (pthread_t)NULL;
        }

        return ret;
}

lb_int32 eui_demo_gauge_clock(void *param)
{
        lb_int32        err = 0;
        lb_uint8        needle_count = 3;
        lb_eui_gauge_needle_t  needles[] = {
            {7, 80, 0xFF4400FF}, {5, 100, 0xFF00FF00}, {3, 130, 0xFFFF0000},
        };
        lb_uint16       angle = 360;
        lb_uint8        line_cnt = 61;
        lb_uint8        label_cnt = 13;
        lb_int16        label_val[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

        if (param == NULL) {
```

```
            printf("Invalid param!\n");
            err = -1;
            goto exit;
        }

        err |= lb_eui_label_set_text_id(D_G_PTOMPT_EFFECT_324,
            lb_eui_common_lang_get_string_id_json("STR_TEXT_GUAGE_PROMPT_7"));
        err |= lb_eui_gauge_set_range(D_G_GAUGE_322, 0, 360);
        err |= lb_eui_gauge_set_scale(D_G_GAUGE_322,
            angle, line_cnt, label_cnt, label_val);
        err |= lb_eui_gauge_set_needle_count(D_G_GAUGE_322, needle_count, needles);
        err |= clock_init();

exit:
        return err;
}
```