



# XJ380 API 标准文档

XJ380 API Specification

C\C++ 版

版权所有© XINGJI Interactive Software 2017 – 2025 保留所有权利。“星际工作室”“XINGJI 工作室”“XINGJI Studios”均为 XINGJI Interactive Software 的别名。“XJ380”“XJ380OS”“XJ380 操作系统”均为 XJ380 操作系统的别名，归 XINGJI Interactive Software 所有。“BrigeEngine”“鹊桥引擎”“bapi”均为 BrigeEngine 的别名，归 XINGJI Interacitve Software 及其旗下工作室 XINGJI Games 所有。“xapi”“XJ380 API”“XJ380 应用程序接口”均为 XJ380 API 别名，归 XINGJI Interactive Software 所有。除适用于 POSIX 标准的 XJ380 API 外最终解释权归 XINGJI 工作室所有。适用于 POSIX 标准的 XJ380 API 最终解释权归 IEEE 所有。本手册由 XINGJI 董事会组织编写。工作室总部地址：太阳公南街 8 号。请 勿邮寄任何快递，寄往此地址的快递我们不会收货(也没法收货)。如有邮寄需要，请联系 XINGJI 工作室董事会。更多信息请参见 XINGJI 工作室官网。

# 目录

## CONTENTS

### CHAPTER 1 – 关于、编译和专有类型

- 1.1 关于
- 1.2 编译为 XJ380 应用程序
- 1.3 专有类型概览

### CHAPTER 2 – XJ380 API (POSIX 版)

- 2.1 简介

### CHAPTER 3 – XJ380 API (XAPI 版)

- 3.1 文本输入输出
  - 3.1.1 xapi\_Output();
  - 3.1.2 xapi\_InputW();
  - 3.1.3 xapi\_Input();
  - 3.1.4 xapi\_EndLine();
  - 3.1.5 xapi\_PrintLine();
  - 3.1.6 xapi\_Printf();
- 3.2 文件操作
  - 3.2.1 XFILE 类型
  - 3.2.2 xapi\_OpenFile();
  - 3.2.3 xapi\_CloseFile();
- 3.3 类型转换
  - 3.3.1 xcr\_char2int();
  - 3.3.2 xcr\_int2char();
  - 3.3.3 xcr\_hex2char();
  - 3.3.4 xcr\_toRGB()
- 3.4 进程
  - 3.3.1 xapi\_fork();
  - 3.3.2 xapi\_execve();

### CHAPTER 4 – XJ380 API (XAPI GUI 版)

- 4.1 创建图形化应用程序
  - 4.1.1 XWINDOW 类型
  - 4.1.2 xapi\_CreateWindow();
  - 4.1.3 xapi\_SetWindowTitle();
  - 4.1.4 xapi\_CloseWindow();
  - 4.1.5 xapi\_SetIcon();

## 4.2 绘图

4.2.1 xapi\_DrawPoint();

4.2.2 xapi\_DrawLine();

4.2.3 xapi\_DrawRect();

4.2.4 xapi\_DrawCircle();

4.2.5 xapi\_DrawText();

## 4.3 插入图片

4.3.1 xapi\_DrawBMP();

4.3.2 xapi\_DrawPNG();

## 4.4 消息处理

4.4.1 消息处理函数

4.4.2 键盘消息

4.4.2.1 CHAR 消息

4.4.3 鼠标消息

4.4.3.1 MOVE 消息

4.4.3.2 LBUTTON 消息

4.4.3.3 RBUTTON 消息

4.4.3.4 MBUTTON 消息

4.4.3.5 ROLLER 消息

4.4.4 控件消息

## 4.5 对 framebuffer 进行操作

4.5.1 xapi\_ReadBuffer();

4.5.2 xapi\_WriteBuffer();

4.5.3 xapi\_WriteBufferA();

## 4.6 控件

4.6.1 按钮控件

4.6.1.1 xapi\_Button();

4.6.1.2 xapi\_StyleButton();

# CHAPTER 5 – BrigeEngine API

## 5.1 简介

# CHAPTER 1

## 关于、编译和专有类型

### ABOUT, COMPILE AND EXCLUSIVE TYPE

本章节将会介绍本篇手册以及 XJ380 API 的相关信息、如何编译为 XJ380 应用程序 (EPF) 以及本手册内的所有专有类型。

#### 1-1 关于

XJ380 API 由 3 (或 4) 部分组成: 兼容 POSIX 的 XJ380 API、XAPI 版 XJ380 API 和带 GUI 的 XAPI 版 XJ380 API (XJ380 Professional Edition 或更高版本默认集成 BrigeEngine 引擎, 可使用 BAPI。详见 BrigeEngine API 标准文档)。其中兼容 POSIX 的 XJ380 API 和 XAPI 版 XJ380 API 仅可创建控制台程序。带 GUI 的 XAPI 版 XJ380 API 和 BrigeEngine BAPI 可用于创建图形化应用程序。这 3 (或 4) 种 API 可以混合使用。

#### 1-2 编译为 XJ380 应用程序

您可以在 <https://www.xingjisoft.top/os/xj380/download> 处下载 XJ380 应用程序编译套件, 或使用 SpaceCode 系列编辑器进行编译 (须安装额外模块)。您也可以自行链接 XAPI 库。请务必使用 C++11 或更高标准。

您需要引入下列头文件的其中至少一个:

头文件名	用途
<b>x3api.h</b>	引入所有 XJ380 API (除 BAPI) (建议使用)
xposix.h	引入 POSIX 版 XJ380 API
xguiapi.h	引入 XAPI (GUI 版)
xtuiapi.h	引入 XAPI (无 GUI 版)
<b>xapi.h</b>	引入所有 XAPI (建议使用)
BridgeEngine.h	引入所有 BAPI
x4api.h	引入所有 XJ380 API

除 bapi 外, 这些头文件均可以在 XJ380 应用程序编译套件中的 include 文件夹找到。  
bapi 相关头文件可以在 BrigeEngine 开发套件中找到。  
请您务必按照要求编写主函数。主函数格式如下:

### 主函数格式

```
int main(int argc, char* argv, char* envp);
```

其中 argv、envp 为传参。

## 1-3 专有类型概览

类型名称	长度（字节）	概述
INT8	1	8 位整型
UINT8	1	无符号 8 位整型
INT16	2	16 位整型
UINT16	2	无符号 16 位整型
INT32	4	32 位整型
UINT32	4	无符号 32 位整型
INT64	8	64 位整型
UINT64	8	无符号 64 位整型
WSTR	-	char 类型字符串，使用 UTF8 编码
XWINDOW	详见 4-1-1	窗口类型
XFILE	详见 3-2-1	文件类型
XCOLOR	6	包含 3 个无符号 8 位整型。
XCOLORA	8	包含 4 个无符号 8 位整型。

表格 1-3-1

# CHAPTER 2

## XJ380 API (POSIX 版)

### XJ380 API (POSIX EDITION)

XJ380 API (POSIX 版) 遵守且兼容小部分 POSIX 标准。

### 2-1 简介

自己看 POSIX 标准和头文件去

# CHAPTER 3

## XJ380 API (XAPI 版)

### XJ380 API (XAPI EDITION)

XJ380 API (XAPI 版) 是由 XINGJI Interactive Software 自主设计的 API 标准。所有 XAPI 均向下兼容低版本。3-3 中的类型转换函数严格意义上并不算做 API，但作为 XAPI 提供的一部分仍被写在这里。

### 3-1 文本输入输出

#### 3-1-1 xapi\_Output();

该函数将会向控制台输出一段字符串。

函数参数

```
void xapi_Output(WSTR str);
```

**str** 将要输出的字符串。

返回值 无

#### 3-1-2 xapi\_InputW();

该函数将会从控制台读取字符串。在读取到字符串前不会继续执行。

函数参数

```
WSTR xapi_InputW(void);
```

返回值 获取的字符串。

#### 3-1-3 xapi\_Input();

该函数将会从控制台读取字符串。在读取到字符串前不会继续执行。

我也不知道跟上面的有什么区别，但是建议使用这个，鬼知道我们哪天会不会突然给 3-1-2 加点什么功能 (((



函数参数

```
WSTR xapi_Input(void);
```

返回值 获取的字符串

### 3-1-4 xapi\_EndLine();

该函数将会向控制台输出换行。

函数参数

```
void xapi_EndLine(void);
```

返回值 无

### 3-1-5 xapi\_PrintLine();

该函数将会向控制台输出一段字符串并换行。

函数参数

```
void xapi_PrintLine(WSTR str);
```

str 将要输出的字符串。

返回值 无

## 3-2 文件操作

### 3-2-1 XFILE 类型

类型结构

```
typedef struct {  
    UINT64 length;  
    void* buffer;  
} XFILE;
```

length 文件长度（单位字节）。

buffer 文件内容。

### 3-2-2 xapi\_OpenFile();

该函数将会打开文件并读取。

函数参数

```
void xapi_OpenFile(  
    WSTR path,  
    XFILE * fsptr  
);
```

**path** 文件路径。

**fsptr** 指向 XFILE 类型结构体的指针。如果执行后为 NULL 则代表读取失败。

**返回值** 无

## 3-2-3 xapi\_CloseFile();

该函数将会关闭文件并释放内存。

函数参数

```
void xapi_CloseFile(XFILE * fsptr);
```

**fsptr** 指向 XFILE 类型结构体的指针。执行后会被设置为空指针。

**返回值** 无

## 3-3 类型转换

### 3-3-1 xcr\_char2int();

该函数将会把字符串转换为整型。

函数参数

```
UINT64 xcr_char2int(WSTR str);
```

**str** 字符串。

**返回值** 整型。

### 3-3-2 xcr\_int2char();

该函数将会把整型转换为字符串。

函数参数

```
WSTR xcr_int2char(UINT64 dec);
```

**dec** 整型。

**返回值** 字符串

### 3-3-3 xcr\_hex2char();

该函数将会把整型转换为 16 进制格式的字符串。

#### 函数参数

```
WSTR xcr_hex2char(UINT64 hex);
```

**hex** 整型。

**返回值** 字符串

## 3-3-4 toRGB();

该宏将会将 3 份数据（分别代表 R G B）转换为 32 位整型格式。

#### 函数参数

```
UINT32 toRGB(  
    UINT8 r,  
    UINT8 g,  
    UINT8 b  
);
```

**R** 整型。代表 R 通道。

**G** 整型。代表 G 通道。

**B** 整型。代表 B 通道。

**返回值** 字符串。

## 3-4 进程

### 3-4-1 xapi\_fork();

该函数会复制当前进程映像至新进程，子进程将从 fork()后开始运行。

#### 函数参数

```
UINT64 xapi_fork(void);
```

**返回值** 父进程：子进程 PID

子进程：0。

### 3-4-2 xapi\_execve()

该函数会使目的可执行文件替换当前进程映像。

## 函数参数

```
UINT64 xapi_execve(  
    WSTR filename,  
    WSTR argv[],  
    WSTR envp[]  
);
```

**filename** 可执行文件路径。

**argv[]** 传给新程序的命令行参数字符串数组。

**envp[]** 传给新程序的环境变量字符串数组。

**返回值** 无：执行成功  
-1：执行失败。

# CHAPTER 4

## XJ380 API (XAPI GUI 版)

### XJ380 API (XAPI GUI EDITION)

本章节将会介绍 XAPI 的 GUI 版本。坐标系以窗口左上角（不含标题栏）为(0, 0)。

## 4-1 创建图形化应用程序

### 4-1-1 XWINDOW 类型

类型结构

```
typedef struct {  
    UINT32 width;  
    UINT32 heigh;  
    WSTR title;  
    UINT8 sets;  
} XWINDOW;
```

**width** 窗口宽度。

**height** 窗口高度（不包含标题栏）。

**title** 标题。

**sets** 窗口参数。如表格 4-1-1-1。可使用多个不同类型的参数，如

参数	说明
NORMAL	使用默认设定。
FRAME_OFF	创建无边框窗口。
FULL_SCR	全屏。
TOP_INDEX	始终置顶。

表格 4-1-1-1

### 4-1-2 xapi\_CreateWindow();

该函数将会创建一个窗口。

函数参数

```
void xapi_CreateWindow(  
    HDLE* handle,  
    XWINDOW* xwin  
);
```

**handle** 指向窗口句柄的指针。即指向 HDLE 类型的变量的指针。

**xwin** 一个指向存储了窗口参数的 XWINDOW 类型的指针。

返回值 无

### 4-1-3 xapi\_SetWindowTitle();

该函数将会设置窗口的标题。

函数参数

```
void xapi_SetWindowTitle(  
    HDLE handle,  
    WSTR str  
);
```

**handle** 窗口句柄。

**str** 标题。

返回值 无

### 4-1-4 xapi\_CloseWindow();

该函数将会关闭窗口，但不会结束程序。

函数参数

```
void xapi_CloseWindow(HDLE handle);
```

**handle** 窗口句柄。

返回值 无

### 4-1-5 xapi\_SetIcon();

该函数将会设置窗口在控制栏和标题栏的图标。

函数参数

```
void xapi_SetIcon(  
    HDLE handle,  
    WSTR path  
);
```

**handle** 窗口句柄。

**path** 图标路径。须为 bmp 格式，大小 16\*16。

返回值 无

## 4-2 绘图

## 4-2-1 xapi\_DrawPoint();

该函数将会在窗口上绘制点。

### 函数参数

```
void xapi_DrawPoint (  
    HDLE handle,  
    UINT32 x,  
    UINT32 y,  
    UINT32 color  
);
```

**handle** 窗口句柄。

**x** 横坐标。

**y** 竖坐标。

**color** 图形的颜色。采用 RGBA（32 位色）格式。

**返回值** 无

## 4-2-2 xapi\_DrawLine();

该函数将会在窗口上绘制线。

### 函数参数

```
void xapi_DrawLine(  
    HDLE handle,  
    UINT32 x1,  
    UINT32 y1,  
    UINT32 x2,  
    UINT32 y2,  
    UINT32 color  
);
```

**handle** 窗口句柄。

**x1** 起始点横坐标。

**y1** 起始点竖坐标。

**x2** 中止点横坐标。

**y2** 中止点竖坐标。

**color** 图形的颜色。采用 RGBA（32 位色）格式。

**返回值** 无

## 4-2-3 xapi\_DrawRect();

该函数将会在窗口上绘制矩形。

## 函数参数

```
void xapi_DrawRect(  
    HDLE handle,  
    UINT32 x1,  
    UINT32 y1,  
    UINT32 x2,  
    UINT32 y2,  
    UINT32 color,  
    bool fill  
);
```

**handle** 窗口句柄。

**x1** 起始点横坐标。

**y1** 起始点竖坐标。

**x2** 中止点横坐标。

**y2** 中止点竖坐标。

**color** 图形的颜色。采用 RGBA（32 位色）格式。

**fill** 是否为实心。

**返回值** 无

## 4-2-4 xapi\_DrawCircle();

该函数将会在窗口上绘制正圆。由于算法问题该函数被暂时移除。



## 4-2-5 xapi\_DrawText();

该函数将会在窗口上绘制字符串。

### 函数参数

```
void xapi_DrawText(  
    HDLE handle,  
    UINT32 x,  
    UINT32 y,  
    WSTR str,  
    UINT32 size,  
    UINT32 color  
);
```

**handle** 窗口句柄。

**x** 横坐标。

**y** 竖坐标。

**str** 字符串。

**size** 字号（请注意单位不是像素）。

**color** 图形的颜色。采用 RGBA（32 位色）格式。

**返回值** 无

## 4-3 插入图片

### 4-3-1 xapi\_DrawBMP();

该函数将会在窗口上绘制 BMP 格式的图片。

### 函数参数

```
void xapi_DrawBMP (  
    HDLE handle,  
    UINT32 x,  
    UINT32 y,  
    UINT32 width,  
    UINT32 height,  
    WSTR path  
);
```

**handle** 窗口句柄。

**x** 横坐标。

**y** 竖坐标。

**width** 图片宽度。会进行拉伸。

**height** 图片高度。会进行拉伸。

**path** 图片路径。  
**返回值** 无

## 4-3-2 xapi\_DrawPNG();

该函数将会在窗口上绘制 PNG 格式的图片。

函数参数

```
void xapi_DrawPNG(  
    HDLE handle,  
    UINT32 x,  
    UINT32 y,  
    UINT32 width,  
    UINT32 height,  
    WSTR path  
);
```

**handle** 窗口句柄。  
**width** 图片宽度。会进行拉伸。  
**height** 图片高度。会进行拉伸。  
**x** 横坐标。  
**y** 竖坐标。  
**path** 图片路径。  
**返回值** 无

## 4-4 消息处理

### 4-4-1 消息处理函数

为了进行人机交互，应用程序需要自行设定一个消息处理函数用于处理除窗口操作外的其他操作。消息处理函数必须为固定格式，并通过 `SetMsgPrcor()` 进行设置。XJ380 回向处理函数传入消息类型码以及 128 位的数据。仅在选中该窗口时才会将消息传入该窗口。

消息处理函数固定格式：

```
void Function(UINT64 Type, UINT64 hData, UINT64 lData) {  
    /* 代码 Code*/  
}
```

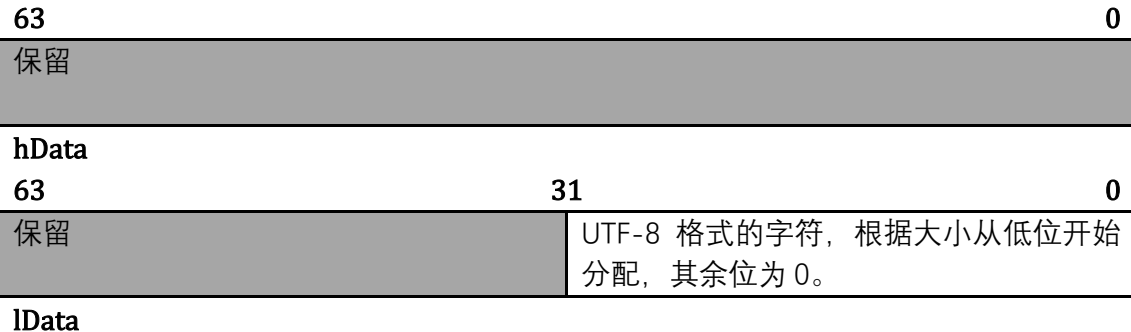
**Type** 消息类型。用于识别消息。  
**hData** 数据（高 64 位）。不同消息的内容分布不同。  
**lData** 数据（低 64 位）。不同消息的内容分布不同。

## 4-4-2 键盘消息

### 4-4-2-1 CHAR 消息

消息识别码（宏）：**MSG\_CHAR**

数据分布结构：



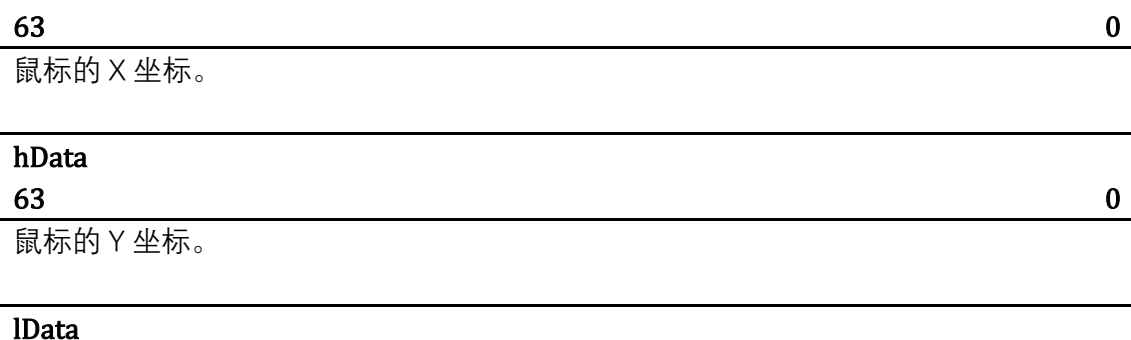
## 4-4-3 鼠标消息

### 4-4-3-1 MOVE 消息

代表鼠标进行了移动。

消息识别码（宏）：**MSG\_MOVE**

数据分布结构：



### 4-4-3-2 LBUTTON 消息

代表左键被按下。

消息识别码（宏）：**MSG\_LBUTTON**

数据分布结构：



#### 4-4-3-3 RBUTTON 消息

代表右键被按下。

消息识别码（宏）：**MSG\_RBUTTON**

数据分布结构：



lData

#### 4-4-3-4 MBUTTON 消息

代表中键被按下

消息识别码（宏）：**MSG\_MBUTTON**

数据分布结构：



lData

#### 4-4-3-5 ROLLER 消息

代表滚轮被滚动。

消息识别码（宏）：**MSG\_ROLLER**

数据分布结构：



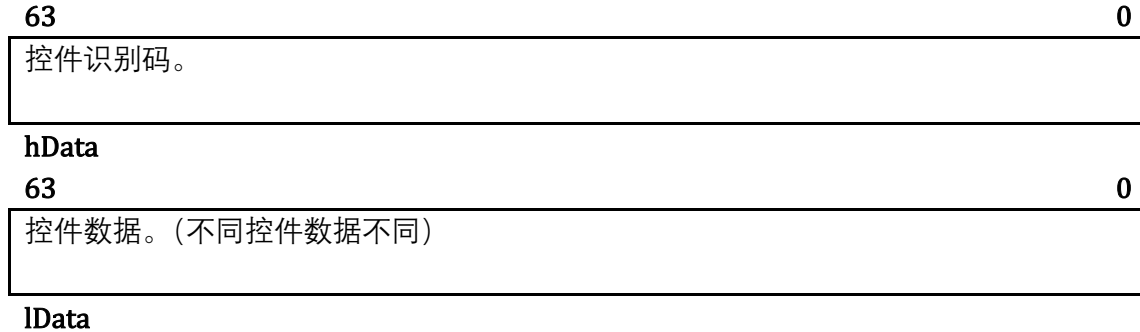
lData

## 4-4-4 控件消息

用于接收控件消息。例如按钮被按下。

消息识别码（宏）：**MSG\_CRL**

数据分布结构：



## 4-5 对 framebuffer 进行操作

用于直接操作窗口的 framebuffer。

### 4-5-1 xapi\_ReadBuffer();

该函数将会获取 framebuffer 中的指定区域。

函数参数

```
void xapi_ReadBuffer(  
    HDLE      handle,  
    UINT32    x,  
    UINT32    y,  
    UINT32    width,  
    UINT32    height,  
    XCOLOR*   buffer  
);
```

**handle** 窗口句柄。

**x** 起始横坐标。

**y** 起始竖坐标。

**width** 宽度。

**height** 高度。

**buffer** 图像。

**返回值** 无

## 4-5-2 xapi\_WriteBuffer();

该函数将会写入 framebuffer 中的指定区域。

### 函数参数

```
void xapi_WriteBuffer(  
    HDLE      handle,  
    UINT32    x,  
    UINT32    y,  
    UINT32    width,  
    UINT32    height,  
    XCOLOR*   buffer  
);
```

**handle** 窗口句柄。

**x** 起始横坐标。

**y** 起始竖坐标。

**width** 宽度。

**height** 高度。

**buffer** 图像。

**返回值** 无

## 4-5-3 xapi\_WriteBufferA();

该函数将会写入 framebuffer 中的指定区域（包含透明色）。

### 函数参数

```
void xapi_WriteBufferA(  
    HDLE      handle,  
    UINT32    x,  
    UINT32    y,  
    UINT32    width,  
    UINT32    height,  
    XCOLORA*  buffer  
);
```

**handle** 窗口句柄。

**x** 起始横坐标。

**y** 起始竖坐标。

**width** 宽度。

**height** 高度。

**buffer** 图像（含 Alpha 通道）。

**返回值** 无

## 4-6 控件

用于更简易的创建用户界面。

### 4-6-1 按钮控件

#### 4-6-1-1 xapi\_Button()

该函数将会在窗口上创建一个按钮控件。

函数参数

```
void xapi_Button(  
    HDLE handle,  
    UINT64 CRLid,  
    WSTR text  
);
```

**handle** 窗口句柄。  
**CRLid** 控件识别码。  
**text** 文本。  
**返回值** 无  
**控件数据** 无

#### 4-6-1-1 xapi\_StyleButton()

该函数将会在窗口上创建一个按钮控件并将指定的 PNG 图片作为按钮样式。

函数参数

```
void xapi_StyleButton(  
    HDLE handle,  
    UINT64 CRLid,  
    WSTR text,  
    WSTR path  
);
```

**handle** 窗口句柄。  
**CRLid** 控件识别码。  
**text** 文本。  
**path** 图片路径。  
**返回值** 无  
**控件数据** 无

# CHAPTER 5

## BridgeEngine API

### BridgeEngine API

本章节将会介绍鹊桥引擎的 BAPI。

### 5-1 简介

BridgeEngine（鹊桥引擎）是由 XINGJI 工作室开发的一款跨平台图形库及 2.5D 游戏引擎。请移步 **BridgeEngine BAPI 标准文档**。