目录

SDK3.0文档概述

设备ID中间带D的才是工程卡, 不是工程卡不支持二次开发

> 注: 看标题头, 确保是工程卡再看其他的

1. SDK3.0协议兼容2.0文档, 连接流程相同.
2. TCP数据包最大长度为9 x 1024字节. xml长度超过9 x 1024 - 4时, 需要做分包处理.
3. SDK数据均采用UTF-8编码格式传输
4. SDK请求信息命令值均是 **kSDKCmdAsk**
5. 此文档非最终版, 如出现新的功能且不在文档信息中, 请更新到最新文档. 如最新文档不存在这些信息, 可以申 请在文档中增加相关文档.

设备ID中间带D的才是工程卡, 不是工程卡不支持二次开发

> 注: 看标题头, 确保是工程卡再看其他的

SDK连接所需的步骤

1. 进行Tcp连接.
2. 进行连接版本协商
3. 进行SDK协商(GetIFVersion), 为了获取后续交互使用的guid
4. 已完成SDK协商, 后续根据SDK协议进行xml数据交互
5. SDK协议数据使用小端字节序
6. SDK接口 数据使用UTF-8编码格式

提问SDK相关问题

1. 首先如果是功能性问题, 那么最初应该使用SDK测试工具验证是否是设备问题.
2. 如果是设备问题, 提出问题的同时需要告知其设备的固件版本号. 可通过**HDPlayer**获取或者SDK的**GetDeviceInfo**来获取.
3. 如果不是设备问题. 那么提问时也需要说明其版本号, 并告知是如何复现该问题和发送的内容. 让解决问题的人员知道复现流程. 那么这问题将很快的进入处理阶段, 而不是让解决人员来不断测试复现这个问题, 这将延迟问题的解决时间.

基本类型定义

```
typedef unsigned long long  huint64;
typedef unsigned int        huint32;
typedef unsigned short      huint16;
typedef unsigned char       huint8;

typedef long long           hint64;
typedef int                 hint32;
typedef short               hint16;
typedef char                hint8;
typedef float               hfloat;
typedef double              hdouble;

#define MAX_DEVICE_ID_LENGHT    15
#define MD5_LENGHT              32
#define LOCAL_TCP_VERSION       0x1000007
#define LOCAL_UDP_VERSION       0x1000007
```

错误码定义

错误码返回 格式如下:

字段	命令包长度(Len)	命令(Cmd)	错误码(Code)
字节数	2字节	2字节	2字节
取值	6	kErrorAnswer	HErrorCode

HErrorCode定义

数值从0开始依次递增, 如正常状态0, 下一个写文件完成1, 流程错误2

```
enum HErrorCode
{
    kSuccess = 0,              ///< 正常状态
    kWriteFinish,              ///< 写文件完成
    kProcessError,             ///< 流程错误
    kVersionTooLow,            ///< 版本过低
    kDeviceOccupa,             ///< 设备被占用
    kFileOccupa,               ///< 文件被占用
    kReadFileExcessive,        ///< 回读文件用户过多
    kInvalidPacketLen,         ///< 数据包长度错误
    kInvalidParam,             ///< 无效的参数
    kNotSpaceToSave,           ///< 存储空间不够
    kCreateFileFailed,         ///< 创建文件失败
    kWriteFileFailed,          ///< 写文件失败
    kReadFileFailed,           ///< 读文件失败
    kInvalidFileData,          ///< 无效的文件数据
    kFileContentError,         ///< 文件内容出错
    kOpenFileFailed,           ///< 打开文件失败
    kSeekFileFailed,           ///< 定位文件失败
    kRenameFailed,             ///< 重命名失败
    kFileNotFound,             ///< 文件未找到
    kFileNotFinish,            ///< 文件未接收完成
    kXmlCmdTooLong,            ///< xml命令过长
    kInvalidXmlIndex,          ///< 无效的xml命令索引值
    kParseXmlFailed,           ///< 解析xml出错
    kInvalidMethod,            ///< 无效的方法名
    kMemoryFailed,             ///< 内存错误
    kSystemError,              ///< 系统错误
    kUnsupportVideo,           ///< 不支持的视频
    kNotMediaFile,             ///< 不是多媒体文件
    kParseVideoFailed,         ///< 解析视频文件失败
    kUnsupportFrameRate,       ///< 不支持的波特率
    kUnsupportResolution,      ///< 不支持的分辨率(视频)
    kUnsupportFormat,          ///< 不支持的格式(视频)
    kUnsupportDuration,        ///< 不支持的时间长度(视频)
    kDownloadFileFailed,       ///< 下载文件失败
    kScreenNodeIsNull,         ///< 显示屏节点为空
    kNodeExist,                ///< 节点已存在
    kNodeNotExist,             ///< 节点不存在
```

```
    kPluginNotExist,          ///< 插件不存在
    kCheckLicenseFailed,      ///< 校验license失败
    kNotFoundWifiModule,      ///< 未找到wifi模块
    kTestWifiUnsuccessful,    ///< 测试wifi模块未
    kRunningError,            ///< 运行错误
    kUnsupportMethod,         ///< 不支持的方法
    kInvalidGUID,             ///< 非法的guid
    kFirmwareFormatError,     ///< 固件格式错误
    kTagNotFound,             ///< 标签不存在
    kAttrNotFound,            ///< 属性不存在
    kCreateTagFailed,         ///< 创建标签失败
    kUnsupportDeviceType,     ///< 不支持的设备型号
    kPermissionDenied,        ///< 权限不足
    kPasswdTooSimple,         ///< 密码太简单
    kUsbNotInsert,            ///< usb未插入
};
```

数据包命令值定义

```
enum HcmdType
{
    kTcpHeartbeatAsk        = 0x005f,   ///< TCP心跳包请求
    kTcpHeartbeatAnswer     = 0x0060,   ///< TCP心跳包反馈
    kSearchDeviceAsk        = 0x1001,   ///< 搜索设备请求
    kSearchDeviceAnswer     = 0x1002,   ///< 搜索设备应答
    kErrorAnswer            = 0x2000,   ///< 出错反馈
    kSDKServiceAsk          = 0x2001,   ///< 版本协商请求
    kSDKServiceAnswer       = 0x2002,   ///< 版本协商应答
    kSDKCmdAsk              = 0x2003,   ///< sdk命令请求
    kSDKCmdAnswer           = 0x2004,   ///< sdk命令反馈
    kGPSInfoAnswer          = 0x3007,   ///< gps信息应答
    kFileStartAsk           = 0x8001,   ///< 文件开始传输请求
    kFileStartAnswer        = 0x8002,   ///< 文件开始传输应答
    kFileContentAsk         = 0x8003,   ///< 携带文件内容的请求
    kFileContentAnswer      = 0x8004,   ///< 写文件内容的应答
    kFileEndAsk             = 0x8005,   ///< 文件结束传输请求
    kFileEndAnswer          = 0x8006,   ///< 文件结束传输应答
    kReadFileAsk            = 0x8007,   ///< 回读文件请求
    kReadFileAnswer         = 0x8008,   ///< 回读文件应答
};
```

协议数据包结构格式

> 协议数据包主要用于:
>
> 1. 文件的下发和回读
> 2. Xml格式的sdk请求和反馈

字段	命令包长度(len)	命令(cmd)
字节数	2字节	2字节

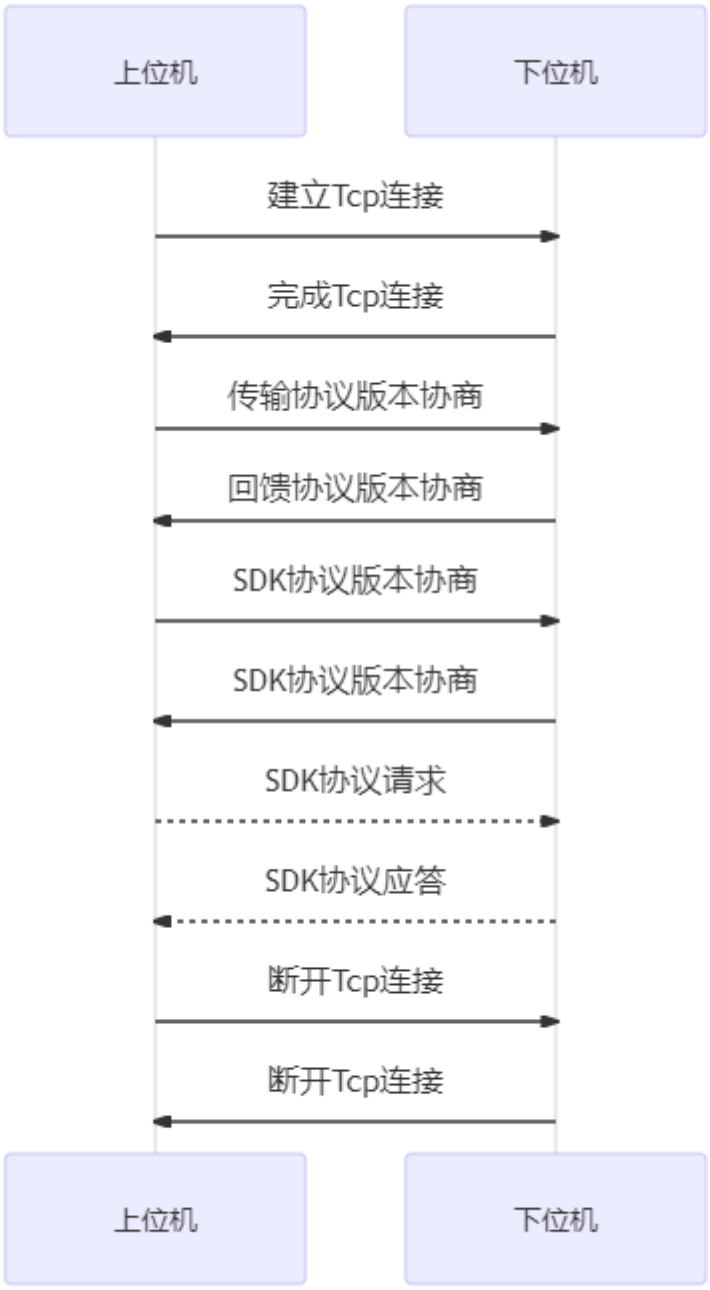
```
#pragma pack(1)
typedef struct HTcpHeader
{
    huint16 len;     ///< 命令包长度
    huint16 cmd;     ///< 命令值
} HTcpHeader;

typedef struct HTcpExt
{
    huint16 len;     ///< 命令包长度
    huint16 cmd;     ///< 命令值
    hint8   ext[0]; ///< 扩展数据起始地址
} HTcpExtAsk, HTcpExtAnswer;
#pragma pack()
```

SDK协议流程

> 注: SDK协议通信不能和发送文件数据交叉发送
>
> 比 如发送文件时, 必须等文件发送完, 在最后接收到发送文件结束确认后才能继续下发SDK信息
>
> 如果在发送文件中突然插入一条SDK信息, 则该连接将被置位非法操作连接, SDK协商将被取消, 可断开连接重新进行SDK协商.

1. 先建立Tcp连接
2. 进行协议版本交互
3. 进行SDK信息交互, 当一段时间未进行SDK信息交互需互发心跳包维持连接
4. 不再进行SDK交互后关闭Tcp连接



版本协商

1. 上位机发送版本协商包, 版本号为 **LOCAL_TCP_VERSION**
2. 下位机收到版本协商包后, 提取版本号,
    - 如版本号小于下位机支持的最低版本号, 返回 **传输协议版本过低的错误信息**
    - 如版本号大于下位机支持的最高版本号, 返回下位机最高版本号, 上位机设置下位机返回 的最高版本号

- 如版本号在下位机支持列表, 则下位机自 动切换到相同传输协议, 并返回 相同的版本号

3. 收到 **传输版本过低** 的错误信息因做断开连接处理

协议版本请求

字段	命令包长度(Len)	命令(Cmd)	版本号(Version)
字节数	2字节	2字节	4字节
取值	8	**kSDKServiceAsk**	LOCAL_TCP_VERSION

协议版本反馈

字段	命令包长度(Len)	命令(Cmd)	版本号(Version)
字节数	2字节	2字节	4字节
取值	8	**kSDKServiceAnswer**	反馈值

SDK版本协商

SDK协议版本协商通过 **GetIFVersion** SDK获取guid值, 此连接后续SDK通讯都需要使用该guid值来完成SDK请求

> 在SDK协议交互前, 需要获取guid值, 这个值在后续SDK交互中, 需要在xml的sdk标签的guid填入.
>
> 注意: SDK版本协商走的是SDK协议交互, SDK协议包头都是一样的, 如下面SDK请求的格式.

SDK协议交互

SDK数据包协议依赖 协议数据包 结构, 每包最大长度都不能找过协议数据包要求

SDK请求

字段	命令包长度(Len)	命令(Cmd)	总长度	索引	SDK接口数据
字节数	2字节	2字节	4字节	4字节	N字节
取值	Len	**kSDKCmdAsk**	Total	index	XmlData

> Len值: Len = 2 + 2 + 4 + 4 + N;
>
> Total值为SDK接口 数据的总长度
>
> Index值为当前包SDK接口 数据的索引,比 如数据大于协议数据包时需要分包,第一包是0,第二包就是xml数据 +已发的xml数据的偏移位置. 如发送200字节Xml,分100字节1包,那么第一包的Index = 0,第二包的Index = 100;
>
> XmlData值为SDK接口 数据,最大长度要求和协议包长度,XmlMaxLength = 9 * 1024 - 2 - 2 - 4 - 4个字节,就是9 * 1027 - 包头的长度.

SDK应答

字段	命令包长度(Len)	命令(Cmd)	总长度	索引	SDK接口数据
字节数	2字节	2字节	4字节	4字节	N字节
取值	Len	**kSDKCmdAnswer**	Total	index	XmlData

错误码反馈格式

字段	命令包长度(Len)	命令(Cmd)	错误码(Code)
字节数	2字节	2字节	2字节
取值	6	**kErrorAnswer**	HErrorCode

SDK版本协商

用于首次SDK协议版本协商进行的SDK接口 数据,因为guid值还不清楚.

在当前连接的SDK通信, **sdk标签中的guid属性** 必须是**SDK版本协商**返回的那个**guid**值.

SDK版本协商请求

要求:

1. 在协商完传输协议版本后, 需发送下面SDK版本请求给下位机进行SDK版本协商
2. 该SDK只有value属性的值可变, 其他均为固定值, 现在使用 1000000

```
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="GetIFVersion">
        <version value="1000000"/>
      </in>
</sdk>
```

SDK版本协商反馈

反馈信息:

1. **sdk**标签的**guid**属性的值, 这个值是当前连接后续进行**SDK**协议交互使用, 需记录保留
2. out标签的result属性的值, 这个值表示状态值, 如kSuccess表示正常状态, 详情可到错误码定义查看

```
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="f7b3b451c4f3cf3310e2d6714fbb50e8">
      <out method="GetIFVersion" result="kSuccess">
        <version value="1000000"/>
      </out>
</sdk>
```

SDK发送文件流程

> 发送文件必须发送完成, 不能在中途插入SDK数据, 否则这将丢失SDK协商. 并返回流程错误的错误码

文件开始发送请求

字段	命令包长度 **(Len)**	命令**(Cmd)**	**md5** 值	文件 大小	文件 类型	文件名
字节 数	2字节	2字节	33字 节	8字节	2字节	2-256 字节
取值	len	**kFileStartAsk**	md5	size	type	name

> len: 2 + 2 + 33 + 8 + 2 + nameLen
>
> md5: 表示下发文件的md5值. 小写
>
> size: 文件大小(字节)
>
> type: 0(图片), 1(视频), 2(字体), 3(固件), 4(基本配置参数), 5(FPGA配置), 128(临时图片文件), 129(临时视频文件)
>
> name: 文件名

注: type常用的就0和1(图片和视频), 临时文件是存放到内存的, 总大小不能超过10MB.

文件开始发送反馈

字段	命令包长度(Len)	命令(Cmd)	错误码	已存在大小
字节数	2字节	2字节	2字节	8字节
取值	8	**kFileStartAnswer**	HErrorCode	existSize

> 当错误码不等于0(kSuccess)时, 需检查错误码, 并发送结束文件包中断传输.
>
> existSize: 当设备已存在时, 不能再发已存在数据, 而是补发缺少的数据(用于断点续传, 减少消耗的发送流量)

文件内容传输

字段	命令包长度(Len)	命令(Cmd)	文件内容
字节数	2字节	2字节	1 ~ (9 * 1024 - 4)字节
取值	len	**kFileContentAsk**	data

> 文件数据包不允许超过每包大小.

文件结束请求

字段	命令包长度(Len)	命令(Cmd)
字节数	2字节	2字节
取值	4	**kFileEndAsk**

> 当完成发送或者结束发送文件时, 必须发送文件结束请求, 用于文件检查和状态校验

文件结束反馈

字段	命令包长度(Len)	命令(Cmd)	错误码
字节数	2字节	2字节	2字节
取值	6	**kFileEndAnswer**	HErrorCode

> 当文件结束反馈回送后, 错误码为0(kSuccess)时, 设备状态才会切换会SDK通信状态, 此时才能进行SDK通信.
>
> 如报错. 请检查对应错误

介绍节目单在xml的结构构成

例子:

```
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="AddProgram">
        <screen timeStamps="0">
            <program guid="##program-guid" type="normal">
                <backgroundMusic/>
                <playControl count="1" disabled="false"/>
                <area alpha="255" guid="##area-guid">
                    <rectangle x="0" height="128" width="128"
y="0"/>
                    <resources>
                        <text guid="text-guid"
singleLine="false">
                            <style valign="middle"
align="center"/>
                            <string>例子</string>
                            <font name="SimSun"
italic="false" bold="false" underline="false" size="28"
color="#ffffff"/>
                        </text>
                    </resources>
                </area>
            </program>
        </screen>
    </in>
</sdk>
```

节目单数据

> 节目单是以xml构成, 其节目标签为SDK节目相关的子标签

注:

> effect标签的停留时间尽量长, 避免不断销毁区域资源

SDK节目相关标签

```
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="##Program">
        <!--这里开始就是节目单标签数据-->
    </in>
</sdk>
```

节目单标签结构

> 节目节点, 区域节点等这些节点都有一些必须属性

```
<!--节目单子标签结构，说明节目单结构用，不包含SDK标签，属性等-->
<!--屏幕节点-->
<screen>
    <!--节目节点，允许多个-->
    <program>
        <!--区域节点，表示这个区域是该节目的子区域，允许多个-->
        <area>
            <!--区域范围，这个区域节点的xy位置和宽高-->
            <rectangle x="##x" height="##height"
width="##width" y="##y"/>
            <!--资源节点，其子节点表示该节目类型和数据，其子节点允许多
个-->
            <resources>
                <!--这里的子标签表示节目，例如text，video，image，
clock等标签，允许多个节目标签-->
            </resources>
        </area>
    </program>
</screen>

<!--program节点必须属性
##guid 节目的唯一标识，该值为空会导致无法更新删除指定节目
##type normal：表示普通节目；template：表示模板节目；html5：表示
HDML5模板节目；默认值：normal
-->
<!--area节点必须属性
##guid 区域的唯一标识
-->
```

节目单标签属性

screen 节点属性

```
<!--screen节点属性说明
##timeStamps 指示屏幕创建的时间戳，当该值与上次不同时会清除当前设备所有节目，
可为格林威治时间1970年01月01日00时00分00秒起至现在的总秒数.
-->
<screen timeStamps="##timeStamps">
    <program>...</program>
</screen>
```

program 节点属性

> 注: 必须要有guid属性, 此属性区别节目节点的唯一标识符. 不可相同
>
> 存在可选子节点
>
> border 边框标签
>
> backgroundMusic 背景音乐标签
>
> playControl 播放控制标签

```
<!--program节点属性说明
##guid 节目的唯一标识，该值为空会导致无法更新删除指定节目
##type 节目的类型 {normal: 表示普通节目; template: 表示模板节目;
html5: 表示HDML5模板节目; 默认值 : normal }

##flag 节目的特殊标签 {realtime: 表示实时节目，当存在该标签的节目时仅
播放该节目; 默认值: 空}(无用属性, 以移除)
-->
<program guid="##guid" type="##type" flag="##flag">
    <!--可选子标签-->
    <border/>
    <backgroundMusic/>
    <playControl/>
    <area>...</area>
</program>
```

border 节目子节点属性

> 使用边框会占用像素, 对应的节目需减去边框占用的像素
>
> 如屏幕大小是128x128, 那么最大区域大小是128x128
>
> 在program的子节点使用边框节点, 边框大小占用1像素时, 则最大区域大小变为126x126.
>
> 如区域节点使用128x128, 在区域节点的子节点使用边框节点, 那么最大区域大小和上面一样变成126x126

```
<!--border 边框功能说明
##index 边框图案的序号(从0开始) 数值参考HDPlayer边框功能
##effect 边框的动画效果 { rotate: 旋转(默认); blink: 闪烁; null:
静止 } (可选)
##speed 边框的速度 { slow: 慢; middle: 适中(默认); fast: 快 } (可
选)
-->
<border index="##index" effect="##effect" speed="##speed"/>
```

backgroundMusic 节目子节点属性

```
<!--backgroundMusic 背景音乐说明
##name 文件名 (允许多个)
-->
<backgroundMusic>
    <file name="##name"/>
</backgroundMusic>
```

playControl 节目子节点属性

```
<!--playControl 播放控制标签
##duration 表示该节目的固定时长 {hh:mm:ss(默认: 节目播放一遍的时间)}
(可选, 与count互斥)
##count 节目连续播放的次数 {[0-999] 默认: 1} (可选, 与duration互斥)
##disabled 节目是否被禁用，如果节目被禁用，则该节目不播放 {"true",
"false"(默认)}

##week.enable 允许播放的星期 { Mon, Tue, Wed, Thur, Fri, Sat,
Sun(多个使用","分隔) }

##date.start 起始日期 { YYYY-MM-DD }
##date.end 终止日期 { YYYY-MM-DD }

##time.start 起始日期 { hh:mm:ss (采用24小时制) }
##time.end 终止日期 { hh:mm:ss (采用24小时制) }

##location.lng 经度 { Double类型 }
##location.lat 纬度 { Double类型 }
##location.range 范围, 单位千米 { Double类型 }
##location.preemptive 抢占式播放，当进入指定区域时，
如果当前播放节目不为定点播放或不在当前区域内，则马上将当前节目停止，切换到
本节目 {"true", "false"(默认)}

##location.playOutside 是否允许区域外播放 {"true"(默认), "false"}
-->
<playControl duration="##duration" count="##count"
disabled="##disabled">
    <!--可选-->
    <week enable="##enable"/>
    <!--可选, 允许多个-->
    <date start="##start" end="##end"/>
    <time start="##start" end="##end"/>
    <location lng="##lng" lat="##lat" range="##range"
preemptive="##preemptive" playOutside="##playOutside"/>
</playControl>
```

area 节点属性

> 注: 必须要有guid属性, 此属性区别区域节点的唯一标识符. 不可相同

多区域时, 当一个节目超过一定数量的区域时(大致10个左右), 并且停留时间短时, 有可能出现闪屏现象, 这时请将不需要翻页的区域的停留时间 **duration** 尽可能设置较高值, 需要翻页的才设置准确值, 避免出现闪屏现象.

```
<!--area节点属性说明
##guid 区域的唯一标识
##alpha 区域的透明度 {0~255 0全透明 255不透明; 默认值: 255}

##rectangle.x 区域位于节目的位置的x轴
##rectangle.y 区域位于节目的位置的y轴
##rectangle.width 区域占节目的宽度
##rectangle.height 区域占节目的高度
-->
<area guid="##guid" alpha="##alpha">
    <rectangle x="##x" height="##height" width="##width"
y="##y"/>
    <!--resources标签无属性, 其子标签为节目标签-->
    <resources>...</resources>
    <!--可选子标签-->
    <border/>
</area>
```

resources 节点属性

> 无节点属性. 其子节点为节目节点

节目节点属性

> 属于资源节点(**resources**)的子节点

text文本节目

```
<!--
##text.guid 文本的唯一标识
##text.singleLine 是否是单行文本 {"true"(默认), "false"}
##text.background 背景颜色，若无此属性则为黑色

string节点为该文本节目的内容

style font effect 节点为功能节点(可选项). 详情请到功能节点属性
-->
<text guid="##guid" singleLine="##singleLine"
background="##background">
    <style valign="##valign" align="##align" />
    <string>文本内容</string>
    <font name="##name" italic="##italic" bold="##bold"
underline="##underline" size="##size" color="##color" />
    <effect inSpeed="##inSpeed" duration="##duration"
in="##in" outSpeed="##outSpeed" out="##out" />
</text>
```

image图片节目

```
<!--
##image.guid 图片的唯一标识符
##fit 图片的填充属性 {
    fill：填充，先将图片等比放大能盖完整区域的比例，再截取中间部分显示。
    center：居中，将图片等比缩小至区域大小，比例不一致时会显示黑边
    stretch：拉伸，可能导致图片变形(默认)
    tile ：平铺
}
##file.name 文件名

file effect 节点为功能节点(可选项)。详情请到功能节点属性
-->
<image guid="##guid" fit="##fit">
    <file name="name" />
    <effect inSpeed="##inSpeed" duration="##duration"
in="##in" outSpeed="##outSpeed" out="##out"/>
</image>
```

video视频节目

```
<!--
##video.guid 视频的唯一标识符
##video.aspectRatio 保持宽高比 {"true", "false"(默认)}
##file.name 文件名

file 节点为功能节点(可选项)。详情请到功能节点属性
-->
<video guid="##guid" aspectRatio="##aspectRatio">
    <file name="name" />
</video>
```

clock时钟节目

```
<!--
##clock.guid 时钟的唯一标识符
##clock.type { digital: 数字时钟 dial: 模拟时钟 } (可选)
##clock.timezone 时区 { 以 “+8:00” 的格式; 默认值:自动适配系统时区 } (可选)
##clock.adjust 时间微调 { "+00:05:00" 向前调时间; "-00:05:00" 向后调时间 默认值: 0 } (可选)
##clock.lang 翻译语言, 默认zh_cn (可选) { "zh_cn", "zh_tw",
"en", "tr", "de", "es", "fa",
 "ja", "ko", "ru", "pl", "id", "ar", "it", "fr", "el", "ms",
"pt", "sr", "th", "vi" }

{
可选字体设置, 不同类型时钟配置:
    dial 直接在 clock节点 增加 font 的子节点, 也可在对应节点增加属性, 优先对应节点的属性
    digital 需在对应的节点增加属性值, fontSize属性可更改字体大小, fontName属性可更改字体
}
子节点皆为可选项
##title.value 标题内容

##date.format 日期的格式, 输入数字 {
0、YYYY/MM/DD
1、MM/DD/YYYY
2、DD/MM/YYYY
3、Jan DD, YYYY
4、DD Jan, YYYY
5、YYYY年MM月DD日( 翻译后的语言)
6、MM月DD日( 翻译后的语言)
}

##week.format 星期的格式, 输入数字 {
0、星期一( 翻译后的语言)
1、Monday
2、Mon
}

##time.format 时间的格式, 输入数字 {
```

```
0、hh:mm:ss
1、hh:mm
2、hh时mm分ss秒（翻译后的语言）
3、hh时mm分（翻译后的语言）
}

时钟选项通用
##color 颜色 格式为 "#RRGGBB"
##display 是否显示 {"true"(默认), "false"}
-->
<clock guid="##guid" type="##type" timezone="##timezone"
adjust="##adjust" lang="##lang">
    <title value="##value" color="##color"
display="##display"/>
    <date format="##format" color="##color"
display="##display"/>
    <week format="##format" color="##color"
display="##display"/>
    <time format="##format" color="##color"
display="##display"/>
    <lunarCalendar color="##color" display="##display"/>
</clock>
```

table表格节目

> 表格自身支持持久性数据，如果需要还原默认数据需在table标签的clearData属性上设置为true

```
<!--
##table.guid 表格的唯一标识符
##table.transparent 背景透明 {"true", "false"(默认)}
##table.clearData 清空数据 {"true", "false"(默认)}
##table.row 行数
##table.col 列数

##effect... 特效节点, 请到功能节点属性查看

##border.color 每个单元格的线的颜色
##border.lineWidth 每个单元格的线的宽度

统一属性: row表示[0, ...)行 col表示[0, ...)列

##size.width 大小宽度表示指定列的宽度
##size.height 大小高度表示指定行的高度

合并单元格, 如果合并内容存在已合并的, 那么将无法进行合并
取消合并的方法, 指定已合并的相同行列, 两两值相等既解除合并
例如解除合并merger的属性:row == mergerRow; col == mergerCol即可
##merger.mergerRow 合并到哪一列
##merger.mergerCol 合并到哪一行

单元格内容
##background.value 默认"#000000", 背景颜色
##background.file 背景图片, 当设置背景图片后背景颜色无效
##text 单元格内容(选其一)
##file 单元格文件(当前只支持图片, 选其一)
-->
<table guid="##guid" transparent="##transparent"
clearData="##clearData" row="##row" col="##col">
    <effect inSpeed="##inSpeed" duration="##duration"
in="##in" outSpeed="##outSpeed" out="##out"/>
    <border color="##color" lineWidth="##lineWidth"/>
    <size row="##row" col="##col" width="##width"
height="##height"/>
    <merger row="##row" col="##col" mergerRow="##mergerRow"
mergerCol="mergerCol"/>
    <cell row="##row" col="col">
```

```
        <style valign="##valign" align="##align" />
        <font name="##name" italic="##italic" bold="##bold"
underline="##underline" size="##size" color="##color" />

        <!--当value存在时, 不需要填file-->
        <background value="##value">
            <file .../>
        </background>

        <!--下面文本或者文件(当前只支持图片)可选一个-->
        <text>...</text>
        <file .../>
    </cell>
</table>
```

功能节点属性

> 附属于节目节点的功能节点属性说明

effect特效节点

> 特效类型

```
enum EffectType
{
    IMMEDIATE_SHOW       = 0,    ///< 直接显示.
    LEFT_PARALLEL_MOVE   = 1,    ///< 向左平移.
    RIGHT_PARALLEL_MOVE  = 2,    ///< 向右平移.
    UP_PARALLEL_MOVE     = 3,    ///< 向上平移.
    DOWN_PARALLEL_MOVE   = 4,    ///< 向下平移.
    LEFT_COVER           = 5,    ///< 向左覆盖.
    RIGHT_COVER          = 6,    ///< 向右覆盖.
    UP_COVER             = 7,    ///< 向上覆盖.
    DOWN_COVER           = 8,    ///< 向下覆盖.
    LEFT_TOP_COVER       = 9,    ///< 左上覆盖.
    LEFT_BOTTOM_COVER    = 10,   ///< 左下覆盖.
    RIGHT_TOP_COVER      = 11,   ///< 右上覆盖.
    RIGHT_BOTTOM_COVER   = 12,   ///< 右下覆盖.
    HORIZONTAL_DIVIDE    = 13,   ///< 水平对开.
    VERTICAL_DIVIDE      = 14,   ///< 垂直对开.
    HORIZONTAL_CLOSE     = 15,   ///< 水平闭合.
    VERTICAL_CLOSE       = 16,   ///< 垂直闭合.
    FADE                 = 17,   ///< 淡入淡出.
    HORIZONTAL_SHUTTER   = 18,   ///< 水平百叶窗.
    VERTICAL_SHUTTER     = 19,   ///< 垂直百叶窗.
    NOT_CLEAR_AREA       = 20,   ///< 不清屏.
    LEFT_SERIES_MOVE     = 21,   ///< 连续左移.
    RIGHT_SERIES_MOVE    = 22,   ///< 连续右移.
    UP_SERIES_MOVE       = 23,   ///< 连续上移.
    DOWN_SERIES_MOVE     = 24,   ///< 连续下移.
    RANDOM               = 25,   ///< 随机特效.
    HT_LEFT_SERIES_MOVE  = 26,   ///< 首尾相接连续左移.
    HT_RIGHT_SERIES_MOVE= 27,   ///< 首尾相接连续右移.
    HT_UP_SERIES_MOVE    = 28,   ///< 首尾相接连续上移.
    HT_DOWN_SERIES_MOVE  = 29,   ///< 首尾相接连续下移.
    EFFECT_COUNTS        = 30,   ///< 特效总数.
};
```

effect特效标签

> 注: effect标签的停留时间尽量长, 避免不断销毁区域资源

```
<!--
##in 入场特效 ID
##out 出场特效 ID
##inSpeed 入场特效速度 { [0-8] 0最快，8最慢 }
##outSpeed 出场特效速度 { [0-8] 0最快，8最慢 }
##duration 停留时间 { [0-9999]单位为0.1秒 }
-->
<effect in="##in" out="##out" inSpeed="##inSpeed"
outSpeed="##outSpeed" duration="##duration"/>
```

font文字功能节点

```
<!--
##name 字体名字 (可选)
##italic 斜体 { "true", "flase"(默认) }
##bold 粗体 { "true", "false"(默认) }
##underline 下划线 { "true", "false"(默认) }
##size 字体大小
##color 字体颜色 { "#FF0000"(默认红色) }
-->
<font name="##name" italic="##italic" bold="##bold"
underline="##underline" size="##size" color="##color"/>
```

style类型功能节点

```
<!--
##valign 垂直对齐方式 { middle: 垂直居中; top: 上对齐; bottom: 下
对齐 }
##align 水平对齐方式 { center : 居中对齐; left: 左对齐; right: 右
对齐 }
-->
<style valign="##valign" align="##align"/>
```

file文件节点

```
<!--指定使用文件
##name 文件名
##md5 文件md5值 (可选)
-->
<file name="##name" md5="##md5"/>
```

开机画面功能

BootLogo开机画面

开机画面相关SDK

GetBootLogo获取开机画面状态

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetBootLogo"/>
</sdk>

<!--反馈
##exist 是否设置了开机画面{"true": 已设置, "false": 未设置}
##md5 开机图片的md5值
##name 开机图片名
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="GetBootLogo" result="kSuccess">
        <logo exist="##exist" md5="##md5" name="##name"/>
    </out>
</sdk>
```

SetBootLogoName设置开机画面

```
<!--请求
##md5 开机图片的md5值，可为空
##name 开机图片名
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetBootLogoName">
        <logo md5="##value" name="##name"/>
    </in>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="SetBootLogoName" result="kSuccess"/>
</sdk>
```

ClearBootLogo清除开机画面

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="ClearBootLogo"/>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="ClearBootLogo" result="kSuccess"/>
</sdk>
```

网络功能

获取网络信息

获取网络信息相关的SDK

GetNetworkInfo获取网络信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetNetworkInfo"/>
</sdk>

<!--反馈
##mode 当前网络设备接入者，优先eth，次级pppoe，最后wifi
##eth.valid 和 ##pppoe.valid 和 ##wifi.valie {"true": 接入，
"false" 未接入}，接入时存在详细信息
##网络相关详细信息在下面对应模块详解
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="GetNetworkInfo" result="kSuccess">
        <network mode="##mode"/>
        <!--eht接入显示详细信息-->
        <eth valid="##valid">
            <enable value="##value"/>
            <dhcp auto="##auto"/>
            <address netmask="##netmask" gateway="##gateway"
ip="##ip" dns="##dns"/>
        </eth>
        <!--pppoe未接入不存在详细信息-->
        <pppoe valid="##valid"/>
        <!--wifi未接入不存在详细信息-->
        <wifi valid="##valid"/>
    </out>
</sdk>
```

GetEth0Info仅获取eth0信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetEth0Info"/>
</sdk>

<!--反馈
##valid 有线网络是否接入 {"true": 接入, "false" 未接入}, 为false时无详细信息
##enable.value {"true": 接入, "false" 未接入}
##dhcp.auto {"true"(dhcp获取ip地址), "false"(静态ip地址)}
##address.ip ip地址
##address.netmask 网络掩码
##address.gateway 网关
##address.dns dns地址
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="GetEth0Info" result="kSuccess">
        <eth valid="##valid">
            <enable value="##value"/>
            <dhcp auto="##auto"/>
            <address ip="##ip" netmask="##netmask"
gateway="##gateway" dns="##dns"/>
        </eth>
    </out>
</sdk>
```

GetPppoeInfo仅获取pppoe(3/4G)信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetPppoeInfo"/>
</sdk>

<!--反馈
##valid 是否有接入 {"true"(有3/4G模块接入), "false"(无3/4G模块接入)}
##enable.value {"true"(有3/4G网络接入), "false"(无3/4G网络接入)}
##apn.value apn值
##manufacturer.value: 模块生产商
##version.value: 模块版本
##model.value: 模块型号
##imei.value: 模块IMEI
##number.value: SIM卡电话号码
##operators.value: 运营商
##signal.value: 信号强度, 取值范围[1, 5]; 1表示信号强度最差; 5表示信号强度最好
##dbm.value: 信号强度 (单位dbm)
##insert.value: SIM卡是否插入, 取值范围{"true"(有SIM卡插入), "false"(无SIM卡插入)}
##status.value: 网络注册状态, 取值范围 {"unregister"(未注册), "register local"(已注册, 本地网络), "searching"(搜索中), "reject"(拒绝注册), "unknow"(未知错误), "register roaming"(已注册, 漫游网络), "init"(初始化状态)}

##network.value: 网络制式, 取值范围 {"init"(初始化状态), "unknow"(未知网络), "2G"(2G), "2.5G"(2.5G), "3GPP"(3GPP家族), "3G TD"(移动3G), "3.5G HSDPA", "3.5G HSUPA", "3.5G HSPAPlus", "4G LTE", "4G TDD", "4G FDD"}

##code.value: 错误码(保留)
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="GetPppoeInfo" result="kSuccess">
        <pppoe valid="##value">
            <enable value="##value"/>
```

```
            <apn value="##value"/>
            <manufacturer value="##value"/>
            <version value="##value"/>
            <model value="##value"/>
            <imei value="##value"/>
            <number value="##value"/>
            <operators value="##value"/>
            <signal value="##value"/>
            <dbm value="##value"/>
            <insert value="##value"/>
            <status value="##value"/>
            <network value="##value"/>
            <code value="##value"/>
        </pppoe>
    </out>
</sdk>
```

GetWifiInfo仅获取wifi信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetWifiInfo"/>
</sdk>

<!--反馈
##wifi.valid取值范围{"true"(有WIFI模块接入), "false"(无WIFI模块接入)};
##enable.value取值范围{"true"(有WIFI网络接入), "false"(无WIFI网络接入)}
##mode.value: wifi工作模式, 取值范围{"ap"(ap模式), "station"(客户端模式)}
##ap.ssid.value: ssid, 固定值(设备ID)
##ap.passwd.value: 密码
##ap.channel.value: 信道
##ap.encryption.value: 加密方式, 固定值"WPA-PSK"
##ap.dhcp: 忽略
##ap.address: 忽略
##station.current.index: 表示当前使用的ap节点在list中的索引值, -1时表示未选中ap节点
##station.list包含多个item, item表示wifi发现和已经保存的ap节点信息
##station.list.item.ssid.value: ap的ssid
##station.list.item.passwd.value: 密码
##station.list.item.signal.value: 信号强度
##station.list.item.apmac.value: ap设备的mac地址
##station.list.item.dhcp.auto: 取值范围{"true"(dhcp获取ip地址), "false"(静态ip地址)}
##station.list.item.address.ip: WIFI网络ip地址
##station.list.item.address.netmask: WIFI网络子网掩码
##station.list.item.address.gateway: 默认路由
##station.list.item.address.dns: dns服务器地址
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="GetWifiInfo" result="kSuccess">
        <wifi valid="##value">
            <enable value="##value"/>
            <mode value="##value"/>
```

```
            <ap>
                <ssid value="##value"/>
                <passwd value="##value"/>
                <channel value="##value"/>
                <encryption value="##value"/>
                <dhcp auto="##value"/>
                <address ip="##value" netmask="##value"
gateway="##value" dns="##value"/>
            </ap>
            <station>
                <current index="##value"/>
                <list>
                    <item>
                        <ssid value="##value"/>
                        <passwd value="##value"/>
                        <signal value="##value"/>
                        <apmac value="##value"/>
                        <dhcp auto="##value"/>
                        <address ip="##value"
netmask="##value" gateway="##value" dns="##value"/>
                    </item>
                </list>
            </station>
        </wifi>
    </out>
</sdk>
```

设置网络信息

设置网络信息相关的SDK

> 修改网络配置时, 有些将导致网络断开.

SetEth0Info设置eth0信息

```
<!--请求
##valid 固定值"true"
##enable.value 固定值"true"
##dhcp.auto {"true"(dhcp获取ip地址), "false"(静态ip地址)}
##address.ip ip地址
##address.netmask 网络掩码
##address.gateway 网关
##address.dns dns地址-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetEth0Info">
        <eth valid="##valid">
            <enable value="##value"/>
            <dhcp auto="##auto"/>
            <address ip="##ip" netmask="##netmask"
gateway="##gateway" dns="##dns"/>
        </eth>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="SetEth0Info" result="kSuccess">
    </out>
</sdk>
```

SetWifiInfo设置wifi信息

```
<!--请求
##mode.value: wifi工作模式, 取值范围{"ap"(ap模式), "station"(客户端模式)}
##ap.ssid.value: ssid, 固定值(设备ID)
##ap.passwd.value: 密码
##ap.channel.value: 信道
##ap.encryption.value: 加密方式, 固定值"WPA-PSK"
##ap.dhcp.auto: 固定值"true"
##ap.address: 固定值ip="0.0.0.0" netmask="0.0.0.0" gateway="0.0.0.0" dns="0.0.0.0"
##station.current.index: 表示当前使用的ap节点在list中的索引值, -1时表示未选中ap节点
##station.list包含多个item, item表示wifi发现和已经保存的ap节点信息
##station.list.item.ssid.value: ap的ssid
##station.list.item.passwd.value: 密码
##station.list.item.signal.value: 信号强度
##station.list.item.apmac.value: ap设备的mac地址
##station.list.item.dhcp.auto: 取值范围{"true"(dhcp获取ip地址), "false"(静态ip地址)}
##station.list.item.address.ip: WIFI网络ip地址
##station.list.item.address.netmask: WIFI网络子网掩码
##station.list.item.address.gateway: 默认路由
##station.list.item.address.dns: dns服务器地址
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetWifiInfo">
        <mode value="##value"/>
        <ap>
            <ssid value="##value"/>
            <passwd value="##value"/>
            <channel value="##value"/>
            <encryption value="##value"/>
            <dhcp auto="##value"/>
            <address gateway="##value" netmask="##value" dns="##value" ip="##value"/>
        </ap>
        <station>
            <current index="##value"/>
```

```
            <list>
                <item>
                    <ssid value="##value"/>
                    <passwd value="##value"/>
                    <signal value="##value"/>
                    <apmac value="##value"/>
                    <dhcp auto="##value"/>
                    <address gateway="##value"
netmask="##value" dns="##value" ip="##value"/>
                </item>
            </list>
        </station>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="SetWifiInfo" result="kSuccess">
    </out>
</sdk>
```

SetApn设置Apn

```
<!--请求
apn.value: apn
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetApn">
        <apn value="##value"/>
    </in>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="SetApn" result="kSuccess"/>
</sdk>
```

测试网络

测试网络相关的SDK

TestAPMode开启测试Ap模式

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="TestAPMode"/>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="TestAPMode" result="kSuccess"/>
</sdk>
```

TestApModeResult测试Ap模式结果

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="TestApModeResult"/>
</sdk>

<!--反馈
##result 返回kInvalidParam时, 则未开启测试模式
返回kNotFoundWifiModule时, 则未找到wifi模块
返回kTestWifiUnsuccessful时, 则测试AP模式失败
返回kSuccess时, 则成功
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="TestApModeResult" result="##result"/>
</sdk>
```

TestApDisconnect停止测试Ap模式

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="TestApDisconnect"/>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="TestApDisconnect" result="kSuccess"/>
</sdk>
```

TestStationMode测试Station模式

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="TestStationMode"/>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="TestStationMode" result="kSuccess"/>
</sdk>
```

TestStationModeResult测试Station模式结果

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="TestStationModeResult"/>
</sdk>

<!--反馈
##result 返回kInvalidParam时, 则未开启测试模式
返回kNotFoundWifiModule时, 则未找到wifi模块
返回kTestWifiUnsuccessful时, 则测试AP模式失败
返回kSuccess时, 则成功
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="0a35b47e0821c4ec26d3075b9737a3b5">
    <out method="TestStationModeResult" result="##result"/>
</sdk>
```

TestStationDisconnect停止测试Station模式

```xml <?xml version='1.0' encoding='utf-8'?>

<?xml version='1.0' encoding='utf-8'?>
```

字体功能

获取或重加载字体

GetAllFontInfo获取字体信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetAllFontInfo"/>
</sdk>

<!--反馈
font.name: 字体名
font.file: 字体文件名
font.bold: 该字体是否支持粗体, 取值范围{"true"(支持), "false"(不支持)}
font.italic: 该字体是否支持斜体, 取值范围{"true"(支持), "false"(不支持)}
font.underline: 该字体是否支持下划线, 取值范围{"true"(支持), "false"(不支持)}
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="fecc9420c96a6a708b35e13f62385e15">
    <out method="GetAllFontInfo" result="kSuccess">
        <fonts>
            <font bold="##bold" italic="##italic" underline="##underline" name="##name" file="##file"/>
        </fonts>
    </out>
</sdk>
```

ReloadAllFonts重新加载所有字体

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="ReloadAllFonts"/>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="fecc9420c96a6a708b35e13f62385e15">
    <out method="ReloadAllFonts" result="kSuccess"/>
</sdk>
```

通用功能

获取设备相关信息

GetDeviceInfo获取设备信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetDeviceInfo"/>
</sdk>

<!--反馈
device.cpu: 设备cpu类型
device.model: 设备类型名称
device.id: 设备ID
device.name: 设备名
version.fpga: fpga固件版本号
version.app: 下位机固件版本号
version.kernel: 内核版本号
screen.width: 屏幕宽度
screen.height: 屏幕高度
screen.rotation: 旋转标志, 取值范围{"0", "90", "180", "270"}
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="fecc9420c96a6a708b35e13f62385e15">
    <out method="GetDeviceInfo" result="kSuccess">
        <device cpu="##cpu" model="##model" name="##name"
id="##id"/>
        <version fpga="##fpga" kernel="##kernel"
app="##app"/>
        <screen width="##width" height="##height"
rotation="##rotation"/>
    </out>
</sdk>
```

GetDeviceName获取设备名

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetDeviceName"/>
</sdk>

<!--反馈
##value 设备名
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="fecc9420c96a6a708b35e13f62385e15">
    <out method="GetDeviceName" result="kSuccess">
        <name value="##value"/>
    </out>
</sdk>
```

GetHardwareInfo获取硬件信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetHardwareInfo"/>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="fecc9420c96a6a708b35e13f62385e15">
    <out method="GetHardwareInfo" result="kSuccess">
        <cpu model="" info=""/>
        <memory model="" size="0" info=""/>
        <storage>
            <flash model="" free="0" total="0" info=""/>
            <extend model="" free="0" total="0" info=""/>
        </storage>
        <eth model="" info=""/>
        <system kernel="" info=""/>
        <ExternInterface>
            <gps model="" valid="false" info=""/>
            <pppoe model="" valid="false" info=""/>
            <wifi model="" valid="false" info=""/>
            <sensor model="" valid="false" info=""/>
        </ExternInterface>
    </out>
</sdk>
```

设置设备相关信息

SetDeviceName设置设备名

```
<!--请求
##value 设备名
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetDeviceName">
        <name value="##value"/>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="fecc9420c96a6a708b35e13f62385e15">
    <out method="SetDeviceName" result="kSuccess"/>
</sdk>
```

获取或设置系统音量

SetSystemVolume设置系统音量

```
<?xml version = "1.0" encoding = "utf-8" ?>
<sdk guid="##GUID">
    <in method="SetSystemVolume" >
        <mode value="##value" />
        <volume percent="##percent" />
        <!-- ploy该节点为可选节点, 模式default时无需该节点 -->
        <ploy>
            <item enable="##enable" start="##start"
percent="##percent"/>
        </ploy>
    </in>
</sdk>
<!--
##value 选择模式default和ploys
##percent 为音量值[0-100]
##ploy.item.enable true为开启该配置项 false为关闭
##ploy.item.start 时间格式hh:mm:ss
##ploy.item.percent 音量
-->

<!--设备回答-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="b040b83a8de5012816e1c2ca1b727305">
    <out method="SetSystemVolume" result="kSuccess"/>
</sdk>
```

GetSystemVolume获取系统音量

```xml
<?xml version = "1.0" encoding = "utf-8" ?>
<sdk guid="##GUID">
    <in method="GetSystemVolume" />
</sdk>

<!--
设备回答
mode标签 value为音量模式
volume标签 percent为音量值
ploy标签 在default模式为无效节点
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="b040b83a8de5012816e1c2ca1b727305">
    <out method="GetSystemVolume" result="kSuccess">
        <mode value="default"/>
        <volume percent="100"/>
        <ploy/>
    </out>
</sdk>
```

获取和设置Tcp服务器

GetSDKTcpServer获取Tcp服务器信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetSDKTcpServer"/>
</sdk>

<!--反馈
##port 端口
##host 主机地址
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="dedef080ba2082110c73404cceb17135">
    <out result="kSuccess" method="GetSDKTcpServer">
        <server port="##port" host="##host"/>
    </out>
</sdk>
```

SetSDKTcpServer设置Tcp服务器

> 用于需要设备向tcp服务器进行连接,而不是主动向设备连接的功能

```
<!--请求
##port 端口
##host 主机地址
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetSDKTcpServer">
        <server port="##port" host="##host"/>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="dedef080ba2082110c73404cceb17135">
    <out result="kSuccess" method="SetSDKTcpServer"/>
</sdk>
```

重启设备

```
<!--请求
##delay 几秒后重启
-->
<?xml version = "1.0" encoding = "utf-8" ?>
<sdk guid="##GUID">
    <in method="Reboot" delay="##delay" />
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="3b03a3534c3a37c76bd610fee8bc8055">
    <out method="Reboot" result="kSuccess"/>
</sdk>
```

关于配置Socket信息

正常情况无需修改. 尽量不修改它, 避免导致Socket被设备不断关闭

GetSocketTimeInfo获取Socket信息

```
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetSocketTimeInfo"/>
</sdk>
```

SetSocketTimeInfo设置Socket信息

```
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetSocketTimeInfo">
        <heartbeat timeout="##timeout" delay="##delay"/>
        <conncet delay="##delay"/>
    </in>
</sdk>
```

许可证功能

许可证信息

GetLicense获取许可证

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetLicense"/>
</sdk>


<!--反馈
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="f96957643084953befcbd2c1a16f7617">
    <out method="GetLicense" result="kSuccess">
        <time value="0"/>
        <code value=""/>
        <old value=""/>
        <set value="0"/>
    </out>
</sdk>
```

设置许可证 (冗余)

> SetLicense 方法, 无效果

清空许可证 (冗余)

> ClearLicense 方法, 无效果

亮度功能

亮度设置

GetLuminancePloy获取亮度配置

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetLuminancePloy"/>
</sdk>


<!--反馈
##mode.value "default"(默认), "ploys"(自 定义模式), "sensor"(传感
器模式)
默认模式
##default.value 亮度值[1, 100]
传感器模式
##sensor.min 最小亮度等级
##sensor.max 最大亮度等级
##sensor.time 亮度调整的间隔时间
自 定义模式
##ploy可包含多个item子节点
##item.enable 该项是否使能, {"true"(使能), "false"(不使能)}
##item.start 改变亮度等级的开始时间, 格式: hh:mm:ss
##item.percent: 亮度等级
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="f96957643084953befcbd2c1a16f7617">
    <out method="GetLuminancePloy" result="kSuccess">
        <mode value="default"/>
        <default value="100"/>
        <sensor min="1" max="100" time="5"/>
        <ploy>
            <item enable="##enable" start="##start"
percent="##percent"/>
        </ploy>
    </out>
</sdk>
```

SetLuminancePloy设置亮度配置

```
<!--请求
##mode.value取值范围{"default"(默认), "ploys"(自 定义模式),
"sensor"(传感器模式)}
默认模式
##default.value: 默认模式时, 设置的亮度等级[1, 100]
自 定义模式
##ploy可以包含多个item子节点, 表示可设置在不同时间段使用不同的亮度等级
##item.enable: 该项是否使能, 取值范围{"true"(使能), "false"(不使
能)}
##item.start: 改变亮度等级的开始时间, 格式: hh:mm:ss
##item.percent: 亮度等级
传感器模式
##sensor.min: 传感器模式最小亮度等级
##sensor.max: 传感器模式最大亮度等级
##sensor.time: 亮度调整的间隔时间, 取值范围[5, 15] (单位秒)

-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetLuminancePloy">
        <mode value="##value"/>
        <default value="##value"/>
        <ploy>
            <item enable="##enable" percent="##percent"
start="##start"/>
        </ploy>
        <sensor max="##max" min="##min" time="##time"/>
    </in>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <out method="SetLuminancePloy" result="kSuccess"/>
</sdk>
```

媒体文件功能

DeleteFiles从设备删除文件

兼容旧版本, 增加md5值, 准确删除指定文件

> 不指定md5值, 和旧版本相同, 但碰到相同name文件时, 删除第一个

```
<!-- 旧版本, 删除首个相同name文件-->
<?xml version="1.0"?>
<sdk guid="##GUID">
    <in method="DeleteFiles">
        <files>
            <file name="##name"/>
        </files>
    </in>
</sdk>


<!-- 新增加属性, 准确删除指定文件-->
<?xml version="1.0"?>
<sdk guid="##GUID">
    <in method="DeleteFiles">
        <files>
            <file name="##name" md5="##md5"/>
        </files>
    </in>
</sdk>

<!--两者反馈
删除成功则没有files标签和子标签, 存在删除文件未找到的则返回 files标签,
子标签为未找到的文件
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="d77658a8565669346a60862dba08f2e1">
    <out result="kSuccess" method="DeleteFiles">
        <files>
            <file result="kFileNotFound" name="##name"/>
        </files>
    </out>
</sdk>
```

根据md5删除文件

使用DeleteFiles相同SDK, 这个是新增功能, 可不使用name属性, 而是使用md5删除文件.

```
<!-- 只使用md5删除文件 -->
<?xml version="1.0"?>
<sdk guid="##GUID">
    <in method="DeleteFiles">
        <files>
            <file md5="##md5"/>
        </files>
    </in>
</sdk>

<!--反馈
删除成功则没有files标签和子标签, 存在删除文件未找到的则返回 files标签,
子标签为未找到的文件
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="d77658a8565669346a60862dba08f2e1">
    <out result="kSuccess" method="DeleteFiles">
        <files>
            <file result="kFileNotFound" md5="##md5"/>
        </files>
    </out>
</sdk>
```

AddFiles添加文件到设备

设备支持http下载

```
<!--请求
##type image和video
##size 文件的大小
##md5 文件的md5值
##name 文件名
##remote 资源位置, http://... https://...
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="AddFiles">
        <files>
            <file type="##type" size="##size" md5="##md5"
name="##name" remote="remote"/>
        </files>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="41f8d777a9d83984e49fce04ea4c77d0">
    <out method="AddFiles" result="kSuccess">
        <files/>
    </out>
</sdk>
```

GetFiles获取设备媒体文件信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetFiles"/>
</sdk>

<!--反馈
##existSize 文件在下位机存在的大小
##md5 文件的md5值
##name 文件名
##size 文件大小
##type 类型
0-图片 1-视频 2-字体 3-固件 4-fpga配置文件
5-设置配置文件 6-项目资源 7-数据 8-临时文件
128- 临时图片文件 129-临时视频文件
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="41f8d777a9d83984e49fce04ea4c77d0">
    <out method="GetFiles" result="kSuccess">
        <files>
            <file existSize="##existSize" type="##type"
md5="##md5" name="##name" size="##size"/>
        </files>
    </out>
</sdk>
```

获取临时内存文件大小

> 如果需要频繁更新图片或视频, 请将上传文件设置为临时图片或临时视频, 避免频繁写入硬盘损耗硬盘寿命

什么是临时文件

> 就是存在于内存的文件, 断电即丢失的那种.

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetRAMSize"/>
</sdk>

<!--反馈
##rawDir.size 临时内存目录的大小
##rawDir.available 剩余可用大小
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="6573188f38ea9d6b3fad98b6783bfa63">
    <out result="kSuccess" method="GetRAMSize">
        <rawDir available="50MB" size="50MB"/>
    </out>
</sdk>
```

屏幕开关功能

配置开关屏速度

GetSpeedForOFFOrON获取SDK开关屏速度

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetSpeedForOFFOrON"/>
</sdk>

<!--反馈
##enable.value SDK开关屏速度限制使能状态
##off.speed SDK关闭屏幕时的速度(单位：千米/小时)
##on.speed SDK打开屏幕时的速度(单位：千米/小时)
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="bd68900da93b59cc553026851cc05afb">
    <out result="kSuccess" method="GetSpeedForOFFOrON">
        <enable value="##value"/>
        <off speed="##speed" />
        <on speed="##speed" />
    </out>
</sdk>
```

SetSpeedForOFFOrON设置SDK开关屏速度

```
<!--请求
##enable.value SDK开关屏速度限制使能状态
##off.speed SDK关闭屏幕时的速度(单位：千米/小时)
##on.speed SDK打开屏幕时的速度(单位：千米/小时)
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetSpeedForOFFOrON">
        <enable value="##value"/>
        <off speed="##speed" />
        <on speed="##speed" />
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="bd68900da93b59cc553026851cc05afb">
    <out result="kSuccess" method="SetSpeedForOFFOrON"/>
</sdk>
```

其他功能

GPS信息上报

GetGpsRespondEnable获取GPS信息上报使能标志

```
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetGpsRespondEnable"/>
</sdk>

<!--反馈
##enable 是否使能 {"true", "false"}
##delay GPS信息上传的间隔 (单位:秒)
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="GetGpsRespondEnable">
        <gps enable="##enable" delay="##delay"/>
    </out>
</sdk>
```

SetGpsRespondEnable设置GPS信息上报使能标志

```
<!--请求
##enable 是否使能 {"true", "false"}
##delay GPS信息上传的间隔 (单位:秒)
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetGpsRespondEnable">
        <gps enable="##enable" delay="##delay"/>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="SetGpsRespondEnable"/>
</sdk>
```

GPS信息报文

当二次开发用户调用“**SetGpsRespondEnable**”接口打开GPS信息上报使能标志后，当控制卡与服务器建立好TCP连接后即会按照二次开发用户设置的间隔时间往服务器周期性的主动上传GPS信息报文。

字段	长度(Len)	命令(Cmd)	纬度	经度	时间	保留
字节数	2字节	2字节	4字节	4字节	4字节	4字节
取值	20	kGPSInfoAnswer	latitude	longitude	seconds	reserved

```
typedef struct HGPSInfoAnswer
{
    huint16 len;         ///<命令包长度
    huint16 cmd;         ///<命令值
    hfloat  latitude;       ///<纬度
    hfloat  longitude;      ///<经度
    huint32 seconds;       ///<时间
    huint32 reserved;       ///<保留
}  HGPSInfoAnswer;
/**  字段说明
 * 1. latitude: 纬度; > 0表示北半球; < 0表示南半球
 * 2. longitude: 经度; >0表示东半球; < 0表示西半球
 * 3. seconds: 距离1970年1月1日 00:00:00的秒数
 * 4. reserved: 保留
 */
```

多屏同步标志

GetMulScreenSync获取多屏同步标志

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetMulScreenSync"/>
</sdk>

<!--反馈
##value 多屏同步使能标志, 取值范围{"true", "false"}
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="GetMulScreenSync">
        <enable value="##value"/>
    </out>
</sdk>
```

SetMulScreenSync设置多屏同步标志

```
<!--请求
##value 多屏同步使能标志，取值范围{"true", "false"}
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetMulScreenSync">
        <enable value="##value"/>
    </in>
</sdk>

<!--反馈
##value 多屏同步使能标志，取值范围{"true", "false"}
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="SetMulScreenSync" />
</sdk>
```

节目播放统计标志

GetPlayProgramCountsEnable获取节目播放统计标志

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetPlayProgramCountsEnable"/>
</sdk>


<!--反馈
##value 节目播放统计标志, 取值范围{"true", "false"}
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess"
method="GetPlayProgramCountsEnable">
        <enable value="##value"/>
    </out>
</sdk>
```

SetPlayProgramCountsEnable 设置节目播放统计

```
<!--请求
##value 节目播放统计标志，取值范围{"true", "false"}
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetPlayProgramCountsEnable">
        <enable value="##value"/>
    </in>
</sdk>


<!--反馈
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess"
method="SetPlayProgramCountsEnable"/>
</sdk>
```

节目播放统计文件名

GetPlayProgramCountsFileName获取节目播放统计文件名

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetPlayProgramCountsFileName"/>
</sdk>

<!--反馈
##file.name: 节目播放统计的文件名
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess"
method="GetPlayProgramCountsFileName">
        <file name="##name"/>
    </out>
</sdk>
```

节目功能

1. 节目构成请到 节目定义结构 查看

节目编辑

GetProgram获取节目信息

```
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetProgram"/>
</sdk>
```

AddProgram添加节目

> 节目相关节点请到 节目结构定义 查看
>
> guid - 全局唯一标识符. 同一个节目不允许存在两个相同的guid

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="AddProgram">
        <!--下面是节目单信息，详情请到节目结构定义查看-->
        <screen timeStamps="0">
            <program guid="##program-guid" type="normal">
                <backgroundMusic/>
                <playControl count="1"/>
                <border effect="rotate" speed="4" index="0"/>
                <area alpha="255" guid="##area-guid">
                    <rectangle x="0" height="128" width="128"
y="0"/>
                    <resources>
                        <text guid="##text-guid"
singleLine="false">
                            <style valign="middle"
align="center"/>
                            <string>例子</string>
                            <font name="SimSun"
italic="false" bold="false" underline="false" size="28"
color="#ffffff"/>
                        </text>
                    </resources>
                </area>
            </program>
        </screen>
        <!--到此处节目单节点结束-->
    </in>
</sdk>


<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="fafaed9fdb9668a5bb06c01f55e95518">
    <out result="kSuccess" method="AddProgram"/>
</sdk>
```

UpdateProgram更新对应节目

```
<!--请求
更新节目根据已有的节目匹配GUID来完成更新的.
需确保##program-guid, ##area-guid, ##text-guid的对应的guid都一致
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="UpdateProgram">
        <!--下面是节目单信息, 详情请到节目结构定义查看-->
        <program guid="##program-guid" type="normal">
            <playControl count="1"/>
            <border index="0" effect="rotate" speed="4"/>
            <area guid="##area-guid" alpha="255">
                <rectangle x="0" y="0" height="128"
width="128"/>
                <resources>
                    <text guid="##text-guid"
singleLine="false">
                        <style align="center"
valign="middle"/>
                        <string>例子</string>
                        <font bold="false" name="SimSun"
size="28" color="#ffffff" italic="false" underline="false"/>
                    </text>
                </resources>
            </area>
        </program>
        <!--到此处节目单节点结束-->
    </in>
</sdk>


<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="fafaed9fdb9668a5bb06c01f55e95518">
    <out result="kSuccess" method="UpdateProgram"/>
</sdk>
```

DeleteProgram删除节目

```
<!--请求
删除节目根据已有的节目匹配GUID来完成删除的.
需确保##program-guid, ##area-guid, ##text-guid的对应的guid都一致
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="DeleteProgram">
        <program guid="##program-guid" type="normal">
            <area guid="##area-guid" alpha="255">
                <resource>
                    <text guid="##text-guid"
singleLine="false">
                        <string>删除节目</string>
                    </text>
                </resource>
            </area>
        </program>
    </in>
</sdk>


<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="fafaed9fdb9668a5bb06c01f55e95518">
    <out result="kSuccess" method="DeleteProgram"/>
</sdk>
```

RealTimeUpdate区域更新SDK

用于更新指定区域, 而不是更新当前节目的所有区域

> 注: 有部分情况会导致更新该区域时判断节目需要全部区域重新播放时, 这时会重新播放全部区域

添加需要区域实时更新的节目演示

> 将需要修改的节目绑定上guid属性. 并给唯一的guid值, 为了 不影响观看, 使用(...)省略无关属性, 演示如下:
>
> 1. 首先先添加节目
> 2. 给需要实时更新的节目添加guid, 可以看到text标签添加了 guid属性. 这个在后续用于实时更新

```
<?xml version = "1.0" encoding = "utf-8" ?>
<sdk guid = "##GUID">
    <in method="AddProgram">
        <screen ...>
            <program ...>
                <playControl .../>
                <area ...>
                    <rectangle .../>
                    <resources>
                        <text guid="c544d34e-fd6c-11ea-8edc-
00005f6af662" name="" singleLine="false">
                            <string>text区域加上guid属性
</string>
                        </text>
                    </resources>
                </area>
            </program>
        </screen>
    </in>
</sdk>
```

RealTimeUpdate 区域更新

区域更新要求guid必须对应的节目必须存在, 并且节目类型为同一类型, 否则会返回错误码

允许更新多节点. 在最新版中该功能是异步功能, 等节目那边更新后才会通知回馈.

如属于远程资源, 那么file标签必须填写远程资源的md5属性, 否则将反馈**kSeekFileFailed**

注: 当处于下载状态时. 而再进行发送一次需要资源下载的实时更新, 则会导致下载失败. 且这次的需要资源下载的任务则被取消.

> text: 要求相同的guid即可更新
>
> image: 要求file标签需要name和图片的md5属性, 允许只使用name属性, 但遇到相同name时只更新第一个

```
<!--
content的子节点是需要更新节目
##guid 属性对应需要更新的节目的guid，要求两者必须相同，不存在则返回错误码
##type 属性为local和remote，local表示资源在本地上，remote表示资源在需要下载，此时文件标签必须存在name和md5属性
-->
<?xml version = "1.0" encoding = "utf-8" ?>
<sdk guid = "##GUID">
    <in method="RealTimeUpdate">
        <content guid="c544d34e-fd6c-11ea-8edc-00005f6af662" type="##type" >
            <text guid="c544d34e-fd6c-11ea-8edc-00005f6af662" singleLine="false">
                <string>Hello World</string>
                <font color="#0000ff"/>
            </text>
        </content>

        <content guid="6e55b916-a9c1-4522-bbfd-63e6c6a63436" type="local">
            <image guid="6e55b916-a9c1-4522-bbfd-63e6c6a63436" fit="stretch">
                <!-- 这个表示资源在本地的 -->
                <file name="test.jpg" md5="449fa54b8ad97ac56dd0f4b1d1f66dbd"/>
            </image>
        </content>

        <content guid="6e55b916-a9c1-4522-bbfd-63e6c6a63436" type="remote">
            <image guid="6e55b916-a9c1-4522-bbfd-63e6c6a63436" fit="stretch">
                <!-- 这个表示资源需要下载的，content的type属性是remote，因需要下载资源，此时file标签需要完备，必须存在md5属性 -->
                <file name="http://127.0.0.1/test.jpg" md5="449fa54b8ad97ac56dd0f4b1d1f66dbd"/>
            </image>
        </content>
```

```
        <!-- 如还有需要更新的, 可继续添加content标签 -->
        <content guid="..." type="##type">
            ...
        </content>
    </in>
</sdk>


<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="09e83c4a6fca57b429482ccbee93bd6a">
    <out method="RealTimeUpdate" result="kSuccess"/>
</sdk>
```

ScreenRotation屏幕旋转

设置屏幕旋转时, 会导致宽高调整和节目重播, 请不要在多屏同步时旋转.

> rotation标签的value值对应顺时针旋转
>
> 0 -> 0°
>
> 1 -> 90°
>
> 2 -> 180°
>
> 3 -> 270°

```
<!--
屏幕旋转请求
##value [0-3], 不属于均返回 kInvalidXmlIndex
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <in method="ScreenRotation">
        <rotation value="##value"/>
    </in>
</sdk>

<!--
屏幕旋转反馈
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="d77e142ea74b179d648e497ea820e6e2">
    <out result="kSuccess" method="ScreenRotation"/>
</sdk>
```

SwitchProgram节目切换

> 请确保关闭 多屏同步

可以使用节目的guid或者使用索引来进行节目切换

```
<!--
节目切换请求，guid和index选一个属性即可
如guid属性和index属性只存在一个时
##guid 节目不存在该guid则返回 kInvalidXmlIndex
##index [0-max) 节目索引，超出max时返回 kInvalidXmlIndex
如两者都存在时，guid属性优先级比index高，当guid找不到时才会使用index
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <in method="SwitchProgram">
        <!-- guid属性和index属性选一个存在即可 -->
        <program guid="##guid" index="##index"/>
    </in>
</sdk>

<!--
节目切换反馈
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="d77e142ea74b179d648e497ea820e6e2">
    <out result="kSuccess" method="SwitchProgram"/>
</sdk>
```

获取当前播放节目的GUID

```
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetCurrentPlayProgramGUID"/>
</sdk>

<!--
##program.guid 当前播放的节目的guid
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="fafaed9fdb9668a5bb06c01f55e95518">
    <out result="kSuccess"
method="GetCurrentPlayProgramGUID">
        <program guid="##guid"/>
    </out>
</sdk>
```

屏幕功能

GetScreenshot和GetScreenshot2屏幕截图

当前屏幕截图有两种方法. 一个是轮询设备是否截图完成, 第二个是异步等待设备回复的截图. 推荐使用第二个

GetScreenshot轮询操作截图(预弃用)

> 操作流程:
>
> 1. 使用GetScreenshot通知下位机开始截图
> 2. 使用GetScreenshotStatus获取图片数据是否存在
> 3. 使用GetScreenshotData获取图片数据

```
<!--
  1. 清理截图数据, 并通知下位机开始截取一张截图
  2. ##width取值范围{-int, int}, 默认(小于等于0则自 适应, 大于帧缓存范围是则自 适应)
  3. ##height取值范围{-int, int}, 默认(小于等于0则自 适应, 大于帧缓存范围是则自 适应)
  -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="GetScreenshot">
        <image width="##width" height="##height"/>
      </in>
</sdk>

<!-- 参数应答
    1. ##result kCreateFileFailed 则启动截图失败, 一般是节目未播放
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <out method="GetScreenshot" result="kSuccess">
    </out>
</sdk>
```

获取下位机截图状态, 用于辅助GetScreenshot(预弃用)

```
<!--
  1. 获取下位机截图状态, 判断下位机是否存在截图数据
  -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="GetScreenshotStatus">
      </in>
</sdk>

<!-- 参数应答
    1. ##result 是kProcessing则没有图片数据, 如发了GetScreenshot则正在截图中, kSuccess则存在截图数据
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <out method="GetScreenshotStatus" result="kSuccess">
    </out>
</sdk>
```

获取下位机截图数据, 用于辅助GetScreenshot(预弃用)

```
<!--
  1. 获取下位机截图数据
  -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="GetScreenshotData">
      </in>
</sdk>

<!-- 参数应答
    1. ##data数据是图片转换为base64的数据, 还原后的数据内容是jpg图片数据
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <out method="GetScreenshotData" result="kSuccess">
        <image data="##data"/>
    </out>
</sdk>
```

GetScreenshot2异步操作截图

> GetScreenshot2是异步截图的, 当你请求后, 你就可以继续做其他事情, 它不会阻塞设备操作.
>
> 同时这和GetScreenshot使用的是不同的缓存, 也就是说你不能使用上面的去检查设备是否有截图数据, 也没必要这样操作
>
> 在设备完成截图操作后, 会自 动向 你反馈数据你这次请求截图的数据, 而不需要再次请求一次. 当然, 这反馈速度是回 复时间和截取屏幕所耗费 的时间的总和的.

```
<!--
  1. 清理截图数据, 并进行下位机截取一张截图, 截图完毕后会进行回复数据
  2. ##width取值范围{-int, int}, 默认(小于等于0则自 适应, 大于帧缓存范围是则自 适应)
  3. ##height取值范围{-int, int}, 默认(小于等于0则自 适应, 大于帧缓存范围是则自 适应)
  -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="GetScreenshot2">
        <image width="##width" height="##height"/>
      </in>
</sdk>


<!-- 参数应答
    1. ##data数据是图片转换为base64的数据, 还原后的数据内容是jpg图片数据
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <out method="GetScreenshot2" result="kSuccess">
        <image data="##data"/>
    </out>
</sdk>
```

ScreenShot屏幕节目(已弃用)

> 仅部分设备支持

```
<!--请求-->
<?xml version = "1.0" encoding = "utf-8" ?>
<sdk guid="##GUID">
    <in method="ScreenShot" />
</sdk>

<!--反馈
##image base64的图片数据
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="624aedc8936411b50636e01248789d98">
    <out method="ScreenShot" result="kSuccess">
        <screenshot image="##image"/>
    </out>
</sdk>
```

屏幕开关功能

开关屏信息

GetSwitchTime获取开关屏信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetSwitchTime"/>
</sdk>

<!--反馈
##open.enable 当前屏幕状态
##ploy.enable 开启时间段控制开关屏 {"true"(开启), "false"(不开启)}
##item.enable 时间段该项是否使能
##item.start 开屏时刻, 格式hh:mm:ss
##item.end 关屏时刻, 格式hh:mm:ss
在item.start 至 item.end时间段内为开屏状态
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="bd68900da93b59cc553026851cc05afb">
    <out result="kSuccess" method="GetSwitchTime">
        <open enable="##enable"/>
        <ploy enable="##enable">
            <item enable="##enable" end="##end"
start="##start"/>
        </ploy>
    </out>
</sdk>
```

SetSwitchTime设置开关屏信息

```
<!--请求
##open.enable固定值"true"
##ploy.enable: 是否开启时间段控制开关屏, 取值范围{"true"(开启), "false"(不开启)}
##item.enable: 该项是否使能, 取值范围{"true"(使能), "false"(不使能)}
##item.start: 开屏时刻, 格式hh:mm:ss
##item.end: 关屏时刻, 格式hh:mm:ss
在item.start 至 item.end时间段内为开屏状态
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetSwitchTime">
        <open enable="##enable"/>
        <ploy enable="##enable">
            <item enable="##enable" end="##end" start="##start"/>
        </ploy>
    </in>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="bd68900da93b59cc553026851cc05afb">
    <out result="kSuccess" method="SetSwitchTime"/>
</sdk>
```

设置开关屏

OpenScreen开屏

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="OpenScreen"/>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="09bf0cceca82bb7c8cb494ddbb136a25">
    <out result="kSuccess" method="OpenScreen"/>
</sdk>
```

CloseScreen关屏

```
<!--关屏-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="CloseScreen"/>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="09bf0cceca82bb7c8cb494ddbb136a25">
    <out result="kSuccess" method="CloseScreen"/>
</sdk>
```

传感器功能

串口透传设置

> 1. Sensor口 支持Uart和RS232以及485
> 2. RS232和485需要转接板
> 3. Sensor口 是Uart接口

Set485Param设置串口设置

```
<!--
1. 设置485参数请求, 关闭则全部设置为空即可
2. 只需要设置一次, 设置项参数会进行保存
3. ##name取值为: "sensor"(表示485设备接sensor口); "gps"(表示485设备接gps口)
4. ##baudRate 波特率, 与对应串口设备相同, 一般为9600
5. ##dateBits 数据位, 与对应串口设备相同, 一般为8
6. ##stopBits 停止位, 与对应串口设备相同, 一般为1
7. ##parity      校验位, 与对应串口设备相同, 一般为n, 为不校验
8. ##timeout  使用ReadDataFrom485的超时时间, 单位为秒(s)
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <in method="Set485Param">
    <device name="##name" baudRate="##baudRate"
dataBits="##dataBits" stopBits="##stopBits"
parity="##parity"/>
    <read timeout="##timeout"/>
  </in>
</sdk>

<!-- 设置485参数应答 -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##guid">
  <out method="Set485Param" result="##result"/>
</sdk>
```

Get485Param获取串口设置

```
<!-- 获取485参数请求 -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
  <in method="Get485Param"/>
</sdk>

<!-- 获取485参数应答 -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##guid">
  <out method="Get485Param" result="##result">
    <device name="##name" baudRate="##baudRate"
dataBits="##dataBits" stopBits="##stopBits"
parity="##parity"/>
    <read timeout="##timeout"/>
  </out>
</sdk>
```

SendDataTo485发送数据到串口设备

> 增加配置校验, 当485串口配置存在空配置和超时时间<=0时, 返回kInvalidParam

1. 发送数据需要以转换为base64, 发送后设备会进行转换会原数据后转发到串口设备. 既支持二进制数据
2. base64数据替换下面 **##data** 标记符

```
<!--
  1. 发送数据给485设备请求
  2. ##data为485数据的base64编码
    -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <in method="SendDataTo485">
    <data value="##data"/>
  </in>
</sdk>

<!-- 发送数据给485设备应答 -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##guid">
  <out method="SendDataTo485" result="##result"/>
</sdk>
```

ReadDataFrom485读取串口设备数据

> 增加配置校验, 当485串口配置存在空配置和超时时间<=0时, 返回kInvalidParam
>
> 注: 设备有1k缓冲区, 超过1k缓存数据还未进行读取操作则会丢失前面的数据. 缓冲数据始终维持1k大小. 直到进行读取后将清空缓冲数据.

```
<!-- 从485设备读取数据请求 -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
  <in method="ReadDataFrom485"/>
</sdk>

<!--
  1. 从485设备读取数据应答
  2. ##data为485数据的base64编码
    -->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##guid">
    <out method="ReadDataFrom485" result="##result">
    <data value="##data"/>
  </out>
</sdk>
```

获取传感器信息功能

GetSensorInfo获取传感器信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetSensorInfo"/>
</sdk>

<!--反馈
##connect 连接状态取值false和true
##luminance 亮度传感器
##temp1 温度传感器
##humidity 湿度传感器
##temp2 温度2传感器
##telecontroller 遥控器
##gps gps
##windSpeed 风速
##windDirection 风向
##noise 噪音
##pressure气压
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="09bf0cceca82bb7c8cb494ddbb136a25">
    <out result="kSuccess" method="GetSensorInfo">
        <luminance connect="##connect"/>
        <temp1 connect="##connect"/>
        <humidity connect="##connect"/>
        <temp2 connect="##connect"/>
        <telecontroller connect="##connect"/>
        <gps connect="##connect"/>
        <windSpeed connect="##connect"/>
        <windDirection connect="##connect"/>
        <noise connect="##connect"/>
        <pressure connect="##connect"/>
    </out>
</sdk>
```

GetGPSInfo获取GPS信息

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetGPSInfo"/>
</sdk>

<!--反馈
##east 东方 {"false", "true"}
##north 北方 {"false", "true"}
##longitude 经度
##latitude 纬度
##counts 计数
##speed 速度
##direction 方向
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="ae26d72998f2e97eb38dcc7e96a15656">
    <out method="GetGPSInfo" result="kSuccess">
        <gps east="##east" north="##north"
longitude="##longitude" latitude="##latitude"
counts="##counts" speed="##speed" direction="##direction"/>
    </out>
</sdk>
```

GetCurrentSensorValue获取当前传感器的值

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetCurrentSensorValue"/>
</sdk>

<!--反馈
##luminance.current 亮度值
##temperature.current 温度值
##humidity.current 湿度值
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0f55d95d39606d0162e03efece728a5d">
    <out method="GetCurrentSensorValue" result="kSuccess">
        <luminance current="##current"/>
        <temperature current="##current"/>
        <humidity current="##current"/>
    </out>
</sdk>
```

串口SDK

GetSerialSDK获取串口SDK配置

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetSerialSDK"/>
</sdk>


<!--反馈
##name 串口接口名 {"sensor", "gps"}
##enable 开启串口SDK功能 {"true", "false"}
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0dda9f5d2f5243075256923b9a8dcde5">
    <out result="kSuccess" method="GetSerialSDK">
        <config name="##name" enable="##enable"/>
    </out>
</sdk>
```

SetSerialSDK设置串口SDK配置

```
<!--请求
##name 串口接口名 {"sensor", "gps"}
##enable 开启串口SDK功能 {"true", "false"}
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetSerialSDK">
        <config name="##name" enable="##enable"/>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0dda9f5d2f5243075256923b9a8dcde5">
    <out result="kSuccess" method="SetSerialSDK"/>
</sdk>
```

串口SDK发送和接收格式

字段	命令包长度(len)	命令(cmd)	xml总长度(total)	xml当前位置(index)	xml crc32	xml
字节数	2字节	2字节	4字节	4字节	4字节	n字节

> 串口SDK数据每包要求不大于9000字节, 超过9000字节进行分包处理

```
#pragma pack(1)
typedef struct HHeader
{
    huint16 len;    ///< 命令包长度
    huint16 cmd;    ///< 命令值
} HHeader;

typedef struct HSerialSDKAsk
{
    HHeader header;
    huint32 total;  ///< xml总长度
    huint32 index;  ///< xml在总xml的位置, 首包为0, 下一包就是切包的位置
    huint32 crc32;  ///< 当前包的xml数据的crc32校验值
    hint8   data[1];
} HSerialSDKAsk, HSerialSDKAnswer;
#pragma pack()
```

时间功能

时间校正功能

GetTimeInfo获取时间校正数据

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetTimeInfo"/>
</sdk>

<!--反馈
##timezone.value: 时区字符串，必须以(UTC+hh:mm)开始，例如:
(UTC+08:00)Beijing,Chongqing,HongKong,Urumchi
##summer.enable: 是否开启夏令时，取值范围{"true"(开启夏令时),
"false"(不开启夏令时)}
##sync.value: 是否开启时间自动同步，取值范围{"none"(不开启自动同步),
"gps_rs232", "gps_usb", "gps_pcie"(gps校时), "network"(网络校
时), "auto"(自动校时), "rf" (rf校时)}
自动校时：有gps接入时自动选择使用gps校时，否则则使用网络校时
##time.value: 当前控制卡时间，格式: "yyyy-mm-dd hh:MM:ss", 例如:
"2017-01-01 00:00:00"
##server.list 服务列表，逗号分隔
##rf.enable.value rf同步是否使能{"false"不使能 "true"使能}
##rf.master.value 主设备{"false", "true"}
##rf.channel.value 通道
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="0f55d95d39606d0162e03efece728a5d">
    <out method="GetTimeInfo" result="kSuccess">
        <timezone value="##value"/>
        <summer enable="##enable"/>
        <sync value="##value"/>
        <time value="##time"/>
        <server list="##list"/>
        <rf>
            <enable value="##value"/>
            <master value="##value"/>
            <channel value="##value"/>
        </rf>
    </out>
</sdk>
```

SetTimeInfo设置时间校正数据

```
<!--请求
##timezone.value: 时区字符串，必须以(UTC+hh:mm)开始，例如:
(UTC+08:00)Beijing,Chongqing,HongKong,Urumchi
##summer.enable: 是否开启夏令时，取值范围{"true"(开启夏令时),
"false"(不开启夏令时)}
##sync.value: 是否开启时间自动同步，取值范围{"none"(不开启自动同步),
"gps"(gps校时), "network"(网络校时), "auto"(自动校时)}
##time.value: 将设置设备的时间，该值在sync.value="none"时生效，其他
情况忽略，格式 yyyy-MM-dd hh:mm:ss
##server.list 服务列表，逗号分隔
##rf.enable.value rf同步是否使能{"false"不使能 "true"使能}
##rf.master.value 主设备{"false", "true"}
##rf.channel.value 通道
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetTimeInfo">
        <timezone value="##value"/>
        <summer enable="##enable"/>
        <sync value="##value"/>
        <time value="##value"/>
        <!--后面新增的，兼容旧版-->
        <server list="##list"/>
        <rf>
            <enable value="##value"/>
            <master value="##value"/>
            <channel value="##value"/>
        </rf>
    </in>
</sdk>

<!--反馈-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="SetTimeInfo"/>
</sdk>
```

升级功能

SDK固件升级

ExcuteUpgradeShell固件升级

```
<!--
固件升级请求
##name 为固件名, 请确保该固件已上传到设备
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="ExcuteUpgradeShell">
        <file name="##name"/>
      </in>
</sdk>

<!--
固件升级反馈
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <out method="ExcuteUpgradeShell" result="kSuccess"/>
</sdk>
```

ExcuteUpgradeShellHttp固件升级

> ExcuteUpgradeShellHttp支持http下载固件后进行升级

```
<!--
固件升级请求
##name 为固件名或者http的路径
##md5 固件的md5值
##size 固件的大小
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="ExcuteUpgradeShellHttp">
        <file name="##name" md5="##md5" size="##size"/>
      </in>
</sdk>

<!--
固件升级反馈
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <out method="ExcuteUpgradeShellHttp" result="kSuccess"/>
</sdk>
```

FirmwareUpgrade固件升级

> 和ExcuteUpgradeShellHttp相同, 返回下载中的状态值不同.

```
<!--
固件升级请求
##name 为固件名或者http的路径
##md5 固件的md5值
##size 固件的大小
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
      <in method="FirmwareUpgrade">
        <file name="##name" md5="##md5" size="##size"/>
      </in>
</sdk>

<!--
固件升级反馈
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <out method="FirmwareUpgrade" result="kSuccess"/>
</sdk>
```

GetUpgradeResult获取固件升级状态

```
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <in method="GetUpgradeResult"/>
</sdk>

<!--
固件升级反馈
##value 为"success"(升级成功), "failed"(升级失败), "upgrading"(升级中) , "none"(没有升级)
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="##GUID">
    <out method="GetUpgradeResult" result="kSuccess">
        <upgrade result="##value"/>
    </out>
</sdk>
```

U盘功能

检查U盘接入状态

CheckUDiskInsert检查U盘是否接入

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="CheckUDiskInsert"/>
</sdk>

<!--反馈
##value {"true"接入 "false"未接入}
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="CheckUDiskInsert">
        <insert value="##value"/>
    </out>
</sdk>
```

U盘功能配置

GetEnableUDiskFunction获取U盘功能配置

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetEnableUDiskFunction"/>
</sdk>

<!--反馈
##effective.value 在U盘什么状态的时候允许被修改 {"all"所有状态
"insert"仅U盘接入时}
##play.video 和image两者需相同, 允许播放U盘根目录视频和图片 {"true"
"false"}
##play.image 和video两者需相同
##import.config 允许导入配置参数 {"true" "false"}
##import.immeProgram 允许直接播放U盘节目 {"true" "false"}
##import.updateProgram 允许更新U盘节目 {"true" "false"}
##import.upgrade 允许升级U盘固件 {"true" "false"}
##storage.enable 允许U盘存储 {"true" "false"}
-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="GetEnableUDiskFunction">
        <effective value="##value"/>
        <play video="##video" image="##image"/>
        <import config="##config" immeProgram="##immeProgram"
updateProgram="##updateProgram" upgrade="##upgrade"/>
        <storage enable="##enable"/>
    </out>
</sdk>
```

SetEnableUDiskFunction开启或设置U盘功能配置

```
<!--请求
##effective.value 在U盘什么状态的时候允许被修改 {"all"所有状态
"insert"仅U盘接入时}
##play.video 和image两者需相同, 允许播放U盘根目录视频和图片 {"true"
"false"}
##play.image 和video两者需相同
##import.config 允许导入配置参数 {"true" "false"}
##import.immeProgram 允许直接播放U盘节目 {"true" "false"}
##import.updateProgram 允许更新U盘节目 {"true" "false"}
##import.upgrade 允许升级U盘固件 {"true" "false"}
##storage.enable 允许U盘存储 {"true" "false"}
-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="SetEnableUDiskFunction">
        <effective value="##value"/>
        <play image="##image" video="##video"/>
        <import upgrade="##upgrade"
updateProgram="##updateProgram" config="##config"
immeProgram="##immeProgram"/>
        <storage enable="##enable"/>
    </in>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="SetEnableUDiskFunction"/>
</sdk>
```

DisableUDiskFunction关闭U盘功能配置

> 在使用SetEnableUDiskFunction时, 将自 动开启U盘功能配置

```
<!--请求-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="DisableUDiskFunction"/>
</sdk>

<!--反馈-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="7a8fe8bb88ccfab7dc57adc9c5f70e2e">
    <out result="kSuccess" method="DisableUDiskFunction"/>
</sdk>
```

扩展功能

自定义绘制模拟时钟属性

颜色计算公式

值为color. 需自 行计算转换为10进制值

R=color & 255; 红色

G = (color >> 8) & 255; 绿色

B = (color >> 8) & 255; 蓝色

> 关闭 AM/PM相关

```
<!-- 1为显示, 其余关闭 -->
<clock Analog_AMPM_d="0">
...
</clock>
```

> MinuteScaleType属性为分钟刻度类型0椭圆 1矩形 2线
>
> MinuteScaleColor属性为绘制分钟刻度颜色
>
> HourHandColor 时针颜色
>
> MinuteHandColor 分针颜色
>
> SecondHandColor 秒针颜色

```
<clock MinuteScaleColor="16711680" ...>
...
</clock>
```

一些问题的解决方案

GetProgram获取的数据用于节目数据时常出现的问题

1. 常见出现黑屏问题: **resources** 标签不正确

> 使用 **GetProgram** 获取的节目数据中, 资源标签是 **resource**, 而添加节目和更新节目的资源标签是带 **s** 的 **resources** 如果资源标签错误. 那么将导致这个节目不存在. 只有区域节目. 这时就会显示只有黑屏了 .

GetProgram的数据

```
<!--这个是请求的数据-->
<?xml version='1.0' encoding='utf-8'?>
<sdk guid="##GUID">
    <in method="GetProgram"/>
</sdk>


<!--这个是返回 的数据-->
<?xml version="1.0" encoding="utf-8"?>
<sdk guid="15811595b4daf72fe80e7e078de6edd4">
    <out result="kSuccess" method="GetProgram">
        <screen ...>
            <program ...>
                <playControl count="1"/>
                <border .../>
                <area ...>
                    <rectangle height="32" x="0" y="0"
width="32"/>
                    <border .../>
                    <!--把这些数据用于更新节目的数据时, 就是这个资源
标签需要注意, 这个是没有s, 正确的更新数据的资源标签是有s的. resources
这种-->
                    <resource>
                        <text ...>
                            <effect .../>
                            <style .../>
                            <string>test</string>
                            <font size="20" name="Arial"
italic="false" underline="false" color="#ffff00"
bold="true"/>
                        </text>
                    </resource>
                </area>
            </program>
        </screen>
    </out>
</sdk>
```

HDPlayer节目和SDK节目的区别

1. HDPlayer绘制的文本节目是由HDPlayer绘制出图片，然后发送的节目是图片，而不是文字.
2. SDK发送的节目是由设备来进行绘制的.
3. 因为绘制设备不同，SDK的节目会需要所有的数据来绘制相应的节目，而HDPlayer已经绘制好了.所以节目数据不包含SDK所有的xml数据.无法用于SDK来更新节目.

> 结论：无法使用HDPlayer的节目来当作SDK节目.因此SDK的更新节目无法用于将HDPlayer的节目更新.

HDPlayer和SDK节目的不同

1. HDPlayer节目都是由自身先进行绘制处理的.(因此不发送绘制数据给设备，只发送节目数据)
2. SDK节目都是将所需数据发送到设备.由设备进行绘制处理.(需要携带所有的绘制数据到节目数据)

HDPlayer为什么不和SDK一样将数据发送到设备处理呢

1. 性能差.电脑性能和设备性能差很显然.用电脑绘制可以使用绘制出质量很高的精致的数据.
2. 减少负担.在当个节目中，内容很多时，根据设备的不同有些设备是无法将SDK的全部数据绘制完的.而HDPlayer会全部绘制后构建节目单即可，同时发送时将已绘制的数据发送下去，这样设备就不需要再次进行绘制.不需要设备再承担绘制这些数据的负担.