|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 【密级程度】 公开 √ 机密 绝密 \_\_   |  | | --- | | **设置项功能** |   文件编号：  版 本 号： V1.0  拟制日期： 2017-09-19  发布日期：   |  |  |  | | --- | --- | --- | | 批准 | 审核 | 编制 | |  |  | 罗清文 | | 日期： | 日期： | 日期：2017-09-19 |  |  |  | | --- | --- | | 发文部门 | 研发部 | | 收文部门 | 研发文控 √ | | 市场部 | | 采购部 | | 品质部 | |

# 目录

[目录 2](#_Toc494274011)

[1. 概述 3](#_Toc494274012)

[1.1. 主要功能 3](#_Toc494274013)

[1.2. 功能描述 3](#_Toc494274014)

[2. 数据包定义 3](#_Toc494274015)

[2.1 亮度设置 3](#_Toc494274016)

[2.2 时间校准 6](#_Toc494274017)

[2.3 开关屏 8](#_Toc494274018)

[2.4 开机画面 12](#_Toc494274019)

[2.5 TCP服务器地址设置 14](#_Toc494274020)

[2.6 获取设备信息 16](#_Toc494274021)

**设置项功能**

1. 概述

## 主要功能

本文主要叙述与下位机设置项相关的内容，主要由：亮度设置、时间校正、开关屏、开机画面、TCP服务器地址设置等部分组成。

## 功能描述

1. 亮度设置：支持默认亮度、自定义亮度、传感器自动调整亮度等模式。
2. 时间校正：支持时间自动校正、时区、夏令时等功能。
3. 开关屏：支持按时段播放。
4. 开机画面：支持自定义开机画面。
5. TCP服务器地址设置：上位机可以通过该功能设置TCP服务器地址，下位机在开机启动后会自动连接(TCP连接)到相应的地址。
6. 数据包定义

## 亮度设置

1. 获取亮度设置请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"GetLuminancePloy"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 获取亮度设置反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"GetLuminancePloy"** result=**"##result"**>

<mode value=**"##value"**/>

<default value=**"##value"**/>

<ploy>

<item enable=**"##value"** start=**"##value"** percent=**"##value"**/>

</ploy>

<sensor min=**"##value"** max=**"##value"** time=**"##value"**/>

</out>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为GetWifiInfo

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

4. out.result取值不为"kSuccess"时, out无子节点

5. mode.value取值范围{"default"(默认), "ploys"(自定义模式), "sensor"(传感器模式)}

6. 亮度等级的取值范围[1, 100]

7. default.value: 默认模式时, 设置的亮度等级

8. 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

9. item.enable: 该项是否使能, 取值范围{"true"(使能), "false"(不使能)}

10.item.start: 改变亮度等级的开始时间, 格式: hh:mm:ss

11.item.percent: 亮度等级

12.sensor.min: 传感器模式最小亮度等级

13.sensor.max: 传感器模式最大亮度等级

14.sensor.time: 亮度调整的间隔时间, 取值范围[5, 15] (单位秒)

-->

1. 设置亮度策略请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"SetLuminancePloy"**>

<mode value=**"##value"**/>

<default value=**"##value"**/>

<ploy>

<item enable=**"##value"** start=**"##value"** percent=**"##value"**/>

</ploy>

<sensor min=**"##value"** max=**"##value"** time=**"##value"**/>

</in>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. mode.value取值范围{"default"(默认), "ploys"(自定义模式), "sensor"(传感器模式)}

3. 亮度等级的取值范围[1, 100]

4. default.value: 默认模式时, 设置的亮度等级

5. 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

6. item.enable: 该项是否使能, 取值范围{"true"(使能), "false"(不使能)}

7. item.start: 改变亮度等级的开始时间, 格式: hh:mm:ss

8. item.percent: 亮度等级

9. sensor.min: 传感器模式最小亮度等级

10.sensor.max: 传感器模式最大亮度等级

11.sensor.time: 亮度调整的间隔时间, 取值范围[5, 15] (单位秒)

-->

1. 设置亮度策略反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"SetLuminancePloy"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"SetLuminancePloy"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

## 时间校准

1. 获取时间校正请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"GetTimeInfo"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 获取时间校正反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"GetTimeInfo"** result=**"##result"**>

<timezone value=**"##value"**/>

<summer enable=**"##value"**/>

<sync value=**"##value"**/>

<time value=**"##value"**/>

</out>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为GetTimeInfo

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

4. out.result取值不为"kSuccess"时, out无子节点

5. timezone.value: 时区字符串, 必须以(UTC+hh:mm)开始, 例如: (UTC+08:00)Beijing,Chongqing,HongKong,Urumchi

6. 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)}

7. 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)}

8. 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

9. 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"

-->

1. 设置时间校正请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"SetTimeInfo"**>

<timezone value=**"##value"**/>

<summer enable=**"##value"**/>

<sync value=**"##value"**/>

<time value=**"##value"**/>

</in>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. timezone.value: 时区字符串, 必须以(UTC+hh:mm)开始, 例如: (UTC+08:00)Beijing,Chongqing,HongKong,Urumchi

3. 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)}

4. 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)}

5. 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

-->

1. 设置时间校正反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"SetTimeInfo"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"SetTimeInfo"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

## 开关屏

1. 获取开关屏请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"GetSwitchTime"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 获取开关屏反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"GetSwitchTime"** result=**"##result"**>

<open enable=**"##value"**/>

<ploy enable=**"##value"**>

<item enable=**"##value"** end=**"##value"** start=**"##value"**/>

</ploy>

</out>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为GetTimeInfo

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

4. out.result取值不为"kSuccess"时, out无子节点

5. open.enable固定值"true"

6. ploy标签可以有多个item子节点

7. 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)}

8. item.enable: 该项是否使能, 取值范围{"true"(使能), "false"(不使能)}

9. item.start: 开屏时刻, 格式hh:mm:ss

10.item.end: 关屏时刻, 格式hh:mm:ss

11.在item.start 至 item.end时间段内为开屏状态

-->

1. 设置开关屏请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"SetSwitchTime"**>

<open enable=**"##value"**/>

<ploy enable=**"##value"**>

<item enable=**"##value"** end=**"##value"** start=**"##value"**/>

</ploy>

</in>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. open.enable固定值"true"

3. ploy标签可以有多个item子节点

4. 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)}

5. item.enable: 该项是否使能, 取值范围{"true"(使能), "false"(不使能)}

6. item.start: 开屏时刻, 格式hh:mm:ss

7. item.end: 关屏时刻, 格式hh:mm:ss

8. 在item.start 至 item.end时间段内为开屏状态

-->

1. 设置开关屏反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"SetSwitchTime"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"SetSwitchTime"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

1. 立即开屏请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"OpenScreen"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 立即开屏反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"OpenScreen"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"OpenScreen"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

1. 立即关屏请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"CloseScreen"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 立即关屏反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"CloseScreen"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"CloseScreen"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

## 开机画面

1. 获取开机画面状态请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"GetBootLogo"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 获取开机画面状态反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"GetBootLogo"** result=**"##result"**>

<logo exist=**"##value"** name=**"##value"** md5=**"##value"**/>

</out>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"GetBootLogo"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

4. out.result取值不为"kSuccess"时, out无子节点

5. logo.exist: 是否设置开机画面, 取值范围{"true"(已设置), "false"(未设置)}

6. logo.name: 开机画面引用图片的名字

7. logo.md5: 开机画面引用图片的md5值

-->

1. 设置开机画面请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"SetBootLogoName"**>

<logo name=**"##value"** md5=**"##value"**/>

</in>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. logo.name: 开机画面引用图片的名字

3. logo.md5: 开机画面引用图片的md5值, 可设置为""

-->

1. 设置开机画面反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"SetBootLogoName"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"SetBootLogoName"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

1. 清除开机画面请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"ClearBootLogo"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 清除开机画面反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"ClearBootLogo"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"ClearBootLogo"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

## TCP服务器地址设置

1. 获取TCP服务器地址信息请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"GetSDKTcpServer"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 获取TCP服务器地址信息反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"GetSDKTcpServer"** result=**"##result"**>

<server host=**"##value"** port=**"##value"**/>

</out>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"GetSDKTcpServer"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

4. server.host: tcp服务器ip或者域名

5. server.port: tcp服务器监听的端口

-->

1. 设置TCP服务器地址请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"SetSDKTcpServer"**>

<server host=**"##value"** port=**"##value"**/>

</in>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. server.host: tcp服务器ip或者域名

3. server.port: tcp服务器监听的端口

-->

1. 设置TCP服务器地址反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"SetSDKTcpServer"** result=**"##result"**/>

</sdk>

<!--

详情:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"SetSDKTcpServer"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

-->

## 获取设备信息

1. 获取设备信息请求(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<in method=**"GetDeviceInfo"**/>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

-->

1. 获取设备信息反馈(参考<<SDK协议流程.docx>>中4.2SDK协议交互小结)

<!-- data的取值为 -->

<?xml version=**"1.0"** encoding=**"utf-8"**?>

<sdk guid=**"##GUID"**>

<out method=**"GetDeviceInfo"** result=**"##result"**>

<device cpu=**"##value"** model=**"##value"** id=**"##value"** name=**"##value"**/>

<version fpga=**"##value"** app=**"##value"** kernel=**"##value"**/>

<screen width=**"##value"** rotation=**"##value"** height=**"##value"**/>

</out>

</sdk>

<!--

详解:

1. sdk.guid为通过SDK协议版本协商交互获取到的guid值

2. out.method固定值为"GetBootLogo"

3. out.result取值参考<<SDK协议流程.docx>>文档中的2.3小结(错误码定义)

4. out.result取值不为"kSuccess"时, out无子节点

5. device.cpu: 设备cpu类型, 取值范围{"Freescale.iMax6", "TI.335x", "ZTE.902c"}

6. device.model: 设备类型, 取值范围{"A601", "A602", "A603", "D1", "D3", "C1i", "C3i", "C30", "C10", "D30", "D20", "D10", "A30", "A30+", "C10+", "C30+"}

7. device.id: 设备ID

8. device.name: 设备名

9. version.fpga: fpga固件版本号

10.version.app: 下位机固件版本号

11.version.kernel: 内核版本号

12.screen.width: 屏幕宽度

13.screen.height: 屏幕高度

14.screen.rotation: 旋转标志, 取值范围{"0", "90", "180", "270"}

-->