FAW HS7

IVI System Design

SM-API User Guide

Released on the: 2017.06.01

Version Number: 0.500

Neusoft Automotives ChangChun

**Version History**

| No. | Version | Section | Brief Description | Date | Rseponsible |
| --- | --- | --- | --- | --- | --- |
| 1 | 0.500 | All | Initial Create | 2017-06-01 | LG |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Content

[1 概述 4](#_Toc5170)

[2 相关文件 4](#_Toc12248)

[3 接口说明 4](#_Toc3141)

[3.1 SM-API使用说明 4](#_Toc2769)

[3.2 查看状态是否存在 4](#_Toc11754)

[3.3 获取事件状态 4](#_Toc6612)

[3.4 设置事件请求 4](#_Toc544)

[3.5 注册响应函数 5](#_Toc525)

[3.6 更新状态 5](#_Toc24838)

[3.7 注册状态改变回调函数 5](#_Toc19536)

[3.8 示例代码 5](#_Toc17538)

[4 Refrences 7](#_Toc30121)

# 概述

本文档主要描述SM-API。SM主要负责应用程序状态信息的管理，应用程序可以使用此接口来设置和获取功能状态信息。

# 相关文件

头文件：/usr/include/appSDK/AppSM.h

库文件：/usr/lib/libappSDK.so

# 接口说明

## SM-API使用说明

约束条件：

* 系统中AppManager进程正常运行；
* 此应用程序通过AppManager启动；
* 使用SM-API前需要首先初始化AppSDK；

## 查看状态是否存在

|  |  |  |
| --- | --- | --- |
| *declare* | bool SM\_isStateExist(std::string state) | |
| *description* | 查看状态是否存在 | |
| *parameter* | std::string state | 事件 |
| *return* | true 存在 false 不存在 | |

状态使用者

## 获取事件状态

|  |  |  |
| --- | --- | --- |
| *declare* | std::string SM\_getState(std::string state) | |
| *description* | 获取事件状态 | |
| *parameter* | std::string state | 事件 |
| *return* | std::string 事件状态或unknown | |
| *note* | Unknown表示更新该事件状态的service还没有启动或完成更新，此时需要不处理或特殊处理。  建议处理方式：  如果是unknown不做任何处理，service启动后update该事件状态时会有回调通知 | |

状态使用者

## 设置事件请求

|  |  |  |
| --- | --- | --- |
| *declare* | void SM\_req2SetState(std::string state,std::string value) | |
| *description* | 设置事件状态请求 | |
| *parameter* | std::string state | 设置事件的类型 |
| std::string value | 设置事件的具体状态 |
| *return* | none | |

状态使用者

## 注册响应函数

|  |  |  |  |
| --- | --- | --- | --- |
| *declare* | void SM\_regReply4Req(SM\_handleReqSet callback) | | |
| *description* | 响应设置事件状态请求 | | |
| *parameter* | SM\_handleReqSet callback | typedef void(\* SM\_handleReqSet)(uint16\_t ampid\_req, std::string key, std::string value); | |
| uint16\_t ampid\_req | 请求设置该状态的app id |
| std::string key | 事件的类型 |
| std::string value | 事件的具体状态 |
| *return* | none | | |

状态拥有者（维护状态）

## 更新状态

|  |  |  |
| --- | --- | --- |
| *declare* | void SM\_updateState(uint16\_t ampid\_req, string state,string value) | |
| *description* | 设置状态后更新该状态 | |
| *parameter* | uint16\_t ampid\_req | 请求设置该状态的app id |
| std::string state | 事件的类型 |
| std::string value | 事件的具体状态 |
| *return* | none | |

状态拥有者（维护状态）

## 注册状态改变回调函数

|  |  |  |  |
| --- | --- | --- | --- |
| *declare* | void SM\_regCb4StateChanged(SM\_handleStateChange callback) | | |
| *description* | 设置状态后广播通知，注册回调 | | |
| *parameter* | SM\_handleStateChange callback | typedef void(\* SM\_handleStateChange)(uint16\_t ampid\_req, uint16\_t ampid\_reply, std::string key, std::string value) | |
| uint16\_t ampid\_req | 请求设置该状态的app id |
| uint16\_t ampid\_reply | 设置该状态的app id（状态由响应的app设置，即谁设置的） |
| std::string key | 事件的类型 |
| std::string value | 事件的具体状态 |
| *return* | none | | |

状态使用者

## 示例代码

以多媒体音量为例

|  |  |
| --- | --- |
| *设置状态回调函数* | void STController::sset\_stateSet(uint16\_t ampid\_req, std::string key, std::string value)  {  logInfo(SSER,"stateinfo.cpp::sset\_stateSet::ampid\_req",ampid\_req,key," =",value,"-->>IN");  //write ......    SM\_updateState(ampid\_req,key,value);    logInfo(SSER,"stateinfo.cpp::sset\_stateSet::ampid\_req",ampid\_req,key," =",value,"-->>OUT");  } |
| *注册设置回调函数* | SM\_regReply4Req(STController::sset\_stateSet); |
| *初始化更新所有状态* | STController::sset\_orgStateUpdate();  void STController::sset\_orgStateUpdate(void)  {  logVerbose(SSER,"stateinfo.cpp::sset\_orgStateUpdate-->>IN");  for (std::map<string,string>::iterator it=key\_orgvalue\_map.begin(); it!=key\_orgvalue\_map.end(); ++it)  {  if(SM\_isStateExist(it->first))  {  string value = STController::sset\_stateGet(it->first);  logInfo(SSER,"stateinfo.cpp::sset\_orgStateUpdate::",it->first," =",value);  SM\_updateState(GetAMPID(),it->first,value);  }  else  {  logWarn(SSER,"stateinfo.cpp::sset\_orgStateUpdate::",it->first,"is not exist");  }  }  logVerbose(SSER,"stateinfo.cpp::sset\_orgStateUpdate-->>OUT");  } |
| *状态改变回调函数* | void stateChangeNotify(uint16\_t ampid\_req, uint16\_t ampid\_reply, std::string key, std::string value)  {  logVerboseF(TBAR,"tbar\_api.cpp::stateChangeNotify::key:%s value:%s ampid\_reply:0x%x ampid\_req:0x%x-->IN",key.c\_str(),value.c\_str(),ampid\_reply,ampid\_req);  //treat  logVerboseF(TBAR,"tbar\_api.cpp::::stateChangeNotify::key:%s value:%s ampid\_reply:0x%x ampid\_req:0x%x-->OUT",key.c\_str(),value.c\_str(),ampid\_reply,ampid\_req);  } |
| *注册状态改变回调函数* | SM\_regCb4StateChanged(stateChangeNotify); |
| *请求改变状态* | SM\_req2SetState("syssetting\_RSE1\_SOFTKEY\_VOL","25"); |
| *获取状态* | SM\_getState("syssetting\_date"); |

# Refrences