**基于SmartDeviceLink的图片展示开发文档**

<V1.0.2>

**2016年03月02日**

1 图片展示逻辑介绍 1

2 类介绍 1

3 实现图片展示 3

4 版本更新 7

# 图片展示逻辑介绍

Gen3添加了对图片的支持，但是需要对图片进行特殊的上传处理，也就是需要用到SDK中的SDLPutFile类把图片上传到TDK。总体的逻辑是：上传图片 -> 获得已经上传的图片的名称 -> 对相应的控件（例如：SDLShow、SDLSetAppIcon等）的图片属性赋值 -> 发送相应的控件到TDK，从而实现图片的添加和展示。

# 类介绍

在此次SDK图片扩展的版本中，存在以下几个处理类：

* **2.1基类**
  + **类名称：FORDSendImageBaseClass**
  + **作用：所有图片处理类需要继承的基类，此类中包含了子类所需要的共有成员属性和方法，子类可以重写基类方法已达到自己的效果。**
* **2.2发送图片初始类**
  + **类名称：FORDSendImageObject**
  + **作用： 所有子类展示图片的起始类，此类中为每一个子类开启一个单独的队列处理，并且通过此类的方法进行图片数据的传递。**
  + **注意：SYNC断开后需要调用类中实例对象方法**

[FORDSendImageObject shareSendImageObjectWithSync:nil];

* **2.3发送图片到TDK处理类**
  + **类名称：FORDPutImageFile**
  + **作用： 此类专用于对图片的上传处理，其上传的数据类型为NSData，图片的名称通过MD5加密而来，支持的图片类型包括：PNG、JPG、BMP三种格式。此类通过Block进行数据的传递和回调通知。**
* **2.4 SDLSetAppIcon图片处理类**
  + **类名称：FORDSendIconImage**
  + **作用： 此类专用于对准备上传的Icon图片数据进行管理，通过数组保存所有的Icon图片数据，依据先进先出、后进后出的原则对数组内的图片一一进行上传，只有当前一个图片数据上传过程结束之后才可进行下一个图片数据的上传动作，直至数组中不存在任何图片数据为止。当收到图片上传成功的回调后，此时sync会对SDLSetAppIcon控件做出发送动作。**
* **2.5 SDLShow图片处理类**
  + **类名称：FORDSendShowImage**
  + **作用：此类专用于对准备上传的show图片数据进行管理，通过数组保存所有的show图片数据，依据先进先出、后进后出的原则对数组内的图片一一进行上传，只有当前一个图片数据上传过程结束之后才可进行下一个图片数据的上传动作，直至数组中不存在任何图片数据为止。当收到图片上传成功的回调后，此时sync会对SDLShow控件做出发送动作。**
* **2.6 SDLSoftButton图片处理类**
  + **类名称：FORDSendSoftButtonImage**
  + **作用：此类专用于对准备上传的softButton图片数据进行管理，通过数组保存所有的softButton图片数据，依据先进先出、后进后出的原则对数组内的图片一一进行上传，只有当前一个图片数据上传过程结束之后才可进行下一个图片数据的上传动作，直至数组中不存在任何图片数据为止。当收到图片上传成功的回调且图片数组中所有图片都已上传，此时sync会对SDLSoftButton数组做出发送动作。需要注意的是，此类中含有一个保存SDLSoftButton控件的数组，此数组的顺序和个数与图片数组中的顺序和个数是需要保持一致的，否则会出现图片和softButton控件不匹配的现象出现。**
* **2.7 SDLAddCommand图片处理类**
  + **类名称：FORDSendCommandImage**
  + **作用：此类专用于对准备上传的addCommand图片数据进行管理，通过数组保存所有的addCommand图片数据，依据先进先出、后进后出的原则对数组内的图片一一进行上传，只有当前一个图片数据上传过程结束之后才可进行下一个图片数据的上传动作，直至数组中不存在任何图片数据为止。当收到图片上传成功的回调后，此时sync会对SDLAddCommand控件做出发送动作。需要注意的是，此类中含有一个保存SDLAddCommand控件的数组，此数组的顺序和个数与图片数组中的顺序和个数是需要保持一致的，否则会出现图片和addCommand控件不匹配的现象出现。**
* **2.8 SDLChoice图片处理类**
  + **类名称：FORDSendPerformInteractionImage**
  + **作用：此类专用于对准备上传的choice图片数据进行管理，通过数组保存所有的choice图片数据，依据先进先出、后进后出的原则对数组内的图片一一进行上传，只有当前一个图片数据上传过程结束之后才可进行下一个图片数据的上传动作，直至数组中不存在任何图片数据为止。SDLChoice图片的处理较其它处理方式有所不同，我们最终想要在GEN3上展示的其实是SDLPerformInteraction控件，而SDLPerformInteraction控件是由SDLCreateInteractionChoiceSet组成，同时后者又是由SDLChoice组成，所以归结到底对SDLPerformInteraction图片的处理其实是对SDLChoice图片的处理，所以当收到choice图片上传成功的回调后，会发出一个通知，通知sync的主干choice图片已处理完毕，可以进行下一步操作。**
* **2.9图片数据字典扩展类**
  + **类名称：NSMutableDictionary+FORDPutImageDictionary**
  + **作用： 对NSMutableDictionary做一个简单的扩展，扩展内容主要是添加一些成员属性。而这些成员属性则是为了保存将要上传到TDK的图片的各类数据值。**
  + **成员属性说明：**
    - imageNameOfMd5String：经过MD5处理过的图片名称，此名称用于控件中图片的名称赋值
    - imageNameOfLocalString：图片本地名称
    - imageLocalOfNetworkPathString：图片本地保存的路径或网络路径
    - displayLayout：屏幕设置的布局类型（例如：LARGE\_GRAPHIC\_ONLY）
    - imagePositionOnScreen：图片在屏幕上展示的位置（例如SDLChoice的位置）
    - imageShowType：图片需要展示的位置

（SendImageShowTypeOfIcon|SendImageShowTypeOfShow|SendImageShowTypeOfSoftButton|SendImageShowTypeOfAddCommand|SendImageShowTypePerformInteraction）

* + - imageSourceType图片的来源位置

（SendImageOfSourceTypeOfBundle|SendImageOfSourceTypeOfSandbox|SendImageOfSourceTypeOfNetwork）

* + - imagePutFileCorrelationID：上传图片数据的RPC的correlationID
    - imageShowCorrelationID：发送控件的RPC的correlationID
    - imagePutState：图片上传的结果：成功|失败

# 实现图片展示

* **3.1 SDLSetAppIcon图片**
  + **第一步设置图片数据**
* NSMutableDictionary \*iconDict = [[NSMutableDictionary alloc]init];
* iconDict.imageNameOfLocalString = @"logo.png";
* iconDict.imageLocalOfNetworkPathString = @"logo.png";
* iconDict.imageSourceType = SendImageOfSourceTypeOfBundle;
* iconDict.imageShowType = SendImageShowTypeOfIcon;
* iconDict.imagePutFileCorrelationID = \_autoIncCorrID++;
* iconDict.imageShowCorrelationID = \_autoIncCorrID++;
  + **第二步传递图片数据（开启了队列模式）**
* [[FORDSendImageObject shareSendImageObjectWithSync:\_syncProxy]putIconImage:iconDict];
  + **第三步发送图片数据到图片上传处理类**
* - (void)sendImageToPutFileClass
* {
* if ([self.sendImageDictArray count] > 0) {
* \_\_weak \_\_typeof(self)weakSelf = self;
* //block
* [self.putImageFileObject setPutIconResultBlock:^(SDLPutFileResponse \*response, NSMutableDictionary \*imageInfoDict) {
* \_\_strong \_\_typeof(weakSelf)strongSelf = weakSelf;
* //judge result if YES=sendRPC NO=fail
* if (response.resultCode == [SDLResult SUCCESS]) {
* //sendRPC
* [strongSelf sendIconRPC:imageInfoDict];
* }else {
* //fail
* }
* //delete finish image info
* if ([strongSelf.sendImageDictArray count] > 0) {
* [strongSelf.sendImageDictArray removeObjectAtIndex:0];
* }
* //the current block is finish, so release
* strongSelf.putImageFileObject.putIconResultBlock = nil;
* //recycle send image info
* [strongSelf sendImageToPutFileClass];
* }];
* //put icon image info
* [self.putImageFileObject putIconImageFile:[self.sendImageDictArray firstObject]];
* }
* }
  + **第四步上传图片**
* /\*\*
* \* get the name of the image through the md5 encryption and image data
* \*
* \* @param imageFileDict image data dictionary
* \*/
* - (void)sendRPCOfPutFile:(NSMutableDictionary \*)imageFileDict
* {
* //lock up: one by one
* @synchronized(self) {
* //if the value of local path is valid
* if (imageFileDict.imageLocalOfNetworkPathString && imageFileDict.imageLocalOfNetworkPathString.length > 0) {
* //image's data
* NSData \*imageData = nil;
* //get image's type (png|jpg|bmp) default is png
* SDLFileType \*fileType = [SDLFileType GRAPHIC\_PNG];
* if ([imageFileDict.imageLocalOfNetworkPathString rangeOfString:@".jpg"].location != NSNotFound || [imageFileDict.imageLocalOfNetworkPathString rangeOfString:@".jpeg"].location != NSNotFound) {
* fileType = [SDLFileType GRAPHIC\_JPEG];
* }else if ([imageFileDict.imageLocalOfNetworkPathString rangeOfString:@".bmp"].location != NSNotFound) {
* fileType = [SDLFileType GRAPHIC\_BMP];
* }
* //judge image's source and deal with itSelf style
* switch (imageFileDict.imageSourceType) {
* case SendImageOfSourceTypeOfBundle:
* {
* //get image from bundle
* UIImage \*sendImage = [UIImage imageNamed:imageFileDict.imageNameOfLocalString];
* //judge image type
* if (fileType == [SDLFileType GRAPHIC\_PNG]) {
* imageData = UIImagePNGRepresentation(sendImage);
* }else if (fileType == [SDLFileType GRAPHIC\_JPEG]) {
* imageData = UIImageJPEGRepresentation(sendImage, 0.01);
* }else if (fileType == [SDLFileType GRAPHIC\_BMP]) {
* if ([imageFileDict.imageNameOfLocalString rangeOfString:@".bmp"].location != NSNotFound) {
* NSString \*bmpBundlePath = [[NSBundle mainBundle] pathForResource:[imageFileDict.imageNameOfLocalString substringToIndex:imageFileDict.imageNameOfLocalString.length-4] ofType:@"bmp"];
* imageData = [NSData dataWithContentsOfFile:bmpBundlePath];
* }
* }
* }
* break;
* case SendImageOfSourceTypeOfSandbox:
* {
* //get image data from sandbox
* imageData = [NSData dataWithContentsOfFile:imageFileDict.imageLocalOfNetworkPathString];
* }
* break;
* case SendImageOfSourceTypeOfNetwork:
* {
* //get image data from network
* dispatch\_async(dispatch\_get\_main\_queue(), ^{
* NSURL \*imageUrl = [NSURL URLWithString:imageFileDict.imageLocalOfNetworkPathString];
* NSURLRequest\* request = [NSURLRequest requestWithURL:imageUrl cachePolicy:NSURLRequestReloadIgnoringLocalCacheData timeoutInterval:10.0];
* [NSURLConnection connectionWithRequest:request delegate:self];
* //get image data block
* \_\_weak \_\_typeof(self)weakSelf = self;
* [self setGetNetworkImageDataBlcok:^(NSData \*imageData) {
* \_\_strong \_\_typeof(weakSelf) strongSelf = weakSelf;
* //send RPC
* [strongSelf sendRPCOfPutFile:imageFileDict withFileType:fileType withImageData:imageData];
* }];
* //don't need to perform
* });
* return;
* }
* break;
* default:
* break;
* }
* //send RPC
* [self sendRPCOfPutFile:imageFileDict withFileType:fileType withImageData:imageData];
* }
* }
* }
* 注意：这里对于网络的图片数据使用NSURLConnection进行数据的获取，需要实现代理回调的方法
* #pragma mark <!-----------network image data receive start
* //image data receive start
* - (void)connection:(NSURLConnection \*)connection didReceiveResponse:(NSURLResponse \*)response
* {
* imageDataOfNetwork = nil;
* imageDataOfNetwork = [[NSMutableData alloc] init];
* [UIApplication sharedApplication].networkActivityIndicatorVisible = YES;
* }
* //image data receive
* - (void)connection:(NSURLConnection \*)connection didReceiveData:(NSData \*)data
* {
* //append data
* [imageDataOfNetwork appendData:data];
* }
* //image data download finish
* - (void)connectionDidFinishLoading:(NSURLConnection \*)connection
* {
* self.getNetworkImageDataBlcok(imageDataOfNetwork);
* [UIApplication sharedApplication].networkActivityIndicatorVisible = NO;
* }
* //image data download fail
* - (void)connection:(NSURLConnection \*)connection didFailWithError:(NSError \*)error
* {
* self.getNetworkImageDataBlcok(imageDataOfNetwork);
* NSLog(@"image data download fail");
* }
* #pragma mark network image receive end-------->
* /\*\*
* \* send RPC of SDLPutFile
* \*
* \* @param imageFileDict image data dictionary
* \* @param fileType SDLFileType
* \* @param imageData image data
* \*/
* - (void)sendRPCOfPutFile:(NSMutableDictionary \*)imageFileDict withFileType:(SDLFileType \*)fileType withImageData:(NSData \*)imageData
* {
* //get image's md5 name through image data
* imageFileDict.imageNameOfMd5String = [FORDPutImageFile getMD5WithData:imageData];
* //send image to sync
* SDLPutFile \*putFile = [[SDLPutFile alloc] init];
* putFile.fileType = fileType;
* putFile.syncFileName = imageFileDict.imageNameOfMd5String;
* putFile.bulkData = imageData;
* putFile.length = [NSNumber numberWithInteger:imageData.length];
* putFile.offset = [NSNumber numberWithInt:0];
* putFile.persistentFile = [NSNumber numberWithBool:NO];
* putFile.correlationID = [NSNumber numberWithInteger:imageFileDict.imagePutFileCorrelationID];
* //send
* [self.putImageSyncProxy sendRPC:putFile];
* }
  + **第五步发送SDLSetAppIcon控件到TDK**
* - (void)sendIconRPC:(NSMutableDictionary \*)imageInfoDict
* {
* SDLSetAppIcon \*appIcon = [SDLSetAppIcon new];
* appIcon.syncFileName = imageInfoDict.imageNameOfMd5String;
* appIcon.correlationID = [NSNumber numberWithInteger:imageInfoDict.imageShowCorrelationID];
* [self.putImageSyncProxy sendRPC:appIcon];
* }
* **3.2 SDLShow图片（内部原理与icon一致，这里只介绍如何调用方法）**
* NSMutableDictionary \*showDict = [[NSMutableDictionary alloc]init];
* showDict.imageNameOfLocalString = @"image5.png";
* showDict.imageLocalOfNetworkPathString = @"image5.png";
* showDict.imageSourceType = SendImageOfSourceTypeOfBundle;
* showDict.imageShowType = SendImageShowTypeOfShow;
* showDict.imagePutFileCorrelationID = \_autoIncCorrID++;
* showDict.imageShowCorrelationID = \_autoIncCorrID++;
* [[FORDSendImageObject shareSendImageObjectWithSync:\_syncProxy]putShowImage:showDict];

**注意：这里涉及到了三种屏幕布局类型：普通（右侧显示一张图片）、大图（整个屏幕显示一张大图片）、两张图（屏幕上显示两张图片）**

**分别为：**

SDLSetDisplayLayout \*displayLayout = [[SDLSetDisplayLayout alloc]init];

displayLayout.displayLayout = @"DOUBLE\_GRAPHIC\_WITH\_SOFTBUTTONS";

displayLayout.displayLayout = @"LARGE\_GRAPHIC\_ONLY";

[\_syncProxy sendRPC:displayLayout];

* **3.3 SDLSoftButton图片（内部原理与icon一致，这里只介绍如何调用方法）**

**参考DEMO中ViewController里面的case MORE\_SHOW\_SOFTBUTTON区域代码**

* **3.4 SDLAddCommand图片（内部原理与icon一致，这里只介绍如何调用方法）**

**参考DEMO中ViewController里面的case MORE\_SHOW\_ADDCOMMAND区域代码**

* **3.5 SDLChoice图片**

**参考DEMO中ViewController里面的case MORE\_SHOW\_PERFORMINTERACTION区域代码**

**注意：此处需要添加一个通知，为了实现Choice图片上传成功后的操作**

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(getSendChoiceImageState:) name:@"FORDSendChoiceImageNotification" object:nil];

# 版本更新

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **修改内容** | **修改人** | **修改时间** | **审核人** |
| V1.0.0 | 创建并完善文档 | 李克 | 16/03/01 |  |
| V1.0.1 | 修改网络图片的数据获取方式 | 李克 | 16/03/02 |  |
| V1.0.2 | 添加图片类实例对象释放方法，解决重新连接USB出现图片不能正常发送问题 | 李克 | 16/03/03 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |