小米穿戴第三方APP能力开放接口文档_v1.4

1.4版本更新日志

兼容Android R

1.3版本更新日志

兼容未来小米穿戴和小米健康合并项目，api无变化，直接替换sdk即可

1.2版本更新日志：

兼容jdk1.7版本，api没有变化

1.1版本更新日志：

1.修改状态订阅和状态查询相关接口返回值，详情请查看文档3.2和3.3部分

2.增加消息通知api，详情请看文档第5部分

1.查询已连接的可穿戴设备（不需要权限）

```
//先获取NodeApi对象
NodeApi api = Wearable.getNodeApi(context);
//调用getConnectedNodes方法获取已连接设备
api.getConnectedNodes().addOnSuccessListener(new OnSuccessListener<List<Node>>()
{
    @Override
    public void onSuccess(List<Node> nodes) {
        //获取当前已连接的设备,目前一次只能连接一个设备
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        //获取已连接设备失败
    }
});
```

2.权限查询与申请

2.1查询授权状态

第一次调用授权会默认授予Permission.DEVICE_MANAGER和Permission.Notify权限

```
//先获取AuthApi对象
AuthApi authApi = Wearable.getAuthApi(context);
//调用checkPermission方法获取某个授权状态
authApi.checkPermission("nodeId",Permission.DEVICE_MANAGER)
        .addOnSuccessListener(new OnSuccessListener<Boolean>() {
            @Override
            public void onSuccess(Boolean result) {
                //如果已授权，result为true，未授权result为false
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
```

```
            public void onFailure(@NonNull Exception e ) {
                //获取授权失败
            }
});
//定义一组权限
Permission[] permissions = new Permission[]
{Permission.DEVICE_MANAGER,Permission.NOTIFY};
//调用checkPermissions接口查询一组权限授权状态
authApi.checkPermissions("nodeId",permissions)
    .addOnSuccessListener(new OnSuccessListener<Boolean[]>() {
        @Override
        public void onSuccess(Boolean[] results) {
            //如果已授权，result为true，未授权result为false,按照请求顺序返回结果
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            //获取授权状态失败
        }
});
```

2.2申请权限

```
//先获取AuthApi对象
AuthApi authApi = Wearable.getAuthApi(context);
//发起授权申请
authApi.requestPermission("nodeId",Permission.DEVICE_MANAGER,Permission.NOTIFY)
        .addOnSuccessListener(new OnSuccessListener<Permission[]>() {
            @Override
            public void onSuccess(Permission[] permissions) {
               //申请权限成功，返回授权成功的权限
            }
        }).addOnFailureListener(new OnFailureListener() {
        @Override
            public void onFailure(@NonNull Exception e) {
                //申请权限失败
            }
        });
```

3.设备管理与状态订阅

3.1支持查询与订阅的设备状态

设备管理与状态订阅	查询结果	订阅事件触发条件	订阅结果
连接状态	1.连接 2.未连接	1.手机与设备连接成功 2.手机与设备断开连接	1.连接成功 2.连接断开 3.连接失败 4.设备被删除
电量状态	电量值（0~100，比如98）	N/A	N/A
充电状态	1.正在充电 2.非充电状态	1.给设备充电 2.充满电 3.停止充电	1.开始充电 2.充电完成 3.停止充电
佩戴状态	1.佩戴中 2.未佩戴	1.手表戴在手腕上 2.摘下手表	1.佩戴 2.未佩戴
睡眠状态	1.睡眠中 2.清醒	1.佩戴手表入睡 2.从入睡状态清醒过来	1.入睡 2.出睡

3.2状态查询（需要申请Permission.DEVICE_MANAGER权限）

```
//先获取NodeApi对象
NodeApi api = Wearable.getNodeApi(context);
//调用query方法，查询不同状态，nodeId通过查询已连接设备得到的设备id
//目前支持查询（连接状态，电量状态，充电状态，佩戴状态，睡眠状态）
api.query("nodeId", DataItem.ITEM_CONNECTION)
     .addOnSuccessListener(new OnSuccessListener<DataQueryResult>() {
         @Override
         public void onSuccess(DataQueryResult result) {
             //查询成功
            boolean connectionStatus = result.isConnected;//DataQueryResult定义了
各种状态的状态值，和DataItem一一对应
         }
     }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            //查询失败
        }
    });
//查询对应的穿戴设备端应用是否安装
api.isWearAppInstalled("nodeId")
    .addOnSuccessListener(new OnSuccessListener<Boolean>() {
        @Override
        public void onSuccess(Boolean result) {
            //查询成功，应用已安装返回true，未安装返回false
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull  Exception e) {
            //查询失败
        }
});
//打开设备端应用
//uri由各个应用自定义，用于打开手表端app的指定页面
api.launchWearApp("nodeId","uri")
```

```
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void var1) {
            //打开穿戴设备端应用成功
        }
    }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull @NotNull Exception var1) {
        //打开穿戴设备端应用失败
    }
});
```

3.3状态订阅（需要申请Permission.DEVICE_MANAGER权限）

```
//先获取NodeApi对象
NodeApi api = Wearable.getNodeApi(context);
//创建监听器
OnDataChangedListener onDataChangedListener = new OnDataChangedListener() {
    @Override
    public void onDataChanged(@NonNull String nodeId, @NonNull DataItem dataItem,
@NonNull DataSubscribeResult data) {
        //收到订阅状态变更通知
        //不同的DataItem对应DataSubscribeResult中不同的status，一一对应
        if(dataItem.getType() == DataItem.ITEM_CONNECTION.getType()){
            int connectionStatus = data.getConnectedStatus();
            if(connectionStatus ==
DataSubscribeResult.RESULT_CONNECTION_CONNECTED){
                //设备连接状态变更为已连接状态
            }
        }

    }
};
//调用subscribe方法，订阅不同状态，nodeId是通过查询已连接设备得到的设备id
//目前支持订阅（连接状态变化，充电状态变化，佩戴状态变化，睡眠状态变化）
api.subscribe("nodeId", DataItem.ITEM_CONNECTION, onDataChangedListener)
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void var1) {
            //添加订阅成功
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull @NotNull Exception var1) {
            //添加订阅失败
        }
});
//调用unsubscribe方法取消监听
api.unsubscribe("nodeId", DataItem.ITEM_CONNECTION)
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void var1) {
            //删除订阅成功
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull @NotNull Exception var1) {
```

```
            //删除订阅失败
        }
});
```

4.应用间消息通信（需要申请Permission.DEVICE_MANAGER权限）

```
//模拟数据，应用可以自定义发送数据
byte[] messageBytes = new byte[1024];
//获取MessageApi对象
MessageApi messageApi = Wearable.getMessageApi(context);
//调用sendMessage方法用来发送数据给穿戴设备端应用
messageApi.sendMessage("nodeId",messageBytes)
     .addOnSuccessListener(new OnSuccessListener<Integer>() {
         @Override
         public void onSuccess(Integer result) {
            //发送数据成功
         }
     })
     .addOnFailureListener(new OnFailureListener() {
         @Override
         public void onFailure(@NonNull Exception e) {
            //发送数据失败
         }
     });

//传见监听器用于监听穿戴设备端发送来的消息
OnMessageReceivedListener onMessageReceivedListener = new
OnMessageReceivedListener() {
    @Override
    public void onMessageReceived(@NotNull String nodeId,  @NotNull byte[]
message) {
        //收到手表端应用发来的消息
    }
};
//监听穿戴设备端应用发来的消息
messageApi.addListener("nodeId", onMessageReceivedListener)
    .addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void var1) {
            //添加消息监听成功
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull @NotNull Exception var1) {
            //添加消息监听失败
        }
});
//取消监听
messageApi.removeListener("nodeId",onMessageReceivedListener).addOnSuccessListene
r(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void var1) {
        //取消消息监听成功
    }
```

```
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull @NotNull Exception var1) {
        //取消消息监听失败
    }
});
```

5.消息通知（需要Permission.NOTIFY权限）

```
//获取NotifyApi
NotifyApi notifyApi = Wearable.getNotifyApi(context);
//发送消息通知，需要提前判断是否有Permission.NOTIFY权限
notifyApi.sendNotify("nodeId","title","message")
        .addOnSuccessListener(new OnSuccessListener<Status>() {
            @Override
            public void onSuccess(Status status) {
                if(status.isSuccess()){
                    //发送通知成功,手表上会看到消息通知内容，表端应用无感知
                }
            }
        }).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull @NotNull Exception var1) {
       //发送通知失败
    }
});
```

6.管理服务连接状态（不需要权限）

```
//获取ServiceApi
ServiceApi serviceApi = Wearable.getServiceApi(context);
//第三方应用与小米穿戴App连接状态可以通过这个监听器来监听
OnServiceConnectionListener onServiceConnectionListener = new
OnServiceConnectionListener() {
    @Override
    public void onServiceConnected() {
        //服务连接成功
    }

    @Override
    public void onServiceDisconnected() {
        //服务断开
    }
};
//注册监听器
serviceApi.registerServiceConnectionListener(onServiceConnectionListener);
//取消监听器
serviceApi.unregisterServiceConnectionListener(onServiceConnectionListener);
```