(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106933636 B
(45)授权公告日 2020.08.18

(21)申请号 201710157645.X

(22)申请日 2017.03.16

(65)同一申请的已公布的文献号
申请公布号 CN 106933636 A

(43)申请公布日 2017.07.07

(73)专利权人 北京奇虎科技有限公司
地址 100088 北京市西城区新街口外大街28号D座112室(德胜园区)

(72)发明人 张炅轩

(74)专利代理机构 北京律诚同业知识产权代理有限公司 11006
代理人 王玉双

(51)Int.Cl.
G06F 9/445(2018.01)

(56)对比文件
CN 101978390 A,2011.02.16
CN 105354081 A,2016.02.24
CN 103902390 A,2014.07.02
CN 106484461 A,2017.03.08

审查员 张莹

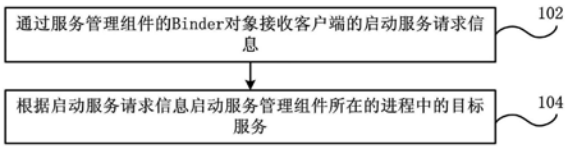
权利要求书4页 说明书23页 附图4页

(54)发明名称

启动插件服务的方法、装置和终端设备

(57)摘要

本发明实施例公开了一种启动插件服务的方法、装置和终端设备。其中方法包括:通过服务管理组件的Binder对象接收客户端的启动服务请求信息;根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务;其中,所述服务管理组件管理其所在的进程中的所有服务,所述服务管理组件与所述客户端运行在不同的进程中。本发明实施例通过为服务端每个进程设置服务管理组件,管理其所在进程中的所有服务,当目标服务与客户端运行在不同的进程中时,可以实现对目标服务的启动,因此在启动插件提供的服务时,既不需要在主程序的AndroidManifest.xml文件中预设服务坑位,也不需要在服务升级时对主程序进行升级的额外处理。



CN 106933636 B

1.一种启动插件服务的方法,其特征在于,包括:

通过服务管理组件的Binder对象接收客户端的启动服务请求信息;

根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务;

其中,所述服务管理组件管理其所在的进程中的所有服务,所述服务管理组件与所述客户端运行在不同的进程中;

所述根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务,包括:

根据所述启动服务请求信息,检测在所述服务管理组件中是否保存有目标服务安装信息;

若在所述服务管理组件中未保存有目标服务安装信息,创建目标服务安装信息,包括:

根据所述启动服务请求信息,检测在所述服务管理组件中是否保存有提供目标服务的插件;

若在所述服务管理组件中保存有提供目标服务的插件,从所述插件中获取目标服务的ServiceInfo对象;

若在所述服务管理组件中未保存有提供目标服务的插件,从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象;

基于所述启动服务请求信息和所述目标服务的ServiceInfo对象,创建目标服务安装信息。

2.根据权利要求1所述的方法,其特征在于,所述通过服务管理组件的Binder对象接收客户端的启动服务请求信息,包括:

根据所述启动服务请求信息调用所述服务管理组件的AIDL接口对象的startService函数。

3.根据权利要求2所述的方法,其特征在于;

所述根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务,包括:

通过所述服务管理组件中继承所述AIDL接口的内部类对象访问所述服务管理组件核心类的startService函数;其中所述服务管理组件的核心类实现所述AIDL接口。

4.根据权利要求1至3任意一项所述的方法,其特征在于,所述根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务,包括:根据所述启动服务请求信息,检测在所述服务管理组件中是否保存有目标服务安装信息;

若在所述服务管理组件中保存有目标服务安装信息,获取所述目标服务安装信息。

5.根据权利要求1所述的方法,其特征在于,所述创建目标服务安装信息之后,还包括:

在所述服务管理组件中保存所述目标服务安装信息。

6.根据权利要求1任意一项所述的方法,其特征在于,所述创建目标服务安装信息之后,还包括:

基于所述目标服务安装信息安装目标服务。

7.根据权利要求6所述的方法,其特征在于,所述获取所述目标服务安装信息之后,还包括:

根据所述目标服务安装信息,检测在所述服务管理组件中是否保存有目标服务;

若在所述服务管理组件中保存有目标服务，调用所述目标服务的onStartCommand函数；

若在所述服务管理组件中未保存有目标服务，基于所述目标服务安装信息安装目标服务。

8.根据权利要求6或7所述的方法，其特征在于，所述基于所述目标服务安装信息安装目标服务，包括：

获取提供目标服务的插件的Context对象；

基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务；

初始化所述目标服务。

9.根据权利要求8所述的方法，其特征在于，所述基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务，包括：

从提供目标服务的插件的Context对象获取所述插件的类加载器；

从所述目标服务安装信息获取目标服务的全类名；

基于所述目标服务的全类名，通过反射机制利用所述插件的类加载器获取所述目标服务的系统类对象；

通过反射机制调用所述目标服务的系统类的newInstance来调用构造函数创建目标服务。

10.根据权利要求8所述的方法，其特征在于，所述初始化所述目标服务，包括：

通过反射机制调用所述目标服务的attachBaseContext的函数；

调用所述目标服务的onCreate函数。

11.根据权利要求10所述的方法，其特征在于，所述基于所述目标服务安装信息安装目标服务之后，还包括：

在所述服务管理组件中保存所述目标服务。

12.根据权利要求11所述的方法，其特征在于，所述基于所述目标服务安装信息安装目标服务之后，还包括：

调用所述目标服务的onStartCommand函数。

13.一种启动插件服务的装置，其特征在于，包括：

接收单元，用于通过服务管理组件的Binder对象接收客户端的启动服务请求信息；

执行单元，用于根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务；

其中，所述服务管理组件管理其所在的进程中的所有服务，所述服务管理组件与所述客户端运行在不同的进程中；

所述执行单元，包括：

第一检测模块，用于根据所述启动服务请求信息，检测在所述服务管理组件中是否保存有目标服务安装信息；

创建模块，用于根据所述第一检测模块的检测结果，响应于在所述服务管理组件中未保存有目标服务安装信息，创建目标服务安装信息；

所述创建模块，具体用于：

根据所述启动服务请求信息，检测在所述服务管理组件中是否保存有提供目标服务的

插件；

若在所述服务管理组件中保存有提供目标服务的插件，从所述插件中获取目标服务的ServiceInfo对象；

若在所述服务管理组件中未保存有提供目标服务的插件，从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象；

基于所述启动服务请求信息和所述目标服务的ServiceInfo对象，创建目标服务安装信息。

14.根据权利要求13所述的装置，其特征在于，所述接收单元，具体用于根据所述启动服务请求信息调用所述服务管理组件的AIDL接口对象的startService函数。

15.根据权利要求14所述的装置，其特征在于，所述执行单元，具体用于通过所述服务管理组件中继承所述AIDL接口的内部类对象访问所述服务管理组件核心类的startService函数；其中，所述服务管理组件的核心类实现所述AIDL接口。

16.根据权利要求15所述的装置，其特征在于，所述执行单元，包括：

获取模块，用于根据所述第一检测模块的检测结果，响应于在所述服务管理组件中保存有目标服务安装信息，获取所述目标服务安装信息。

17.根据权利要求15或16所述的装置，其特征在于，所述执行单元，还包括：

第一存储模块，用于在所述服务管理组件中保存所述目标服务安装信息。

18.根据权利要求17所述的装置，其特征在于，所述执行单元，还包括：

安装模块，用于在所述创建模块创建目标服务安装信息之后，基于所述目标服务安装信息安装目标服务。

19.根据权利要求18所述的装置，其特征在于，所述执行单元，还包括：

第二检测模块，用于根据所述目标服务安装信息，检测在所述服务管理组件中是否保存有目标服务；

执行模块，用于根据所述第二检测模块的检测结果，响应于在所述服务管理组件中保存有目标服务，调用所述目标服务的onStartCommand函数；

所述安装模块，还用于根据所述第二检测模块的检测结果，响应于在所述服务管理组件中未保存有目标服务，基于所述目标服务安装信息安装目标服务。

20.根据权利要求19所述的装置，其特征在于，所述安装模块，具体用于：

获取提供目标服务的插件的Context对象；

基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务；

初始化所述目标服务。

21.根据权利要求20所述的装置，其特征在于，所述安装模块基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务，具体用于：

从提供目标服务的插件的Context对象获取所述插件的类加载器；

从所述目标服务安装信息获取目标服务的全类名；

基于所述目标服务的全类名，通过反射机制利用所述插件的类加载器获取所述目标服务的系统类对象；

通过反射机制调用所述目标服务的系统类的newInstance来调用构造函数创建目标服务。

22.根据权利要求20所述的装置，其特征在于，所述安装模块初始化所述目标服务，具体用于：

通过反射机制调用所述目标服务的attachBaseContext的函数；

调用所述目标服务的onCreate函数。

23.根据权利要求22所述的装置，其特征在于，所述执行单元，还包括：

第二存储模块，用于在所述服务管理组件中保存所述目标服务。

24.根据权利要求22所述的装置，其特征在于，所述执行模块，还用于在所述安装模块基于所述目标服务安装信息安装目标服务之后，调用所述目标服务的onStartCommand函数。

25.一种终端设备，其特征在于，包括：处理器和存储器；其中，

所述存储器用于存储权利要求1至12任意一项所述的启动插件服务的方法；

所述处理器用于执行所述启动插件服务的方法。

启动插件服务的方法、装置和终端设备

技术领域

[0001] 本发明属于计算机技术领域，特别是涉及一种启动插件服务的方法、装置和终端设备。

背景技术

[0002] 插件是一种遵循一定规范的应用程序接口编写出来的程序，目前Android系统使用插件提供的服务的方法包括：启动服务(Start Service)、停止服务(Stop Service)、绑定服务(Bind Service)和解除绑定服务(Unbind Service)。在Android系统中使用插件提供的服务主要采用两种方案来实现：一种是通过在主程序中编写一个服务中介，利用这个服务中介通过反射机制来调用插件，其具体实现方法为：首先在主程序的AndroidManifest.xml文件中声明此服务中介，然后在主程序中编写此服务中介，此服务中介的所有函数都是通过反射机制调用插件内部的函数来实现；另一种是通过在主程序中预设一些服务坑位，利用这些服务坑位来加载插件提供的服务，其具体实现方法为：首先预先在主程序的AndroidManifest.xml文件中声明多个虚拟的服务注册信息，即服务坑位，然后为插件提供的服务分配相适配的服务坑位，将此服务加载到服务坑位所对应的进程空间中运行。

[0003] 上述现有的两种方案虽然都能够实现对插件提供的服务的使用，然而，在具体实现过程中，我们发现上述现有的两种方案都存在着一定的局限性：第一种方案，要升级插件提供的服务必须先升级主程序，只有在升级了主程序后才能够使用升级后的服务，需要进行主程序升级的额外处理；第二种方案，由于预设的服务坑位数量有限，而服务常驻于后台，生命周期较长，这样很快预设的服务坑位就会被用完，而如果预设坑位过多又会导致启动速度受到影响，当第三方或者自己通过Binder获取主程序应用的信息时，有可能会因信息过多而出现TransactionTooLarge的异常。

发明内容

[0004] 本发明实施例要解决的一个技术问题是：提供一种启动插件服务的方法、装置和终端设备，可以在不需要设置坑位且不需要进行额外处理的情况下，启动插件提供的服务。

[0005] 为解决上述技术问题，根据本发明实施例的一个方面，提供一种启动插件服务的方法，包括：

[0006] 通过服务管理组件的Binder对象接收客户端的启动服务请求信息；

[0007] 根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务；

[0008] 其中，所述服务管理组件管理其所在的进程中的所有服务，所述服务管理组件与所述客户端运行在不同的进程中。

[0009] 在基于本发明上述方法的另一个实施例中，所述通过服务管理组件的Binder对象接收客户端的启动服务请求信息，包括：

[0010] 根据所述启动服务请求信息调用所述服务管理组件的AIDL接口对象的

startService函数。

[0011] 在基于本发明上述方法的另一个实施例中,所述根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务,包括:

[0012] 通过所述服务管理组件中继承所述AIDL接口的内部类对象访问所述服务管理组件核心类的startService函数;其中所述服务管理组件的核心类实现所述AIDL接口。

[0013] 在基于本发明上述方法的另一个实施例中,所述根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务,包括:

[0014] 根据所述启动服务请求信息,检测在所述服务管理组件中是否保存有目标服务安装信息;

[0015] 若在所述服务管理组件中保存有目标服务安装信息,获取所述目标服务安装信息;

[0016] 若在所述服务管理组件中未保存有目标服务安装信息,创建目标服务安装信息。

[0017] 在基于本发明上述方法的另一个实施例中,所述创建目标服务安装信息,包括:

[0018] 根据所述启动服务请求信息,检测在所述服务管理组件中是否保存有提供目标服务的插件;

[0019] 若在所述服务管理组件中保存有提供目标服务的插件,从所述插件中获取目标服务的ServiceInfo对象;

[0020] 若在所述服务管理组件中未保存有提供目标服务的插件,从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象;

[0021] 基于所述启动服务请求信息和所述目标服务的ServiceInfo对象,创建目标服务安装信息。

[0022] 在基于本发明上述方法的另一个实施例中,所述创建目标服务安装信息之后,还包括:

[0023] 在所述服务管理组件中保存所述目标服务安装信息。

[0024] 在基于本发明上述方法的另一个实施例中,所述创建目标服务安装信息之后,还包括:

[0025] 基于所述目标服务安装信息安装目标服务。

[0026] 在基于本发明上述方法的另一个实施例中,所述获取所述目标服务安装信息之后,还包括:

[0027] 根据所述目标服务安装信息,检测在所述服务管理组件中是否保存有目标服务;

[0028] 若在所述服务管理组件中保存有目标服务,调用所述目标服务的onStartCommand函数;

[0029] 若在所述服务管理组件中未保存有目标服务,基于所述目标服务安装信息安装目标服务。

[0030] 在基于本发明上述方法的另一个实施例中,所述基于所述目标服务安装信息安装目标服务,包括:

[0031] 获取提供目标服务的插件的Context对象;

[0032] 基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务;

[0033] 初始化所述目标服务。

[0034] 在基于本发明上述方法的另一个实施例中,所述基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务,包括:

[0035] 从提供目标服务的插件的Context对象获取所述插件的类加载器;

[0036] 从所述目标服务安装信息获取目标服务的全类名;

[0037] 基于所述目标服务的全类名,通过反射机制利用所述插件的类加载器获取所述目标服务的系统类对象;

[0038] 通过反射机制调用所述目标服务的系统类的newInstance来调用构造函数创建目标服务。

[0039] 在基于本发明上述方法的另一个实施例中,所述初始化所述目标服务,包括:

[0040] 通过反射机制调用所述目标服务的attachBaseContext的函数;

[0041] 调用所述目标服务的onCreate函数。

[0042] 在基于本发明上述方法的另一个实施例中,所述基于所述目标服务安装信息安装目标服务之后,还包括:

[0043] 在所述服务管理组件中保存所述目标服务。

[0044] 在基于本发明上述方法的另一个实施例中,所述基于所述目标服务安装信息安装目标服务之后,还包括:

[0045] 调用所述目标服务的onStartCommand函数。

[0046] 根据本发明实施例的另一个方面,提供一种启动插件服务的装置,包括:

[0047] 接收单元,用于通过服务管理组件的Binder对象接收客户端的启动服务请求信息;

[0048] 执行单元,用于根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务;

[0049] 其中,所述服务管理组件管理其所在的进程中的所有服务,所述服务管理组件与所述客户端运行在不同的进程中。

[0050] 在基于本发明上述装置的另一个实施例中,所述接收单元,具体用于根据所述启动服务请求信息调用所述服务管理组件的AIDL接口对象的startService函数。

[0051] 在基于本发明上述装置的另一个实施例中,所述执行单元,具体用于通过所述服务管理组件中继承所述AIDL接口的内部类对象访问所述服务管理组件核心类的startService函数;其中,所述服务管理组件的核心类实现所述AIDL接口。

[0052] 在基于本发明上述装置的另一个实施例中,所述执行单元,包括:

[0053] 第一检测模块,用于根据所述启动服务请求信息,检测在所述服务管理组件中是否保存有目标服务安装信息;

[0054] 获取模块,用于根据所述第一检测模块的检测结果,响应于在所述服务管理组件中保存有目标服务安装信息,获取所述目标服务安装信息;

[0055] 创建模块,用于根据所述第一检测模块的检测结果,响应于在所述服务管理组件中未保存有目标服务安装信息,创建目标服务安装信息。

[0056] 在基于本发明上述装置的另一个实施例中,所述创建模块,具体用于:

[0057] 根据所述启动服务请求信息,检测在所述服务管理组件中是否保存有提供目标服

务的插件；

[0058] 若在所述服务管理组件中保存有提供目标服务的插件，从所述插件中获取目标服务的ServiceInfo对象；

[0059] 若在所述服务管理组件中未保存有提供目标服务的插件，从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象；

[0060] 基于所述启动服务请求信息和所述目标服务的ServiceInfo对象，创建目标服务安装信息。

[0061] 在基于本发明上述装置的另一个实施例中，所述执行单元，还包括：

[0062] 第一存储模块，用于在所述服务管理组件中保存所述目标服务安装信息。

[0063] 在基于本发明上述装置的另一个实施例中，所述执行单元，还包括：

[0064] 安装模块，用于在所述创建模块创建目标服务安装信息之后，基于所述目标服务安装信息安装目标服务。

[0065] 在基于本发明上述装置的另一个实施例中，所述执行单元，还包括：

[0066] 第二检测模块，用于根据所述目标服务安装信息，检测在所述服务管理组件中是否保存有目标服务；

[0067] 执行模块，用于根据所述第二检测模块的检测结果，响应于在所述服务管理组件中保存有目标服务，调用所述目标服务的onStartCommand函数；

[0068] 所述安装模块，还用于根据所述第二检测模块的检测结果，响应于在所述服务管理组件中未保存有目标服务，基于所述目标服务安装信息安装目标服务。

[0069] 在基于本发明上述装置的另一个实施例中，所述安装模块，具体用于：

[0070] 获取提供目标服务的插件的Context对象；

[0071] 基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务；

[0072] 初始化所述目标服务。

[0073] 在基于本发明上述装置的另一个实施例中，所述安装模块基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务，具体用于：

[0074] 从提供目标服务的插件的Context对象获取所述插件的类加载器；

[0075] 从所述目标服务安装信息获取目标服务的全类名；

[0076] 基于所述目标服务的全类名，通过反射机制利用所述插件的类加载器获取所述目标服务的系统类对象；

[0077] 通过反射机制调用所述目标服务的系统类的newInstance来调用构造函数创建目标服务。

[0078] 在基于本发明上述装置的另一个实施例中，所述安装模块初始化所述目标服务，具体用于：

[0079] 通过反射机制调用所述目标服务的attachBaseContext的函数；

[0080] 调用所述目标服务的onCreate函数。

[0081] 在基于本发明上述装置的另一个实施例中，所述执行单元，还包括：

[0082] 第二存储模块，用于在所述服务管理组件中保存所述目标服务。

[0083] 在基于本发明上述装置的另一个实施例中，所述执行模块，还用于在所述安装模

块基于所述目标服务安装信息安装目标服务之后，调用所述目标服务的onStartCommand函数。

[0084] 根据本发明实施例的再一个方面，提供的一种终端设备，包括：处理器和存储器；其中，

[0085] 所述存储器用于存储上述任一实施例所述的启动插件服务的方法的程序；

[0086] 所述处理器用于执行所述存储器中存储的所述启动插件服务的方法的程序。

[0087] 基于本发明上述实施例提供的启动插件服务的方法、装置和终端设备，通过为服务端每个进程设置服务管理组件，管理其所在进程中的所有服务，当通过服务管理组件的Binder对象接收到客户端的启动服务请求信息后，根据此启动服务请求信息启动服务管理组件所在的进程中的目标服务，当目标服务与客户端运行在不同的进程中时，可以实现对目标服务的启动，由于本发明实施例是通过设置于每一进程中的服务管理组件对进程中的所有服务进行管理，因此其既不需要在主程序的AndroidManifest.xml文件中预设服务坑位，也不需要在服务升级时对主程序进行升级的额外处理，即可实现对插件提供的服务的启动。

附图说明

[0088] 构成说明书的一部分的附图描述了本发明的实施例，并且连同描述一起用于解释本发明的原理。

[0089] 参照附图，根据下面的详细描述，可以更加清楚地理解本发明，其中：

[0090] 图1是本发明实施例启动插件服务的方法的一个实施例的流程图。

[0091] 图2是本发明实施例启动插件服务的方法的另一个实施例的流程图。

[0092] 图3是本发明实施例启动插件服务的方法的又一个实施例的流程图。

[0093] 图4是本发明实施例启动插件服务的方法的再一个实施例的流程图。

[0094] 图5是本发明实施例启动插件服务的装置的一个实施例的结构图。

[0095] 图6是本发明实施例启动插件服务的装置的另一个实施例的结构图。

[0096] 图7是本发明实施例启动插件服务的装置的又一个实施例的结构图。

[0097] 图8是与本发明实施例提供的终端设备相关的手机的部分结构的框图。

具体实施方式

[0098] 现在将参照附图来详细描述本发明的各种示例性实施例。应注意到：除非另外具体说明，否则在这些实施例中阐述的部件的相对布置、数字表达式和数值不限制本发明的范围。

[0099] 同时，应当明白，为了便于描述，附图中所示出的各个部分的尺寸并不是按照实际的比例关系绘制的。

[0100] 以下对至少一个示例性实施例的描述实际上仅仅是说明性的，决不作为对本发明及其应用或使用的任何限制。

[0101] 对于相关领域普通技术人员已知的技术、方法和设备可能不作详细讨论，但在适当情况下，所述技术、方法和设备应当被视为说明书的一部分。

[0102] 应注意到：相似的标号和字母在下面的附图中表示类似项，因此，一旦某一项在一

个附图中被定义，则在随后的附图中不需要对其进行进一步讨论。

[0103] 图1是本发明实施例启动插件服务的方法的一个实施例的流程图。如图1所示，该实施例的方法包括：

[0104] S102，通过服务管理组件的Binder对象接收客户端的启动服务请求信息。

[0105] S104，根据启动服务请求信息启动服务管理组件所在的进程中的目标服务。

[0106] 其中，目标服务为插件提供的服务，例如为Android系统的一个后台服务，服务所在的进程通常会在AndroidManifest.xml文件中的<Service>标签中的android:process中进行指定。客户端为调用服务的一方，实施服务的一方为服务端，例如清理插件要调用备份插件中的备份服务来做清理前的备份，则清理插件的一方为客户端，提供备份服务的一方为服务端，客户端与服务端运行在不同的进程中。在服务端的每一个进程中都设有一个服务管理组件，通过服务管理组件来管理其所在的进程中的所有服务。

[0107] 在一个具体实施例中，启动服务请求信息可以包括：Intent对象、Context对象和Messenger对象。其中，Intent是Android系统中对执行一次操作的动作、动作涉及数据和附加数据的描述，Android系统根据此Intent的描述，负责找到对应的组件，将Intent传递给调用的组件，完成组件的调用。Context是描述一个应用程序环境的信息，通过Context可以获取应用程序的资源和类，也包括一些应用级别操作。Messenger是由客户端传递来的用来标记客户端的唯一标识，也可用做与客户端的发送消息服务。

[0108] 具体实现中，操作S102可以根据启动服务请求信息调用服务管理组件的AIDL接口对象的startService函数，操作S104可以通过服务管理组件中继承AIDL接口的内部类对象访问服务管理组件核心类的startService函数来启动目标服务，其中服务管理组件的核心类实现AIDL接口。

[0109] 在一个具体实施例中，IPluginServiceServer接口对象为服务管理组件的AIDL接口对象，IPluginServiceServer接口是服务管理组件的核心接口，PluginServiceServer类是服务管理组件的核心类，是IPluginServiceServer接口的具体实现，服务管理组件通过核心接口与核心类共同实现服务端的服务调度、提供等工作。由于IPluginServiceServer接口对象是Binder对象的Proxy，因此需要在目标服务所在的进程中实现IPluignServiceServer.Stub类，IPluignServiceServer.Stub类是PluginServiceServer类的内部类，继承了IPluginServiceServer接口，实现其基础功能，并最终跳转到PluginServiceServer类中进行相应的处理；其中IPluginServiceServer接口声明了startService函数，内部类IPluignServiceServer.Stub定义了startService函数，PluginServiceServer类实现了startService函数。

[0110] 其中，在服务管理组件中实现IPluignServiceServer.Stub类，并对其初始化的操作包括：创建或获取PluginServiceServer对象，创建或获取PluginServiceServer.Stub对象，及调用PluginServiceServer.Stub中的任一方法，跳转到PluginServiceServer。首先，检测在服务管理组件中是否保存有PluginServiceServer对象；若在服务管理组件中保存有PluginServiceServer对象，获取PluginServiceServer对象；否则，若在服务管理组件中未保存有PluginServiceServer对象，创建PluginServiceServer对象；然后，检测在服务管理组件中是否保存有PluginServiceServer.Stub对象；若在服务管理组件中保存有PluginServiceServer.Stub对象，获取PluginServiceServer.Stub对象；否则，若在服务管

理组件中未保存有PluginServiceServer.Stub对象，创建PluginServiceServer.Stub对象；最后，调用PluginServiceServer.Stub中的任一方法，跳转到PluginServiceServer。

[0111] 具体地，创建PluginServiceServer对象可以通过调用IPluginHost接口或IPluginClient接口的fetchServiceServer函数获取IPluginServiceServer接口对象来实现。其具体代码的示例如下：

```
public IPluginServiceServer fetchServiceServer() throws RemoteException {
    if (mServiceMgr == null) {
        mServiceMgr = new PluginServiceServer(mContext);
    }
    return mServiceMgr.getService();
}
```

[0112]

[0113] 其中IPluginHost接口为常驻进程的接口，而IPluginClient接口为除常驻进程外其它进程的接口，mServiceMgr对象的类为PluginServiceServer，mServiceMgr对象通过调用getService函数获取IPluginServiceServer.Stub类。

[0114] 创建PluginServiceServer.Stub对象可以通过调用getService函数来实现。其具体代码的示例如下：

```
public IPluginServiceServer getService() {
    if (mStub == null) {
        mStub = new PluginServiceServer.Stub();
    }
    return mStub;
}
```

IPluginServiceServer.Stub 类的实现的具体代码的示例如下：

```
class Stub extends IPluginServiceServer.Stub {
```

[0115]

```
    @Override
    public ComponentName startService(Intent intent, Messenger client)
throws RemoteException {
        synchronized (LOCKER) {
            return PluginServiceServer.this.startServiceLocked(intent,
client);
        }
    }
```

```
        @Override
        public int stopService(Intent intent, Messenger client) throws
    RemoteException {
            synchronized (LOCKER) {
                return PluginServiceServer.this.stopServiceLocked(intent);
            }
        }

        @Override
        public int bindService(Intent intent, IServiceConnection conn, int flags,
    Messenger client) throws RemoteException {
            synchronized (LOCKER) {
[0116]          return PluginServiceServer.this.bindServiceLocked(intent,
    conn, flags, client);
            }
        }

        @Override
        public boolean unbindService(IServiceConnection conn) throws
    RemoteException {
            synchronized (LOCKER) {
                return PluginServiceServer.this.unbindServiceLocked(conn);
            }
        }
    }
```

[0117] 在一个具体实施例中，startService函数的具体代码的示例如下：

[0118]
```
    ComponentName startServiceLocked(Intent intent, Messenger client) {
        intent = cloneIntentLocked(intent);
```

```
            ServiceRecord sr = retrieveServiceLocked(intent);
            if (sr == null) {
                return null;
            }

            if (!installServiceIfNeededLocked(sr)) {
                return null;
            }

            sr.service.onStartCommand(intent, 0, 0);
            sr.startRequested = true;

            ComponentName cn = intent.getComponent();
            return cn;
        }
```

[0119]

[0120] 其中，在该函数中首先复制了一份Intent对象，如果客户端和目标服务在同一个进程中，一旦目标服务修改了Intent对象，客户端也会对应被修改，这样会出现不可预知的问题，因此需要在函数中首先复制一份Intent对象来做各种操作，以防止出现上面的问题。复制Intent对象的具体代码的示例如下：

[0121] `private Intent cloneIntentLocked(Intent intent){`

[0122] `return new Intent(intent);`

[0123] `}`

[0124] 然后该函数基于Intent对象通过retrieveServiceLocked函数来获得ServiceRecord对象，其中ServiceRecord对象记录了目标服务的安装信息，其中在获得ServiceRecord对象的过程中可能会涉及插件信息的获取。之后若目标服务未安装，则通过installServiceIfNeededLocked函数尝试安装目标服务。最后调用目标服务的onStartCommand函数，以及直接返回目标服务的全类名ComponentName，表示目标服务已经成功启动。

[0125] 基于本发明实施例提供的启动插件服务的方法，通过为服务端每个进程设置服务管理组件，管理其所在进程中的所有服务，当通过服务管理组件的Binder对象接收到客户端的启动服务请求信息后，根据此启动服务请求信息启动服务管理组件所在的进程中的目标服务，当目标服务与客户端运行在不同的进程中时，可以实现对目标服务的启动，由于本发明实施例是通过设置于每一进程中的服务管理组件对进程中的所有服务进行管理，因此其既不需要在主程序的AndroidManifest.xml文件中预设服务坑位，也不需要在服务升级

时对主程序进行升级的额外处理,即可实现对插件提供的服务的启动。

[0126] 图2是本发明实施例启动插件服务的方法的另一个实施例的流程图。如图2所示,该实施例的方法包括:

[0127] S202,通过服务管理组件的Binder对象接收客户端的启动服务请求信息。

[0128] S204,根据启动服务请求信息,检测在服务管理组件中是否保存有目标服务安装信息。

[0129] 若在服务管理组件中保存有目标服务安装信息,执行操作S206;否者,若在服务管理组件中未保存有目标服务安装信息,执行操作S208。

[0130] S206,获取目标服务安装信息。

[0131] S208,创建目标服务安装信息。

[0132] 具体实现中,操作S208可以首先根据启动服务请求信息,检测在服务管理组件中是否保存有提供目标服务的插件,若在服务管理组件中保存有提供目标服务的插件,则从插件中获取目标服务的ServiceInfo对象,否则,若在服务管理组件中未保存有提供目标服务的插件,则从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象,最后基于启动服务请求信息和所获得的目标服务的ServiceInfo对象,创建目标服务安装信息。

[0133] 进一步,在操作S208之后,该实施例的启动插件服务的方法还包括:操作S210,基于目标服务安装信息安装目标服务。

[0134] 具体实现中,在操作S208之后,还可以包括:在服务管理组件中保存目标服务安装信息的操作,以在下一次启动目标服务时可以直接从服务管理组件中获取目标服务安装信息。

[0135] 在一个具体实施例中,操作S204至操作S208,首先是以目标服务的全类名ComponentName和启动服务请求信息中的Intent对象,在服务管理组件的ServiceRecord缓存表中寻找与其相匹配的ServiceRecord对象,其中ServiceRecord缓存表中记录有服务管理组件所在的进程中所有已安装的服务的安装信息,ServiceRecord对象为目标服务安装信息,若在服务管理组件的ServiceRecord缓存表中找到ServiceRecord对象,获取ServicedRecord对象,表示目标服务已经安装,若在服务管理组件的ServiceRecord缓存表中未找到ServiceRecord对象,表示目标服务尚未安装,创建ServicedRecord对象;然后从目标服务的全类名ComponentName中获取插件名,通过与插件名对应的插件化框架判断在服务管理组件中是否存在该插件,若在服务管理组件中存在该插件,则可以使用它提供的服务,此时从插件中获取目标服务的ServiceInfo对象,例如从插件中预设的ComponentList对象中获取目标服务的ServiceInfo对象,其中ComponentList对象保存有插件的PackageInfo对象,若在服务管理组件中不存在该插件,则需要下载该插件,从插件的安装包中获取目标服务的ServiceInfo对象;最后基于启动服务请求信息和目标服务的ServiceInfo对象创建目标服务的ServiceRecord对象,并将其保存到ServiceRecord缓存表和系统的Intent.FilterComparison表中,其中Intent.FilterComparison表用来进行Intent对比。上述操作的具体代码的示例如下:

[0136]

```
private ServiceRecord retrieveServiceLocked(Intent service) {
    ComponentName cn = service.getComponent();
    ServiceRecord sr = mServicesByName.get(cn);
    if (sr != null) {
        return sr;
    }
```

```
            sr = mServicesByIntent.get(service);
            if (sr != null) {
                return sr;
            }

            String pn = cn.getPackageName();
            if (!MP.isPluginExist(pn)) {
                return null;
            }

            ComponentList col = Factory.queryPluginComponentList(pn);
            if (col == null) {
                return null;
            }
            ServiceInfo si = col.getService(cn.getClassName());
            if (si == null) {
                return null;
            }

            Intent.FilterComparison fi = new Intent.FilterComparison(service);
            sr = new ServiceRecord(cn, fi, si);
            mServicesByName.put(cn, sr);
            mServicesByIntent.put(fi, sr);
            return sr;
        }
```

[0137]

[0138] 其中，ServiceRecord类包括：name，用来记录服务的全类名，以系统的ComponentName对象来封装，其中ComponentName对象的Key为插件名，Value为类全名；plugin，用来记录服务所属的插件的名字，也即为name.getPackageName()的结果，以字符串来表示；className，用来记录服务的类全名，也即为name.getClassName()的结果，以字符串来表示；intent，用来记录一些服务中的过滤条件，例如Action、IntentFilter等，以方

便之后读取，并做筛选；serviceInfo，用来记录插件的AndroidManifest.xml中描述的服务信息，为系统的ServiceInfo对象；service，用来记录服务对象，是在服务安装成功后创建的，为核心字段；startRequested，用来记录是否已启动服务，并调用onStartCommand，在解除绑定服务和停止服务时使用。

[0139] 图3是本发明实施例启动插件服务的方法的又一个实施例的流程图。如图3所示，该实施例的方法包括：

[0140] S302，通过服务管理组件的Binder对象接收客户端的启动服务请求信息。

[0141] S304，根据启动服务请求信息，检测在服务管理组件中是否保存有目标服务安装信息。

[0142] 若在服务管理组件中保存有目标服务安装信息，执行操作S306；否者，若在服务管理组件中未保存有目标服务安装信息，执行操作S308。

[0143] S308，创建目标服务安装信息。

[0144] S306，获取目标服务安装信息。

[0145] S310，根据目标服务安装信息，检测在服务管理组件中是否保存有目标服务。

[0146] 若在服务管理组件中保存有目标服务，执行操作314；否则，若在服务管理组件中未保存有目标服务，执行操作S312。

[0147] S312，基于目标服务安装信息安装目标服务。

[0148] S314，调用目标服务的onStartCommand函数。

[0149] 进一步，在操作S308之后，还执行操作S312，基于目标服务安装信息安装目标服务的操作。

[0150] 具体实现中，在操作S308之后，还可以包括：在服务管理组件中保存目标服务安装信息的操作，以在下一次启动目标服务时可以直接从服务管理组件中获取目标服务安装信息。

[0151] 进一步，在操作S312之后，还执行操作S314，调用目标服务的onStartCommand函数的操作。

[0152] 具体实现中，在操作S312之后，还可以包括：在服务管理组件中保存目标服务的操作，以在下一次启动目标服务时可以直接从服务管理组件中获取目标服务。

[0153] 在一个具体实施例中，操作S310至操作S314，首先根据所获得的ServiceRecord对象判断在服务管理组件中是否存在对应的服务，其中ServiceRecord缓存表中不仅记录有服务管理组件所在的进程中所有已安装的服务的安装信息，还记录有服务管理组件所在的进程中所有准备安装的服务的安装信息，若在服务管理组件中存在对应的服务，表示之前已经成功安装了该服务，直接调用onStartCommand函数即可，若在服务管理组件中不存在对应的服务，表示之前并未成功安装该服务，需要开始尝试安装该服务。其具体代码的示例如下：

```
private boolean installServiceIfNeededLocked(ServiceRecord sr) {
    if (sr.service != null) {
        return true;
    }
    return installServiceLocked(sr);
}
```

[0154]

[0155] 图4是本发明实施例启动插件服务的方法的再一个实施例的流程图。如图4所示，该实施例提供了一种基于目标服务安装信息安装目标服务的方法，该方法包括：

[0156] S402，获取提供目标服务的插件的Context对象。

[0157] S404，基于目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务。

[0158] S406，初始化目标服务。

[0159] 具体实现中，操作S404可以包括：从提供目标服务的插件的Context对象获取插件的类加载器，从目标服务安装信息获取目标服务的全类名，基于目标服务的全类名，通过反射机制利用插件的类加载器获取目标服务的系统类对象，通过反射机制调用目标服务的系统类的newInstance来调用构造函数创建目标服务。

[0160] 具体实现中，操作S406可以包括：通过反射机制调用目标服务的attachBaseContext的函数；以及调用目标服务的onCreate函数。

[0161] 其中，在安装目标服务之后，在服务管理组件中保存目标服务的操作，具体是在服务管理组件中保存初始化后的目标服务，以在下一次启动目标服务时能够直接从服务管理组件中获取初始化后的目标服务。

[0162] 在一个具体实施例中，操作402至操作406，首先获取提供目标服务的插件的Context对象，以通过它来获取ClassLoader对象，以及将其作为目标服务的BaseContext，然后从提供目标服务的插件的Context对象获取插件的ClassLoader对象，以便通过反射机制来创建目标服务，之后通过反射机制来创建目标服务；其具体包括以下三个步骤：从目标服务的ServiceRecord对象的ServiceInfo中的Name字段获取目标服务的全类名，通过反射机制利用插件的ClassLoader对象来获取目标服务的Class对象，即系统类对象，再通过反射机制调用目标服务的系统类的newInstance来调用构造函数创建目标服务；之后通过反射机制调用目标服务的attachBaseContext的函数，最后调用目标服务的onCreate函数，实现对目标服务的初始化。其具体代码的示例如下：

[0163]

```
private boolean installServiceLocked(ServiceRecord sr) {
    Context plgc = Factory.queryPluginContext(sr.plugin);
    if (plgc == null) {
        return false;
    }

    ClassLoader cl = plgc.getClassLoader();
    if (cl == null) {
        return false;
    }

    Service s = (Service) cl.loadClass(sr.serviceInfo.name).newInstance();
    if (s == null) {
        return false;
    }

    try {
        attachBaseContextLocked(s, plgc);
    } catch (Throwable e) {
        return false;
    }

    s.onCreate();

    sr.service = s;
    return true;
}
```

[0164]

[0165] 其中,attachBaseContext函数的具体代码的示例如下：

```
private void attachBaseContextLocked(ContextWrapper cw, Context c) throws
NoSuchMethodException,                    InvocationTargetException,
IllegalAccessException {
    if (mAttachBaseContextMethod == null) {
[0166]          mAttachBaseContextMethod                              =
ContextWrapper.class.getDeclaredMethod("attachBaseContext",
Context.class);
        mAttachBaseContextMethod.setAccessible(true);
    }
    mAttachBaseContextMethod.invoke(cw, c);
[0167]
}
```

[0168] 在该函数中，参数ContextWrapper是目标服务，而目标服务的父类是系统的ContextWrapper类，参数Context则是之前获取的插件的Context对象。

[0169] 图5是本发明实施例启动插件服务的装置的一个实施例的结构图。如图5所示，该实施例的装置包括：接收单元510和执行单元520。其中：

[0170] 接收单元510，用于通过服务管理组件的Binder对象接收客户端的启动服务请求信息。

[0171] 执行单元520，用于根据启动服务请求信息启动服务管理组件所在的进程中的目标服务。

[0172] 其中，服务管理组件管理其所在的进程中的所有服务，服务管理组件与客户端运行在不同的进程中。

[0173] 具体实现中，接收单元510可以具体用于根据启动服务请求信息调用服务管理组件的AIDL接口对象的startService函数，执行单元520可以具体用于通过服务管理组件中继承所述AIDL接口的内部类对象访问服务管理组件核心类的startService函数来启动目标服务，其中服务管理组件的核心类实现AIDL接口。

[0174] 基于本发明实施例提供的启动插件服务的装置，通过为服务端每个进程设置服务管理组件，管理其所在进程中的所有服务，当通过服务管理组件的Binder对象接收到客户端的启动服务请求信息后，根据此启动服务请求信息启动服务管理组件所在的进程中的目标服务，当目标服务与客户端运行在不同的进程中时，可以实现对目标服务的启动，由于本发明实施例是通过设置于每一进程中的服务管理组件对进程中的所有服务进行管理，因此其既不需要在主程序的AndroidManifest.xml文件中预设服务坑位，也不需要在服务升级时对主程序进行升级的额外处理，即可实现对插件提供的服务的启动。

[0175] 图6是本发明实施例启动插件服务的装置的另一个实施例的结构图。如图6所示，与图5的实施例相比，在该实施例中，启动插件服务的装置的执行单元620还包括：第一检测模块621、获取模块622和创建模块623。其中：

[0176] 第一检测模块621，用于根据启动服务请求信息，检测在服务管理组件中是否保存

有目标服务安装信息。

[0177] 获取模块622，用于根据第一检测模块621的检测结果，响应于在服务管理组件中保存有目标服务安装信息，获取目标服务安装信息。

[0178] 创建模块623，用于根据第一检测模块621的检测结果，响应于在服务管理组件中未保存有目标服务安装信息，创建目标服务安装信息。

[0179] 具体实现中，创建模块623可以具体用于：首先根据启动服务请求信息，检测在服务管理组件中是否保存有提供目标服务的插件，若在服务管理组件中保存有提供目标服务的插件，则从插件中获取目标服务的ServiceInfo对象，若在服务管理组件中未保存有提供目标服务的插件，则从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象，最后基于启动服务请求信息和所获得的目标服务的ServiceInfo对象，创建目标服务安装信息。

[0180] 进一步，如图6所示，执行单元620还包括：安装模块624，用于在创建模块623创建目标服务安装信息之后，基于目标服务安装信息安装目标服务。

[0181] 具体实现中，执行单元620还可以包括：第一存储模块，用于在服务管理组件中保存目标服务安装信息，以在下一次启动目标服务时可以直接从服务管理组件中获取目标服务安装信息。

[0182] 图7是本发明实施例启动插件服务的装置的又一个实施例的结构图。如图7所示，与图6的实施例相比，在该实施例中，启动插件服务的装置的执行单元720还包括：第二检测模块725和执行模块726。其中：

[0183] 第二检测模块725，用于根据目标服务安装信息，检测在服务管理组件中是否保存有目标服务。

[0184] 执行模块726，用于根据第二检测模块725的检测结果，响应于在服务管理组件中保存有目标服务，调用目标服务的onStartCommand函数。

[0185] 安装模块724，还用于根据第二检测模块725的检测结果，响应于在服务管理组件中未保存有目标服务，基于目标服务安装信息安装目标服务。

[0186] 具体实现中，安装模块724可以具体用于：获取提供目标服务的插件的Context对象，基于目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务，以及初始化所述目标服务。

[0187] 具体地，安装模块724基于目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务，具体用于：从提供目标服务的插件的Context对象获取插件的类加载器，从目标服务安装信息获取目标服务的全类名，基于目标服务的全类名，通过反射机制利用插件的类加载器获取目标服务的系统类对象，通过反射机制调用目标服务的系统类的newInstance来调用构造函数创建目标服务。

[0188] 具体地，安装模块724初始化目标服务，具体用于：通过反射机制调用目标服务的attachBaseContext的函数，以及调用目标服务的onCreate函数。

[0189] 进一步，执行单元720还包括：第二存储模块，用于在服务管理组件中保存目标服务，以在下一次启动目标服务时可以直接从服务管理组件中获取目标服务。

[0190] 具体地，在安装模块724初始化目标服务之后，第二存储模块用于保存初始化后的目标服务，以在下一次启动目标服务时能够直接从服务管理组件中获取初始化后的目标服

务。

[0191] 进一步，执行模块726，还用于在安装模块724基于目标服务安装信息安装目标服务之后，调用目标服务的onStartCommand函数。

[0192] 另外，本发明实施例还提供了终端设备，如图8所示，为了便于说明，仅示出了与本发明实施例相关的部分，具体技术细节未揭示的，请参照本发明实施例方法部分。该终端设备可以为包括手机、平板电脑、PDA(Personal Digital Assistant，个人数字助理)、POS(Point of Sales，销售终端)、车载电脑等任意终端设备，以终端设备为手机为例：

[0193] 图8是与本发明实施例提供的终端设备相关的手机的部分结构的框图。参考图8，手机包括：射频(Radio Frequency，RF)电路810、存储器820、输入单元830、显示单元840、传感器850、音频电路860、无线保真(wireless-fidelity，Wi-Fi)模块870、处理器880、以及电源890等部件。本领域技术人员可以理解，图8中示出的手机结构并不构成对手机的限定，可以包括比图示更多或更少的部件，或者组合某些部件，或者不同的部件布置。

[0194] 下面结合图8对手机的各个构成部件进行具体的介绍：

[0195] RF电路810可用于收发信息或通话过程中，信号的接收和发送，特别地，将基站的下行信息接收后，给处理器880处理；另外，将设计上行的数据发送给基站。通常，RF电路810包括但不限于天线、至少一个放大器、收发信机、耦合器、低噪声放大器(Low Noise Amplifier，LNA)、双工器等。此外，RF电路810还可以通过无线通信与网络和其他设备通信。上述无线通信可以使用任一通信标准或协议，包括但不限于全球移动通讯系统(Global System of Mobile communication，GSM)、通用分组无线服务(General Packet Radio Service，GPRS)、码分多址(Code Division Multiple Access，CDMA)、宽带码分多址(Wideband Code Division Multiple Access，WCDMA)、长期演进(Long Term Evolution，LTE)、电子邮件、短消息服务(Short Messaging Service，SMS)等。

[0196] 存储器820可用于存储软件程序以及模块，处理器880通过运行存储在存储器820的软件程序以及模块，从而执行手机的各种功能应用以及数据处理。存储器820可主要包括存储程序区和存储数据区，其中，存储程序区可存储操作系统、至少一个功能所需的应用程序(比如声音播放功能、图像播放功能等)等；存储数据区可存储根据手机的使用所创建的数据(比如音频数据、电话本等)等。此外，存储器820可以包括高速随机存取存储器，还可以包括非易失性存储器，例如至少一个磁盘存储器件、闪存器件、或其他易失性固态存储器件。

[0197] 输入单元830可用于接收输入的数字或字符信息，以及产生与手机的用户设置以及功能控制有关的键信号输入。具体地，输入单元830可包括触控面板831以及其他输入设备832。触控面板831，也称为触摸屏，可收集用户在其上或附近的触摸操作(比如用户使用手指、触笔等任何适合的物体或附件在触控面板831上或在触控面板831附近的操作)，并根据预先设定的程式驱动相应的连接装置。可选的，触控面板831可包括触摸检测装置和触摸控制器两个部分。其中，触摸检测装置检测用户的触摸方位，并检测触摸操作带来的信号，将信号传送给触摸控制器；触摸控制器从触摸检测装置上接收触摸信息，并将它转换成触点坐标，再送给处理器880，并能接收处理器880发来的命令并加以执行。此外，可以采用电阻式、电容式、红外线以及表面声波等多种类型实现触控面板831。除了触控面板831，输入单元830还可以包括其他输入设备832。具体地，其他输入设备832可以包括但不限于物理键

盘、功能键(比如音量控制按键、开关按键等)、轨迹球、鼠标、操作杆等中的一种或多种。

[0198] 显示单元840可用于显示由用户输入的信息或提供给用户的信息以及手机的各种菜单。显示单元840可包括显示面板841,可选的,可以采用液晶显示器(Liquid Crystal Display,LCD)、有机发光二极管(Organic Light-Emitting Diode,OLED)等形式来配置显示面板841。进一步的,触控面板831可覆盖显示面板841,当触控面板831检测到在其上或附近的触摸操作后,传送给处理器880以确定触摸事件的类型,随后处理器880根据触摸事件的类型在显示面板841上提供相应的视觉输出。虽然在图8中,触控面板831与显示面板841是作为两个独立的部件来实现手机的输入和输入功能,但是在某些实施例中,可以将触控面板831与显示面板841集成而实现手机的输入和输出功能。

[0199] 手机还可包括至少一种传感器850,比如光传感器、运动传感器以及其他传感器。具体地,光传感器可包括环境光传感器及接近传感器,其中,环境光传感器可根据环境光线的明暗来调节显示面板841的亮度,接近传感器可在手机移动到耳边时,关闭显示面板841和/或背光。作为运动传感器的一种,加速计传感器可检测各个方向上(一般为三轴)加速度的大小,静止时可检测出重力的大小及方向,可用于识别手机姿态的应用(比如横竖屏切换、相关游戏、磁力计姿态校准)、振动识别相关功能(比如计步器、敲击)等;至于手机还可配置的陀螺仪、气压计、湿度计、温度计、红外线传感器等其他传感器,在此不再赘述。

[0200] 音频电路860、扬声器861,传声器862可提供用户与手机之间的音频接口。音频电路860可将接收到的音频数据转换后的电信号,传输到扬声器861,由扬声器861转换为声音信号输出;另一方面,传声器862将收集的声音信号转换为电信号,由音频电路860接收后转换为音频数据,再将音频数据输出处理器880处理后,经RF电路810以发送给比如另一手机,或者将音频数据输出至存储器820以便进一步处理。

[0201] WiFi属于短距离无线传输技术,手机通过WiFi模块870可以帮助用户收发电子邮件、浏览网页和访问流式媒体等,它为用户提供了无线的宽带互联网访问。虽然图8示出了WiFi模块870,但是可以理解的是,其并不属于手机的必须构成,完全可以根据需要在不改变发明的本质的范围内而省略。

[0202] 处理器880是手机的控制中心,利用各种接口和线路连接整个手机的各个部分,通过运行或执行存储在存储器820内的软件程序和/或模块,以及调用存储在存储器820内的数据,执行手机的各种功能和处理数据,从而对手机进行整体监控。可选的,处理器880可包括一个或多个处理单元;优选的,处理器880可集成应用处理器和调制解调处理器,其中,应用处理器主要处理操作系统、用户界面和应用程序等,调制解调处理器主要处理无线通信。可以理解的是,上述调制解调处理器也可以不集成到处理器880中。

[0203] 手机还包括给各个部件供电的电源890(比如电池),优选的,电源可以通过电源管理系统与处理器880逻辑相连,从而通过电源管理系统实现管理充电、放电、以及功耗管理等功能。

[0204] 尽管未示出,手机还可以包括摄像头、蓝牙模块等,在此不再赘述。

[0205] 在本发明实施例中,该终端设备所包括的处理器880还具有以下功能:

[0206] 通过服务管理组件的Binder对象接收客户端的启动服务请求信息;

[0207] 根据启动服务请求信息启动服务管理组件所在的进程中的目标服务;

[0208] 其中,服务管理组件管理其所在的进程中的所有服务,服务管理组件与客户端运

行在不同的进程中。

[0209] 本发明实施例提供了以下技术方案：

[0210] 1、一种启动插件服务的方法，包括：

[0211] 通过服务管理组件的Binder对象接收客户端的启动服务请求信息；

[0212] 根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务；

[0213] 其中，所述服务管理组件管理其所在的进程中的所有服务，所述服务管理组件与所述客户端运行在不同的进程中。

[0214] 2、根据1所述的方法，所述通过服务管理组件的Binder对象接收客户端的启动服务请求信息，包括：

[0215] 根据所述启动服务请求信息调用所述服务管理组件的AIDL接口对象的startService函数。

[0216] 3、根据2所述的方法，所述根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务，包括：

[0217] 通过所述服务管理组件中继承所述AIDL接口的内部类对象访问所述服务管理组件核心类的startService函数；其中所述服务管理组件的核心类实现所述AIDL接口。

[0218] 4、根据1至3任意一项所述的方法，所述根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务，包括：

[0219] 根据所述启动服务请求信息，检测在所述服务管理组件中是否保存有目标服务安装信息；

[0220] 若在所述服务管理组件中保存有目标服务安装信息，获取所述目标服务安装信息；

[0221] 若在所述服务管理组件中未保存有目标服务安装信息，创建目标服务安装信息。

[0222] 5、根据4所述的方法，所述创建目标服务安装信息，包括：

[0223] 根据所述启动服务请求信息，检测在所述服务管理组件中是否保存有提供目标服务的插件；

[0224] 若在所述服务管理组件中保存有提供目标服务的插件，从所述插件中获取目标服务的ServiceInfo对象；

[0225] 若在所述服务管理组件中未保存有提供目标服务的插件，从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象；

[0226] 基于所述启动服务请求信息和所述目标服务的ServiceInfo对象，创建目标服务安装信息。

[0227] 6、根据4或5所述的方法，所述创建目标服务安装信息之后，还包括：

[0228] 在所述服务管理组件中保存所述目标服务安装信息。

[0229] 7、根据4至6任意一项所述的方法，所述创建目标服务安装信息之后，还包括：

[0230] 基于所述目标服务安装信息安装目标服务。

[0231] 8、根据7所述的方法，所述获取所述目标服务安装信息之后，还包括：

[0232] 根据所述目标服务安装信息，检测在所述服务管理组件中是否保存有目标服务；

[0233] 若在所述服务管理组件中保存有目标服务，调用所述目标服务的onStartCommand函数；

[0234] 若在所述服务管理组件中未保存有目标服务，基于所述目标服务安装信息安装目标服务。

[0235] 9、根据7或8所述的方法，所述基于所述目标服务安装信息安装目标服务，包括：

[0236] 获取提供目标服务的插件的Context对象；

[0237] 基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务；

[0238] 初始化所述目标服务。

[0239] 10、根据9所述的方法，所述基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务，包括：

[0240] 从提供目标服务的插件的Context对象获取所述插件的类加载器；

[0241] 从所述目标服务安装信息获取目标服务的全类名；

[0242] 基于所述目标服务的全类名，通过反射机制利用所述插件的类加载器获取所述目标服务的系统类对象；

[0243] 通过反射机制调用所述目标服务的系统类的newInstance来调用构造函数创建目标服务。

[0244] 11、根据9所述的方法，所述初始化所述目标服务，包括：

[0245] 通过反射机制调用所述目标服务的attachBaseContext的函数；

[0246] 调用所述目标服务的onCreate函数。

[0247] 12、根据8至11任意一项所述的方法，所述基于所述目标服务安装信息安装目标服务之后，还包括：

[0248] 在所述服务管理组件中保存所述目标服务。

[0249] 13、根据8至12任意一项所述的方法，所述基于所述目标服务安装信息安装目标服务之后，还包括：

[0250] 调用所述目标服务的onStartCommand函数。

[0251] 14、一种启动插件服务的装置，包括：

[0252] 接收单元，用于通过服务管理组件的Binder对象接收客户端的启动服务请求信息；

[0253] 执行单元，用于根据所述启动服务请求信息启动所述服务管理组件所在的进程中的目标服务；

[0254] 其中，所述服务管理组件管理其所在的进程中的所有服务，所述服务管理组件与所述客户端运行在不同的进程中。

[0255] 15、根据14所述的装置，所述接收单元，具体用于根据所述启动服务请求信息调用所述服务管理组件的AIDL接口对象的startService函数。

[0256] 16、根据15所述的装置，所述执行单元，具体用于通过所述服务管理组件中继承所述AIDL接口的内部类对象访问所述服务管理组件核心类的startService函数；其中，所述服务管理组件的核心类实现所述AIDL接口。

[0257] 17、根据16所述的装置，所述执行单元，包括：

[0258] 第一检测模块，用于根据所述启动服务请求信息，检测在所述服务管理组件中是否保存有目标服务安装信息；

[0259] 获取模块，用于根据所述第一检测模块的检测结果，响应于在所述服务管理组件中保存有目标服务安装信息，获取所述目标服务安装信息；

[0260] 创建模块，用于根据所述第一检测模块的检测结果，响应于在所述服务管理组件中未保存有目标服务安装信息，创建目标服务安装信息。

[0261] 18、根据17所述的装置，所述创建模块，具体用于：

[0262] 根据所述启动服务请求信息，检测在所述服务管理组件中是否保存有提供目标服务的插件；

[0263] 若在所述服务管理组件中保存有提供目标服务的插件，从所述插件中获取目标服务的ServiceInfo对象；

[0264] 若在所述服务管理组件中未保存有提供目标服务的插件，从提供目标服务的插件的安装包中获取目标服务的ServiceInfo对象；

[0265] 基于所述启动服务请求信息和所述目标服务的ServiceInfo对象，创建目标服务安装信息。

[0266] 19、根据17或18所述的装置，所述执行单元，还包括：

[0267] 第一存储模块，用于在所述服务管理组件中保存所述目标服务安装信息。

[0268] 20、根据17至19任意一项所述的装置，所述执行单元，还包括：

[0269] 安装模块，用于在所述创建模块创建目标服务安装信息之后，基于所述目标服务安装信息安装目标服务。

[0270] 21、根据20所述的装置，所述执行单元，还包括：

[0271] 第二检测模块，用于根据所述目标服务安装信息，检测在所述服务管理组件中是否保存有目标服务；

[0272] 执行模块，用于根据所述第二检测模块的检测结果，响应于在所述服务管理组件中保存有目标服务，调用所述目标服务的onStartCommand函数；

[0273] 所述安装模块，还用于根据所述第二检测模块的检测结果，响应于在所述服务管理组件中未保存有目标服务，基于所述目标服务安装信息安装目标服务。

[0274] 22、根据20或21所述的装置，所述安装模块，具体用于：

[0275] 获取提供目标服务的插件的Context对象；

[0276] 基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务；

[0277] 初始化所述目标服务。

[0278] 23、根据22所述的装置，所述安装模块基于所述目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务，具体用于：

[0279] 从提供目标服务的插件的Context对象获取所述插件的类加载器；

[0280] 从所述目标服务安装信息获取目标服务的全类名；

[0281] 基于所述目标服务的全类名，通过反射机制利用所述插件的类加载器获取所述目标服务的系统类对象；

[0282] 通过反射机制调用所述目标服务的系统类的newInstance来调用构造函数创建目标服务。

[0283] 24、根据22所述的装置，所述安装模块初始化所述目标服务，具体用于：

[0284] 通过反射机制调用所述目标服务的attachBaseContext的函数；

[0285] 调用所述目标服务的onCreate函数。

[0286] 25、根据21至24任意一项所述的装置，所述执行单元，还包括：

[0287] 第二存储模块，用于在所述服务管理组件中保存所述目标服务。

[0288] 26、根据21至25任意一项所述的装置，所述执行模块，还用于在所述安装模块基于所述目标服务安装信息安装目标服务之后，调用所述目标服务的onStartCommand函数。

[0289] 27、一种终端设备，包括：处理器和存储器；其中，

[0290] 所述存储器用于存储1至13任意一项所述的启动插件服务的方法；

[0291] 所述处理器用于执行所述启动插件服务的方法。

[0292] 本说明书中各个实施例均采用递进的方式描述，每个实施例重点说明的都是与其它实施例的不同之处，各个实施例之间相同或相似的部分相互参见即可。对于系统实施例而言，由于其与方法实施例基本对应，所以描述的比较简单，相关之处参见方法实施例的部分说明即可。

[0293] 可能以许多方式来实现本发明的方法和装置、设备。例如，可通过软件、硬件、固件或者软件、硬件、固件的任何组合来实现本发明的方法和装置、设备。用于所述方法的步骤的上述顺序仅是为了进行说明，本发明的方法的步骤不限于以上具体描述的顺序，除非以其它方式特别说明。此外，在一些实施例中，还可将本发明实施为记录在记录介质中的程序，这些程序包括用于实现根据本发明的方法的机器可读指令。因而，本发明还覆盖存储用于执行根据本发明的方法的程序的记录介质。

[0294] 本发明的描述是为了示例和描述起见而给出的，而并不是无遗漏的或者将本发明限于所公开的形式。很多修改和变化对于本领域的普通技术人员而言是显然的。选择和描述实施例是为了更好说明本发明的原理和实际应用，并且使本领域的普通技术人员能够理解本发明从而设计适于特定用途的带有各种修改的各种实施例。

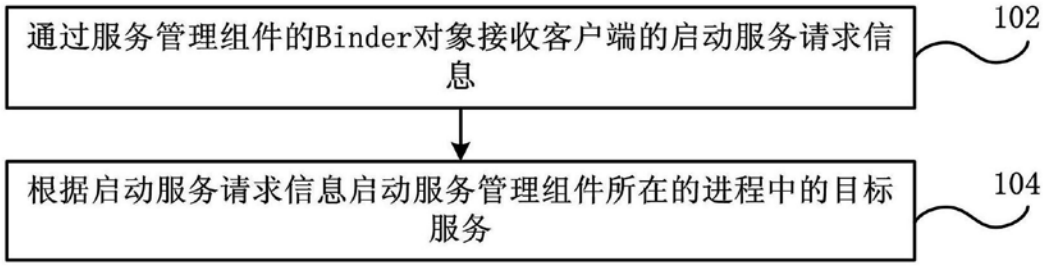

通过服务管理组件的Binder对象接收客户端的启动服务请求信息
102
根据启动服务请求信息启动服务管理组件所在的进程中的目标服务
104

图1

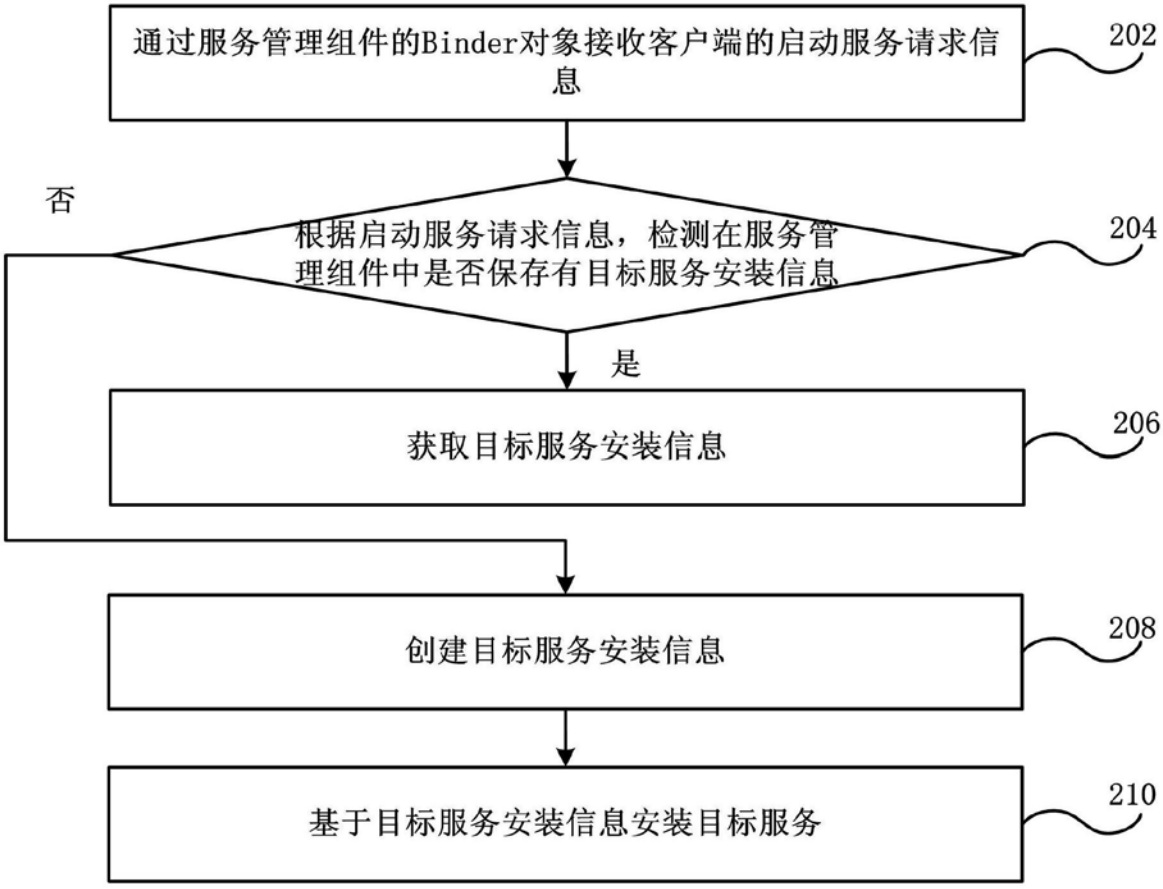

通过服务管理组件的Binder对象接收客户端的启动服务请求信息
202
否
根据启动服务请求信息，检测在服务管理组件中是否保存有目标服务安装信息
204
是
获取目标服务安装信息
206
创建目标服务安装信息
208
基于目标服务安装信息安装目标服务
210

图2

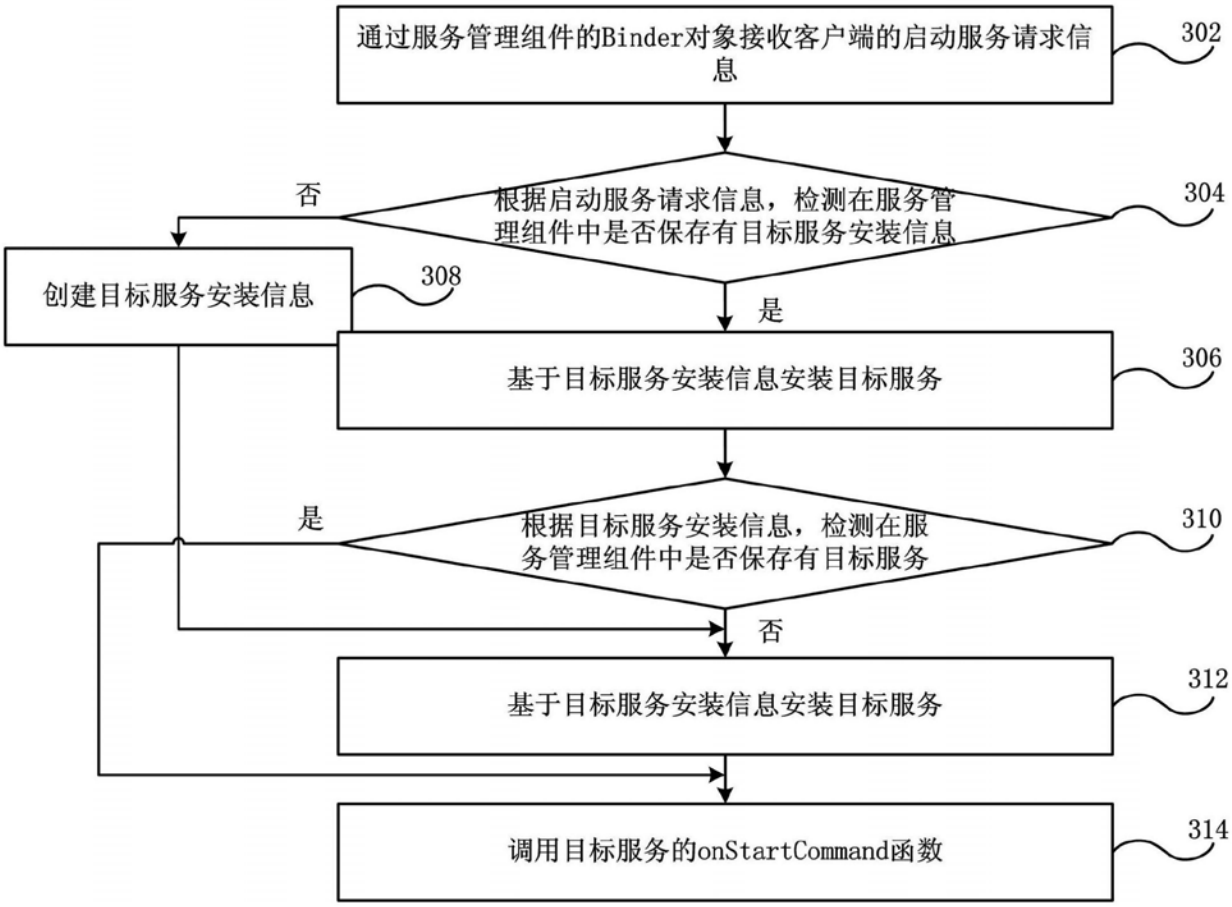

通过服务管理组件的Binder对象接收客户端的启动服务请求信息
302
根据启动服务请求信息，检测在服务管理组件中是否保存有目标服务安装信息
304
否
是
创建目标服务安装信息
308
基于目标服务安装信息安装目标服务
306
根据目标服务安装信息，检测在服务管理组件中是否保存有目标服务
310
是
否
基于目标服务安装信息安装目标服务
312
调用目标服务的onStartCommand函数
314

图3

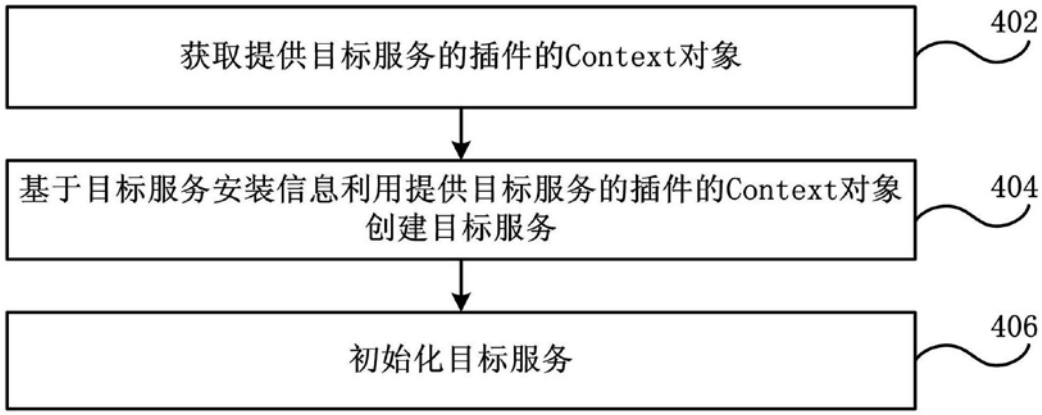

获取提供目标服务的插件的Context对象
402
基于目标服务安装信息利用提供目标服务的插件的Context对象创建目标服务
404
初始化目标服务
406

图4


接收单元
510
执行单元
520

图5

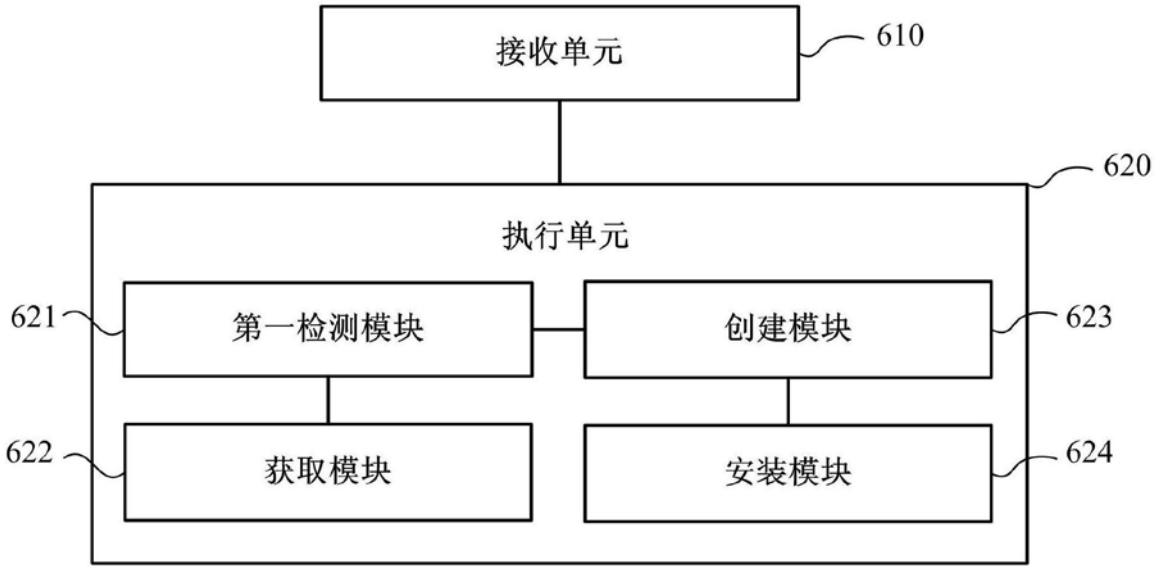

接收单元
610
620
执行单元
621
第一检测模块
创建模块
623
622
获取模块
安装模块
624

图6

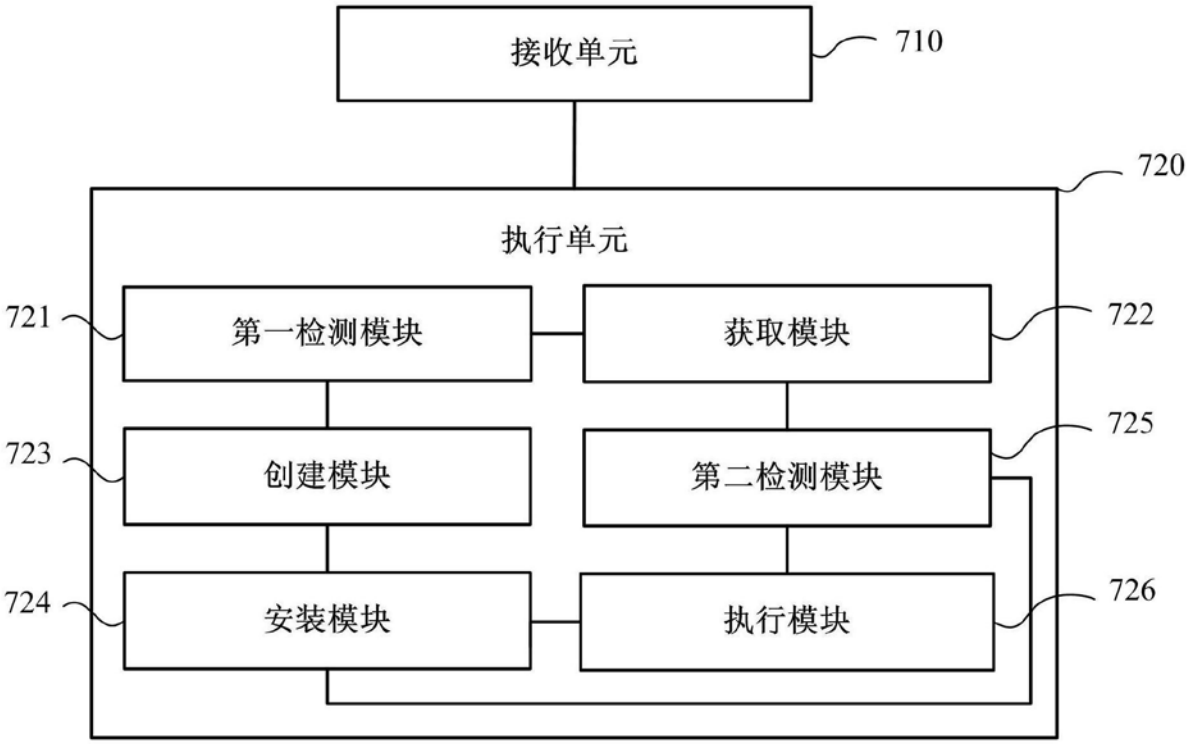

接收单元
710
720
执行单元
721
第一检测模块
获取模块
722
723
创建模块
第二检测模块
725
724
安装模块
执行模块
726

图7

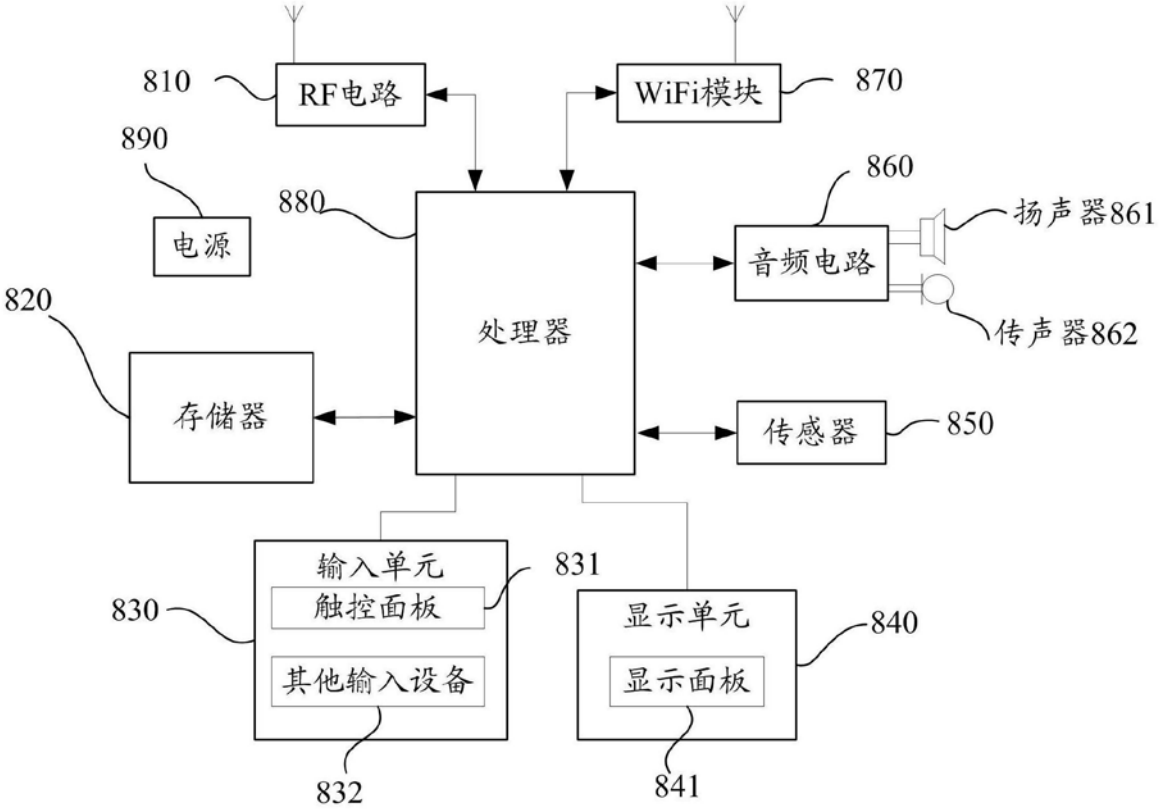

810
RF电路
WiFi模块
870
890
电源
880
处理器
860
音频电路
扬声器861
传声器862
820
存储器
传感器
850
输入单元
831
830
触控面板
其他输入设备
832
显示单元
840
显示面板
841

图8