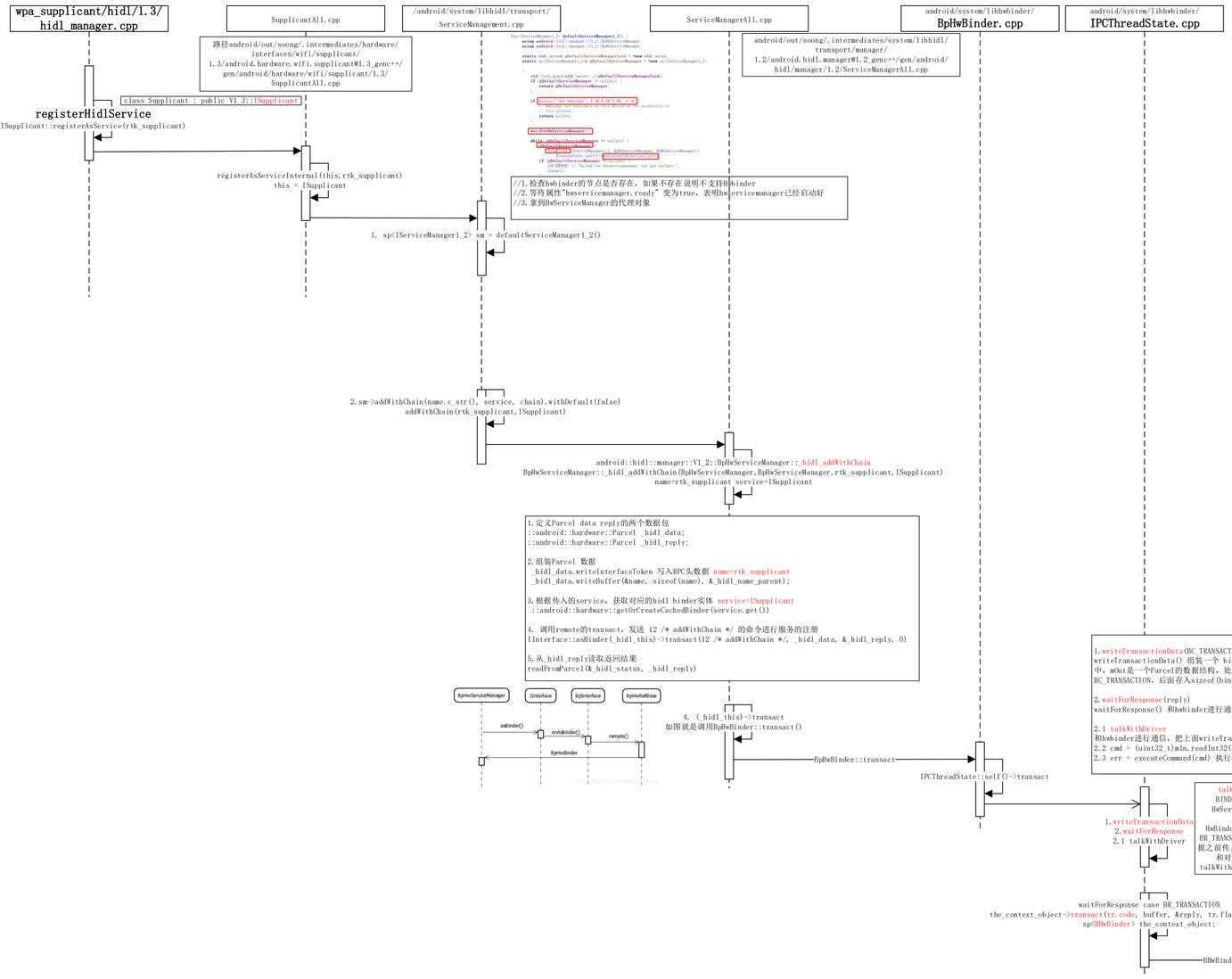
Native层注册HIDL Services



总结

1.服务进程先获得一个 ProcessState()的对象

2.获取HwServiceManager的代理对象HwBpServiceManager，主要通过new BpHwBinder(0)得到。

3.调用HwBpServiceManager的addWithChain，组装一个Parce数据，传入服务名称、服务实体对象一BHwBinder、执行12 /* addWithChain */

4.先通过IPCThreadThread的writeTransactionData()把上面的Parcel数据写入mOut，用来进行发送

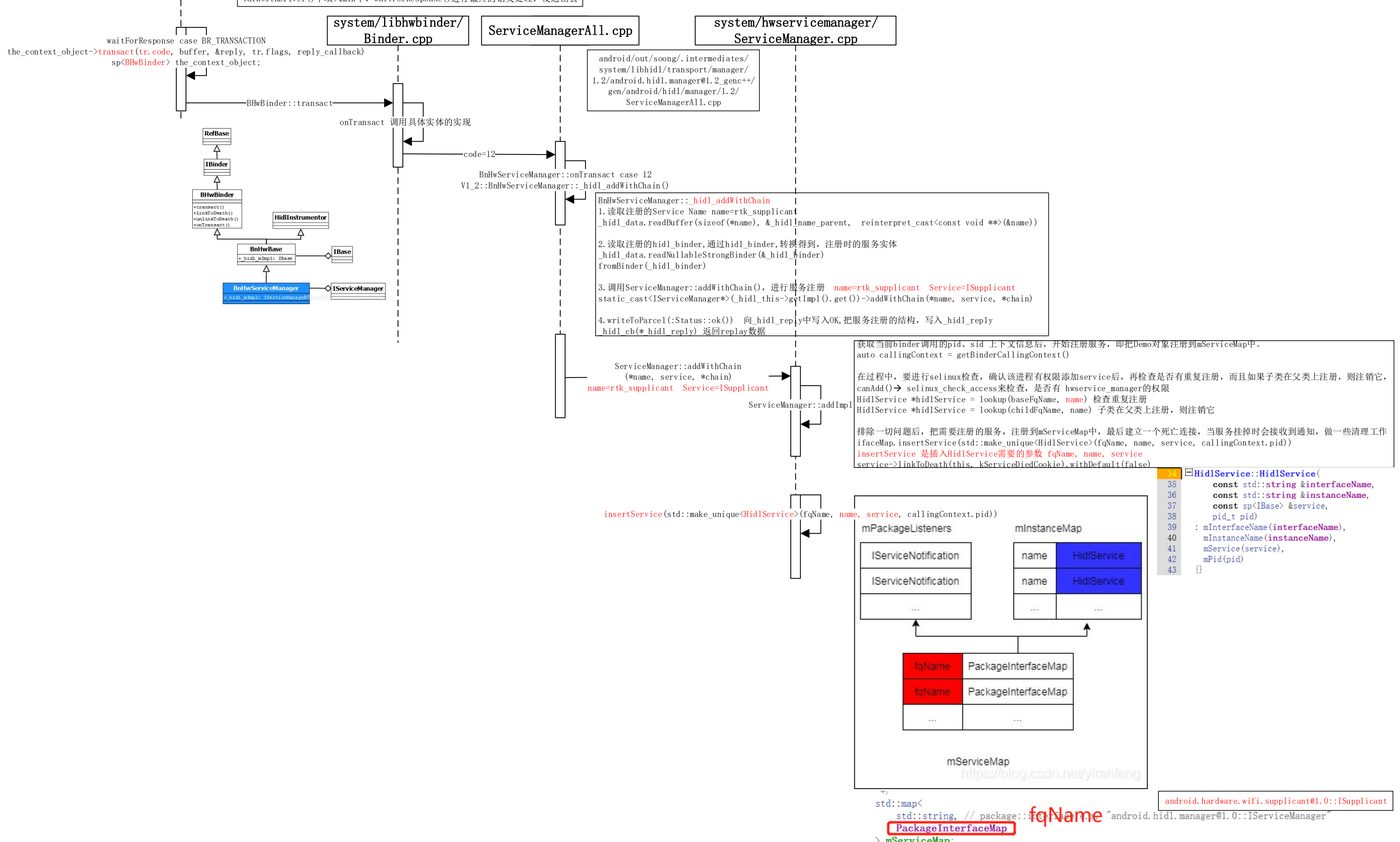
5.通过IPCThreadThread的talkWithDriver()与HwBinder驱动通信，传递语义BINDER_WRITE_READ

6.HwServiceManager有个for循环，调用epoll_wait()监控hwbinder的变化

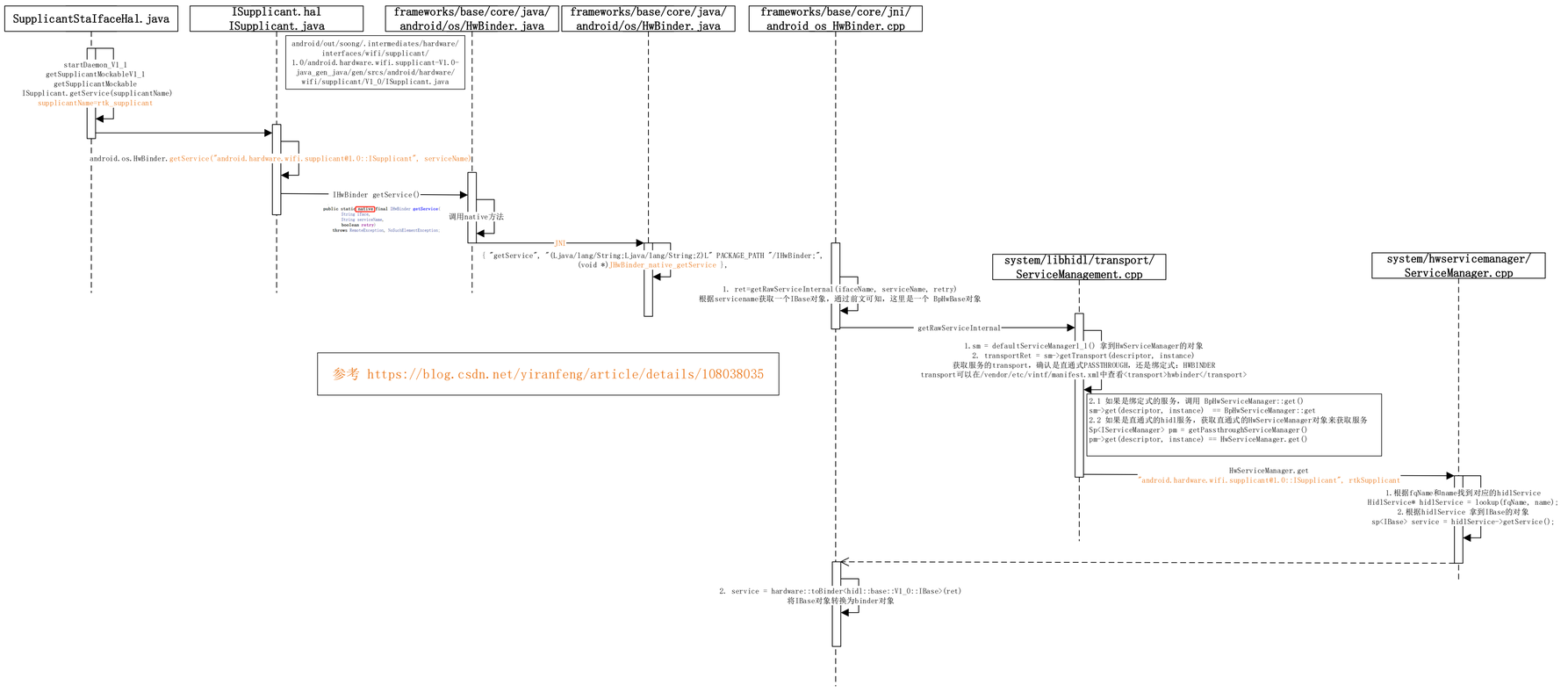
7.waitForResponse()收到BR_TRANSACTION响应码后，最终后调用executeCommand()进行命令执行，根据逻辑处理，最终会调用到BnHwServiceManager::onTransact()，根据传入的code=12,调用addWithChain，最终把服务名称和实体对象插入到mServiceMap这个map中

8.注册成功后，把注册结果写入reply，返回reply信息

9.最终完成服务的注册流程



JAVA层注册HIDL Service获取




```
struct PackageInterfaceMap {
    InstanceMap &getInstanceMap();
    const InstanceMap &getInstanceMap() const;

    /**
     * Finds a HidlService with the desired name. If none,
     * returns nullptr. HidlService::getService() might also be nullptr
     * if there are registered IServiceNotification objects for it. Return
     * value should be treated as a temporary reference.
     */
    HidlService *lookup(
        const std::string &name);
    const HidlService *lookup(
        const std::string &name) const;

    void insertService(std::unique_ptr<HidlService> &&service);

    void addPackageListener(sp<IServiceNotification> listener);
    bool removePackageListener(const wp<IBase>& who);
    bool removeServiceListener(const wp<IBase>& who);

    void sendPackageRegistrationNotification(
        const hidl_string &fqName,
        const hidl_string &instanceName);

private:
    InstanceMap mInstanceMap{};

    std::vector<sp<IServiceNotification>> mPackageListeners{};
};

using InstanceMap = std::map<
    std::string, // instance name e.x. "manager"
    std::unique_ptr<HidlService>
>;
```

rtk_supplicant

```
struct HidlService {
    HidlService(const std::string &interfaceName,
                const std::string &instanceName,
                const sp<IBase> &service,
                const pid_t pid);
    HidlService(const std::string &interfaceName,
                const std::string &instanceName)
    : HidlService(
        interfaceName,
        instanceName,
        nullptr,
        static_cast<pid_t>(IServiceManager::PidConstant::NO_PID))
    {}
    virtual ~HidlService() {}
```

android.hardware.wifi.supplicant@1.0::ISupplicant

ISupplicant