

WS73V100 Android 星闪协议移植

用户指南

文档版本 02

发布日期 2025-01-21

前言

概述

本文档介绍了 WS73V100 SDK 及在 Android 平台下的星闪协议移植指导，帮助用户快速实现对 Android 系统的星闪协议移植。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
WS73	V100

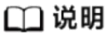
读者对象

本文档主要适用于以下工程师：

- 技术支持工程师
- 产品测试工程师
- 产品开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不可避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不可避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不可避免则可能导致轻微或中度伤害的具有低等级风险的危害。
 须知	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
 说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
02	2025-01-21	- 更新 “1 源码准备” 章节内容。 - 更新 “2 源码适配” 章节内容。 - 更新 “4.3.3 HID 断连” 章节内容。
01	2024-03-29	第一次正式版本发布。

目 录

前言 i

1 源码准备 1

2 源码适配 3

2.1 frameworks 3

2.1.1 SystemServiceRegistry.java 3

2.1.1.1 变更类型 3

2.1.1.2 同步方式 3

2.1.1.3 修改说明 3

2.1.1.4 修改内容 3

2.1.2 Context.java 4

2.1.2.1 变更类型 4

2.1.2.2 同步方式 4

2.1.2.3 修改说明 4

2.1.2.4 修改内容 4

2.1.3 PackageManager.java 4

2.1.3.1 变更类型 4

2.1.3.2 同步方式 4

2.1.3.3 修改说明 5

2.1.3.4 修改内容 5

2.1.4 Process.java 5

2.1.4.1 变更类型 5

2.1.4.2 同步方式 5

2.1.4.3 修改说明 5

2.1.4.4 修改内容 5

2.1.5 UserManager.java 6

2.1.5.1 变更类型	6
2.1.5.2 同步方式	6
2.1.5.3 修改说明	6
2.1.5.4 修改内容	6
2.1.6 Settings.java	7
2.1.6.1 变更类型	7
2.1.6.2 同步方式	7
2.1.6.3 修改说明	7
2.1.6.4 修改内容	7
2.1.7 nearlink	10
2.1.7.1 变更类型	10
2.1.7.2 同步方式	10
2.1.7.3 修改说明	10
2.1.7.4 修改内容	10
2.1.8 config.xml	11
2.1.8.1 变更类型	11
2.1.8.2 同步方式	11
2.1.8.3 修改说明	11
2.1.8.4 修改内容	11
2.1.9 symbols.xml	11
2.1.9.1 变更类型	11
2.1.9.2 同步方式	12
2.1.9.3 修改说明	12
2.1.9.4 修改内容	12
2.1.10 enums.proto	12
2.1.10.1 变更类型	12
2.1.10.2 同步方式	12
2.1.10.3 修改说明	12
2.1.10.4 修改内容	12
2.1.11 AndroidManifest.xml	13
2.1.11.1 变更类型	13
2.1.11.2 同步方式	13
2.1.11.3 修改说明	13

2.1.11.4 修改内容	13
2.1.12 platform.xml	15
2.1.12.1 变更类型	15
2.1.12.2 同步方式	15
2.1.12.3 修改说明	15
2.1.12.4 修改内容	15
2.1.13 SettingsLib	15
2.1.13.1 变更类型	15
2.1.13.2 同步方式	16
2.1.13.3 修改说明	16
2.1.13.4 修改内容	16
2.1.14 metrics_constants.proto	16
2.1.14.1 变更类型	16
2.1.14.2 同步方式	16
2.1.14.3 修改说明	16
2.1.14.4 修改内容	16
2.1.15 ActivityManagerService.java	19
2.1.15.1 变更类型	19
2.1.15.2 同步方式	19
2.1.15.3 修改说明	19
2.1.15.4 修改内容	19
2.1.16 PackageManagerService.java	19
2.1.16.1 变更类型	19
2.1.16.2 同步方式	20
2.1.16.3 修改说明	20
2.1.16.4 修改内容	20
2.1.17 NearlinkManagerService.java	20
2.1.17.1 变更类型	20
2.1.17.2 同步方式	20
2.1.17.3 修改说明	20
2.1.17.4 修改内容	20
2.1.18 NearlinkService.java	20
2.1.18.1 变更类型	20

2.1.18.2 同步方式	20
2.1.18.3 修改说明	21
2.1.18.4 修改内容	21
2.1.19 SystemServer.java	21
2.1.19.1 变更类型	21
2.1.19.2 同步方式	21
2.1.19.3 修改说明	21
2.1.19.4 修改内容	21
2.1.20 Android.bp	22
2.1.20.1 变更类型	22
2.1.20.2 同步方式	22
2.1.20.3 修改说明	22
2.1.20.4 修改内容	22
2.1.21 com_android_internal_os_Zygote.cpp	22
2.1.21.1 变更类型	22
2.1.21.2 同步方式	22
2.1.21.3 修改说明	22
2.1.21.4 修改内容	22
2.1.22 UidMap.cpp	23
2.1.22.1 变更类型	23
2.1.22.2 同步方式	23
2.1.22.3 修改说明	23
2.1.22.4 修改内容	23
2.2 packages	23
2.2.1 nearlink.apk	23
2.2.1.1 变更类型	23
2.2.1.2 同步方式	23
2.2.1.3 修改说明	23
2.2.1.4 修改内容	23
2.3 system	24
2.3.1 android_filesystem_config.h	24
2.3.1.1 变更类型	24
2.3.1.2 同步方式	24

2.3.1.3 修改说明	24
2.3.1.4 修改内容	24
2.3.2 init.rc	24
2.3.2.1 变更类型	24
2.3.2.2 同步方式	25
2.3.2.3 修改说明	25
2.3.2.4 修改内容	25
2.3.3 binder	25
2.3.3.1 变更类型	25
2.3.3.2 同步方式	25
2.3.3.3 修改说明	25
2.3.3.4 修改内容	25
2.3.4 sepolicy	27
2.3.4.1 变更类型	27
2.3.4.2 同步方式	27
2.3.4.3 同步方式	27
2.3.4.4 修改说明	27
2.3.4.5 修改内容	27
3 编译	29
3.1 编译	29
3.2 lint 更新	29
4 接口说明	31
4.1 星闪开关	31
4.1.1 接口列表	31
4.1.2 打开星闪	31
4.1.2.1 Framework 层接口	31
4.1.2.2 使用方法	32
4.1.2.3 Demo 实例	32
4.1.3 关闭星闪	33
4.1.3.1 Framework 层接口	33
4.1.3.2 使用方法	33

4.1.3.3 Demo 实例	33
4.1.4 获取星闪状态	34
4.1.4.1 Framework 层接口	34
4.1.4.2 使用方法	34
4.1.4.3 Demo 实例	34
4.2 设备扫描	35
4.2.1 接口列表	35
4.2.2 开始扫描	35
4.2.2.1 Framework 层接口	35
4.2.2.2 使用方法	36
4.2.2.3 Demo 实例	36
4.2.3 停止扫描	36
4.2.3.1 Framework 层接口	36
4.2.3.2 使用方法:	36
4.2.3.3 Demo 实例	37
4.3 设备连接和断连	37
4.3.1 接口列表	37
4.3.2 HID 连接	37
4.3.2.1 Framework 层接口	37
4.3.2.2 使用方法	38
4.3.2.3 Demo 实例	38
4.3.3 HID 断连	39
4.3.3.1 Framework 层接口	39
4.3.3.2 使用方法	39
4.3.3.3 Demo 实例	39
4.3.4 获取已配对设备列表	40
4.3.4.1 Framework 层接口	40
4.3.4.2 使用方法	40
4.3.4.3 Demo 实例	40

1 源码准备

获取 WS73 SDK 软件发布包，解压获取对应 Android 版本的源码以及对应主控的类型的协议栈库文件。

— application	# 用户态二进制文件和示例代码
— bin	# 星闪可执行程序
— lib	# 星闪可执行程序依赖库
— sample	# WS73 BLE/SLE示例代码
— sle_android	# 星闪android协议实现，按支持的Android版本发布
— build	# SDK构建所需配置文件和编译脚本
— config	# WS73 默认配置文件
— driver	# WS73的驱动文件
— android_autoconfig.h	# Android配置文件
— android.config	# Android配置文件
— bsle	# WS73 BLE/SLE驱动文件
— Makefile	# WS73 驱动编译入口
— platform	# WS73 平台文件
— wifi	# WS73 WiF驱动文件
— firmware	# WS73 驱动依赖固件
— e	# USB2.0/UART版本固件
— us	# SDIO/USB版本固件
— include	# API头文件存放目录

```
|— Makefile          # Makefile编译总入口
|— open_source       # 存放开源组件及相关patch
|— output            # 编译时生成的目标文件和中间文件
```

获取 sdk 包中 application/sle_android/对应 Android 版本的源码。

2 源码适配

📖 说明

需要按照 SDK 发布包中的源码进行对比合入，该章节主要介绍星闪安卓各模块代码的功能。

2.1 frameworks

2.2 packages

2.3 system

2.1 frameworks

Android frameworks 层星闪修改，为上层 APP 提供星闪相关接口，使用星闪相关能力。

2.1.1 SystemServiceRegistry.java

2.1.1.1 变更类型

修改

2.1.1.2 同步方式

代码比对合入

2.1.1.3 修改说明

注册 NearlinkManager，将其加入到 SystemServiceRegistry 的缓存中，供外部使用时懒加载

2.1.1.4 修改内容

```
frameworks/base/core/java/android/app/SystemServiceRegistry.java
import android.nearlink.NearlinkManager;
```

```
registerService(Context.NEARLINK_SERVICE, NearlinkManager.class,
    new CachedServiceFetcher<NearlinkManager>() {
        @Override
        public NearlinkManager createService(ContextImpl ctx) {
            return new NearlinkManager(ctx);
        }
    });
```

2.1.2 Context.java

2.1.2.1 变更类型

修改

2.1.2.2 同步方式

代码比对合入

2.1.2.3 修改说明

增加星闪服务名称定义，使用 Context.getSystemService()调用获取 NearlinkManager 时需要传入此关键字

2.1.2.4 修改内容

frameworks/base/core/java/android/content/Context.java

/**

```
 * Use with {@link #getSystemService(String)} to retrieve a
 * {@link android.nearlink.NearlinkManager} for using Nearlink.
 *
 * @see #getSystemService(String)
 */
```

```
public static final String NEARLINK_SERVICE = "nearlink";
```

2.1.3 PackageManager.java

2.1.3.1 变更类型

修改

2.1.3.2 同步方式

代码比对合入

2.1.3.3 修改说明

定义设备是否支持星闪特性的关键字，用于 `getSystemAvailableFeatures` 和 `hasSystemFeature` 时判断系统是否支持星闪的入参。注意：需要在系统特性配置文件（公版 11： `handheld_core_hardware.xml`）中增加支持星闪的 feature

2.1.3.4 修改内容

`frameworks/base/core/java/android/content/pm/PackageManager.java`

`/**`

```
* Feature for {@link #getSystemAvailableFeatures} and
* {@link #hasSystemFeature}: The device is capable of communicating with
* other devices via Nearlink.
*/
@SdkConstant(SdkConstantType.FEATURE)
public static final String FEATURE_NEARLINK = "android.hardware.nearlink";
```

2.1.4 Process.java

2.1.4.1 变更类型

修改

2.1.4.2 同步方式

代码比对合入

2.1.4.3 修改说明

定义星闪进程的用户组

 说明

此处需要保证和系统已有的 UID 不冲突

2.1.4.4 修改内容

`frameworks/base/core/java/android/os/Process.java`

`/**`

```
* Defines the UID/GID for the Nearlink service process.
* @hide
*/
public static final int NEARLINK_UID = 1081;
```

2.1.5 UserManager.java

2.1.5.1 变更类型

修改

2.1.5.2 同步方式

代码比对合入

2.1.5.3 修改说明

UserManager 中增加星闪用户权限控制相关常量定义

2.1.5.4 修改内容

frameworks/base/core/java/android/os/UserManager.java

将 XXX

/**

```
* Specifies if a user is disallowed from configuring nearlink.  
* This does <em>not</em> restrict the user from turning nearlink on or off.  
* The default value is <code>>false</code>.  
* <p>This restriction doesn't prevent the user from using nearlink. For disallowing usage of  
* nearlink completely on the device, use {@link #DISALLOW_NEARLINK}.  
* <p>This restriction has no effect in a managed profile.  
*  
* <p>Key for user restrictions.  
* <p>Type: Boolean  
* @see DevicePolicyManager#addUserRestriction(ComponentName, String)  
* @see DevicePolicyManager#clearUserRestriction(ComponentName, String)  
* @see #getUserRestrictions()  
*/
```

```
public static final String DISALLOW_CONFIG_NEARLINK = "no_config_nearlink";
```

/**

```
* Specifies if nearlink is disallowed on the device.  
*  
* <p> This restriction can only be set by the device owner and the profile owner on the  
* primary user and it applies globally - i.e. it disables nearlink on the entire device.  
* <p>The default value is <code>>false</code>.  
* <p>Key for user restrictions.  
* <p>Type: Boolean  
* @see DevicePolicyManager#addUserRestriction(ComponentName, String)  
* @see DevicePolicyManager#clearUserRestriction(ComponentName, String)  
* @see #getUserRestrictions()  
*/
```

```
public static final String DISALLOW_NEARLINK = "no_nearlink";
```

@StringDef(value = 注解中增加 DISALLOW_CONFIG_NEARLINK 和
DISALLOW_NEARLINK, 定义权限控制方法可使用的常量

@StringDef(value = {

```
    DISALLOW_MODIFY_ACCOUNTS,  
    ...  
    DISALLOW_CONFIG_NEARLINK,  
    DISALLOW_NEARLINK,  
    ....  
    KEY_RESTRICTIONS_PENDING,  
})
```

2.1.6 Settings.java

2.1.6.1 变更类型

修改

2.1.6.2 同步方式

代码比对合入

2.1.6.3 修改说明

增加星闪系统设置相关常量定义

2.1.6.4 修改内容

frameworks/base/core/java/android/provider/Settings.java

import java.util.Locale;

MOVED_TO_SECURE_THEN_GLOBAL 中增加自定义 Key

MOVED_TO_SECURE_THEN_GLOBAL.add(Global.NEARLINK_ON);

MOVED_TO_GLOBAL 中增加自定义 Key

MOVED_TO_GLOBAL.add(Settings.Global.NEARLINK_ON);

定义 Nearlink 系统设置相关常量

/**

* Constant for use in AIRPLANE_MODE_RADIOS to specify Nearlink radio.

*/

public static final String RADIO_NEARLINK = "nearlink";

/**

* An integer representing the Nearlink Class of Device (CoD).

*


```

    * @hide
    */
    public static final String NEARLINK_CLASS_OF_DEVICE = "nearlink_class_of_device";
    /**
     * A Long representing a bitmap of profiles that should be disabled when nearlink starts.
     * See {@link android.nearlink.NearlinkProfile}.
     * @hide
     */
    public static final String NEARLINK_DISABLED_PROFILES = "nearlink_disabled_profiles";
    /**
     * A semi-colon separated list of Nearlink interoperability workarounds.
     * Each entry is a partial Nearlink device address string and an integer representing
     * the feature to be disabled, separated by a comma. The integer must correspond
     * to a interoperability feature as defined in "interop.h" in /system/nl.
     * <p>
     * Example: <br/>
     * "00:11:22,0;01:02:03:04,2"
     * @hide
     */
    public static final String NEARLINK_INTEROPERABILITY_LIST =
"nearlink_interoperability_list";
    /**
     * Settings to allow SLE scans to be enabled even when Nearlink is turned off for
     * connectivity.
     * @hide
     */
    public static final String SLE_SCAN_ALWAYS_AVAILABLE = "sle_scan_always_enabled";
    /**
     * The length in milliseconds of a SLE scan window in a low-power scan mode.
     * @hide
     */
    public static final String SLE_SCAN_LOW_POWER_WINDOW_MS =
"sle_scan_low_power_window_ms";
    /**
     * The length in milliseconds of a SLE scan window in a balanced scan mode.
     * @hide
     */
    public static final String SLE_SCAN_BALANCED_WINDOW_MS =
"sle_scan_balanced_window_ms";
    /**
     * The length in milliseconds of a SLE scan window in a low-latency scan mode.
     * @hide
     */

```

```

public static final String SLE_SCAN_LOW_LATENCY_WINDOW_MS =
    "sle_scan_low_latency_window_ms";
/**
 * The length in milliseconds of a SLE scan interval in a low-power scan mode.
 * @hide
 */
public static final String SLE_SCAN_LOW_POWER_INTERVAL_MS =
    "sle_scan_low_power_interval_ms";
/**
 * The length in milliseconds of a SLE scan interval in a balanced scan mode.
 * @hide
 */
public static final String SLE_SCAN_BALANCED_INTERVAL_MS =
    "sle_scan_balanced_interval_ms";
/**
 * The length in milliseconds of a SLE scan interval in a low-latency scan mode.
 * @hide
 */
public static final String SLE_SCAN_LOW_LATENCY_INTERVAL_MS =
    "sle_scan_low_latency_interval_ms";
/**
 * The mode that SLE scanning clients will be moved to when in the background.
 * @hide
 */
public static final String SLE_SCAN_BACKGROUND_MODE =
"sle_scan_background_mode";
/**{@hide} */
public static final String
    NEARLINK_INPUT_DEVICE_PRIORITY_PREFIX = "nearlink_input_device_priority_";

```

增加函数获取 HIDHost priority Key

```

/**
 * Get the key that retrieves a nearlink Input Device's priority.
 * @hide
 */
public static final String getNearlinkHidHostPriorityKey(String address) {
    return NEARLINK_INPUT_DEVICE_PRIORITY_PREFIX +
address.toUpperCase(Locale.ROOT);
}

```

2.1.7 nearlink

2.1.7.1 变更类型

新增

2.1.7.2 同步方式

目录直接拷贝

2.1.7.3 修改说明

Android frameworks 层新增星闪功能实现

2.1.7.4 修改内容

frameworks/base/core/java/android/nearlink

文件列表：

NearlinkAdapter.java

NearlinkAddress.java

NearlinkAnnounceCallback.java

NearlinkAnnounceCallbackConstants.java

NearlinkAnnounceCallbackWrapper.java

NearlinkAnnounceParam.java

NearlinkAnnouncer.java

NearlinkAnnounceSettings.java

NearlinkAppearance.java

NearlinkAuthInfoEvt.java

NearlinkCommCallback.java

NearlinkConnection.java

NearlinkConnectionCallbackWrapper.java

NearlinkConstant.java

NearlinkDevice.java

NearlinkErrorCode.java

NearlinkHidHost.java

NearlinkManager.java

NearlinkProfile.java

NearlinkPublicData.java

NearlinkSeekCallback.java

NearlinkSeekCallbackConstants.java

NearlinkSeeker.java

NearlinkSeekFilter.java

NearlinkSeekParams.java

```
NearlinkSeekResultInfo.java
NearlinkSsapClient.java
NearlinkSsapClientCallback.java
NearlinkSsapDescriptor.java
NearlinkSsapProperty.java
NearlinkSsapReadByUuid.java
NearlinkSsapServer.java
NearlinkSsapServerCallback.java
NearlinkSsapService.java
NearlinkUtils.java
```

2.1.8 config.xml

2.1.8.1 变更类型

修改

2.1.8.2 同步方式

代码比对合入

2.1.8.3 修改说明

配置是否开启非法的 MAC 地址检测功能和星闪状态保存功能。

2.1.8.4 修改内容

frameworks/base/core/res/res/values/config.xml

配置是否开启非法的 MAC 地址检测功能

```
<!-- Boolean indicating if current platform need do one-time nearlink address re-
validation -->
```

```
<bool name="config_nearlink_address_validation">false</bool>
```

配置是否支持保存星闪的状态

```
<bool name="config_supportNearlinkPersistedState">true</bool>
```

2.1.9 symbols.xml

2.1.9.1 变更类型

修改

2.1.9.2 同步方式

代码比对合入

2.1.9.3 修改说明

增加 config_nearlink_address_validation 和 config_supportNearlinkPersistedState 在 Java 侧的类型定义，**注意：对应 2.1.8 config.xml 中的定义，需要保持一致。**

2.1.9.4 修改内容

frameworks/base/core/res/res/values/symbols.xml

config_nearlink_address_validation（配置是否开启非法的 MAC 地址检测功能）在 Java 侧的类型定义映射

```
<java-symbol type="bool" name="config_nearlink_address_validation" />
```

config_supportNearlinkPersistedState（配置是否支持保存星闪的状态）在 Java 侧的类型定义映射

```
<java-symbol type="bool" name="config_supportNearlinkPersistedState" />
```

2.1.10 enums.proto

2.1.10.1 变更类型

新增

2.1.10.2 同步方式

代码比对合入

2.1.10.3 修改说明

增加星闪连接和断连原因相关枚举常量定义

2.1.10.4 修改内容

frameworks/base/core/proto/android/nearlink/enums.proto

```
syntax = "proto2";
```

```
package android.nearlink;
```

```
option java_outer_classname = "NearlinkProtoEnums";
```

```
option java_multiple_files = true;
```

```
// Nearlink connection states.
```

```
enum ConnectionStateEnum {
```

```
    CONNECTION_STATE_DISCONNECTED = 0;
    CONNECTION_STATE_CONNECTING = 1;
    CONNECTION_STATE_CONNECTED = 2;
    CONNECTION_STATE_DISCONNECTING = 3;
}
// Nearlink Adapter Enable and Disable Reasons
enum EnableDisableReasonEnum {
    ENABLE_DISABLE_REASON_UNSPECIFIED = 0;
    ENABLE_DISABLE_REASON_APPLICATION_REQUEST = 1;
    ENABLE_DISABLE_REASON_AIRPLANE_MODE = 2;
    ENABLE_DISABLE_REASON_DISALLOWED = 3;
    ENABLE_DISABLE_REASON_RESTARTED = 4;
    ENABLE_DISABLE_REASON_START_ERROR = 5;
    ENABLE_DISABLE_REASON_SYSTEM_BOOT = 6;
    ENABLE_DISABLE_REASON_CRASH = 7;
    ENABLE_DISABLE_REASON_USER_SWITCH = 8;
    ENABLE_DISABLE_REASON_RESTORE_USER_SETTING = 9;
}
```

2.1.11 AndroidManifest.xml

2.1.11.1 变更类型

修改

2.1.11.2 同步方式

代码比对合入

2.1.11.3 修改说明

定义星闪在 Android 系统内部的广播类型以及星闪服务需要的系统权限

2.1.11.4 修改内容

frameworks/base/core/res/AndroidManifest.xml

Android 系统星闪广播类型：

<!-- For Nearlink -->

```
<protected-broadcast android:name="android.nearlink.intent.DISCOVERABLE_TIMEOUT" />
<protected-broadcast android:name="android.nearlink.adapter.action.STATE_CHANGED" />
<protected-broadcast
android:name="android.nearlink.adapter.action.ANNOUNCE_MODE_CHANGED" />
<protected-broadcast
```

```

android:name="android.nearlink.adapter.action.DISCOVERY_STARTED" />
    <protected-broadcast
android:name="android.nearlink.adapter.action.DISCOVERY_FINISHED" />
    <protected-broadcast
android:name="android.nearlink.adapter.action.LOCAL_NAME_CHANGED" />
    <protected-broadcast
android:name="android.nearlink.adapter.action.NEARLINK_ADDRESS_CHANGED" />
    <protected-broadcast
android:name="android.nearlink.adapter.action.CONNECTION_STATE_CHANGED" />
    <protected-broadcast android:name="android.nearlink.device.action.UUID" />
    <protected-broadcast android:name="android.nearlink.device.action.MAS_INSTANCE" />
    <protected-broadcast android:name="android.nearlink.device.action.ALIAS_CHANGED" />
    <protected-broadcast android:name="android.nearlink.device.action.FOUND" />
    <protected-broadcast android:name="android.nearlink.device.action.DISAPPEARED" />
    <protected-broadcast android:name="android.nearlink.device.action.CLASS_CHANGED" />
    <protected-broadcast android:name="android.nearlink.device.action.ACL_CONNECTED" />
    <protected-broadcast
android:name="android.nearlink.device.action.ACL_DISCONNECT_REQUESTED" />
    <protected-broadcast android:name="android.nearlink.device.action.ACL_DISCONNECTED" />
    <protected-broadcast android:name="android.nearlink.device.action.NAME_CHANGED" />
    <protected-broadcast android:name="android.nearlink.device.action.NAME_FAILED" />
    <protected-broadcast
android:name="android.nearlink.adapter.action.PAIR_STATE_CHANGED" />
    <protected-broadcast android:name="android.nearlink.device.action.PAIRING_REQUEST" />
    <protected-broadcast android:name="android.nearlink.device.action.PAIRING_CANCEL" />
    <protected-broadcast
android:name="android.nearlink.device.action.CONNECTION_ACCESS_REPLY" />
    <protected-broadcast
android:name="android.nearlink.device.action.CONNECTION_ACCESS_CANCEL" />
    <protected-broadcast
android:name="android.nearlink.device.action.CONNECTION_ACCESS_REQUEST" />
    <protected-broadcast android:name="android.nearlink.device.action.SDP_RECORD" />
    <protected-broadcast
android:name="android.nearlink.device.action.BATTERY_LEVEL_CHANGED" />
    <protected-broadcast android:name="android.nearlink.devicepicker.action.LAUNCH" />
    <protected-broadcast
android:name="android.nearlink.devicepicker.action.DEVICE_SELECTED" />
    <protected-broadcast
android:name="android.nearlink.input.profile.action.CONNECTION_STATE_CHANGED" />

```

星闪相关系统权限声明：

<!-- Allows applications to connect to paired Nearlink devices.

<p>Protection level: normal

```
-->
<permission android:name="android.permission.NEARLINK"
    android:protectionLevel="normal" />
<!-- Allows applications to discover and pair nearlink devices.
    <p>Protection level: normal
-->

<permission android:name="android.permission.NEARLINK_ADMIN"
    android:protectionLevel="normal" />
<!-- @SystemApi Allows applications to pair nearlink devices without user interaction, and to
    allow or disallow phonebook access or message access.
    This is not available to third party applications. -->

<permission android:name="android.permission.NEARLINK_PRIVILEGED"
    android:protectionLevel="signature|privileged" />
```

2.1.12 platform.xml

2.1.12.1 变更类型

修改

2.1.12.2 同步方式

代码比对合入

2.1.12.3 修改说明

增加星闪相关权限组定义

2.1.12.4 修改内容

frameworks/base/data/etc/platform.xml

```
<permission name="android.permission.NEARLINK" >
    <group gid="nearlink" />
    <group gid="wakelock" />
    <group gid="uhid" />
</permission>
```

2.1.13 SettingsLib

2.1.13.1 变更类型

新增

2.1.13.2 同步方式

目录直接拷贝

2.1.13.3 修改说明

SettingsLib 增加星闪相关 Setting 接口实现

2.1.13.4 修改内容

星闪图标文件:

frameworks/base/packages/SettingsLib/res/drawable/ic_settings_nearlink.png

Settings 星闪相关默认英文资源定义:

frameworks/base/packages/SettingsLib/res/values/nl_strings.xml

Settings 星闪相关默认中文资源定义:

frameworks/base/packages/SettingsLib/res/values-zh-rCN/nl_strings.xml

Settingslib 星闪功能实现:

frameworks/base/packages/SettingsLib/src/com/android/settingslib/nearlink

2.1.14 metrics_constants.proto

2.1.14.1 变更类型

修改

2.1.14.2 同步方式

代码比对合入

2.1.14.3 修改说明

定义 UI 跳转时统计数据的常量，**注意：每个常量的值是否已被占有。**

2.1.14.4 修改内容

```
//          | -- --Nearlink start          | -- -----  
  
// OPEN: Settings > Nearlink  
// CATEGORY: SETTINGS  
// OS: 6.0  
NEARLINK = 1751;  
// OPEN: Choose Nearlink device (ex: when sharing)
```

```
// CATEGORY: SETTINGS
// OS: 6.0
NEARLINK_DEVICE_PICKER = 1752;
// OBSOLETE
NEARLINK_DEVICE_PROFILES = 1753;
// OPEN: QS Nearlink tile shown
// ACTION: QS Nearlink tile tapped
// SUBTYPE: 0 is off, 1 is on
// CATEGORY: QUICK_SETTINGS
// OS: 6.0
QS_NEARLINK = 1754;
// OPEN: QS Nearlink detail panel
// CATEGORY: QUICK_SETTINGS
// OS: 6.0
QS_NEARLINK_DETAILS = 1755;
// ACTION: QS Nearlink detail panel > Nearlink toggle
// SUBTYPE: 0 is off, 1 is on
// CATEGORY: QUICK_SETTINGS
// OS: 6.0
QS_NEARLINK_TOGGLE = 1756;
// ACTION: Settings > Nearlink > Toggle
// SUBTYPE: 0 is off, 1 is on
// CATEGORY: SETTINGS
// OS: 6.0
ACTION_NEARLINK_TOGGLE = 1757;
// ACTION: Settings > Nearlink > Overflow > Refresh
// CATEGORY: SETTINGS
// OS: 6.0
ACTION_NEARLINK_SCAN = 1758;
// ACTION: Settings > Nearlink > Overflow > Rename this device
// CATEGORY: SETTINGS
// OS: 6.0
ACTION_NEARLINK_RENAME = 1759;
// ACTION: Settings > Nearlink > Overflow > Show received files
// CATEGORY: SETTINGS
// OS: 6.0
ACTION_NEARLINK_FILES = 1760;
// OPEN: Settings > Nearlink > Rename this device
DIALOG_NEARLINK_RENAME = 1761;
// OPEN: Settings > Nearlink > Paired device profile
DIALOG_NEARLINK_PAIRING_DEVICE_PROFILE = 1762;
// OPEN Settings > Nearlink > Attempt to connect to device that shows dialog
NEARLINK_DIALOG_FRAGMENT = 1763;
```

```

// ACTION: Settings > Connected devices > Nearlink -> Available devices
ACTION_SETTINGS_NEARLINK_PAIR = 1764;
// ACTION: Settings > Connected devices > Nearlink -> Paired devices
ACTION_SETTINGS_NEARLINK_CONNECT = 1765;
// ACTION: Settings > Connected devices > Nearlink -> Connected device
ACTION_SETTINGS_NEARLINK_DISCONNECT = 1766;
// ACTION: Settings > Connected devices > Nearlink -> Error dialog
ACTION_SETTINGS_NEARLINK_CONNECT_ERROR = 1767;
// ACTION: Settings > Connected devices > Nearlink master switch Toggle
ACTION_SETTINGS_MASTER_SWITCH_NEARLINK_TOGGLE = 1768;
// OPEN: Settings->Connected Devices->Nearlink->(click on details link for a paired device)
// CATEGORY: SETTINGS
// OS: O DR
NEARLINK_DEVICE_DETAILS = 1769;
// OPEN: Settings->Connected Devices->Nearlink->(click on details link for a paired device)
// -> Edit name button.
// CATEGORY: SETTINGS
// OS: O DR
DIALOG_NEARLINK_PAIED_DEVICE_RENAME = 1770;
// OPEN: Settings > Connected devices > Nearlink > Pair new device
// CATEGORY: SETTINGS
// OS: O DR
NEARLINK_PAIRING = 1771;
// OPEN: Settings->Connected Devices->Nearlink->(click on details link for a paired device)
// -> Forget button.
// CATEGORY: SETTINGS
// OS: O DR
DIALOG_NEARLINK_PAIED_DEVICE_FORGET = 1772;
// ACTION: Logged when user tries to pair a Nearlink device without name from Settings app
// CATEGORY: SETTINGS
// OS: O MR
ACTION_SETTINGS_NEARLINK_PAIR_DEVICES_WITHOUT_NAMES = 1773;
// OPEN: Settings > Connected Devices > Nearlink
// CATEGORY: SETTINGS
// OS: P
NEARLINK_FRAGMENT = 1774;
// OPEN: Settings -> Developer Options -> Disable Nearlink A2DP hardware
// offload
// CATEGORY: SETTINGS
// OS: P
DIALOG_NEARLINK_DISABLE_A2DP_HW_OFFLOAD = 1775;
//          | -- --Nearlink end          | -- -----

```

2.1.15 ActivityManagerService.java

2.1.15.1 变更类型

修改

2.1.15.2 同步方式

代码比对合入

2.1.15.3 修改说明

将星闪服务进程加到后台运行白名单中

2.1.15.4 修改内容

导入 NearLink UID

```
import static android.os.Process.NEARLINK_UID;
```

mBackgroundAppldWhitelist 中增加 NEARLINK_UID

```
int[] mBackgroundAppldWhitelist = new int[] {  
    NEARLINK_UID,  
    BLUETOOTH_UID  
};
```

isCallerSystem 增加 NEARLINK_UID 的处理

```
final boolean isCallerSystem;  
switch (UserHandle.getAppId(callingUid)) {  
    case ROOT_UID:  
    case SYSTEM_UID:  
    case PHONE_UID:  
    case NEARLINK_UID:  
    case BLUETOOTH_UID:  
    case NFC_UID:  
    case SE_UID:
```

2.1.16 PackageManagerService.java

2.1.16.1 变更类型

修改

2.1.16.2 同步方式

代码比对合入

2.1.16.3 修改说明

为星闪系统服务增加默认 UID

2.1.16.4 修改内容

增加 NEARLINK_UID 定义

```
private static final int NEARLINK_UID = Process.NEARLINK_UID;
```

PackageManagerService 中给 nearlink 增加系统默认 UID

```
mSettings.addSharedUserLPw("android.uid.nearlink", NEARLINK_UID,
```

```
ApplicationInfo.FLAG_SYSTEM, ApplicationInfo.PRIVATE_FLAG_PRIVILEGED);
```

2.1.17 NearlinkManagerService.java

2.1.17.1 变更类型

增加

2.1.17.2 同步方式

文件直接拷贝

2.1.17.3 修改说明

增加星闪 MangerService，用于获取星闪服务

2.1.17.4 修改内容

```
frameworks/base/services/core/java/com/android/server/NearlinkManagerService.java
```

2.1.18 NearlinkService.java

2.1.18.1 变更类型

增加

2.1.18.2 同步方式

文件直接拷贝

2.1.18.3 修改说明

增加星闪系统服务启动入口

2.1.18.4 修改内容

frameworks/base/services/core/java/com/android/server/NearlinkService.java

2.1.19 SystemServer.java

2.1.19.1 变更类型

增加

2.1.19.2 同步方式

文件直接拷贝

2.1.19.3 修改说明

增加启动星闪系统服务。**注意：需要在系统特性配置文件(公版 11: handheld_core_hardware.xml)中增加支持星闪的 feature 。**

2.1.19.4 修改内容

frameworks/base/services/java/com/android/server/SystemServer.java

// Skip Nearlink if we have an emulator kernel

```
if (isEmulator) {
    Slog.i(TAG, "No Nearlink Service (emulator)");
} else if (mFactoryTestMode == FactoryTest.FACTORY_TEST_LOW_LEVEL) {
    Slog.i(TAG, "No Nearlink Service (factory test)");
} else if (!context.getPackageManager().hasSystemFeature
    (PackageManager.FEATURE_NEARLINK)) {
    Slog.i(TAG, "No Nearlink Service (Nearlink Hardware Not Present)");
} else {
    t.traceBegin("StartNearlinkService");
    mSystemServiceManager.startService(NearlinkService.class);
    t.traceEnd();
}
```

2.1.20 Android.bp

2.1.20.1 变更类型

修改

2.1.20.2 同步方式

代码比对合入

2.1.20.3 修改说明

新增 nearlink aidl 的编译目录，framework-non-updatable-sources 中增加 nearlink 的 AIDL “:libnearlink-binder-aidl”

2.1.20.4 修改内容

```
frameworks/base/Android.bp
":libnearlink-binder-aidl",
```

2.1.21 com_android_internal_os_Zygote.cpp

2.1.21.1 变更类型

修改

2.1.21.2 同步方式

代码比对合入

2.1.21.3 修改说明

函数 CalculateCapabilities 增加星闪服务进程的 capabilities 权限设置

2.1.21.4 修改内容

```
frameworks/base/core/jni/com_android_internal_os_Zygote.cpp
if (multiuser_get_app_id(uid) == AID_NEARLINK) {
    capabilities |= (1LL << CAP_WAKE_ALARM);
    capabilities |= (1LL << CAP_NET_ADMIN);
    capabilities |= (1LL << CAP_NET_RAW);
    capabilities |= (1LL << CAP_NET_BIND_SERVICE);
    capabilities |= (1LL << CAP_SYS_NICE);
}
```

2.1.22 UidMap.cpp

2.1.22.1 变更类型

修改

2.1.22.2 同步方式

代码比对合入

2.1.22.3 修改说明

UidMap::sAidToUidMapping 增加星闪服务进程 AID 和 UID 的映射，**注意 UID 需要和 2.1.4 Process.java 中定义的 nearlink uid 保持一致**

2.1.22.4 修改内容

```
frameworks/base/cmds/statsd/src/packages/UidMap.cpp
{"AID_NEARLINK", 1081},
```

2.2 packages

Service 层实现，提供星闪服务能力，通过 JNI 和星闪协议栈进行交互。

2.2.1 nearlink.apk

2.2.1.1 变更类型

新增

2.2.1.2 同步方式

目录直接拷贝，注意：需要从 WS73 SDK 包中获取对应主控类型的星闪协议栈库文件，替换 packages/apps/Nearlink/jni/lib 下的 libsle_host.so。SDK 包中库文件位置：application/lib/XXX，XXX 为对应的主控。

2.2.1.3 修改说明

新增星闪服务层实现

2.2.1.4 修改内容

```
packages/apps/Nearlink:
```


Nearlink

```

├── jni
│   ├── lib
│   └── res
│   └── src
├── Android.bp
├── Android.mk
└── AndroidManifest.xml

```

2.3 system

此目录修改包括 AID 定义、Selinux 的设置、启动脚本设置以及 frameworks 和 services 层之间的 binder 接口定义。

2.3.1 android_filesystem_config.h

2.3.1.1 变更类型

修改

2.3.1.2 同步方式

代码比对合入

2.3.1.3 修改说明

新增新增 AID 定义，**注意：需要和 2.1.4 Process.java 中 public static final int NEARLINK_UID = 1081;定义保持一致，且不能和系统其他 AID 冲突。**

2.3.1.4 修改内容

```

system/core/libcutils/include/private/android_filesystem_config.h
#define AID_NEARLINK 1081          /* nearlink subsystem*/

```

2.3.2 init.rc

2.3.2.1 变更类型

修改

2.3.2.2 同步方式

代码比对合入

2.3.2.3 修改说明

启动过程中创建星闪运行需要的文件目录，并设置文件目录权限。

2.3.2.4 修改内容

system/core/rootdir/init.rc

on post-fs-data 下增加如下脚本：

```
mkdir /data/misc/nearlink 0770 nearlink nearlink
```

2.3.3 binder

2.3.3.1 变更类型

新增

2.3.3.2 同步方式

目录直接拷贝

2.3.3.3 修改说明

新增星闪 binder 接口定义，用于 frameworks 层通过 binder 和 service 交互。

2.3.3.4 修改内容

system/nl

文件如下：

system

```
| - nl
  | - binder
    | - android
      | - nearlink
        | - INearlink.aidl
        | - INearlinkAnnounceCallback.aidl
        | - INearlinkCallback.aidl
```

```
| -- INearlinkConnection.aidl  
  
| -- INearlinkConnectionCallback.aidl  
  
| -- INearlinkDiscovery.aidl  
  
| -- INearlinkEnableCallback.aidl  
  
| -- INearlinkHidHost.aidl  
  
| -- INearlinkManager.aidl  
  
| -- INearlinkManagerCallback.aidl  
  
| -- INearlinkProfileServiceConnection.aidl  
  
| -- INearlinkSeekCallback.aidl  
  
| -- INearlinkSsapClient.aidl  
  
| -- INearlinkSsapClientCallback.aidl  
  
| -- INearlinkSsapServer.aidl  
  
| -- INearlinkSsapServerCallback.aidl  
  
| -- INearlinkStateChangeCallback.aidl  
  
| -- NearlinkAddress.aidl  
  
| -- NearlinkAnnounceParam.aidl  
  
| -- NearlinkAuthInfoEvt.aidl  
  
| -- NearlinkDevice.aidl  
  
| -- NearlinkPublicData.aidl  
  
| -- NearlinkRemoteDevice.aidl  
  
| -- NearlinkSeekFilter.aidl  
  
| -- NearlinkSeekParams.aidl  
  
| -- NearlinkSeekResultInfo.aidl  
  
| -- NearlinkSsapReadByUuid.aidl  
  
| -- NearlinkSsapService.aidl  
  
| -- Android.bp  
  
| -- Android.bp
```

2.3.4 sepolicy

2.3.4.1 变更类型

新增 + 修改

2.3.4.2 同步方式

代码比对合入

2.3.4.3 同步方式

比较合入，**修改关键字**：nearlink

2.3.4.4 修改说明

星闪相关 Selinux 权限定义。

2.3.4.5 修改内容

system/sepolicy

prebuilts

```
| -- api
|   |-- 26.0
|   |-- 27.0
|   |-- 28.0
|   |-- 29.0
|   |-- 30.0
|   |--private
|     |-- compat
|     |-- file_contexts
|     |-- logd.te
|     |-- nearlink.te
|     |-- seapp_contexts
|     |-- service_contexts
|   |--public
|     |-- app.te
```

```
| -- file.te  
| -- installd.te  
| -- nearlink.te  
| -- service.te
```

3 编译

3.1 编译

3.2 lint 更新

3.1 编译

执行编译脚本，进行代码编译。

以公版 AOSP 11 为例，主要步骤如下：

- `source ./build/envsetup.sh`
- `lunch` 选择编译版本 `aosp_arm-userdebug`
- `make -j32 2>&1 | tee bigfish.log`

3.2 lint 更新

Android 11 中，对于系统层的代码进行了更严格的 Lint 检查，编译过程中可能由于代码使用不规范等，在执行 `make update-api` 时上报大量错误，需要根据提示进行 lint 的屏蔽处理。

- 根据编译错误，拷贝对应的 lint 文件
- `cp out/soong/.intermediates/frameworks/base/system-api-stubs-docs-non-updatable/android_common/api_lint_baseline.txt frameworks/base/non-updatable-api/system-lint-baseline.txt`
`cp out/soong/.intermediates/frameworks/base/system-api-stubs-docs/android_common/api_lint_baseline.txt frameworks/base/api/system-lint-baseline.txt`
- 根据报错信息，对应执行提示的命令，更新屏蔽信息
- `make api-stubs-docs-update-current-api`

```
make api-stubs-docs-non-updatable-update-current-api  
make system-api-stubs-docs-non-updatable-update-current-api  
make system-api-stubs-docs-update-current-api  
make update-api
```

4 接口说明

- 4.1 星闪开关
- 4.2 设备扫描
- 4.3 设备连接和断连

4.1 星闪开关

4.1.1 接口列表

接口	功能	所属类
public boolean enable()	打开星闪	android.nearlink.NearlinkAdapter
public boolean disable()	关闭星闪	android.nearlink.NearlinkAdapter
public boolean isEnabled()	判断星闪是否打开	android.nearlink.NearlinkAdapter

4.1.2 打开星闪

4.1.2.1 Framework 层接口

```
/**
 * 启动星闪协议栈
 *
 * @return true 启动指令已下发成功 false 启动指令下发失败
```



```
*/  
  
@RequiresPermission(Manifest.permission.NEARLINK_ADMIN)  
public boolean enable()
```

所属类路径：android.nearlink.NearlinkAdapter

4.1.2.2 使用方法

- 获取 NearlinkManager NearlinkManager nIService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nIAdapter = nIService.getAdapter();
- 调用接口使能星闪 nIAdapter.enable();

4.1.2.3 Demo 实例

```
swOpen.setOnCheckedChangeListener(new  
CompoundButton.OnCheckedChangeListener() {
```

```
    @Override  
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
        if (nIAdapter != null) {  
            boolean enabled = nIAdapter.isEnabled();  
            Log.e(TAG, "[onCheckedChanged] nIAdapter enabled:" + enabled);  
            if (isChecked) {  
                if (!enabled) {  
                    boolean enable = nIAdapter.enable();  
                    Log.e(TAG, "[onCheckedChanged] nIAdapter enable result:" +  
enable);  
  
                    Toast.makeText(MainActivity.this, "打开星闪",  
Toast.LENGTH_SHORT).show();  
                }  
            } else {  
                if (enabled) {  
                    boolean disable = nIAdapter.disable();  
                    Log.e(TAG, "[onCheckedChanged] nIAdapter disable result:" +  
disable);  
  
                    Toast.makeText(MainActivity.this, "关闭星闪",  
Toast.LENGTH_SHORT).show();  
                }  
            }  
        } else {  
            Log.e(TAG, "[onCheckedChanged] nIService null");  
        }  
    }  
}
```

```
    }  
    });
```

4.1.3 关闭星闪

4.1.3.1 Framework 层接口

```
/**  
 * 停止星闪协议栈  
 *  
 * @return true 关闭指令下发成功 false 关闭指令下发失败  
 */  
@RequiresPermission(Manifest.permission.NEARLINK_ADMIN)  
public boolean disable()
```

所属类路径：android.nearlink.NearlinkAdapter

4.1.3.2 使用方法

- 获取 NearlinkManager NearlinkManager nIService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nIAdapter = nIService.getAdapter();
- 调用接口关闭星闪功能 nIAdapter.disable();

4.1.3.3 Demo 实例

```
swOpen.setOnCheckedChangeListener(new  
CompoundButton.OnCheckedChangeListener() {  
  
    @Override  
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
        if (nIAdapter != null) {  
            boolean enabled = nIAdapter.isEnabled();  
            Log.e(TAG, "[onCheckedChanged] nIAdapter enabled:" + enabled);  
            if (isChecked) {  
                if (!enabled) {  
                    boolean enable = nIAdapter.enable();  
                    Log.e(TAG, "[onCheckedChanged] nIAdapter enable result:" +  
enable);  
  
                    Toast.makeText(MainActivity.this, "打开星闪",  
Toast.LENGTH_SHORT).show();  
                }  
            } else {
```

```
        if (enabled) {
            boolean disable = nIAdapter.disable();
            Log.e(TAG, "[onCheckedChanged] nIAdapter disable result:" +
disable);

            Toast.makeText(MainActivity.this, "关闭星闪",
Toast.LENGTH_SHORT).show();
        }
    }
} else {
    Log.e(TAG, "[onCheckedChanged] nIService null");
}
}
});
```

4.1.4 获取星闪状态

4.1.4.1 Framework 层接口

```
/**
 * 检查Nearlink是否处于STATE_ON
 *
 * @return 是 true 否 false
 */
@RequiresPermission(Manifest.permission.NEARLINK)
public boolean isEnabled()
```

所属类路径: android.nearlink.NearlinkAdapter

4.1.4.2 使用方法

- 获取 NearlinkManager NearlinkManager nIService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nIAdapter = nIService.getAdapter();
- 调用接口获取星闪是否使能 nIAdapter.isEnabled();

4.1.4.3 Demo 实例

```
protected void onResume() {
    super.onResume();
    Log.e(TAG, "[onResume] nIAdapter:" + nIAdapter);
    if (nIAdapter != null) {
```

```
swOpen.setChecked(nlAdapter.isEnabled());
String localName = nlAdapter.getName();
Log.e(TAG, "[onResume] localName:" + localName);
if (!TextUtils.isEmpty(localName)) {
    updateDeviceName(localName);
}
String address = nlAdapter.getAddress();
Log.e(TAG, "[onResume] address:" + address);
if (!TextUtils.isEmpty(address)) {
    updateDeviceAddress(address);
}
}
```

4.2 设备扫描

4.2.1 接口列表

接口	功能	所属类
public void startSeek(List filters, NearlinkSeekCallback seekCallback)	启动星闪设备扫描，扫描结果通过 seekCallBack 通知	android.nearlink.NearlinkSeeker
public void stopSeek(NearlinkSeekCallback seekCallback)	停止星闪设备扫描	android.nearlink.NearlinkSeeker

4.2.2 开始扫描

4.2.2.1 Framework 层接口

```
/**
 * 启动星闪设备扫描
 *
 * @param filters      扫描结果过滤器
 * @param seekCallback 扫描结果回调
 */
public void startSeek(List<NearlinkSeekFilter> filters, NearlinkSeekCallback seekCallback) {
```

所属类路径：android.nearlink.NearlinkAdapter

4.2.2.2 使用方法

- 获取 NearlinkManager NearlinkManager nIService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nIAdapter = nIService.getAdapter();
- 获取 NearlinkSeeker NearlinkSeeker nearlinkSeeker = nIAdapter.getNearlinkSeeker();
- 调用接口启动扫描 nearlinkSeeker.startSeek(null, mSeekCallback);

4.2.2.3 Demo 实例

Button btn_startseek = findViewById(R.id.btn_startseek);

```
btn_startseek.setOnClickListener((view) -> {  
    Log.e(TAG, "[OnClick] btn_startseek");  
    if (nIAdapter != null) {  
        NearlinkSeeker nearlinkSeeker = nIAdapter.getNearlinkSeeker();  
        nearlinkSeeker.startSeek(null, mSeekCallback);  
    }  
    Log.e(TAG, "[OnClick] btn_startseek end");  
});
```

4.2.3 停止扫描

4.2.3.1 Framework 层接口

```
/**  
 * 停止星闪设备扫描  
 *  
 * @param seekCallback 与启动seek同一个callback  
 */  
public void stopSeek(NearlinkSeekCallback seekCallback)
```

所属类路径：android.nearlink.NearlinkAdapter

4.2.3.2 使用方法：

- 获取 NearlinkManager NearlinkManager nIService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nIAdapter = nIService.getAdapter();

- 获取 NearlinkSeeker NearlinkSeeker nearlinkSeeker = nlAdapter.getNearlinkSeeker();
- 调用接口停止扫描 nearlinkSeeker.stopSeek(mSeekCallback);

4.2.3.3 Demo 实例

```
Button btn_stopseek = findViewById(R.id.btn_stopseek);

btn_stopseek.setOnClickListener((view) -> {
    Log.e(TAG, "[OnClick] btn_stopseek");
    if (nlAdapter != null) {
        NearlinkSeeker nearlinkSeeker = nlAdapter.getNearlinkSeeker();
        nearlinkSeeker.stopSeek(mSeekCallback);
    }
    Log.e(TAG, "[OnClick] btn_stopseek end");
});
```

4.3 设备连接和断连

4.3.1 接口列表

接口	功能	所属类
public boolean connect(NearlinkDevice device)	HID 连接, 返回值表示接口调用成功, 连接状态需要通过系统广播获取	android.nearlink.NearlinkHidHost
public boolean disconnect(NearlinkDevice device)	断开 HID 连接, H 返回值表示接口调用成功, 连接状态需要通过系统广播获取	android.nearlink.NearlinkHidHost
public Set<NearlinkDevice> getPairedDevices()	获取已配对设备列表	android.nearlink.NearlinkAdapter

4.3.2 HID 连接

4.3.2.1 Framework 层接口

/**

```

* 进行HID设备连接
*
* @param device 待连接的远端HID设备信息
*/
public boolean connect(NearlinkDevice device)

```

所属类路径: android.nearlink. NearlinkHidHost

4.3.2.2 使用方法

- 获取 NearlinkManager NearlinkManager nlService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nlAdapter = nlService.getAdapter();
- 获取 NearlinkHidHost 实例 adapter.getProfileProxy(context, new HidHostServiceListener(), NearlinkProfile.HID_HOST);
- 调用接口进行 HID 设备连接 hidHost.connect(device);

4.3.2.3 Demo 实例

```
NearlinkDevice device = nlAdapter.getRemoteDevice(deviceInfo.address);
```

```

findViewById(R.id.btn_hid_connect).setOnClickListener(v -> {
    Log.e(TAG, "onClick: btn_hid_connect");
    nlAdapter.getProfileProxy(OperationsActivity.this, new
NearlinkProfile.ServiceListener() {
        @Override
        public void onServiceConnected(int profile, NearlinkProfile nearlinkProfile) {
            Log.e(TAG, "onServiceConnected: profile = NearlinkProfile.HID_HOST == "
+ profile);

            hidHost = (NearlinkHidHost) nearlinkProfile;
            boolean connect = hidHost.connect(device);

            String connectResult = connect ? "Hid连接成功" : "Hid连接失败";

            showToast(connectResult);
            UILog.log(TAG, "Hid connect: [connect ret] " + connect);
        }
        @Override
        public void onServiceDisconnected(int profile) {
            Log.e(TAG, "onServiceDisconnected: profile = " + profile);
        }
    }, NearlinkProfile.HID_HOST);
});

```

4.3.3 HID 断连

4.3.3.1 Framework 层接口

```
/**
 * 断开HID设备连接
 *
 * @param device 待断开连接的远端HID设备信息
 */
public boolean disconnect(NearlinkDevice device)
```

所属类路径： android.nearlink.NearlinkHidHost

4.3.3.2 使用方法

- 获取 NearlinkManager NearlinkManager nlService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nlAdapter = nlService.getAdapter();
- 获取 NearlinkHidHost 实例 adapter.getProfileProxy(context, new HidHostServiceListener(), NearlinkProfile.HID_HOST);
- 调用接口断开 HID 设备连接 hidHost.disconnect(device);

4.3.3.3 Demo 实例

```
findViewById(R.id.btn_hid_disconnect).setOnClickListener(v -> {
    Log.e(TAG, "onClick: btn_hid_disconnect");
    if (hidHost != null) {
        boolean disconnect = hidHost.disconnect(device);

        String disconnectResult = disconnect ? "Hid断联成功" : "Hid断联失败";
        showToast(disconnectResult);
        UILog.log(TAG, "Hid disconnect: [disconnect ret] " + disconnect);
    } else {
        showToast("请先连接HID");
    }
});
```


4.3.4 获取已配对设备列表

4.3.4.1 Framework 层接口

```
/**
 * 获取配对设备列表
 *
 * @return 配对设备列表
 */
public List<NearlinkDevice> getPairedDevices()
```

所属类路径：android.nearlink.NearlinkAdapter

4.3.4.2 使用方法

- 获取 NearlinkManager NearlinkManager nlService = (NearlinkManager) MainActivity.this.getSystemService(Context.NEARLINK_SERVICE);
- 获取 NearlinkAdapter NearlinkAdapter nlAdapter = nlService.getAdapter();
- 调用接口获取已配对设备列表
nlAdapter.getNearlinkConnection().getPairedDevices()

4.3.4.3 Demo 实例

```
findViewById(R.id.btnPairDevices).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (nlAdapter != null && nearlinkAddress != null && nearlinkDevice != null) {
            try {
                List<NearlinkDevice> list =
                    nlAdapter.getNearlinkConnection().getPairedDevices();
                UILog.log(TAG, "[getPairedDevices] list:" + list);
            } catch (Exception e) {
                e.printStackTrace();
                Log.e(TAG, "[getPairedDevices] e:" + e.toString());
            }
        }
    }
});
```