ATV-RU Поиск

Интеграция для авторов приложений

Документ для разработчиков Android TV приложений: как сделать так, чтобы контент вашего приложения появлялся в нашей поисковой ленте, и как принимать голосовой запрос «прямо внутрь» приложения.

slavenja.ru · slavenja@ya.ru
TVVoiceSearch · ATV-RU Поиск

Если автор Android TV приложения хочет, чтобы контент появлялся в нашей ленте — следуйте шагам ниже. Документ покрывает два сценария: (а) появление в общей ленте поиска через ContentProvider; (б) приём голосового запроса прямо в открытое приложение.

Часть I. Появление в ленте — ContentProvider

1. Создать ContentProvider

```
content://com.yourapp.search/search_suggest_query/<запрос>?limit=10
```

Возвращает `Cursor` с результатами.

2. searchable.xml в res/xml/

```
<searchable xmlns:android="http://schemas.android.com/apk/res/android"
    android:label="@string/app_name"
    android:searchSuggestAuthority="com.yourapp.search"
    android:searchSuggestIntentAction="android.intent.action.VIEW"
    android:includeInGlobalSearch="true" />
```

3. AndroidManifest.xml

```
<provider
    android:name=".SearchSuggestProvider"
    android:authorities="com.yourapp.search"
    android:exported="true" />

<activity android:name=".SearchActivity" android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.SEARCH" />
    </intent-filter>
    <meta-data
        android:name="android.app.searchable"
        android:resource="@xml/searchable" />
</activity>
```

4. Колонки курсора

Обязательные

Колонка	Что
`SUGGEST_COLUMN_TEXT_1`	Название — «Интерстеллар»
`SUGGEST_COLUMN_INTENT_DATA`	Deep link для открытия — `yourapp://movie/123`

Желательные (делают карточку красивее)

Колонка	Пример
`SUGGEST_COLUMN_TEXT_2`	Фантастика, драма
`suggest_icon_1`	URL вертикального постера

`suggest_result_card_image`	URL широкого постера (16:9)
`suggest_production_year`	2014
`suggest_rating_score`	8.7
`suggest_rating_style`	−1 (по умолчанию 0–10), 5 (звёзды 0–5), 6 (проценты 0–100)
`suggest_duration`	10140000 (мс)
`suggest_intent_data_id`	TMDB ID, например «157336»
`suggest_intent_extra_data`	«movie» или «tv»

Если переданы `suggest_intent_data_id` с TMDB ID и `suggest_intent_extra_data` с типом (`movie`/`tv`) — наше приложение автоматически подтянет постер и рейтинг из TMDB, даже если провайдер их не вернул. Полезно, если у автора приложения нет своих постеров.

5. Минимальный пример

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
                    String[] selectionArgs, String sortOrder) {
    String query = uri.getLastPathSegment();

    MatrixCursor cursor = new MatrixCursor(new String[]{
        "_id",
        SearchManager.SUGGEST_COLUMN_TEXT_1,
        SearchManager.SUGGEST_COLUMN_TEXT_2,
        SearchManager.SUGGEST_COLUMN_INTENT_DATA,
        "suggest_icon_1",
        "suggest_production_year",
        "suggest_rating_score",
        "suggest_duration"
    });

    for (Movie movie : searchMovies(query)) {
        cursor.addRow(new Object[]{
            movie.id,
            movie.title,                    // «Интерстеллар»
            movie.genre,                    // «Фантастика, драма»
            "yourapp://movie/" + movie.id,  // deep link
            movie.posterUrl,                // постер
            movie.year,                     // «2014»
            movie.rating,                   // «8.7»
            movie.durationMs                // 10140000
        });
    }
    return cursor;
}
```

Часть II. Голосовой поиск прямо в активное приложение

Помимо «обычного» сценария (наша лента собирает все источники), у нас есть фича **foreground-redirect**: если включена настройка «приоритет активного приложения» (PRO, по умолчанию выключена) и в момент нажатия кнопки голосового поиска на переднем плане уже работает совместимое приложение — мы не открываем свою ленту, а отправляем распознанный запрос **прямо внутрь** этого приложения, в его собственный поиск.

Для пользователя это работает как «нативный» голосовой поиск приложения. Для автора приложения — бесплатный голосовой ввод без интеграции с распознаванием.

Ключевая часть: ваше приложение должно уметь принять запрос через intent из внешнего источника. Как именно — выбирает автор. Ниже — пять способов, которые мы уже поддерживаем (реализованы в `FG_HANDLERS` в `VoiceSearchActivity.java`); каждый соответствует реальному пакету. Любой из них автор может применить у себя.

Регистрация в нашем коде (один раз на пакет):

```
FG_HANDLERS.put("com.yourapp", (pkg, query) -> {
    // соберите ваш intent тут
});
```

Способ 1. ACTION_VIEW + URL с query (как SmartTube)

Подходит, если у вас уже есть веб-вариант поиска — просто принимайте обычный URL поисковой выдачи.

```
// Наш код
new Intent(Intent.ACTION_VIEW,
    Uri.parse("https://www.youtube.com/results?search_query=" + Uri.encode(q)))
    .setPackage("org.smarttube.stable");
```

Что нужно от автора: intent-filter на `ACTION_VIEW` для `https://your.domain` или собственной схемы; внутри — извлечение `search_query` / `query` / `q` из query string. SmartTube в `IntentExtractor` так и делает.

Способ 2. ACTION_WEB_SEARCH с extra «query» (Яндекс Браузер TV)

Стандартный intent `android.intent.action.WEB_SEARCH`, ключ `"query"`.

```
new Intent("android.intent.action.WEB_SEARCH")
    .setPackage("com.yandex.browser.tv")
    .putExtra("query", q);
```

Способ 3. ACTION_SEARCH + SearchManager.QUERY (OTT Navigator, Filmix)

Самый стандартный путь Android. Если у вас уже есть `<activity android:name=".SearchActivity">` с intent-filter `android.intent.action.SEARCH` и `<meta-data android:name="android.app.searchable">` — этого достаточно. Чтение запроса в Activity:

```
String query = getIntent().getStringExtra(SearchManager.QUERY);
```

С нашей стороны:

```
new Intent(Intent.ACTION_SEARCH)
    .setPackage("studio.scillarium.ottnavigator") // или "net.filmix.filmix"
    .putExtra(SearchManager.QUERY, q);
```

Это самый рекомендуемый способ — стандартный Android, ничего нового изобретать не нужно, та же `SearchActivity` работает и для системного поиска, и для нашего.

Способ 4. Deeplink с собственной схемой (VK Видео TV)

Если у вас есть собственная навигация по deeplink-ам (`yourapp://...`) — добавьте маршрут на поиск:

```
new Intent(Intent.ACTION_VIEW,
    Uri.parse("vkvideo://vk.com/search?query=" + Uri.encode(q)))
    .setPackage("com.vk.tv");
```

В коде VK Видео TV это `BaseLinkRedirect`, который читает `Set { "video", "vk.com" }` как допустимые authority и `getQueryParameter("query")` как сам запрос.

Способ 5. Deeplink через rutube:// (RuTube)

Похоже на способ 4, чуть жёстче — конкретная authority и path:

```
new Intent(Intent.ACTION_VIEW,
    Uri.parse("rutube://rutube.ru/search?query=" + Uri.encode(q)))
    .setPackage("ru.rutube.app");
```

В RuTube `DeeplinkRootNavigator` читает `getQueryParameter("query")` (`SearchIntents.EXTRA_QUERY`); плюсы трактуются как пробелы автоматически Android-ом.

Что выбрать автору

- **Если у вас уже есть SearchActivity для системного поиска** — никаких новых правок, у вас уже всё работает по способу 3.
- **Если ничего нет, начинаете с нуля** — реализуйте способ 3 (ACTION_SEARCH + SearchManager.QUERY). Это минимум 10 строк кода.
- **Если у вас есть собственная навигация по URL** (browser-like приложение, VK/RuTube-стиль) — добавьте маршрут `/search?query=...`, способ 4 или 5.
- **Если поиск у вас веб-страница** — способ 1.

Каждый из способов мы умеем вызывать. Главное условие — чтобы в активном приложении именно эта Activity с поиском подняла запрос. Не пакет вообще, а именно роутер intent-а.

Не подойдут (проверено)

- **Lampa, Lumpa, Prisma, KinoTrend, Okko, Кинопоиск** — `ACTION_SEARCH` либо игнорируется, либо приложение читает только URI path segments (`movie/<id>`), а не текст. Для них работает обычный fallback — поднимаем блок провайдера в начале нашей ленты.
- **Kodi** — нативный движок крэшится на `FLAG_ACTIVITY_CLEAR_TASK`, который Android прикручивает при `setPackage` для VIEW. Нужен другой путь без CLEAR_TASK.
- **SeasonHit TV** — `TvActivity` имеет ACTION_SEARCH-фильтр, но запрос не читается; фильтр существует только для Katniss-подсказок через ContentProvider, а не для внешних intent-ов.

Что нужно от автора, чтобы мы добавили его пакет

- Реализовать одну из схем выше (любую, какая удобна).
- Сообщить нам пакет и схему — мы добавим запись в `FG_HANDLERS`. Это коммит в две строки на нашей стороне.
- На пользователях с включённой опцией «приоритет активного приложения» это начнёт работать сразу после обновления нашего APK.

Частая ошибка — Activity не exported

Если приложение **внутри** реализует поиск-Activity, которая корректно читает `SearchIntents.EXTRA_QUERY` / `SearchManager.QUERY` из intent extras, но **в манифесте** объявлена без `android:exported="true"` и без `intent-filter` — Android отбивает внешние запуски с `SecurityException: not exported from uid ….` Наш FG_HANDLER в этом случае срабатывает в логах как `[FG_REDIRECT] startActivity упал`.

Минимальная правка манифеста, чтобы починить:

```
<activity
    android:name=".ui.tv.activities.ActivityTvSearch"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.SEARCH" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
    <meta-data
        android:name="android.app.searchable"
        android:resource="@xml/searchable" />
</activity>
```

Реальный кейс — LazyMedia Deluxe 3.447: `ActivityTvSearch.onCreate` уже корректно делает `getIntent().getStringExtra(SearchIntents.EXTRA_QUERY)`, но активити не exported → запуск из ATV-RU поверх неё ловит `SecurityException`. Ждём обновления манифеста от автора.

По вопросам: slavenja@ya.ru — поможем с тестированием.
Сайт проекта: slavenja.ru