直播 SDK

直播列表



腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2026 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

直播列表
- 直播列表（Android）
- 直播列表（Web&H5 Vue3）
- 直播列表（iOS)
- 直播列表（Web&H5 React）
- 直播列表（HarmonyOS）
- 直播列表（Flutter）

直播列表
直播列表（Android）

最近更新时间：2026-11-16 17:07:30

TUILiveKit 的直播列表能力基于 LiveListView，以竖向分页信息流的形式展示多个直播间，观众上下滑动即可切换，可快速集成到您的应用中。

> ⓘ 本文档介绍基于 UI 组件 的直播列表接入方式（含完整交互 UI）。若您只需观看单个直播间，请参考 观众观看；若您需通过 Core SDK 自行搭建界面，请参考 观众观看 (Core SDK)。

功能概览

- **直播间信息流**：以竖向分页容器展示多个直播间，每一页都是完整的观看界面。
- **上下滑动切换**：观众上下滑动即可切换到上一个 / 下一个直播间，播放与进房状态由组件自动管理。
- **首屏置顶**：构造时传入的 liveID 会被置顶到列表第一位，保证从分享链接、推送通知进入时首屏看到的就是目标直播间。
- **分页加载**：滑动到列表末尾前自动回调 `onLoadMore`，可对接 LiveListStore 或您自己的业务服务端追加下一页数据。

直播间切换	直播观看





组件说明

`LiveListView` 是一个竖向分页的直播间信息流容器，内部基于 `ViewPager2` + `FragmentStateAdapter` 实现；每一页内部都由 `PlayerView` 承载，因此单个直播间的播放能力（拉流、进房、连麦、弹幕、礼物、点赞）与 观众观看 文档完全一致。

> **说明：**
> LiveListView 内部使用 `ViewPager2` + `FragmentStateAdapter`，因此承载它的 Activity 必须继承自 `FragmentActivity`（推荐 `AppCompatActivity`），否则会在构造时抛出类型强转异常。

快速接入

步骤1：开通服务

参考 开通服务 文档开通**体验版**或**大规模直播版**套餐。

步骤2：代码集成

参考 准备工作 接入 `TUILiveKit`，并完成 AtomicXCore 登录。

> **注意：**

> 创建 LiveListView 之前，请确保已完成登录，否则拉流与进房会失败。

步骤3：接入 LiveListView 滑动切换直播间

在您的直播列表观看页 Activity 中创建 LiveListView，并为其注入直播间列表数据：

Kotlin

```
import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import com.trtc.uikit.livekit.livelist.LiveListView
import io.trtc.tuikit.atomicxcore.api.CompletionHandler
import io.trtc.tuikit.atomicxcore.api.live.LiveListStore

class AudienceLiveListActivity : AppCompatActivity() {

    companion object {
        private const val TAG = "AudienceLiveListActivity"
        private const val FETCH_COUNT = 6
        const val EXTRA_LIVE_ID = "liveID"
    }

    private lateinit var liveListView: LiveListView

    private var isLoading = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val liveID = intent.getStringExtra(EXTRA_LIVE_ID).orEmpty()

        // 1. 创建 LiveListView，构造时传入的 liveID 会作为首屏直播间
        //    重写 onLoadMore，在观众滑动到列表末尾前加载更多数据
        liveListView = object : LiveListView(this, liveID = liveID) {
            override fun onLoadMore() {
                loadMore()
            }
        }
```

```
        // 2. 将 LiveListView 添加到界面
        setContentView(liveListView)

        // 3. 拉取首屏直播间列表并设置给 LiveListView
        refresh()
    }

    private fun refresh() {
        if (isLoading) return
        fetch(isRefresh = true, callback = object : CompletionHandler {
            override fun onSuccess() {

liveListView.setData(LiveListStore.shared().liveState.liveList.value)
                isLoading = false
            }

            override fun onFailure(code: Int, desc: String) {
                Log.e(TAG, "refresh failed, code:$code, desc:$desc")
                isLoading = false
            }
        })
    }

    private fun loadMore() {
        if (isLoading) return
        val cursor =
LiveListStore.shared().liveState.liveListCursor.value
        if (cursor.isEmpty()) return

        val oldSize =
LiveListStore.shared().liveState.liveList.value.size
        fetch(isRefresh = false, callback = object : CompletionHandler {
            override fun onSuccess() {
                val newList =
LiveListStore.shared().liveState.liveList.value
                if (newList.size > oldSize) {
                    liveListView.appendData(newList.subList(oldSize,
newList.size))
                }
                isLoading = false
```

```
            }

            override fun onFailure(code: Int, desc: String) {
                Log.e(TAG, "loadMore failed, code:$code, desc:$desc")
                isLoading = false
            }
        })
    }

    private fun fetch(isRefresh: Boolean, callback: CompletionHandler) {
        isLoading = true
        val cursor =
LiveListStore.shared().liveState.liveListCursor.value
        val fetchCursor = if (isRefresh || cursor.isEmpty()) "" else
cursor
        LiveListStore.shared().fetchLiveList(fetchCursor, FETCH_COUNT,
object : CompletionHandler {
            override fun onSuccess() {
                callback.onSuccess()
            }

            override fun onFailure(code: Int, desc: String) {
                callback.onFailure(code, desc)
            }
        })
    }
}
```

说明：

- LiveListView 只负责渲染，不负责拉取数据。直播间列表可以来自 AtomicXCore 的 LiveListStore，也可以来自您自己的业务服务端，最终通过 `setData` / `appendData` 注入即可。
- 构造时传入的 liveID 会被自动置顶到列表第一位，保证观众首屏看到的就是目标直播间。
- 观众滑动到距离列表末尾还剩 2 个直播间时，onLoadMore 会被回调，请在此时追加下一页数据。
- LiveListView 从窗口移除时会自动停止播放并退出直播间，无需您手动调用 `stopPlay` / `leaveChannel` 。
- 观众处于连麦中时，组件会自动禁用滑动切换，避免误触导致连麦中断。

步骤4：跳转到直播列表页面

在您需要跳转到直播列表观看页时，可参考如下代码示例：

Kotlin

```kotlin
// 跳转到「滑动切换直播间」观看页
fun enterLiveListPage(context: Context, liveID: String) {
    val intent = Intent(context, AudienceLiveListActivity::class.java)
    intent.putExtra(AudienceLiveListActivity.EXTRA_LIVE_ID, liveID)
    context.startActivity(intent)
}
```

组件 API 说明

LiveListView

`LiveListView` 继承自 `FrameLayout`，是一个竖向分页的直播间信息流容器。

方法 / 属性	说明
LiveListView(context, liveID)	构造方法。liveID 会作为首屏直播间（置顶到列表第一位）；不传则展示列表第一条。
setData(liveList: List<LiveInfo>)	设置直播间列表数据。内部会自动按 liveID 去重，并将构造传入的 liveID 置顶。
appendData(liveList: List<LiveInfo>)	在列表尾部追加直播间数据，用于分页加载。已存在的直播间会被更新而非重复插入。
onLoadMore()	可重写的方法，观众滑动到距离列表末尾还剩 2 个直播间时回调，用于加载下一页数据。

直播间列表数据

组件自身不负责拉列表。列表来自 `AtomicXCore` 的 `LiveListStore`：

能力	接口
拉取列表	`LiveListStore.shared().fetchLiveList(cursor, count, CompletionHandler)`
列表数据	`LiveListStore.shared().liveState.liveList.value`

分页游标	`LiveListStore.shared().liveState.liveListCursor.value`（为空表示没有更多数据）

下一步

恭喜您，现在您已经成功集成了直播列表。接下来，您可以实现**主播开播**、**观众观看**等功能，可参考下表：

功能	描述	集成指引
观众观看	使用 PlayerView 观看单个直播间，包含拉流、进房与互动能力。	观众观看
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
礼物系统	实现自定义礼物素材配置、计费系统对接、PK 场景互动等功能。	礼物系统

常见问题

创建 LiveListView 时崩溃，提示 ClassCastException?

LiveListView 内部使用 `ViewPager2` + `FragmentStateAdapter`，需要 `FragmentActivity` 环境。请将承载 LiveListView 的 Activity 改为继承 `AppCompatActivity`（或 `FragmentActivity`）。

调用了 setData 后列表仍然为空?

请先确认：

- 已调用 `setData()` 注入数据，且列表中的 liveID 不为空；
- 已完成 AtomicXCore 登录，`fetchLiveList` 回调为 `onSuccess`；
- 列表数据来自您自己的业务服务端时，已转换为 LiveInfo 对象。

LiveListView 本身不会拉取列表。

观众连麦时还能上下滑动切换直播间吗?

不能。观众进入 CO_GUEST 状态后组件会自动禁用滑动切换，避免误触导致连麦中断；连麦结束后恢复。

版权所有：腾讯云计算（北京）有限责任公司

直播列表（Web&H5 Vue3）

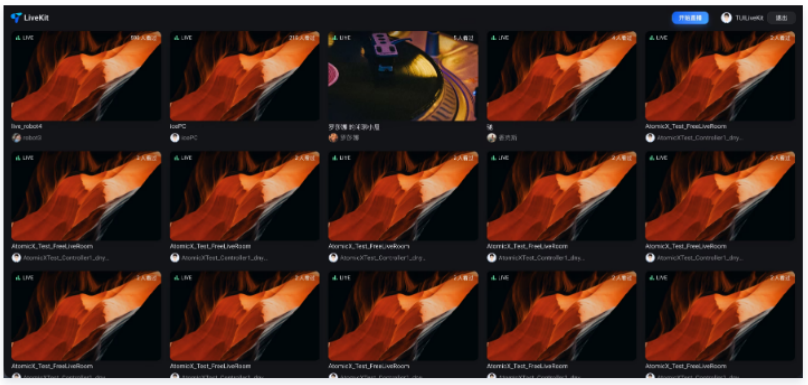
最近更新时间：2026-11-17 17:54:30

TUILiveKit 的直播列表能力基于 LiveList 组件，以竖向分页信息流的形式展示多个直播间，PC 与 H5 自适应布局。可快速集成到您的应用中。

> ⓘ **集成提示：**
> 本文档介绍基于 UI 组件 的直播列表接入方式（含完整交互 UI）。若您只需观看单个直播间，请参考 观众观看；若您需通过 Core SDK 自行搭建界面，请参考 观众观看 (Core SDK)。

功能概览

- **直播间列表**：展示封面、房间名、主播信息与观看数；PC / H5 自适应卡片网格。
- **下拉刷新**：H5 端支持下拉刷新，PC 端滚动到底部加载更多。
- **自动分页**：滑动接近列表底部时自动续拉下一页，游标为空时显示**没有更多**。

Web	H5
	

组件说明

`LiveList` 从 `@tencentcloud/livekit-web-vue3` 包根具名导出，内部渲染 `tuikit-atomicx-vue3` 的列表组件并转发点击事件。组件列表数据由 `LiveListState` 状态驱动；组件本身不进房、不拉流。

快速接入

步骤1：服务

参考 开通服务 文档开通**体验版**或**大规模直播版**套餐。

步骤2：代码集成

参考 准备工作 接入 TUILiveKit，并完成 AtomicXCore 登录。

> ⚠ **注意：**
> 创建 LiveList 组件之前，请确保已完成登录，否则拉取列表数据会失败。

步骤3：接入 LiveList 组件

```
<template>
  <LiveList
    brand-name="LiveKit"
    logo-url="/logo.svg"
    @live-room-click="handleRoomClick"
    @start-live="handleStartLive"
    @logout="handleLogout"
  />
</template>

<script setup lang="ts">
import { useRouter } from 'vue-router';
import { LiveList } from '@tencentcloud/livekit-web-vue3';

const router = useRouter();

function handleRoomClick(liveInfo: { liveId: string }) {
  // 接入 LivePlayer 组件后，跳转到播放页
  router.push({ path: '/live-player', query: { liveId: liveInfo.liveId }
});
}
function handleStartLive() {
  // 接入 LivePusher 组件后，跳转到开播页
  router.push('/live-pusher');
}
function handleLogout() {
  // 调用业务侧登出逻辑，之后跳转登录页
  router.push('/login');
}
</script>
```

组件 API 说明

Props

属性	类型	必填	生效端	说明
`logoUrl`	`string`	否	PC、H5	顶部品牌 logo 地址。
`brandName`	`string`	否	PC、H5	顶栏品牌名称。

Events

事件	参数	生效端	说明
`liveRoomClick`	`LiveInfo`	PC、H5	点击房间卡片后触发，将返回参数 `liveInfo.liveId` 携带跳转到播放页。
`startLive`	无	PC、H5	点击直播列表内的开播按钮后触发，路由跳转到 LivePusher 组件。
`logout`	`() => void`	PC、H5	需自行调用 `logout` 接口退出登录，再卸载组件/路由跳转。

下一步

恭喜您，现在您已经成功集成了直播列表。接下来，您可以实现主播开播、观众观看等功能，可参考下表：

功能	描述	集成指引
观众观看	使用 `LivePlayer` 观看单个直播间，包含拉流、进房与互动能力。	观众观看
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播

直播列表（iOS）

最近更新时间：2026-11-16 17:07:30

TUILiveKit 的直播列表能力基于 LiveListView，以竖向分页信息流的形式展示多个直播间，观众上下滑动即可切换，可快速集成到您的应用中。

> ⓘ 本文档介绍基于 UI 组件 的直播列表接入方式（含完整交互 UI）。若您只需观看单个直播间，请参考 观众观看；若您需通过 Core SDK 自行搭建界面，请参考 观众观看 (Core SDK)。

功能概览

- **直播间信息流**：以竖向分页容器展示多个直播间，每一页都是完整的观看界面。
- **上下滑动切换**：观众上下滑动即可切换到上一个 / 下一个直播间，播放与进房状态由组件自动管理。
- **首屏置顶**：构造时传入的 liveID 会被置顶到列表第一位，保证从分享链接、推送通知进入时首屏看到的就是目标直播间。
- **分页加载**：滑动到列表末尾前自动回调 `onLoadMore`，可对接 LiveListStore 或您自己的业务服务端追加下一页数据。

直播间切换	直播观看
	

组件说明

`LiveListView` 继承自 `UIView`，是一个竖向分页的直播间信息流容器，内部基于 `UICollectionView` 竖向分页实现；每一页内部都由 `PlayerView` 承载，因此单个直播间的播放能力（拉流、进房、连麦、弹幕、礼物、点赞）与 观众观看 文档完全一致。

> **说明：**
> 宿主 App 必须声明 audio 后台模式（`UIBackgroundModes` → `audio`，即 `Target` → `Signing & Capabilities` → `Background Modes` → `Audio, AirPlay, and Picture in Picture`）。未声明时 App 进入后台会被系统挂起，观众端将听不到直播间声音。

快速接入

步骤1：服务

参考 开通服务 文档开通**体验版**或**大规模直播版**套餐。

步骤2：代码集成

参考 准备工作 接入 TUILiveKit，并完成 AtomicXCore 登录。

> **注意：**
> 创建 LiveListView 之前，请确保已完成登录，否则拉流与进房会失败。

步骤3：接入 LiveListView 滑动切换直播间

在您的直播列表观看页 Activity 中创建 LiveListView，并为其注入直播间列表数据：

Kotlin

```
import AtomicXCore
import SnapKit
import TUILiveKit
import UIKit

class AudienceLiveListViewController: UIViewController {

    private let liveListView: LiveListView
    private let pageSize = 6

    private var isLoading = false
```

```
    init(liveID: String = "") {
        // 1. 创建 LiveListView，传入的 liveID 会作为首屏直播间
        liveListView = LiveListView(liveID: liveID)
        super.init(nibName: nil, bundle: nil)
    }

    @available(*, unavailable)
    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func loadView() {
        // 2. 将 LiveListView 作为控制器的根视图
        view = liveListView
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // 3. 设置加载更多回调，观众滑动到列表末尾前会触发
        liveListView.onLoadMore = { [weak self] in
            self?.loadMore()
        }

        // 4. 拉取首屏直播间列表并设置给 LiveListView
        refresh()
    }

    private func refresh() {
        guard !isLoading else { return }
        fetch(cursor: "") { [weak self] in

self?.liveListView.setData(LiveListStore.shared.state.value.liveList)
        }
    }

    private func loadMore() {
        guard !isLoading else { return }
        let cursor = LiveListStore.shared.state.value.liveListCursor
```

```
        guard !cursor.isEmpty else { return }
        let oldCount = LiveListStore.shared.state.value.liveList.count
        fetch(cursor: cursor) { [weak self] in
            let list = LiveListStore.shared.state.value.liveList
            guard list.count > oldCount else { return }
            self?.liveListView.appendData(Array(list[oldCount..
<list.count]))
        }
    }

    private func fetch(cursor: String, completion: @escaping () -> Void)
{
        isLoading = true
        LiveListStore.shared.fetchLiveList(cursor: cursor, count:
pageSize) { [weak self] result in
            DispatchQueue.main.async {
                self?.isLoading = false
                switch result {
                case .success:
                    completion()
                case .failure(let error):
                    print("fetchLiveList failed:\
(error.localizedDescription)")
                }
            }
        }
    }
}
```

说明：

- LiveListView 只负责渲染，不负责拉取数据。直播间列表可以来自 AtomicXCore 的 LiveListStore，也可以来自您自己的业务服务端，最终通过 `setData` / `appendData` 注入即可。
- 构造时传入的 liveID 会被自动置顶到列表第一位，保证观众首屏看到的就是目标直播间。
- 观众滑动到距离列表末尾还剩 2 个直播间时，`onLoadMore` 会被回调，请在此时追加下一页数据。
- LiveListView 从窗口移除时会自动停止播放并退出直播间，无需您手动调用 `stopPlay` / `leaveChannel`。
- 观众处于连麦中时，组件会自动禁用滑动切换，避免误触导致连麦中断。

步骤4：导航路由到直播列表页面

在您需要跳转到直播列表观看页时，可参考如下代码示例：

Swift

```
// 跳转到「滑动切换直播间」观看页
let liveListVC = AudienceLiveListViewController(liveID: "您的直播房间 id")
liveListVC.modalPresentationStyle = .fullScreen
present(liveListVC, animated: false)
```

组件 API 说明

LiveListView

`LiveListView` 继承自 `UIView`，是一个竖向分页的直播间信息流容器。

方法 / 属性	说明
init(liveID: String = "")	构造方法。liveID 会作为首屏直播间（置顶到列表第一位）；不传则展示列表第一条。
setData(_ list: [LiveInfo])	设置直播间列表数据。内部会自动按 liveID 去重，并将构造传入的 liveID 置顶。
appendData(_ list: [LiveInfo])	在列表尾部追加直播间数据，用于分页加载。已存在的直播间会被更新而非重复插入。
onLoadMore	闭包回调，观众滑动到距离列表末尾还剩 2 个直播间时触发，用于加载下一页数据。

直播间列表数据

组件自身不负责拉列表。列表来自 `AtomicXCore` 的 `LiveListStore`：

能力	接口
拉取列表	`LiveListStore.shared.fetchLiveList(cursor:count:completion:)`
列表数据	`LiveListStore.shared.state.value.liveList`
分页游标	`LiveListStore.shared.state.value.liveListCursor`（为空表示没有更多数据）

下一步

恭喜您，现在您已经成功集成了直播列表。接下来，您可以实现**主播开播**、**观众观看**等功能，可参考下表：

功能	描述	集成指引
观众观看	使用 `PlayerView` 观看单个直播间，包含拉流、进房与互动能力。	观众观看
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
礼物系统	实现自定义礼物素材配置、计费系统对接、PK 场景互动等功能。	礼物系统

常见问题

App 退到后台后直播间没有声音?

请检查宿主 App 是否已声明 audio 后台模式：`Target` → `Signing & Capabilities` → `Background Modes` → 勾选 `Audio, AirPlay, and Picture in Picture`。未声明时 iOS 会在 App 进入后台后挂起音频会话。

调用了 setData 后列表仍然为空?

请先确认：

- 已调用 setData() 注入数据，且列表中的 liveID 不为空；
- 已完成 AtomicXCore 登录，`fetchLiveList` 回调为 `.success`；
- 列表数据来自您自己的业务服务端时，已转换为 LiveInfo 对象。

`LiveListView` 本身不会拉取列表。

观众连麦时还能上下滑动切换直播间吗?

不能。观众进入 CO_GUEST 状态后组件会自动禁用滑动切换，避免误触导致连麦中断；连麦结束后恢复。

直播列表（Web&H5 React）

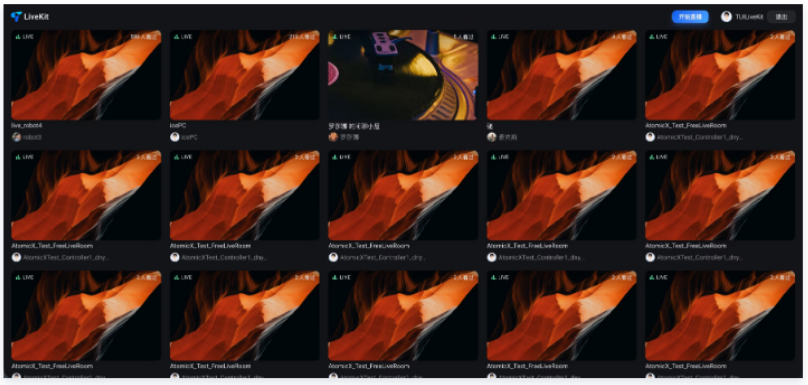
最近更新时间：2026-11-17 17:54:30

TUILiveKit 的直播列表能力基于 LiveList 组件，以竖向分页信息流的形式展示多个直播间，PC 与 H5 自适应布局。可快速集成到您的应用中。

> ⓘ **集成提示：**
> 本文档介绍基于 UI 组件 的直播列表接入方式（含完整交互 UI）。若您只需观看单个直播间，请参考 观众观看；若您需通过 Core SDK 自行搭建界面，请参考 观众观看 (Core SDK)。

功能概览

- **直播间列表**：展示封面、房间名、主播信息与观看数；PC / H5 自适应卡片网格。
- **下拉刷新**：H5 端支持下拉刷新，PC 端滚动到底部加载更多。
- **自动分页**：滑动接近列表底部时自动续拉下一页，游标为空时显示**没有更多**。

Web	H5
	

组件说明

`LiveList` 从 `@tencentcloud/livekit-web-react` 包根具名导出，内部渲染 `tuikit-atomicx-react` 的列表组件并转发点击事件。组件列表数据由 `LiveListState` 状态驱动；组件本身不进房、不拉流。

快速接入

步骤1：服务

参考 开通服务 文档开通**体验版**或**大规模直播版**套餐。

步骤2：代码集成

参考 准备工作 接入 TUILiveKit，并完成 AtomicXCore 登录。

> ⚠ **注意：**
> 创建 LiveList 组件之前，请确保已完成登录，否则拉取列表数据会失败。

步骤3：接入 LiveList 组件

```
import { useNavigate } from 'react-router-dom';
import { loginClient } from 'tuikit-atomicx-react';
import { LiveList } from '@tencentcloud/livekit-web-react';

export function LiveListPage() {
  const navigate = useNavigate();

  function handleRoomClick(liveInfo: { liveId: string }) {
    // 接入 LivePlayer 组件后，跳转到播放页
    navigate({ pathname: '/live-player', search: `?
liveId=${liveInfo.liveId}` });
  }
  function handleStartLive() {
    // 接入 LivePusher 组件后，跳转到开播页
    navigate('/live-pusher');
  }
  async function handleLogout() {
    // 调用业务侧登出逻辑，之后跳转登录页
    await loginClient.logout();
    navigate('/login');
  }

  return (
    <LiveList
      brandName="LiveKit"
      logoUrl="/logo.svg"
      onLiveRoomClick={handleRoomClick}
      onStartLive={handleStartLive}
      onLogout={handleLogout}
    />
  );
}
```

组件 API 说明

Props

属性	类型	必填	生效端	说明
`logoUrl`	`string`	否	PC、H5	顶部品牌 logo 地址。
`brandName`	`string`	否	PC、H5	顶栏品牌名称。

回调

回调	参数	生效端	说明
`onLiveRoomClick`	`LiveInfo`	PC、H5	点击房间卡片后触发，将返回参数 `liveInfo.liveId` 携带跳转到播放页。
`onStartLive`	无	PC、H5	点击直播列表内的开播按钮后触发，路由跳转到 LivePusher 组件。
`onLogout`	`() => void`	PC、H5	需自行调用 `logout` 接口退出登录，再卸载组件/路由跳转。

下一步

恭喜您，现在您已经成功集成了直播列表。接下来，您可以实现主播开播、观众观看等功能，可参考下表：

功能	描述	集成指引
观众观看	使用 `LivePlayer` 观看单个直播间，包含拉流、进房与互动能力。	观众观看
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播

直播列表（HarmonyOS）

最近更新时间：2026-11-13 14:41:00

`TUILiveKit` 的直播列表能力基于 `LiveListView`，以竖向分页信息流的形式展示多个直播间，观众上下滑动即可切换，可快速集成到您的 HarmonyOS 应用中。

功能概览

- **直播间信息流**：以竖向分页容器展示多个直播间，每一页都是完整的观看界面。
- **上下滑动切换**：观众上下滑动即可切换到上一个 / 下一个直播间，播放与进房状态由组件自动管理。
- **首屏置顶**：构造时传入的 liveID 会被置顶到列表第一位，保证从分享链接、推送通知进入时首屏看到的就是目标直播间。
- **分页加载**：滑动到列表末尾前自动回调 onLoadMore，可对接 LiveListStore 或您自己的业务服务端追加下一页数据。

直播间切换	直播观看
	

组件说明

`LiveListView` 是 `ArkUI @Component struct`，统一从 'tuilivekit' 导入，作为竖向分页的直播间信息流容器；每一页内部都由 PlayerView 承载，因此单个直播间的播放能力（拉流、进房、连麦、弹幕、礼物、点赞）与 观众观看 文档完全一致。

> **说明：**
> HarmonyOS 端的 LiveListView 没有 `setData` / `appendData` 方法：列表数据通过 @Prop liveList 由宿主的 @State 状态驱动注入，状态变化时组件自动刷新并按 liveID 合并去重。

快速接入

步骤1：开通服务

参考 开通服务 文档开通**体验版**或**大规模直播版**套餐。

步骤2：代码集成

参考 准备工作 接入 `TUILiveKit`，并完成 AtomicXCore 登录。

> **注意：**
> - 创建 LiveListView 之前，请确保已完成登录，否则拉流与进房会失败。
> - 宿主模块需在 `module.json5` 中声明 `ohos.permission.INTERNET`、`ohos.permission.GET_NETWORK_INFO`、`ohos.permission.CAMERA`、`ohos.permission.MICROPHONE`。组件 HAR 自身不声明这些权限。

步骤3：接入 LiveListView 滑动切换直播间

在您的直播列表观看页中创建 `LiveListView`，并把 `@State` 状态中的直播间列表传给组件：

ArkTS

```
import { LiveInfo, LiveListStore } from '@tencentcloud/atomicxcore'
import { LiveListView } from 'tuilivekit'

@Entry
@Component
struct AudienceLiveListPage {
  liveID: string = ''
  @State private liveList: LiveInfo[] = []
  private readonly liveListStore: LiveListStore = LiveListStore.shared()
  private readonly pageSize: number = 6
```

```
  private isLoading: boolean = false

  aboutToAppear(): void {
    // 1. 拉取首屏直播间列表（cursor 为空表示从头开始）
    this.fetch('')
  }

  private fetch(cursor: string): void {
    if (this.isLoading) {
      return
    }
    this.isLoading = true
    this.liveListStore.fetchLiveList(cursor, this.pageSize).then(() => {
      this.isLoading = false
      // 2. 从 LiveListStore 读取最新列表，更新 @State 驱动组件刷新
      this.liveList = this.liveListStore.liveState.liveList.getAllData()
    }).catch((error: Error) => {
      this.isLoading = false
      console.error(`fetchLiveList failed: ${error.message}`)
    })
  }

  private loadMore(): void {
    const cursor = this.liveListStore.liveState.liveListCursor
    if (cursor.length === 0) {
      return
    }
    this.fetch(cursor)
  }

  build() {
    LiveListView({ liveID: this.liveID, liveList: this.liveList,
onLoadMore: () => this.loadMore() })
  }
}
```

步骤4：跳转到直播列表页面

在直播列表页点击直播间卡片时，可通过 `NavPathStack` 跳转到滑动切换观看页；在 Navigation 的 `navDestination` 构造器中按路由名创建对应页面：

Kotlin

```
@Provide('navigationStack') private navigationStack: NavPathStack = new
NavPathStack()

Navigation(this.navigationStack) {
  WaterfallViewController({
    onPlay: (liveID: string) => this.navigationStack.pushPath({ name:
'player', param: liveID }),
  })
}
.mode(NavigationMode.Stack)
.navDestination(this.navigationDestination)
```

组件 API 说明

LiveListView

`LiveListView` 是 `ArkUI` 组件，作为竖向分页的直播间信息流容器。

属性	说明
liveID: string	首屏直播间 ID；不传则展示列表第一条。
@Prop liveList: LiveInfo[]	直播间列表数据，由宿主的 `@State` 状态驱动；数据变化时组件自动刷新。
onLoadMore?: () => void	观众滑动到距离列表末尾还剩 2 个直播间时触发，用于加载下一页数据。

直播间列表数据

组件自身不负责拉列表。列表来自 `@tencentcloud/atomicxcore` 的 `LiveListStore`：

能力	接口
拉取列表	`LiveListStore.shared().fetchLiveList(cursor, count)`，返回 `Promise<void>`
列表数据	`LiveListStore.shared().liveState.liveList.getAllData()`
分页游标	`LiveListStore.shared().liveState.liveListCursor`（为空表

	示没有更多数据）

下一步

恭喜您，现在您已经成功集成了直播列表。接下来，您可以实现观众观看、主播开播、礼物系统等功能，可参考下表：

功能	描述	集成指引
观众观看	使用 PlayerView 观看单个直播间，包含拉流、进房与互动能力。	观众观看
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播

常见问题

LiveListView 传入了数据但列表为空？

请确认传入的是宿主的 `@State` 变量（例如 `LiveListView({ liveID, liveList: this.liveList, onLoadMore })`），并且在列表拉取成功后真正更新了该变量。HarmonyOS 端没有 `setData` / `appendData` 方法，组件不会自行拉取数据。

首屏没有展示指定的直播间？

请确认已把 `liveID` 传给 LiveListView，并在拉取首屏数据后更新 `@State liveList`；组件会把该 `liveID` 置顶到列表第一位。

观众连麦时还能上下滑动切换直播间吗？

不能。观众进入 `coGuest` 状态后组件会自动禁用滑动切换，避免误触导致连麦中断；连麦结束后恢复。

直播列表（Flutter）

最近更新时间：2026-11-13 14:41:00

`TUILiveKit` 的直播列表能力基于 `LiveListView`，以竖向分页信息流的形式展示多个直播间，观众上下滑动即可切换，可快速集成到您的 `Flutter` 应用中。

> **集成提示：**
> 本文档介绍基于 **UI 组件** 的观众观看方式（含完整交互 UI）。若您需通过 **Core SDK** 自行搭建观众端界面，请参考 观众观看 (Core SDK)。

功能概览

- **直播间信息流**：以竖向分页容器展示多个直播间，每一页都是完整的观看界面。
- **上下滑动切换**：观众上下滑动即可切换到上一个 / 下一个直播间，播放与进房状态由组件自动管理。
- **首屏置顶**：构造时传入的 liveID 会被置顶到列表第一位，保证从分享链接、推送通知进入时首屏看到的就是目标直播间。
- **分页加载**：滑动到列表末尾前自动回调 `onLoadMore`，可对接 LiveListStore 或您自己的业务服务端追加下一页数据。

直播间切换	直播观看





组件说明

`LiveListView` 是一个 `StatelessWidget`，统一从 `package:tencent_live_uikit/tencent_live_uikit.dart` 导入，作为竖向分页的直播间信息流容器；每一页内部都由 PlayerView 承载，因此单个直播间的播放能力（拉流、进房、连麦、弹幕、礼物、点赞）与 观众观看 文档完全一致。
组件采用「Widget + Controller」结构：`LiveListViewController` 负责注入列表数据；在控制器尚未挂载时调用 `setData` / `appendData`，数据会被缓存，组件挂载后自动生效。

> **说明：**
> 宿主 App 需声明摄像头与麦克风权限；iOS 端还需在 Podfile 中打开 `PERMISSION_CAMERA=1`、`PERMISSION_MICROPHONE=1`，否则连麦时权限请求会直接失败。

快速接入

步骤1：开通服务

参考 开通服务 文档开通**体验版**或**大规模直播版**套餐。

步骤2：代码集成

参考 准备工作 接入 TUILiveKit。

> ⚠ **注意：**
> 创建 LiveListView 之前，请确保已完成登录，否则拉流与进房会失败。

步骤3：接入 LiveListView 滑动切换直播间

在您的直播列表观看页中创建 `LiveListView`，并通过 `LiveListViewController` 注入直播间列表数据：

```
import 'package:atomic_x_core/atomicxcore.dart';
import 'package:flutter/material.dart';
import 'package:tencent_live_uikit/tencent_live_uikit.dart';

class AudienceLiveListPage extends StatefulWidget {
  const AudienceLiveListPage({super.key, this.liveID = ''});

  final String liveID;

  @override
  State<AudienceLiveListPage> createState() =>
_AudienceLiveListPageState();
}

class _AudienceLiveListPageState extends State<AudienceLiveListPage> {
  static const int pageSize = 6;

  final LiveListViewController liveListController =
LiveListViewController();
  final LiveListStore liveListStore = LiveListStore.shared;
  bool isLoading = false;
  int lastAppliedCount = 0;

  @override
  void initState() {
    super.initState();
    // 1. 拉取首屏直播间列表
    refresh();
  }

  Future<void> refresh() async {
    if (isLoading) return;
```

```
    isLoading = true;
    final CompletionHandler handler =
        await liveListStore.fetchLiveList(cursor: '', count: pageSize);
    isLoading = false;
    if (!handler.isSuccess) {
      debugPrint('fetchLiveList failed: ${handler.errorCode}');
      return;
    }
    // 2. 全量设置列表数据
    final List<LiveInfo> list = liveListStore.liveState.liveList.value;
    lastAppliedCount = list.length;
    liveListController.setData(list);
  }

  Future<void> loadMore() async {
    if (isLoading) return;
    final String cursor = liveListStore.liveState.liveListCursor.value;
    if (cursor.isEmpty) return;
    isLoading = true;
    final int oldCount = lastAppliedCount;
    final CompletionHandler handler =
        await liveListStore.fetchLiveList(cursor: cursor, count:
pageSize);
    isLoading = false;
    if (!handler.isSuccess) {
      debugPrint('fetchLiveList failed: ${handler.errorCode}');
      return;
    }
    // 3. 只把新增的一页追加到列表尾部
    final List<LiveInfo> list = liveListStore.liveState.liveList.value;
    if (list.length > oldCount) {
      liveListController.appendData(list.sublist(oldCount));
      lastAppliedCount = list.length;
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
```

```
      body: LiveListView(
        controller: liveListController,
        liveID: widget.liveID,
        onLoadMore: loadMore,
      ),
    );
  }
}
```

步骤4：跳转到直播列表页面

在您需要跳转到直播列表观看页时，可参考如下代码示例：

```
import 'package:flutter/material.dart';

import 'audience_live_list_page.dart';

void enterLiveListPage(BuildContext context, String liveID) {
  Navigator.of(context).push(
    MaterialPageRoute<void>(
      builder: (_) => AudienceLiveListPage(liveID: liveID),
      fullscreenDialog: true,
    ),
  );
}
```

组件 API 说明

LiveListView 与 LiveListViewController

`LiveListView` 是一个 `StatelessWidget`，作为竖向分页的直播间信息流容器。

方法 / 属性	说明
LiveListView({controller, liveID, initialLiveID, onLoadMore})	构造方法。liveID 会作为首屏直播间；同时传入 `initialLiveID` 时以 `initialLiveID` 优先。
onLoadMore	观众滑动到距离列表末尾还剩 2 个直播间时触发，用于加载下一页数据。
LiveListViewController.setData(List<	全量设置直播间列表数据；未挂载时数据会被缓存。

LiveInfo> list)	
LiveListViewController.appendData(List<LiveInfo> list)	在列表尾部追加直播间数据，用于分页加载。
LiveListViewController.isAttached	控制器是否已挂载到 LiveListView。

直播间列表数据

组件自身不负责拉列表。列表来自 AtomicXCore 的 LiveListStore：

能力	接口
拉取列表	`LiveListStore.shared.fetchLiveList(cursor: String, count: int)`，返回 `Future<CompletionHandler>`
列表数据	`LiveListStore.shared.liveState.liveList.value`
分页游标	`LiveListStore.shared.liveState.liveListCursor.value`（为空表示没有更多数据）

下一步

恭喜您，现在您已经成功集成了直播列表。接下来，您可以实现观众观看、主播开播、礼物系统等功能，可参考下表：

功能	描述	集成指引
观众观看	使用 PlayerView 观看单个直播间，包含拉流、进房与互动能力。	观众观看
主播开播	主播开播全流程功能，包括开播前的准备和开播后的各种互动。	主播开播
礼物系统	实现自定义礼物素材配置、计费系统对接、PK 场景互动等功能。	礼物系统

常见问题

LiveListView 设置了数据但列表为空?

请确认已把 `LiveListViewController` 同时传给 `LiveListView(controller: ...)`，并调用 setData 注入数据；LiveListView 本身不会拉取列表。数据也可来自您自己的业务服务端。

首屏没有展示指定的直播间?

请确认已把 liveID 传给 LiveListView（或使用 initialLiveID），并在拉取首屏数据后调用 setData；组件会把该 liveID 置顶到列表第一位。

观众连麦时还能上下滑动切换直播间吗?

不能。观众进入 coGuest 状态后组件会自动禁用滑动切换，避免误触导致连麦中断；连麦结束后恢复。

版权所有：腾讯云计算（北京）有限责任公司