蚂蚁科技

接入 HarmonyOS NEXT
使用指南

文档版本：20250728

蚂蚁集团
ANT GROUP

法律声明

蚂蚁集团版权所有©2022，并保留一切权利。

商标声明

免责声明

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	**危险** 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	**警告** 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	**注意** 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	**说明** 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	**单击设置> 网络> 设置网络类型。**
粗体	表示按键、菜单、页面名称等UI元素。	**在结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.接入 HarmonyOS NEXT

1.1. 在控制台创建应用并获取 HarmonyOS NEXT config 配置文件

要使用 mPaaS，您首先需要在 mPaaS 控制台创建应用并下载配置文件。

前置条件

您需要确保拥有开发者账号。账号注册的更多信息，请参见 账号注册（PC端）。

创建 mPaaS 应用

1. 登录 mPaaS 控制台。

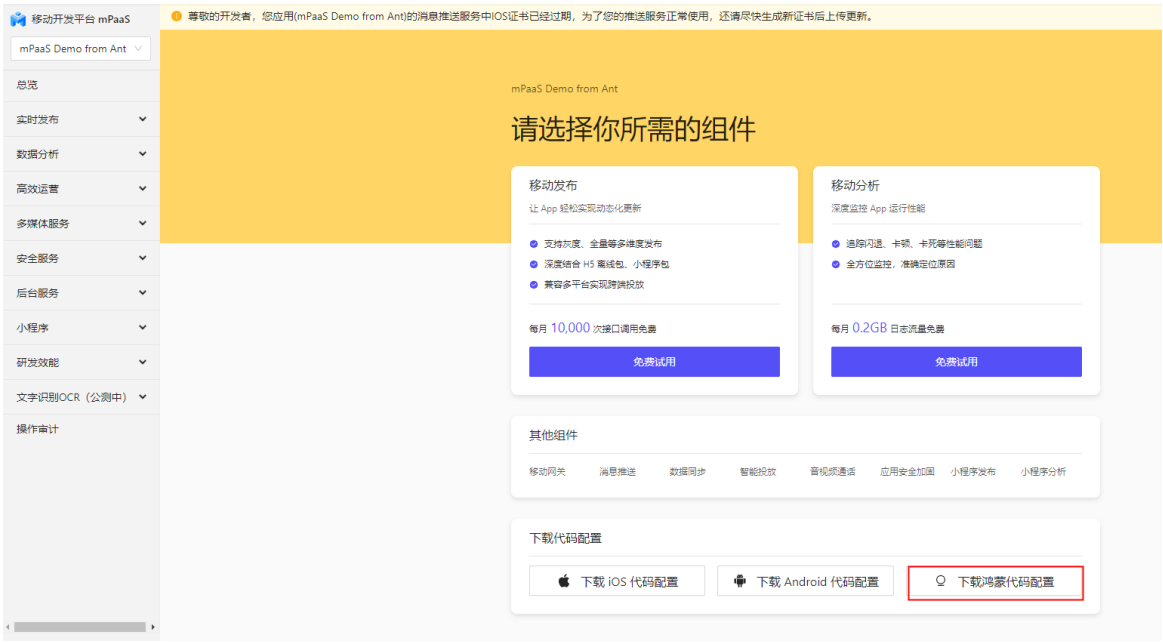
> **说明**
> 如果您在其他平台（如蚂蚁科技开放平台）使用 mPaaS，请登录对应平台的 mPaaS 控制台。

2. 单击 **创建应用** 按钮。

3. 完善应用信息。
   i. 输入应用名称。示例：`mPaaS Demo`。
   ii. 单击 `+` 上传应用图标。您可以忽略此步骤，此时应用将使用默认图标。
4. 单击 **创建** 按钮，完成应用创建。您可以看到应用列表，列表里包含刚才创建的应用。

下载配置文件

1. 在应用列表页，单击应用名称，您将看到以下页面：
   - 左上方展示应用名称，您可以在此切换应用。
   - 页面左侧展示 mPaaS 提供的组件服务列表。



2. 单击 **下载鸿蒙代码配置** ，进入 **代码配置** 页面。
3. 在 **代码配置** 页面，单击 **下载配置**，下载 `.config` 格式的应用配置文件到本地，为后续开发做准备。

文件内容为 `JSON` 格式，示例如下：

```
{
 "absBase64Code":"",
 "appId":"570DA89281533",
 "appKey":"570DA89281533_HARMONY",
 "base64Code":"",
 "packageName":"com.mpaas.demo",
 "rootPath":"mpaas/harmony/570DA89281533-shaojian",
 "v6Base64Code":"",
 "workspaceId":"shaojian",
 "syncport":"443",
 "syncserver":"msync.mpaas.cn-hangzhou.aliyuncs.com",
 "pushPort":"443",
 "pushGW":"push.mpaas.cn-hangzhou.aliyuncs.com",
 "rpcGW":"https://mgw.mpaas.cn-hangzhou.aliyuncs.com/mgw.htm",
 "logGW":"https://mdap.mpaas.cn-hangzhou.aliyuncs.com",
 "mpaasapi":"https://mpaasapi.mpaas.cn-hangzhou.aliyuncs.com/mgw.htm",
 "mrtcserver":"wss://mrtc.mpaas.cn-hangzhou.aliyuncs.com/ws",
 "mdc":"https://mdc.mpaas.cn-hangzhou.aliyuncs.com",
 "mpaasConfigVersion":"V_1.0",
 "mpaasConfigEnv":"ONEX_CLOUD",
 "mpaasConfigPluginExpired":"",

"mpaasConfigLicense":"rycSP0m9EH7shIngOpuN4xI17G07vpOrpNF+1Lve5xQxBgubiDxHPbon0v5Qr7STX
hfOSA893sTbkdINxS7mzbAZF/QeqNyuyUR72Hx1ylKep0Pw4cROX3FXyhximn9kziQCT+f/NfUKQw2aI3IWNnnZ
l/GVGGBwPG1YHywfyVEWRGx/uRc8RTzAq7VLEhduyktgNoLK1DhQiWbQKREeeITz8PvpnrzLGWPS+mM9yrmv1/Y
OA15OdCSr9Zo9EmKM52yo4hxvD4dS0GfiuTwLjLBGCiI3lkGIGuOOUnumMGKfpqSRjav4lLAzmqsGS7yzwo7gvy
2PaSSmqw=="
}
```


1.2. 接入 mPaaS 能力

本文介绍如何基于 ohpmrc 快速接入 mPaaS。

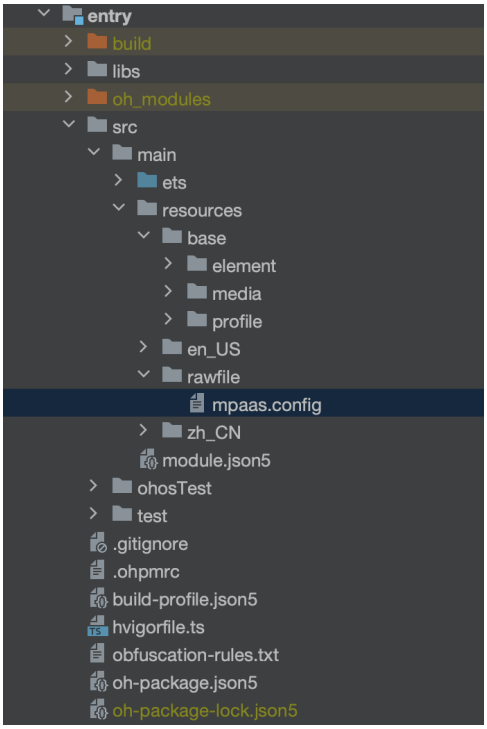
前置条件

- 已安装 HarmonyOS NEXT 最新版开发环境，支持 API 12 以上版本。
- 已有鸿蒙 3.0.0.22 以上版本真机或模拟器，模拟器使用场景请参考 鸿蒙模拟器。
- 已在控制台创建应用，并下载了 HarmonyOS NEXT 版本 `.config` 配置文件。更多信息，参见 在控制台创建应用并获取 HarmonyOS NEXT config 配置文件。

> **说明**
>
> 对于已经接入 beta 版本的客户，以下步骤可以帮助您实现平滑升级。

接入步骤

1. 将 `.config` 配置文件重命名为 `mpaas.config` 并拷贝到 App 主工程的 `entry` 的 `resource` 目录下的 `rawfile` 中。



2. 安装 mppm 插件。

   i. 安装和配置 Node.js 环境。

   升级 mppm 插件之前请确保已正确安装和配置 Node.js 环境。Node.js 版本不低于 v18，可使用命令 `node -v` 查看 Node.js 版本。更多关于使用 Node.js 的信息请参见 Node.js 官网。

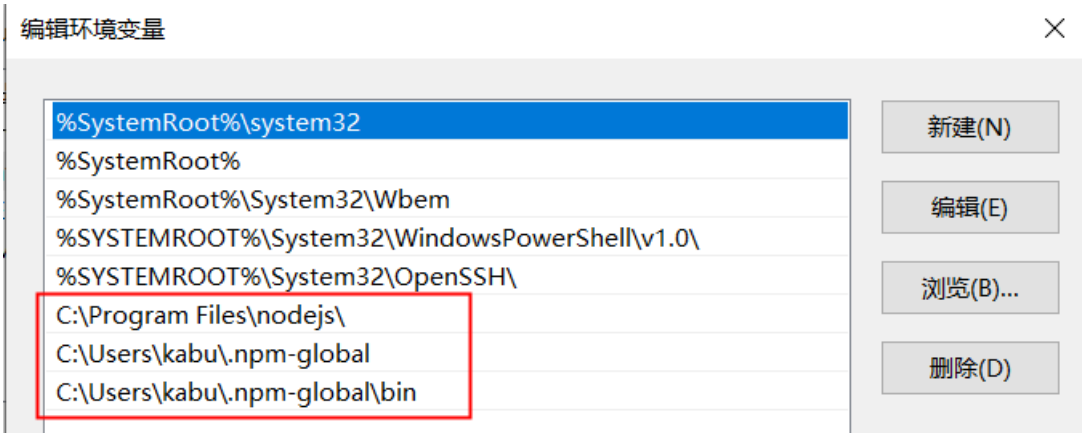
   ii. 安装 mppm。

   在命令行执行以下命令即可安装。

   > **说明**
   >
   > - 更多关于 npm 的使用请参见 npm 主页。
   > - 最新插件版本请参考 mPaaS 插件的 mppm 发布说明。

```
npm install @alipay-inc/oh-mpaas-cli -g
```

iii. Windows 客户还需配置 node 相关环境变量包括 `npm-global` 、 `npm-global/bin` ，如下图所示。可以使用 `npm config get prefix` 命令来查看环境变量，通常会返回一个路径如： `C:\Users\<用户名>\.npm-global` 。



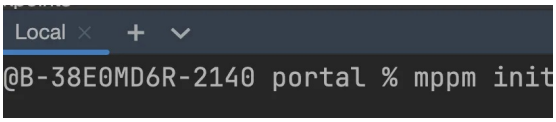
iv. 安装完成之后，通过 `mppm -v` 命令检查当前安装版本。

v. 将 `.config` 配置文件重命名为 `mpaas.config` 并拷贝到 App 主工程的 `entry` 的 `resource` 目录下的 `rawfile` 中。

3. 初始化工程。

使用 `mppm init` 命令初始化工程，该过程会为您配置 mPaaS 鸿蒙版的一些配置。安装 SDK 需在 DevEco Studio terminal 中执行。

i. 在 DevEco Studio terminal 中打开主工程目录执行 `mppm init` 命令。如图

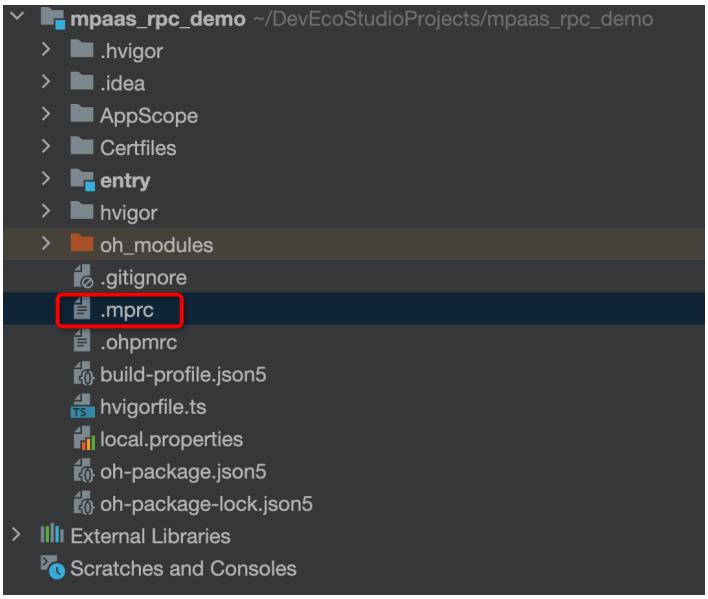


ii. 选择基线。

选择 10.2.3 基线，按 **回车键** 继续。

```
? Please select a baseline version of mPaaS SDK: (Use arrow keys)
❯ Baseline Name: 10.2.3  Latest Version: 3
```

该步骤完成后，mppm 会在工程根目录下创建 `.mprc` 文件用于描述基线信息，如图所示。




```
{
  "version": 12,
  "baseline": "10.2.3"
}
```

iii. 选择需安装的组件。（对于已经安装过组件，会自动选中）

通过 **方向键** 切换光标，通过 **空格键** 选择/取消 ，按下 **回车键** 继续执行下面内容。

```
? Please select modules you want to install of mPaaS SDK: (Press <space> to select, <a> to toggle all, <i> to invert selection, and <enter> to proceed)
 ◉ H5容器
 ◉ 框架
 ◉ 移动网关
❯◯ 移动同步
 ◯ 国密
 ◯ 移动推送
 ◉ 升级
(Move up and down to reveal more choices)
```

> **重要**
>
> 基于鸿蒙新版本命名规范要求，之前对接过 **数据同步** 的客户需要将原基线 `@mpaas/sync-service` 更新为 `@mpaas/sync_service` 。

iv. 如果 Windows 用户碰到以下报错：

```
Failed to clean dependencies in root dir
start install
 Installing dependency for root project
Failed to install dependencies in root project status: null error: Error: spawnSync C:\Users\kabu\Documents\mps2\mps\ ENOENT
Failed to install dependencies in root project
 Installing dependency for module: entry
Failed to install dependencies in module: entry status: null error: Error: spawnSync C:\Users\kabu\Documents\mps2\mps\entry ENOENT
Failed to install dependencies in module: entry
 Installing dependency for module: push
Failed to install dependencies in module: push status: null error: Error: spawnSync C:\Users\kabu\Documents\mps2\mps\push ENOENT
```

可以手动执行 `ohpm clean` 以及 `ohpm install` 来拉取 SDK。

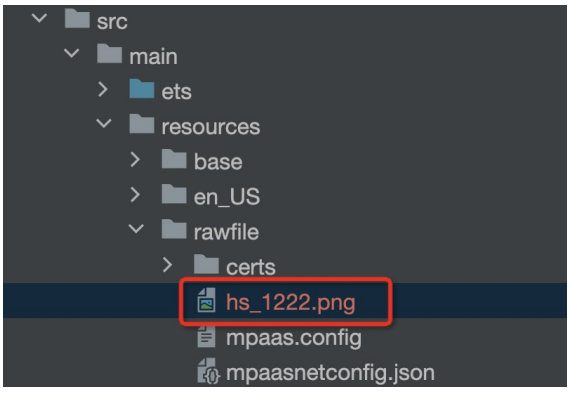
v. 当前 mppm 仅支持在自动将依赖安装到 `entry` 目录下，后续请根据发布日志来升级 mppm。完成之后，相关依赖会放在 `entry module` 下的 `oh-pakacage.json5` 中。

```
"dependencies": {
    "@mpaas/rpc": "1.0.240904203320",
    "@mpaas/framework": "1.0.240722195448",
    "@mpaas/sync_service": "1.0.240911193552",
    "@mpaas/gm": "1.0.240702194534",
    "@mpaas/push": "1.0.240724124844",
    "@mpaas/upgrade": "1.0.240909120426",
    "@mpaas/safekeyboard": "1.0.240829094956",
    "@mpaas/scanapp": "1.0.240722200423",
    "@mpaas/datacenter": "1.0.240909151445",
    "@mpaas/udid": "1.0.240722105823",
    "@mpaas/mpsecurestorage": "1.0.240806155218",
    "@mpaas/configservice": "1.0.240726163920",
    "@mpaas/hriver": "1.0.240912170641"
},
```

4. 获取安全图片。

获取安全图片首先需要在 `rawfile` 下配置 `mpaas.config` 文件。

执行 `mppm fetch-image` 命令，通过 `--finger` 和 `--secret` 选项以配置签名和 `appsecret`，其中 `finger` 为 App 签名的指纹，`secret` 为 mPaaS 控制台中 `appsecret` 的值，若未配置这两个选项会导致拉取失败。拉取成功后，图片会放在 `entry` 下的 `rawfile` 目录中。



选项

- `--finger` 配置 `fingerprint`。
- `--secret` 配置 `appsecret`。

示例

```
mppm fetch-image --finger xxxxxx --secret xxxx
```

获取 finger pirnt 的三种方式

方法一：通过证书获取

1. 将 `.p7b` 中的 `distribution-certificate` 字段单独拷贝出来命名为 `xxx.cer`，其中 `\n` 字符需替换为 **换行**。

> **重要**
>
> `.p7b` 是开发者在申请这个证书时发放的。

2. 利用 keytool 工具（DevEco Studio\jbr\bin\keytool.exe）打印对应的证书的指纹 `keytool -printcert -file xxx.cer` 。

方法二：通过代码获取

```
import bundleManager from '@ohos.bundle.bundleManager';

let info =
bundleManager.getBundleInfoForSelfSync(bundleManager.BundleFlag.GET_BUNDLE_INFO_WITH_SIGN
URE_INFO);
let finger = info.signatureInfo.fingerprint;//该值为fingerpirnt
```

方法三：通过 bm 命令获取 （需手机中已安装该 hap）

```
hdc shell
bm dump -n bundleName | grep fing
```

1.3. 初始化 mPaaS

引入依赖

1. 在项目的 `.ohpmrc` 中添加如下仓库。

```
@mpaas:registry=https://mpaas-ohpm.oss-cn-hangzhou.aliyuncs.com/meta
```

2. 参考接入文档中 `oh-package.json5` 中添加 `"@mpaas/framework":"0.0.2"` 安装框架依赖。
3. 参考接入文档中 `oh-package.json5` 中添加 `"@mpaas/cpp-shared":"1.0.0"` 安装 `libc++_shared.so` 依赖，如果您引入了第三方其他依赖包含了此二进制文件，请不要重复安装。

下载配置

在 mPaaS AppCenter 控制台上，下载 HarmonyOS NEXT 相关的配置文件，改名为 `mpaas.config` 并放入 `entry` 工程的 `rawfile` 中。

初始化与使用

参看 鸿蒙官方文档 AbilityStage 组件容器 手动新建一个 ArkTs 文件 `EntryAbilityStage`。

在 `AbilityStage` 的 `onCreate` 回调初始化框架。

```
import AbilityStage from '@ohos.app.ability.AbilityStage';
import { MPFramework } from '@mpaas/framework';

export default class EntryAbilityStage extends AbilityStage {
  async onCreate() {
    const app = this.context;
    MPFramework.create(app);
    const instance: MPFramework = MPFramework.instance;
    const ctx: Context = instance.context
    // 进行其他组件的初始化
  }
}
```

切记要在 `module.json5` 中注册该 `AbilityStage` 。

```
{
  "module": {
    "name": "entry",
    "type": "entry",
    "description": "$string:module_desc",
    "mainElement": "EntryAbility",
    "srcEntry": "./ets/EntryAbilityStage.ets",
  }
}
```

> **说明**
> `srcEntry` 对应的值以实际路径为准。

相关 API

- **获取 udid**：`MPFramework.instance.udid`

> **重要**
> 该接口为异步接口，调用时添加 await。

- **设置/获取 userId**：`MPFramework.instance.userId`
- **设置/获取 appSecret**：`MPFramework.instance.appSecret`

1.4. 使用 mppm 工具

mppm 是 mPaaS 提供的 SDK 管理工具，可用于清理依赖缓存、安装依赖、生成必要初始化代码等功能。

安装 mppm 工具

```
npm install @alipay-inc/oh-mpaas-cli -g
```

查看工具版本

```
mppm --version
```

目前工具版本为 `v2.0.0` 。

清除鸿蒙工程缓存

```
mppm clean
```

此步骤会清除 `hvigor` 缓存和 `oh_modules` 下的缓存，执行完成后需要重新执行 `sync` 和 `ohpm install` 操作。

安装鸿蒙依赖

```
mppm install
```

协助用户执行 `ohpm install` ，会为每一个 module 都执行这个步骤。

生成安全图片

```
mppm fetch-image --finger xxxxxx --secret xxxx
```

`--finger` 和 `--secret` 选项用来配置签名和 `appsecret` ，如果配置这两个选项会导致生成图片失败。拉取信息成功后，图片会放在 `entry` 下的 `rawfile` 目录中。

选项

- `--finger` 配置 `fingerprint`
- `--secret` 配置 `appsecret`
- `--pack` 配置 `包名` （非必须），默认情况下会从工程中的 `app.json5` 文件中读取
- `--appid` 配置 `appid` （非必须），默认情况下会从mpaas.config文件中读取
- `--workSpaceId` 配置 `workSpaceId` （非必须），默认情况下会从mpaas.config文件中读取

升级基线

```
mppm upgrade
```

升级当前基线到最新版本，升级中会询问是否安装/卸载 SDK，详情请参考 初始化工程。

安装组件

```
mppm sdk
```

安装基线 SDK，适用于已经完成 mPaaS 初始化的操作。切换到新基线的最新版本，中间会询问是否安装/卸载 SDK，详情请参考 初始化工程。

选项

`--custom` 使用定制基线，后面跟上定制基线的基线名。

示例

`mppm sdk --custom 10.2.3.isec`

同步基线

```
mppm sync
```

同步 SDK，适用于手动修改基线版本。`mppm init` 之后，会自动生成 `.mprc` 配置文件。

```
{
  "version": 3,
  "baseline": "10.2.3.test"
}
```

修改该文件后，通过 `mppm sync` 可以刷新工程中到依赖为 `.mprc` 中对应版本的基线依赖，此过程不会询问安装 SDK。

1.5. HarmonyOS NEXT 适配说明

1.5.1. 推送功能更新适配

背景

由于鸿蒙的元能力与窗口组件新增了安全机制（应用退出后台后，不允许执行 Ability 之间的拉起），因此当前 App 离线状态下点击通知栏之后，无法直接跳转到应用内页面。具体组件启动规则，参考鸿蒙组件启动规则。

> **说明**
> 此次更新删除了 mPaaS 中间层，修复了上述问题。

客户端更新与接入

鸿蒙客户端基线升级之后，用户只需直接配置需要跳转的目标页面即可。例如，需要点击通知栏之后跳转到 `PushLandingAbility` 页面，需在 `module.json5` 文件中进行如下配置。

```
{
  "name": "PushLandingAbility",
  "srcEntry": "./ets/pushability/PushLandingAbility.ets",
  "description": "$string:EntryAbility_desc",
  "icon": "$media:icon",
  "label": "$string:LandingAbility_label",
  "startWindowIcon": "$media:startIcon",
  "startWindowBackground": "$color:start_window_background",
  "exported": true,
  "skills": [
    {
      "actions": [""],
      "uris": [
        {
          "scheme": "jump",
          "host": "com.mpaas.harmony.push",
          "path": "landing"
        }
      ]
    }
  ]
},
```

> **重要**
> mPaaS 通过消息中的 URI 匹配具体的页面，actions 需要设置为空（不能缺少该字段）。

在跳转的目标页面中，用户可以使用 mPaaS 提供的工具类 PushMsgHandler 解析消息通知中包含的数据（使用时，需要确保服务端已经升级），具体代码参考如下：

```
import { MPPushMsg,PushMsgHandler,msgOutput } from '@mpaas/push';

export class PushLandingAbility extends UIAbility {
  private TAG: string = "PushLandingAbility"

  async onCreate(want: Want, launchParam: AbilityConstant.LaunchParam): Promise<void> {
    super.onCreate(want, launchParam)

    let msg: msgOutput | undefined = PushMsgHandler.parsePushMsg(want)
    let msg_id: string | undefined = msg?.msg_id
    let msg_data: MPPushMsg | undefined = msg?.msg_data
  }

  ......
}
```

- `msg_id` 对应消息 ID。
- `msg_data` 对应具体的消息数据。

另外， 升级后的 SDK 在 NcAbility 内部做了配置页面的销毁动作，因此 **开发者需要删除在配置页面中 onForeground 阶段的 this.context.terminateSelf 代码**，配置示例如下：

```
export default class MpaasBridgeMsgAbility extends MpaasNcAbility {
  static tag: string = "MpaasBridgeMsgAbility"
  onCreate(want: Want, launchParam: AbilityConstant.LaunchParam): void {
    hilog.info(0x0000, MpaasBridgeMsgAbility.tag, "start MpaasBridgeMsgAbility onCreate
")
    super.onCreate(want, launchParam)
  }

  onForeground() {
    console.log('MpaasBridgeMsgAbility onBackground');
  }

}
```

> **重要**
>
> **风险点提示 1：** 如果升级客户端组件之后，没有删除上述销毁的代码，那么跳转动作会失败。
>
> **风险点提示 2：** 升级客户端组件之后，如果没有升级服务端，保持现有的消息解析方式不变，不要使用 PushMsgHandler 解析消息，否则会解析失败。

服务端更新内容

服务端通过 pushVersion 字段的值（在客户端调用初始化上报接口时会上报该字段）来判断和适配不同版本的鸿蒙推送 SDK，升级前的 pushVersion 字段值为 `2.5.0`，升级后的 pushVersion 字段值为 `3.0.0`。

针对 **升级前的 SDK**，服务端发送的推送消息结构体如下：

```
{
  "pushOptions": {
    "testMessage": true,
    "biTag": "0#11#1.0.0#c_0_0_8314646940c9f7ceb2e0Grm&49bd2fa97769abdf",
    "collapseKey": -1,
    "ttl": 180
  },
  "payload": {
    "notification": {
      "badge": {
        "addNum": 1
      },
      "notifyId": 1568946632,
      "category": "MARKETING",
      "title": "推送测试-noUrl000",
      "body": "推送测试-noUrl000",
      "clickAction": {
        "actionType": 1,
        "data": {
          "com_harmony_push": {
            "bData": "
{\"harmonyos_badge_add_num\":\"1\",\"pushStyle\":0,\"silent\":false,\"notifyType\":\"noti
\",\"action\":\"0\",\"id\":\"console_1709535017366\",\"params\":
{\"ALIMpsChannel\":\"HARMONYOS\"},\"url\":\"https://www.baidu.com\",\"test_msg\":\"1\"}",

            "k": "c_0_0_8314646940c9f7ceb2e0Grm"
          }
        },
        "action": "com.mpaas.harmony.push",
        "uri": "mpaasscheme://com.mpaas.harmony.push/longlink"
      }
    }
  },
  "target": {
    "token": ["MAMzLgNB-
HcFV6YAstOIywAAAGQAAAAAAAHmQeccAFJz604RQSgRll3LPikX2SogGoJtfDDs4lG2Vaz5MwV18jA9UsYKH4oNbT
IDVIjY9SzGrm"]
  }
}
```

针对 升级后的 SDK，服务端发送的推送消息结构体如下：

```
{
  "pushOptions": {
    "testMessage": true,
    "biTag": "0#11#1.0.0#c_0_0_8315cb56301994ccb601Grm&104aca1b4f59abd6",
    "collapseKey": -1,
    "ttl": 180
  },
  "payload": {
    "notification": {
      "badge": {
        "addNum": 1
      },
      "notifyId": 1568946632,
      "category": "MARKETING",
      "title": "推送测试-noUrl000",
      "body": "推送测试-noUrl000",
      "clickAction": {
        "actionType": 1,
        "data": {
          "com_harmony_push": {
            "bData": "
{\"harmonyos_badge_add_num\":\"1\",\"pushStyle\":0,\"silent\":false,\"notifyType\":\"noti
\",\"action\":\"1\",\"id\":\"console_1709535017366\",\"params\":
{\"ALIMpsChannel\":\"HARMONYOS\"},\"url\":\"jump://com.mpaas.harmony.push/landing\",\"tes
msg\":\"1\"}",
            "k": "c_0_0_8315cb56301994ccb601Grm"
          }
        },
        "action": "",
        "uri": "jump://com.mpaas.harmony.push/landing"
      }
    }
  },
  "target": {
    "token": ["MAMzLgNB-
HcFV6YAstOIywAAAGQAAAAAAAHmQeccAFJz604RQSgRll3LPikX2SogGoJtfDDs4lG2Vaz5MwV18jA9UsYKH4oNbT
IDVIjY9SzGrm"]
  }
}
```

区别

- 服务端对于升级前的 SDK，URI 传递的是一个固定的配置页面的地址：mpaasscheme://com.mpaas.harmony.push/longlink，客户端点击通知栏之后会跳转到固定的配置页面（即继承了 NcAbility 的页面），然后由该配置页面去拉起不同的页面（页面地址在 bData 里面的 URL 字段）。
- 服务端对于升级后的 SDK，URI 传递的是需要跳转的页面的地址：jump://com.mpaas.harmony.push/landing，点击后由鸿蒙系统直接跳转。

升级步骤

此次更新涉及到服务端升级与客户端升级，**建议先升级服务端，后升级客户端。**

> **说明**
> 服务端升级之后，会兼容新旧两种客户端版本。

服务端升级版本：1.35.0 及其以上版本

客户端基线版本：10.2.3.13 及其以上版本基线

鸿蒙基线接入方式可参考文档：接入 HarmonyOS NEXT

> **重要**
> **风险点提示：** 如果先升级客户端，则 **不要删除原有的配置页面** （继承 NcAbility 的配置页面），如果删除了继承 NcAbility 的配置页面，但是没有升级服务端，那么客户端将无法处理通知栏的点击跳转事件。

升级改动点

此次升级之后，客户端点击通知栏之后直接跳转到消息中包含的 URI 对应的应用内页面，不经过 NcAbility 页面中转，同时提供了数据解析工具，帮助用户在目标页面解析消息数据。

> **重要**
> **风险点提示：** 鸿蒙 Push Kit 能力暂不支持点击通知消息直接跳转到浏览器并且打开 HTTP 链接，如果服务端推送的消息包含了网页类型的跳转地址，升级之后将无法自动打开浏览器打开链接，需要用户自行适配此种消息的跳转。详情请参考鸿蒙文档：点击消息动作。

兼容情况与说明

服务端升级，客户端不升级

由于升级之后的服务端可以兼容老版本客户端组件，因此客户端不升级则维持现有方式。

客户端升级，服务端不升级

若客户端升级之后，保持继承了 NcAbility 的配置页面存在，那么依然会走原有的跳转逻辑。