

嵌码步骤

要实现小程序的性能监控，需要在小程序包中嵌入基调听云小程序JS探针，具体操作步骤如下。

1. 新建应用名称。

在应用页面，单击右上角的添加应用按钮进入新建应用页面，输入小程序名称，保存，小程序名称不可重复。

新建应用



2. 配置代码，支持微信/支付宝/钉钉/百度/京东/飞书小程序。

1. 在项目根目录建立 agent 文件夹，将探针 mp_agent_<探针版本号>.js，文件名根据不同端小程序进行修改，复制到 agent 目录下，并新建 init.js，

- 微信小程序项目中集成，解压 mp_agent-wechat_<探针版本号>.zip，将 mp_agent_<探针版本号>.js 加入到项目中；
- 支付宝小程序项目中集成，解压 mp_agent-alipay_<探针版本号>.zip，将 mp_agent-alipay_<探针版本号>.js 加入到项目中；
- 钉钉小程序项目中集成，解压 mp_agent-dd_<探针版本号>.zip，将 mp_agent-dd_<探针版本号>.js 加入到项目中。
- 百度小程序项目中集成，解压 mp_agent-baidu_<探针版本号>.zip，将 mp_agent-baidu_<探针版本号>.js 加入到项目中。
- 京东小程序项目中集成，解压 mp_agent-jd_<探针版本号>.zip，将 mp_agent-jd_<探针版本号>.js 加入到项目中。
- 飞书小程序项目中集成，解压 mp_agent-feishu_<探针版本号>.zip，将 mp_agent-feishu_<探针版本号>.js 加入到项目中。

目录结构：

```
|-- agent
   |-- mp_agent_<探针版本号>.js // 文件名根据不同端小程序进行修改
   |-- init.js
```

2. 修改 init.js

```
const agent = require('./mp_agent_<探针版本号>.js');
agent.config({
  beacon: <数据上传服务器地址>,
  key: <应用key>,
  id: <账号id>,
  sampleRate: 1
});

module.exports = agent;
```

3. 在 app.js 首行引入探针

```
// 根据实际路径填写
import './agent/init.js';
// ...
```

4. 如果在相关页面调用探针接口时, 可以import探针对象使用

例如, pages/index/index.js 中需要调用探针接口, 引入探针示例如下:

```
// 引入探针对象, 名称设置为Tingyun(可自定义), 路径填写实际路径
import Tingyun from '../..//agent/init.js';

Page({
  callTingyunAgent() {
    // const action = Tingyun.newAction({
    //   // ...
    // })
  }
})
```

注意:

- 探针若放到其他目录下, 后面的代码探针地址务必同步更改。

3. 配置request合法域名。

复制域名地址后, 在微信小程序后台的**开发设置>服务器域名>request合法域名**中加入基调听云接收探针数据的服务器地址, 如下图:



如需购买服务器资源及域名，可前往[腾讯云](#)购买。有可使用官方推出的[微信云开发](#)或[微信云托管](#)，无需服务器及域名配置即可上线小程序，[立即开通](#)

request合法域名

以 https:// 开头。可填写多个域名，域名间请用 ; 分割

socket合法域名

以 wss:// 开头。可填写多个域名，域名间请用 ; 分割

uploadFile合法域名

以 https:// 开头。可填写多个域名，域名间请用 ; 分割

downloadFile合法域名

以 https:// 开头。可填写多个域名，域名间请用 ; 分割

udp合法域名

以 udp:// 开头。可填写多个域名，域名间请用 ; 分割

tcp合法域名

以 tcp:// 开头。可填写多个域名，域名间请用 ; 分割

保存并提交

取消

配置完成，小程序发布后，即可在基调听云小程序控制台看到实时性能数据。

合规配置

为了确保您的应用符合个人信息保护合规要求，建议在用户明确同意隐私政策之后，再初始化【基调听云用户体验监控 SDK】进行数据采集与上报。

推荐方案

1. 首屏控制：在用户未点击“同意”按钮前，不执行 SDK 的初始化逻辑。
2. 状态持久化：用户同意后，业务侧需在本地缓存（Storage）中记录同意状态，以便后续启动时自动初始化。
3. 动态加载：建议使用 `require` 方式动态引入 SDK，确保在用户同意前不会触发探针代码。

示例代码：

```
// 本示例仅供参考，具体根据实际业务实现。
// 1. app.js 只需 import 当前文件，是否启动探针由本文件控制。
// 2. 首次启动时，业务侧先用 wx.getPrivacySetting 判断是否需要授权。
// 3. 用户点击官方同意按钮后，再调用 startAgent()。
// 4. 本示例用 privacyConsent 作为本地同步标记，便于后续冷启动时尽早启动探针。
let started = false;

function hasPrivacyConsent() {
  return wx.getStorageSync('privacyConsent') === 'granted';
}

function startAgent() {
  if (started) {
    return;
  }
}
```

```
// 延迟加载探针本体，避免在同意隐私前提前执行探针代码。
const agent = require('./tingyun-agent.js');

agent.config({
  beacon: <数据上传服务器地址>,
  key: <应用key>,
  id: <账号id>,
  sampleRate: 1
});

started = true;
}

// 若业务侧已在同意回调后写入 privacyConsent='granted'，则后续冷启动时自动启动探针。
if (hasPrivacyConsent()) {
  startAgent();
}

export {
  hasPrivacyConsent,
  startAgent,
};
```