

【S项目·前端】中后台 Slardar 监控建设 Copy

Info Collector [Export is restricted for this type of content.]

1. 背景

目前供应链中后台各应用系统的监控比较松散，也没有一定的规范。存在一下一些问题：

- 应用接了 Slardar，但监控指标、告警、数据看板没有使用起来
- 缺少一个全局视角去观察，给整体 Review 造成了一定麻烦

因此需要一个规范化的监控约束，提升项目质量。

2. 主应用接入收敛

2.1 核心思想

- 所有子应用 slardar 配置收敛为相同配置，在主应用中管理。（如有特殊需求，可在主应用中匹配子应用名称做差异化配置）
- 指定同学负责管理 dbmp 监控配置，收集处理各方提出的监控需求。

2.2 接入方式对比

	方案一（建议，已有线上项目在用）	方案二（当前）
接入方式	<p>在主应用中使用 garfish 的 slardar 插件 统一接入</p> <ul style="list-style-type: none">在应用的 AppInfo 中配置 SlardarOptionsSlardar Plugin 会自动化为子应用初始化 Slardar 实例，并保证上报至该 Slardar 实例的数据是该子应用的数据	<p>子应用分别接入 Slardar</p> <ul style="list-style-type: none">在子应用中使用 Slardar Plugin 针对当前子应用隔离上报数据只上报子应用相关数据至对应的 Slardar 实例上
优缺点	<ul style="list-style-type: none">主应用 Slardar 上报的数据不会包括子应用的相关数据监控的管控统一交给主应用管理，Slardar Plugin 升级维护都放在主应用中子应用仅需提供 Slardar 的配置信息即可确定是否要为当前子应用提供增加了子应用开始渲染到最终展示的维度的三个指标：MFFP、MFFCP、MFLCP	<ul style="list-style-type: none">子应用 Slardar 的实例初始化和封装都维护在子应用上不方便做统一的 Slardar Plugin 的升级操作子应用相关数据会上报至主应用中主要是 Slardar Plugin 1.0 中提供的能力，目前该能力在 2.0 中仍然保留，但是不建议用户使用方案二，推荐在主应用中完成 Slardar 的接入
子应用配置位置	可选方式：	子应用各自配置

	<ul style="list-style-type: none">在主应用中统一配置（原则上子应用配置保持一致，特殊场景下可匹配子应用name做差异化配置）子应用各自配置<ul style="list-style-type: none">在goofy子应用额外信息中配置（自定义插件需要import，goofy中只能配置json，所以不支持集成自定义插件）在子应用中配置，通过主子应用通信传递配置（优点：自治；缺点：不利于配置收敛）	
slardar实例调用方式	子应用根组件props里的slardarInStance 属性	<code>import slardar from '@jupiter/plugin-runtime/slardar';</code>
本地调试（一般用不上）	由于slardar配置是包含在子应用列表配置中的，目前仅支持通过jupiter.config配置子应用列表 不支持mock（当前使用）、远程异步获取等方式	

参考文档：[📖 Garfish 常见问题](#)（Jupiter Garfish 接入 Slardar Plugin 最佳实践）

环境演示：

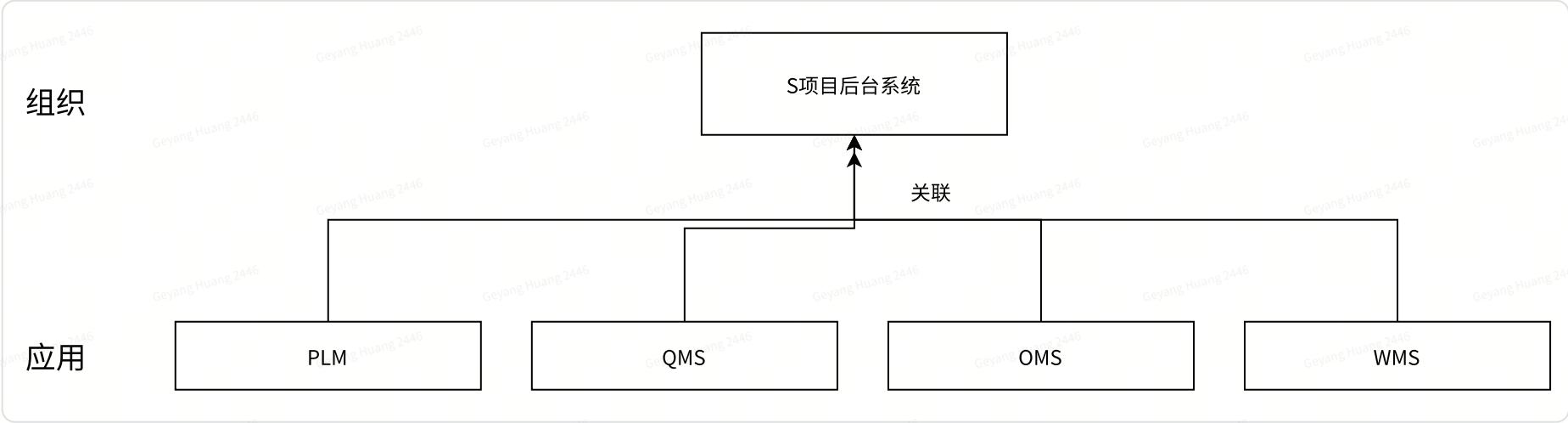
Main 应用 https://slardar-us.bytedance.net/node/web/url_list?bid=s_operation_main&lang=zh

wms 子应用 https://slardar-us.bytedance.net/node/web/url_list?env=test-bqf&bid=s_operation_wms&lang=zh&site_type=web&start_time=1665306181&end_time=1665565381&layout=normal

（注意将环境切换到 test-bqf）

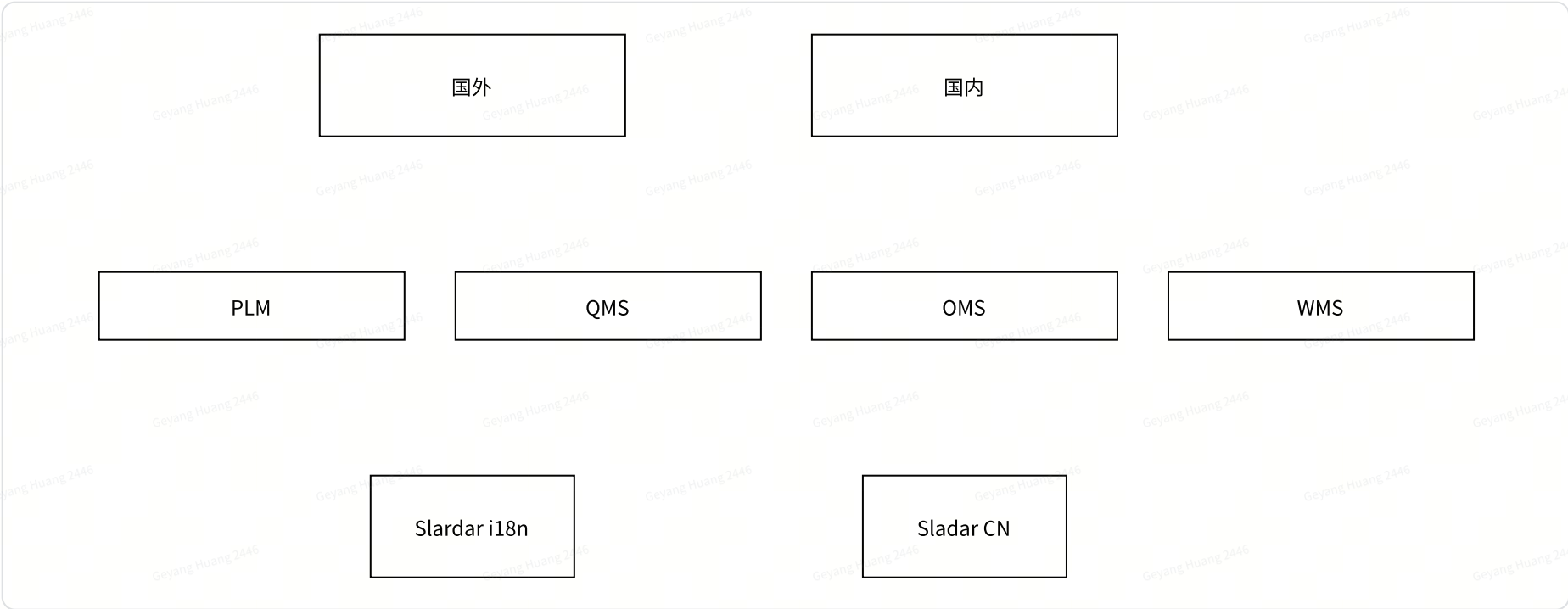
3. 应用管理方式

3.1 应用关联组织



各应用统一接入到『S项目后台』系统的 Slardar 组织中，组织关联之后，可以从组织面板统一配置告警规则、告警触达方式；同时组织维度可以观察各应用的整体质量、告警大盘以及建立全局看板，便于应用的管理和质量的 Review。

3.2 国内/国外业务分离



各应用通过『组织』进行管理，国内外分为不同的组织。BID一致，国外（目前）上传到 Slardar i18n，用 SlardarUS 平台管理；国内使用 Slardar CN，用 Slardar CN 平台管理。两者数据隔离。

组织 id	组织名	URL	
s_operation_fe	S项目后台系统	https://slardar-us.bytedance.net/node/web_org/overview?oid=s_operation_fe&start_time=1666337963&end_time=1666597163&lang=zh&env=production	SlardarUS
s_operation_fe_cn	国内服饰供应链后台系统	https://slardar.bytedance.net/node/web_org/overview?oid=s_operation_fe_cn&start_time=1666340089&end_time=1666599289&lang=zh	SlardarCN

4. 监控 & 告警

4.1 组织视角

所有后台子应用关联到同一个组织当中，告警规则从组织的模板中建立。在组织中下发的告警规则，会自动同步到应用当中。

监控指标和告警规则详见：[📖 前端监控指标 & 告警规则](#)














4.2 告警配置

4.2.1 告警方式

基于[📖 前端监控指标 & 告警](#)，设置了 P0、P1、P2 的告警分级，这里通过组织视图统一收敛了报警方式。


⊕ 新建报警方式模板

模板名称搜索

名称	创建人	创建时间	操作
<div>—</div> P0 报警触达	lixuezhi.lucas	2022-10-24 07:59:41	<div></div>
<div>⊕ 新增接收项</div>			
bid	应用名称	报警方式	操作
s_operation_fe_finance	S-财务库存-FE	<div> Lark  电话</div>	<div></div>
		<div>  电话</div>	
共 1 条 < 1 > 10 条/页			
<div>+</div> P1 报警触达	lixuezhi.lucas	2022-10-24 12:28:04	<div></div>
<div>+</div> P2 报警触达	lixuezhi.lucas	2022-10-24 12:28:15	<div></div>
共 3 条 < 1 > 10 条/页			

4.2.2 告警规则

子应用统一接入组织，通过组织视图配置了统一的告警规则。

 不需要子应用单独配置，由 Slardar 组织面板完成统一配置收敛工作

4.2.3 告警触达配置

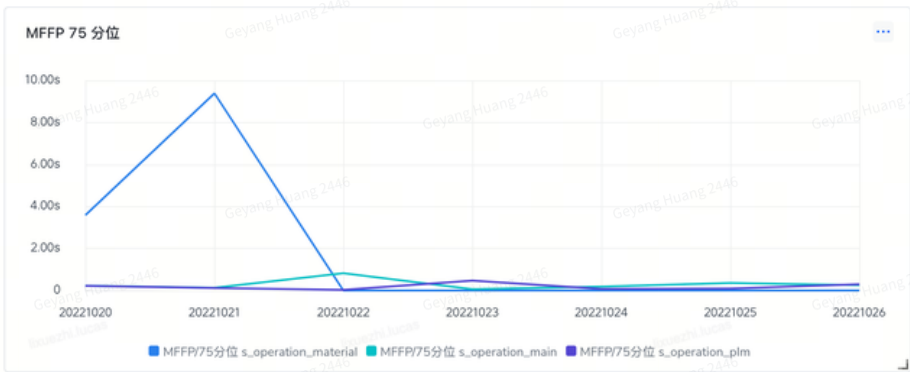
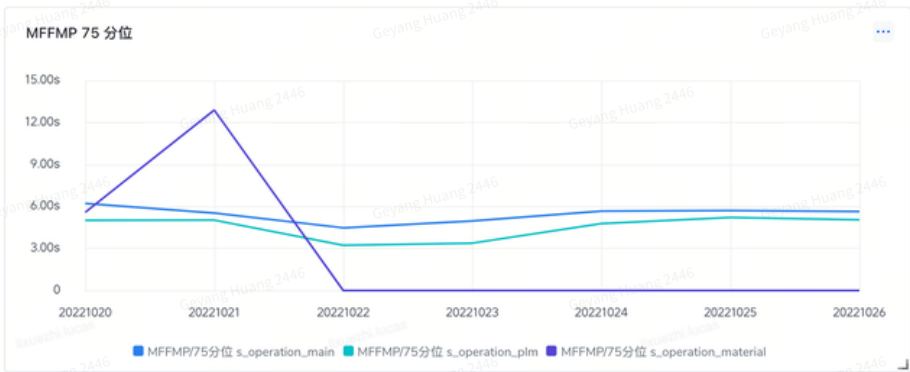
由于告警触达是统一配置，因此需要收集各系统负责同学的信息，后面通过组织面板统一进行下发。

[👁 中后台 Slardar 应用信息收集](#)

5. 数据看板

5.1 组织视角全局看板

从组织维度有一个全局看板，以 BID 作为维度进行拆分，方便阅读后台系统大盘。设定每周五上午 11 点推送到供应链前端群。



MFFMP 75 分位 周同比			
分组	20221020	20221021	20221022
bid	MFFMP/75分位	MFFMP/75分位	MFFMP/75分位
s_operation_main	6.22s (6.73%)	5.53s (-9.16%)	4.48s (-10.56%)
s_operation_material	5.59s (-19.40%)	12.89s (119.28%)	--
s_operation_plm	5.01s (7.18%)	5.03s (-4.93%)	3.23s (100.00%)

MFFP 75 分位 周同比波动			
分组	20221020	20221021	20221022
bid	MFFP/75分位	MFFP/75分位	MFFP/75分位
s_operation_main	230.80ms (-42.70%)	130.15ms (-53.65%)	820.58ms (1642.20%)
s_operation_material	3.59s (1.67%)	9.39s (349.06%)	--
s_operation_plm	225.65ms (395.39%)	115.60ms (0.52%)	31.50ms (100.00%)



https://slardar-us.bytedance.net/node/web_org/kanban/detail/63?oid=s_operation_fe&start_time=1666367060&end_time=1666626260&lang=zh&env=production&dashboardLayout=0&layout=contentOnly&href=https%3A%2F%2Fslardar-us.bytedance.net%2Fnode%2Fweb_org%2Fkanban%2Fdetail%2F63%3Foid%3Ds_operation_fe%26start_time%3D1666367060%26end_time%3D1666626260%26lang%3Dzh%26env%3Dproduction

5.2 子应用看板建设

数据看板图表建设基于 [前端监控告警治理](#)，由以下类型图表构成：

- ☐ JS 异常数（折线图）；
- ☐ API 调用错误率（折线图）；
- ☐ API 调用（业务错误）错误率（折线图）；
- ☐ LCP 75 分位数据（折线图）；
- ☐ TTI 75 分位数据（折线图）；
- ☐ API 75 分位耗时数据（折线图）；
- ☐ 静态资源错误数（柱状图）；
- ☐ PV & UV（折线图）；
- ☐ MFFP 75 分位数据（折线图）；
- ☐ MFLCP 75 分位数据（折线图）；
- ☐ MFFCP 75 分位数据（折线图）；

6. 接入步骤

6.1 主应用配置



所有配置均在主应用内完成，无需关注子应用。

在 @s_op/configs 中设置如下变量：

packages/configs/src/constants/slardar.js

```
1  const { isCn, isSg, isVa } = require('../envs');
2
3  const getDomain = () => {
4    if (isCn) {
5      return '';
6    }
7    if (isSg) {
8      return 'https://mon-sg.byteoversea.com/';
9    }
10   if (isVa) {
11     return 'https://mon-sg.byteoversea.com/';
12   }
13   return '';
14 };
15
16 module.exports = {
17   BID_PREFIX: 's_operation',
18   DOMAIN: getDomain(),
19 };
20
```

packages/configs/src/constants/index.js

```
1  exports.HOST = require('./host');
2  exports.THEME = require('./theme.overwrite');
3  exports.APP_ID = require('./app-id');
4  +exports.SLARDAR = require('./slardar');
5
```

6.1.1 配置主应用接入 slardar

安装 @jupiter/plugin-garfish、@byted-garfish/slardar-plugin

主应用 slardar 关键配置 apps/main/src/util/init-slardar.tsx

```
1  import { createBrowserClient } from '@slardar/web';
2  import { SlardarPlugin } from '@byted-garfish/slardar-plugin';
3  import { Garfish } from '@jupiter/plugin-garfish';
4  import { User } from '@api/op/data/user';
5
6  const slardarInstance = window.location.href.includes('localhost')
7    ? undefined
8    : createBrowserClient();
9
10 export function initSlardar() {
11   if (window.location.href.includes('localhost')) {
12     return;
13   }
14 }
```



```

14 Garfish.usePlugin(
15   SlardarPlugin({
16     createBrowserClient,
17     slardarOptions: {
18       getSlardarInstance: () => slardarInstance!,
19       config: {
20         bid: `${SLARDAR_BID_PREFIX}_main`,
21         domain: SLARDAR_DOMAIN,
22         env: SLARDAR_BUILD_TYPE,
23         release: BUILD_INFO?.build_version,
24       },
25     },
26   }),
27 );
28 }
29
30 export function setExtraSlardarCtx(user?: User) {
31   if (window.location.href.includes('localhost')) {
32     return;
33   }
34
35   slardarInstance?.set({
36     userId: user?.user_id,
37   });
38
39   slardarInstance?.context?.merge({
40     nickname: user?.nickname,
41     employee_id: user?.employee_id,
42     // 多租户id
43   });
44 }
45

```

在 jupiter 中配置全局环境变量

```

1 import { COMMON } = require('@s_op/configs')
2
3 export default defineConfig({
4   source: {
5     globalVars: {
6       SLARDAR_BID_PREFIX: COMMON.SLARDAR.BID_PREFIX,
7       SLARDAR_DOMAIN: COMMON.SLARDAR.DOMAIN,
8       SLARDAR_BUILD_TYPE: ENV.DEVELOP_ENV,
9     }
10   }
11 })

```

然后在主应用根组件 App 中执行 初始化

```

1 import { defineConfig } from '@jupiter/plugin-runtime';
2 import { initSlardar } from './util/init-slardar';
3
4 +initSlardar();
5
6 const App = () => {
7   // ...
8 }
9

```

```
10 export default App;
```

6.1.2 配置子应用接入 slardar

子应用 slardar 关键配置 apps/main/src/App.tsx

```
1  import { defineConfig } from '@jupiter/plugin-runtime';
2  import { initSlardar } from './util/init-slardar';
3
4  initSlardar();
5
6  const App = () => {
7    // ...
8  }
9
10 if (!window.location.href.includes('localhost')) {
11   defineConfig(App, {
12     masterApp: {
13       manifest: {
14         goofyConfig: {
15           url: GOOFY_CONFIG_PATH, // 主应用的路由地址
16           maxRetry: 3, // 失败最大重试次数
17           handleData: apps =>
18             apps.map(o => ({
19               ...o,
20               slardarOptions: {
21                 config: {
22                   bid: `${SLARDAR_BID_PREFIX}_${o.name}`,
23                   domain: SLARDAR_DOMAIN,
24                   env: SLARDAR_BUILD_TYPE,
25                   release: BUILD_INFO?.build_version,
26                 },
27               },
28             })),
29           },
30         },
31       },
32     });
33   }
34
35
36 export default App;
```

如果想要在本地开发模式下想要调试slardar接入，由于garfish slardar plugin v2不支持 mock 子应用列表，要改为在jupiter.config 中配置 runtime.features.masterApp.apps

1. apps/main/config/index.js 禁用获取子应用列表的 mock 请求
2. 在 jupiter.config.ts 中配置子应用列表

```
1  import { data as apps } from './config/mock/get-garfish-app-list';
2  import { COMMON } = require('@s_op/configs')
3
4  export default defineConfig({
5    // ...
6    runtime: {
7      features: {
8        masterApp: {
```



```
9      manifest: {
10        apps: apps.map(o => ({
11          ...o,
12          slardarOptions: {
13            config: {
14              bid: `${COMMON.SLARDAR.BID_PREFIX}_${o.name}`,
15              domain: COMMON.SLARDAR.DOMAIN,
16              env: IS_BOE ? 'test' : 'production',
17              release: BUILD_INFO?.build_version,
18            },
19          },
20        })),
21      },
22    },
23  },
24 },
25 }
```


6.1.3 验证

在 Chrome 网络面板筛选上报域名，根据query中的bid识别应用，确认有上报请求发出。

6.4 创建子应用的 slardar 应用并加入组织

以 `${SLARDAR_BID_PREFIX}_${子应用名}` 为格式创建 Sladar 子应用，并填写在 [国中后台 Slardar 应用信息收集](#) 中，后续统一加入到组织当中。

6.5 子应用接入告警规则

 只需填写 [国中后台 Slardar 应用信息收集](#) 信息，后续统一下发告警规则。

当 Slardar 应用加入了『S供应链后台系统』的组织之后，可以从组织面板给当前组织下的应用『下发』告警规则。完成『下发』之后，子应用则完成了通用告警规则的配置。

报警管理

报警统计

批量创建

模板名称搜索

新建报警模板

导入报警

任何批量操作都会将所有下属报警同步

批量更新

批量暂停

批量启动

名称

Bid

P2

MFFMP/75分位报警

s_operati

e_finance

批量创建报警

报警配置1

+

请选择 BID 和环境

S-财务库存-FE

production

⊕

S项目-后台系统-面辅...

production

⊕ ⊖

S项目-后台系统-PLM...

production

⊕ ⊖

报警配置

```
1 1
2 2 "origin_oid": "s_operation_fe",
3 3 "origin_org_template_id": 43,
4 4 "id": 43,
5 5 "name": "MFFMP/75分位报警",
6 6 "is_close": false,
7 7 "methods": [
8 8   "lark"
9 9 ],
10 10 "receivers": [
11 11   "lixuezhi.lucas"
12 12 ],
13 13 "feedback_duty_name": "",
14 14 "note": "",
15 15 "category": "performance",
16 16 "strategy_list": [
17 17   {
18 18     "measure": {
19 19       "type": "monomial",
20 20       "raw_measure_list": [
21 21         {
22 22           "measure_name": "browser_perf.custom.mffmp.pct75"
```

创建时间

操作

2-10-24 07:32:06

🔗

📄

🗑️

批量创建

最近一个月报警次数

操作

0

🔛

🔗

📄

🗑️

✖️

共 1 条

<

1

>

20 条/页

共 1 条

<

1

>

10 条/页

6.6 子应用接入看板

子应用需要做的是从『[这里](#)』打开，选择复制『全局看板』到自己的应用，即可。



6.7 验证数据是否上报

7. 扩展功能

7.1 关联版本号（主应用集中配置，子应用无需关注）

/src/App.tsx

```
1 import { getSlardarConfig } from '@util/init-slardar';
2
3 const appendGarfishAppLifeCycle = (applist: Array<any>) => {
4   // eslint-disable-next-line @typescript-eslint/ban-ts-comment
5   // @ts-expect-error
6   const garrModules = window.gfdatav1?.garrModules?.data || [];
7   return applist.map(app => ({
8     ...app,
9     slardarOptions: {
10       config: getSlardarConfig(
11         app.name,
12         garrModules.find((o: any) => o.name === app.name)?.version,
13       ),
14     },
15     active: () => {
16       /* intend to be empty */
17     },
18     deactivate: () => {
19       /* intend to be empty */
20     },
21   }));
22 };
23
24 // runtime config
25 defineConfig(App, {
26   masterApp: {
27     protectVariable: ['nickname', 'employee_id'],
28     apps: appendGarfishAppLifeCycle(DevAppEntries),
29     manifest: {
30       goofyConfig: {
31         url: GOOFY_CONFIG_PATH, // 主应用的路由地址
32         maxRetry: 3, // 失败最大重试次数
```

```

33     handleData: (apps: Array<AppInfo>) => {
34         if (!IS_DEV) {
35             return appendGarfishAppLifeCycle(apps);
36         }
37         return apps;
38     },
39 },
40 },
41 },
42 });

```

initSlardar.tsx

```

1  export const getSlardarConfig = (app_name: string, app_version?: string) => ({
2      bid: `${slardarEnv?.bid_prefix}_${app_name.replaceAll('-', '_')}`,
3      domain: slardarEnv.domain,
4      env: slardarEnv?.env,
5      // app—version是子应用版本号, slardarEnv?.release 是主应用版本号
6      release: app_version ?? slardarEnv?.release,
7  });
8
9  export function initSlardar() {
10     // if (window.location.href.includes('localhost')) {
11     //     return;
12     // }
13     // eslint-disable-next-line react-hooks/rules-of-hooks
14     Garfish.usePlugin(
15         SlardarPlugin({
16             createBrowserClient,
17             slardarOptions: {
18                 getSlardarInstance: () => slardarInstance,
19                 config: getSlardarConfig('main'),
20             },
21         }),
22     );
23 }
24

```

7.2 配置 sourcemap 上传

sourcemap 上传功能是通过webpack的插件支持的，需要在jupiter.config.ts中配置

```

1  import { ENV, COMMON } from '@s_op/configs';
2  const SlardarWebpackPlugin = require('@slardar/webpack-plugin');
3  {
4      webpack: (config, { appendPlugins }) => {
5          if (ENV.IS_PROD) {
6              // prod
7              // upload slardar sourcemap
8              appendPlugins([
9                  new SlardarWebpackPlugin({
10                     bid: `${COMMON.SLARDAR.BID_PREFIX}_main`,
11                     include: ['./dist/static/js'],
12                     release: BUILD_INFO?.build_version,
13                     clear_after_upload: true,
14                     region: ENV.CUSTOM_CDN_REGION || 'sg',
15                 }),
16             ]);

```

```
17     }
18   }
19 }
```

子应用的配置方式可选：

1. **【首选】** 配置放在仓库根目录config/jupiter.base.config.ts中，子应用继承 jupiter 配置
2. **【过渡】** 在子应用的jupiter.config.ts中单独配置

备注：bid的前缀变量从 @s_op/configs 中取

7.3 自定义上报

自定义上报的关键就是如何拿到 slardar 实例。

主应用的 slardar 实例在 init-slardar 进行了创建并导出。

子应用中获取自己的 slardar 实例，可以通过两种方式：

1. 根组件 App 的props.slardarInstance 即子应用的 slardar 实例（可以通过ReactContext传递），此逻辑已封装在@s_op/hooks 中

添加依赖

```
1
```

在根组件App.tsx中注入 slardar 实例

```
1  import { GlobalContextProvider } from '@s_op/hooks/src/useGlobalContext';\
2  import { SlardarInstance } from '@s_op/utls/src/types/module';
3
4  type Props = {
5    userInfo: {
6      nickname: string;
7      employee_id: string;
8    };
9    slardarInstance: SlardarInstance;
10 };
11
12
13 const App = (props: Props) => {
14   const { userInfo, slardarInstance } = props;
15
16   return (
17     <GlobalContextProvider value={{ userInfo, slardar: slardarInstance }}>
18       <Layout className="wms-layout">
19         { /* ... */ }
20       </Layout>
21     </GlobalContextProvider>
22   )
23 }
```

在组件中调用 slardar实例进行上报

```
1  import { useEffect } from 'react';
2  import useGlobalContext from '@s_op/hooks/src/useGlobalContext';
```

```

3
4
5 export default () => {
6   const { userInfo, slardar } = useGlobalContext();
7
8   useEffect(() => {
9     slardar('sendEvent', {
10      name: 'example',
11      metrics: {
12        count: 1,
13      },
14      categories: {},
15    });
16   }, []);
17
18   return (
19     <div className="wms-page">
20       <div className="wms-page-header">
21         <h2 className="wms-page-title">
22           容器列表
23         </h2>
24       </div>
25     </div>
26   </div>
27 );
28 };
29

```

2. 通过 window.Garfish.appInfos 根据应用的名字匹配得到 appInfo，appInfo.props.slardarInstance 即子应用自己的 slardar 实例，此逻辑已封装在@s_op/utils中

```

1
2 import { SlardarTracker, ApiTracker } from '@s_op/utils';
3
4 export const tracker = new SlardarTracker('wms');
5 export const slardar = tracker.getSlardarInstance();
6
7
8 const initialState: IState = {
9   list: [],
10  loading: false,
11 };
12 const slice = reduce.createSlice<IState>({
13   name: 'meta/container',
14   initialState,
15 });
16
17 export default slice;
18
19 // ...
20
21 export const getContainerList = slice.define(
22   (set, get) => async () => {
23     set(draft => {
24       draft.loading = true;
25     });
26     const { query, pagination } = get();
27     const params = {
28       query_param: query,
29

```

```

29     page_info: {
30         page_no: pagination.current,
31         page_size: pagination.pageSize,
32     },
33 };
34 slardar('sendEvent', {
35     name: 'before request',
36     metrics: {
37         count: 1,
38     },
39     categories: {
40     },
41 });
42
43 const apiTracker = new ApiTracker(tracker, 'getContainerList').start();
44 const result = await metaService.ListContainer(params);
45
46 setState(draft => {
47     draft.loading = false;
48 });
49 if (!result) {
50     apiTracker.fail('error info');
51     return;
52 }
53
54 apiTracker.success();
55 // ...
56 },
57 );

```

7.4 请求异常上报

/src/_/utils/slardar.ts

```

1 import { SlardarTracker } from '@s_op/utils';
2
3 export const tracker = new SlardarTracker('wms');
4 export const slardar = tracker.getSlardarInstance();
5

```

apps/wms/src/api/index.ts

```

1 import { i18n } from '@jupiter/plugin-runtime/i18n';
2
3 import commonFetch, { OptionConfig } from '@s_op/common-fetch';
4 import { Message } from '@arco-design/web-react';
5 import { slardar } from '../util/slardar';
6 import MetaService from './meta';
7 export const metaService = new MetaService<OptionConfig>({
8     baseURL: '/bff/wms/dbmp',
9     request: commonInterceptor,
10 });
11
12 function commonInterceptor(
13     params: {
14         url: string;
15         method: 'GET' | 'DELETE' | 'POST' | 'PUT' | 'PATCH';
16         data?: any;

```



```

17     params?: any;
18     headers?: any;
19 },
20 options: OptionConfig,
21 ): Promise<any> {
22     const { url, ...rest } = params;
23     return new Promise(resolve => {
24
25         commonFetch(url, rest, {
26             ...options,
27             slardar,
28         })
29         .then(resp => {
30             if (resp?.base_resp?.code === 0) {
31                 resolve(resp);
32             } else {
33                 Message.error(
34                     resp?.base_resp?.message ||
35                     i18n.t('wms_pc_fe_api_system_error', '系统错误'),
36                 );
37                 resolve(null);
38             }
39         })
40         .catch(() => {
41             Message.error(i18n.t('wms_pc_fe_api_system_error', '系统错误'));
42             resolve(null);
43         });
44     });
45 }
46

```

7.5 白屏监控

7.6 url id 聚合

7.7 过滤冗余上报

8. 接入计划

具体看 [国slardar配置迁移](#)

第一阶段 DDL 11.4（延迟到 11.15）

- ☒ ~~主应用、wms接入（appname匹配wms）~~
- ☒ ~~代码层面接入~~
- ☒ ~~slardar 后台应用配置~~
- ☒ ~~拉各模块POC确认接入时间，指定接入负责人~~

第二阶段 DDL 11.11（延迟到11.18）

- ☒ ~~provider封装，提供给子应用获取slardar实例（加上用户信息）~~
- ☒ ~~其他子应用陆续接入~~
 - ☒ ~~代码层面接入~~

☒ slardar 后台应用配置

各模块接入进度 [国中后台 Slardar 应用信息收集](#)

第三阶段 其他主应用及关联子应用接入 DDL 11.25（可能会延迟一周）

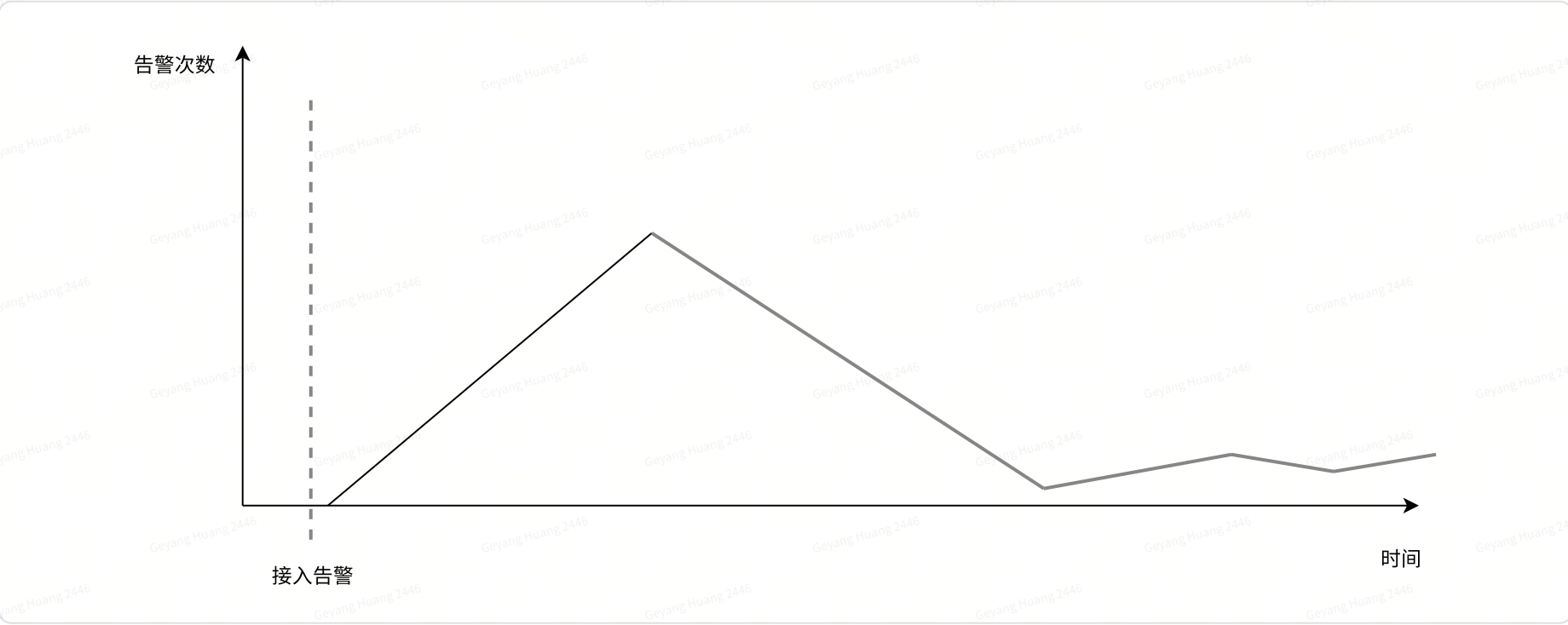
☒ portal

☒ attract

9. 后续计划

当后台系统全部接入 Slardar 并加入对应的 Slardar 组织之后，便得到了一定能力的基础监控。在监控建立初期的一段时间，可能会经历一段『阵痛期』：

- 个别接口调用链路长，查询基数大，拉高了整体接口的 75 分位，导致每天都有超时告警。
- 页面加载太慢，经常超过阈值，告警频发。
- 有不少重复的告警，工作不断被打断，增加了心智负担。
 - LoadComponent (A,b,c)
 - /DETAIL/:id
- 业务系统还未大范围使用，75 分位的随机性很大。
- 知道是 API 异常，但能获取的上下文比较单薄（缺少 LogID、请求参数等上下文信息），排查依旧困难。



告警的阈值和规则，需要不断地根据业务的规模等实际情况去进行调节和优化。因此，后续的计划会基于业务的进程、告警大盘围，围绕着治理优化为目标持续建设整个监控体系。

执行时间	任务	备注
11.1 ~ 11.21	持续协助后台系统各应用接入告警监控、数据看板，完成基础告警的落地	
11.1 ~ 11.15	完善监控建设，包括但不限于：	

	<div><div><input type="checkbox"/> 增加 common-fetch 的自定义上报，补充 logid 等上下文信息</div><div><input type="checkbox"/> 增加子应用渲染失败自定义上报</div><div><input type="checkbox"/> 补充 Path 聚合，合并不必要的 PID</div><div><input type="checkbox"/> 持续优化重复的 JS Error，合并同类型错误</div><div><input type="checkbox"/> 收集各应用接入后的告警情况，根据实际情况持续优化告警阈值</div></div>	
11.21 开始	逐步开始 Review 每周监控情况，治理每一个类型的错误	

10. 参考文档

📖【S项目·前端】PDA Slardar 监控建设

Garfish 插件 plugin-slardar

Slardar React 集成 <https://slardar.cn.goofy.app/docs/sdk/advanced/react>

📖接入slardar plugin2.0问题反馈

📖Garfish 常见问题

📖前端监控概述