# 业务平台审批流接入相关

**startupParameterConfig.ts**

获取策略文件boot.js和/config/config${type}.json

/\*\*

\* 获取配置文件配置

\*/

export const getParamConfig = (configType) => {

/\*\*

\* JSFPP-2296 前端策略规则更新

\* 租户类型统一使用"产品类型\_客户表示"的方式 例如 GCGL\_YANGO

\* 在指定租户策略获取失败时优先获取产品统一策略,如果仍然获取失败,使用默认租户策略

\*/

const type = configType ? `\_${configType}` : ""

return request({

method: "get",

url: `/config/config${type}.json`

}).catch((e) => {

if (configType?.indexOf("\_") != -1) {

return getParamConfig(configType.split("\_")[0])

} else {

return getParamConfig("default")

}

})

}

/\*\*

\* 获取boot.js配置

\*/

export const getBootJs = () => {

const timestamp = new Date().getDate();

return loadJs("/config/boot.js?t="+timestamp, "boot").then(res => {

return res.globalConfig

});

}

async function loadJs(

url: string,

moduleName: string,

opts = { async: true, container: "body" }

) {

const node = findScriptNode(url);

if (node != null && node.dataset.hasLoaded == "true") {

return await waitAndGet(moduleName, LOAD\_TIMEOUT);

} else {

try {

await loadScript("script", url, { type: "text/javascript", ...opts });

return moduleName ? window[moduleName] : undefined;

} catch (err) {

throw new Error(`load module ${moduleName} failed`);

}

}

}

const sleep = function (ms: number) {

return new Promise((resolve) => setTimeout(resolve, ms));

};

async function waitAndGet(moduleName: string, timeout: number) {

await sleep(timeout || WAIT\_TIMEOUT);

let result = window[moduleName];

return result;

}

function findScriptNode(url: string): HTMLScriptElement {

const nodes = document.querySelectorAll("script");

for (let node of nodes) {

if (node.src && node.src.indexOf(url) > 0) {

return node;

}

}

return null;

}

function findLinkNode(url: string): HTMLLinkElement {

const nodes = document.querySelectorAll("link");

for (let node of nodes) {

if (node.href && node.href.indexOf(url) > 0) {

return node;

}

}

return null;

}

async function loadScript(nodeName, url, props) {

return new Promise(function (resolve, reject) {

const containerName = props.container || "head";

var container = document.getElementsByTagName(containerName)[0];

var node = document.createElement(nodeName);

for (let propName in props) {

node[propName] = props[propName];

}

node.charset = "utf-8";

node.async = !!props.async;

node.timeout = LOAD\_TIMEOUT;

node.src = url;

node.onload = function () {

node.dataset.hasLoaded = "true";

console.log(`load script success, url: ${url}`);

node.onerror = node.onload = null;

resolve(url);

};

node.onerror = function () {

console.log(`load script failed, url: ${url}`);

node.onerror = node.onload = null;

reject(url);

};

container.appendChild(node);

});

}

**业务代码**

<iframe :src="visaConfirmIframeUrl"

frameborder="0"

style="height: 100%; width: 100%"

></iframe>

**获取当前环境配置信息**

await this.getStrategyConfig()

this.version = this.getWorkFloVersion()

if (this.version === "v2") {

this.montageVisaChangeUrl(res)

}

**拼接 iframe**

async montageVisaChangeUrl(res) {

if (res?.flowTaskId) {

const item = await bimoneApi.getTasksByFlowTaskId(

this.projectId,

res?.flowTaskId

)

const host =

process.env.VUE\_APP\_REAL\_ENV === "dev"

? "http://bimdev.glodon.com"

: ""

this.visaChangeIframeUrl = `${host}/mobile-apps/workflow-center/index.html?local\_token=${getUrlVar(

"userToken"

)}&source=jsf&ignoreScopes=application,sign,comment,role,print&hiddenPageTitle=true&tenantId=${

this.$route?.query?.tenantId

}&returnType=iframe&themeColor=%23ff7373#/processTrack?processInstanceId=${

item?.xmglInstanceId

}&hiddenFlowChart=true&hiddenUserPhone=true`

}

},