鸿蒙cmcasdk接入指南

### cmcasdk-default.tgz 使用前必读

1. 从华为官网下载DevEco Studio(开发工具版本号是DevEco Studio 5.0.2 Release)，新建或打开已有的HarmonyOs工程。
2. 右键项目名称，选择new->Folder 新建一个名为libs的文件夹。将cmcasdk-default.tgz复制到libs文件夹下。
3. 在文件夹下oh-package.json5 添加"dependencies": { 'cmcasdk': "file:./src/libs/cmcasdk-default.tgz"}
4. 执行ohpm install ，执行成功后
5. 在module.json5 "requestPermissions": 中添加使用网络的权限{"name"："ohos.permission.INTERNET",}
6. API支持14 （5.0.2）

### 接口调用

#### 获取token

1. /\*\*
2. \* 获取token
3. \* @param url 请求地址
4. \* @param msisdn 手机号码
5. \* @returns
6. \*/
7. export async function tokenPostRequest(url: string, msisdn:string ): Promise<http.HttpResponse> {}

接口调用示例

1. .onClick(async () => {
2. try {
3. const response = await tokenPostRequest(
4. this.url, //传入的IP地址
5. this.phoneNumber //手机
6. );
7. if (response.responseCode === 200) {
8. let rspData = JSON.parse(response.result.toString()) as RespData<GetToken>;
9. let tokenRes = parseResponseGetToken(rspData)//解析出来token
10. promptAction.showToast({ message: '请求成功！获取到的token是===》' + tokenRes });
11. this.responseData = tokenRes;
12. console.info('tokenRes========>' + tokenRes);
14. }
15. } catch (error) {
16. this.responseData = error.message;
17. }

#### 申请证书

1. /\*\*\*
2. \* 获取公钥
3. \*/
4. export async function getPubKey() : Promise<string>{}
5. /\*\*
6. \* 获取私钥
7. \* @returns
8. \*/
9. export async function getPriKey() : Promise<string>{}
10. /\*\*
11. \* 申请证书
12. \* @param url
13. \* @param token
14. \* @param data （
15. \* algType: string; 算法类型
16. deviceType: string;v设备类型
17. imei: string;设备imei
18. imsi: string;设备imsi
19. misdn: string;手机号码
20. publicKey: string;公钥
21. transactionID: string;）时间ID
22. \* @returns
23. \*/
24. export async function applyCertPostRequest(url: string, token: string, data: object): Promise<http.HttpResponse> {

调用示例

1. .onClick(async () => {
2. ToastUtil.showLong("申请证书中请稍后。。。")
3. // 加载状态层
4. // if(StrUtil.isEmpty(await PreferencesUtil.getStringSync(CERT)) && !StrUtil.equal(this.phoneNumber,await PreferencesUtil.getStringSync(MSISDN))){
5. let pubKey = await getPubKey(); //公钥
7. let applyCertBean = new ApplyCertBean();
8. let applyCert: BaseBean<ApplyCertBean> = new BaseBean();
9. applyCertBean.misdn = this.phoneNumber;
10. applyCertBean.transactionID = "20250221103647665199";
11. applyCertBean.algType = "1";
12. applyCertBean.deviceType = 'M2007J22C';
13. applyCertBean.imei = "a40174a8b94cb592";
14. applyCertBean.imsi = "a40174a8b94cb592";
15. applyCertBean.publicKey = pubKey;
16. applyCert.data = applyCertBean; //拼接data
17. try {
18. const response = await applyCertPostRequest(
19. this.url,
20. this.responseData, applyCert
21. );
22. if (response.responseCode === 200) {
23. this.isLoading = false // 无论成功失败都关闭加载
24. let rspCertData = JSON.parse(response.result.toString()) as RespData<GetCert>;
25. let cert = parseResponseGetCert(rspCertData) //解析出来证书
26. this.certTextInfo = cert
27. PreferencesUtil.putSync(CERT, cert);
28. }
29. } catch (error) {
30. this.isLoading = false // 无论成功失败都关闭加载
31. this.certTextInfo = error.message;
32. }

#### 获取算法类型

1. /\*\*
2. \* 获取算法类型
3. \* @param url
4. \* @param qrString 二维码信息
5. \* @returns
6. \*/
7. export async function getAlgTypePostStrRequest(url: string, qrString: string): Promise<http.HttpResponse> {—}

调用示例

1. .onClick(async () => {
2. const algTypeResp = await getAlgTypePostStrRequest(
3. this.url,
4. this.mStrQrtext
5. );
6. if (algTypeResp.responseCode === 200) {
8. let rspAlgTypeData = JSON.parse(algTypeResp.result.toString()) as RespData<GetAlgType>;
9. this.rspAlgTypeData = JSON.stringify(algTypeResp.result);
10. let getPlatFormData = parseResponseGetPlatFormName(rspAlgTypeData);//解析出来的平台名称
11. promptAction.showToast({ message: '平台名称是！' + getPlatFormData });
12. }
13. });

#### 查询证书

1. /\*\*
2. \* 查询证书
3. \* @param url
4. \* @param data {
5. \* msisdn: string;手机号码
6. algType: number;算法类型
7. certNo: string;证书序列号
8. \* }
9. \* @returns
10. \*/
11. export async function QueryCertRequest(url: string, data: object): Promise<http.HttpResponse> {

调用示例

1. .onClick(async () => {
2. let certNoData = await getCertNo(this.certTextInfo);
3. // 使用 await 等待请求完成（看似同步）
4. let queryCertBean = new QueryCertBean();
5. queryCertBean.certNo = certNoData;
6. queryCertBean.algType = 1;
7. queryCertBean.msisdn = this.phoneNumber;
8. let queryCertBeanData: BaseBean<QueryCertBean> = new BaseBean()
9. queryCertBeanData.data = queryCertBean;
10. const queryCertResp = await QueryCertRequest(
11. this.url,
12. queryCertBeanData
13. );
14. if (queryCertResp.responseCode === 200) {
16. this.rspQueryData = JSON.stringify(queryCertResp.result);
17. promptAction.showToast({ message: '查询证书的结果是====》' + queryCertResp.result });
18. }
19. });

#### 扫码登录

1. /\*\*
2. \* 扫码登录
3. \* @param url
4. \* @param qrCode 二维码信息
5. \* @param userId userID
6. \* @returns
7. \*/
8. export async function QrLoginPostRequest(url: string,qrCode :string,userId:string): Promise<http.HttpResponse> {

#### 加密

1. /\*\*
2. \* 加密
3. \* @param orgData 原文
4. \* @param pinStr 6位PIN
5. \* @returns
6. \*/
7. export async function Encrypted(orgData :string,pinStr :string): Promise<string> {

调用示例

1. .onClick(async () => {
3. let str = await Encrypted(this.encryptedStrData, this.setPin);
4. promptAction.showToast({ message: `加密后的数据是后的数据是: ${await str}` });
5. this.dataStr = str;
6. });

#### 解密

1. /\*\*
2. \* 解密
3. \* @param cipherText 加密后的数据
4. \* @param strPin PIN码
5. \* @returns
6. \*/
7. export async function DecryptData(cipherText :string,strPin : string): Promise<string> {

调用示例

1. .onClick(async () => {
2. let decCryptData = DecryptData(this.dataStr, this.setPin);
3. promptAction.showToast({ message: '解密的结果是====》' + await decCryptData });
4. this.decCryptData = await decCryptData;
5. });

#### 设置PIN

1. /\*\*
2. \* 设置PIN
3. \* @param pinStr
4. \* @returns
5. \*/
6. export async function SettingPIN(pinStr: string): Promise<string> {}

调用示例

1. .onClick(async () => {
2. let str = SettingPIN(this.setPin);
3. promptAction.showToast({ message: `` + await str });
4. }).width(320);

#### 重置PIN

1. /\*\*
2. \* 重置PIN
3. \* @param pinStr
4. \* @returns
5. \*/
6. export async function ReSettingPIN(pinStrData: string): Promise<string> {}

示例代码

1. .onClick(async () => {
2. let reset = ReSettingPIN(this.reSetPIN);
3. promptAction.showToast({ message: `` + await reset });
4. });

#### 修改PIN

1. /\*\*
2. \* 修改PIN
3. \* @param orgPinStr 原PIN
4. \* @param endPin 修改后的PIN
5. \* @returns
6. \*/
7. export async function ChangerPIN(orgPinStr: string,endPin:string): Promise<string> {}

调用示例

1. Button("修改PIN").width(320)
2. .onClick(async () => {
3. let str = ChangerPIN(this.oldPIN,this.newPIN);
4. promptAction.showToast({ message: `` + await str });
5. });