智 方 便
iAM Smart

"iAM Smart" Sandbox Programme

Authentication and Form Filling

智 方 便 iAM Smart | SANDBOX Programme

Agenda

- Authentication Overview
- Account Linkup and Creation
- Form Filling (with Service Login and Anonymous)
- UI Requirements
- Practical Tips
- Quiz

智方便 iAM Smart | SANDBOX Programme

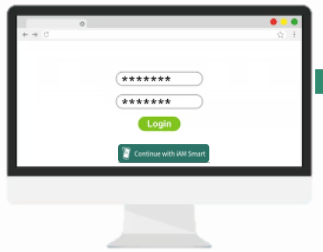
Disclaimer

Please be aware that the video is intended for preliminary introduction. It shall not be followed as the technical instruction. “iAM Smart” Sandbox Programme would not guarantee the correctness and timeliness of data which could be possibly affected by the modification of development. Development team shall follow the guidelines and policies or enquire to related professionals if any safety apprehension.

Authentication Overview

Authentication API

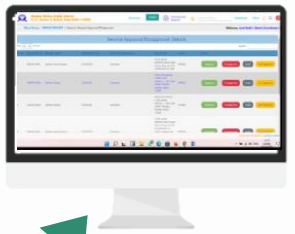
HCP Online Service Login Page



"iAM Smart" QR Page/ Broker Page

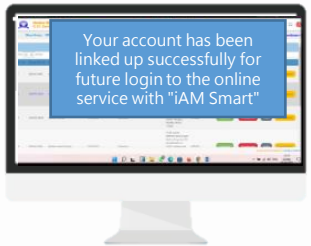
"iAM Smart" Authorisation Page

HCP Online Service Main Page



Page for user input account username & password for the link up

Display a message to inform the user that his/ her account has been **linked up** for the future login with "iAM Smart"



From "iAM Smart" APP

Open ID (tokenised id) found?

Yes

No

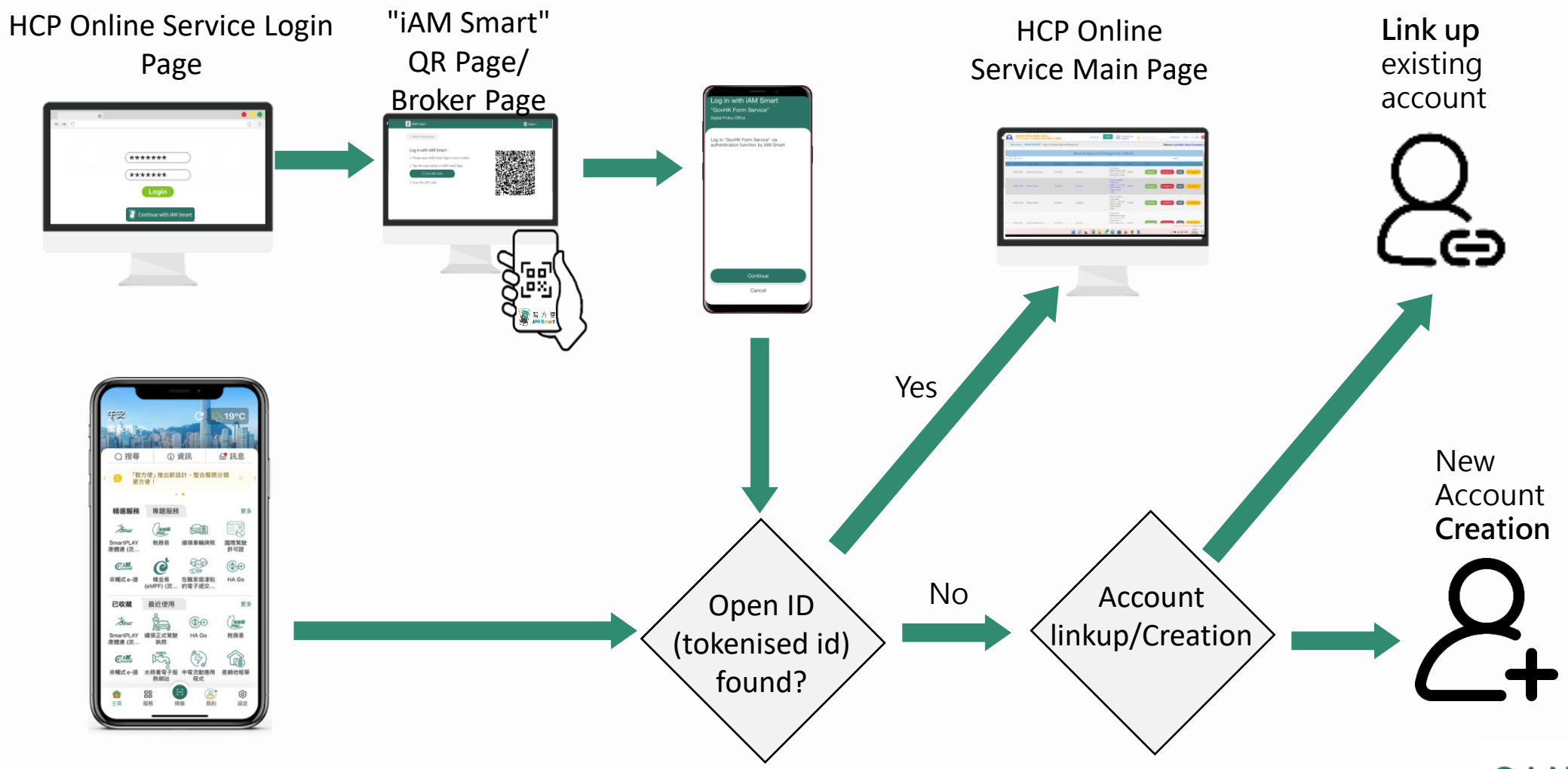
Create New Account/ Link up?

Link up existing account

New Account **Creation**

Form Filling API and complete account **creation** process

Authentication API

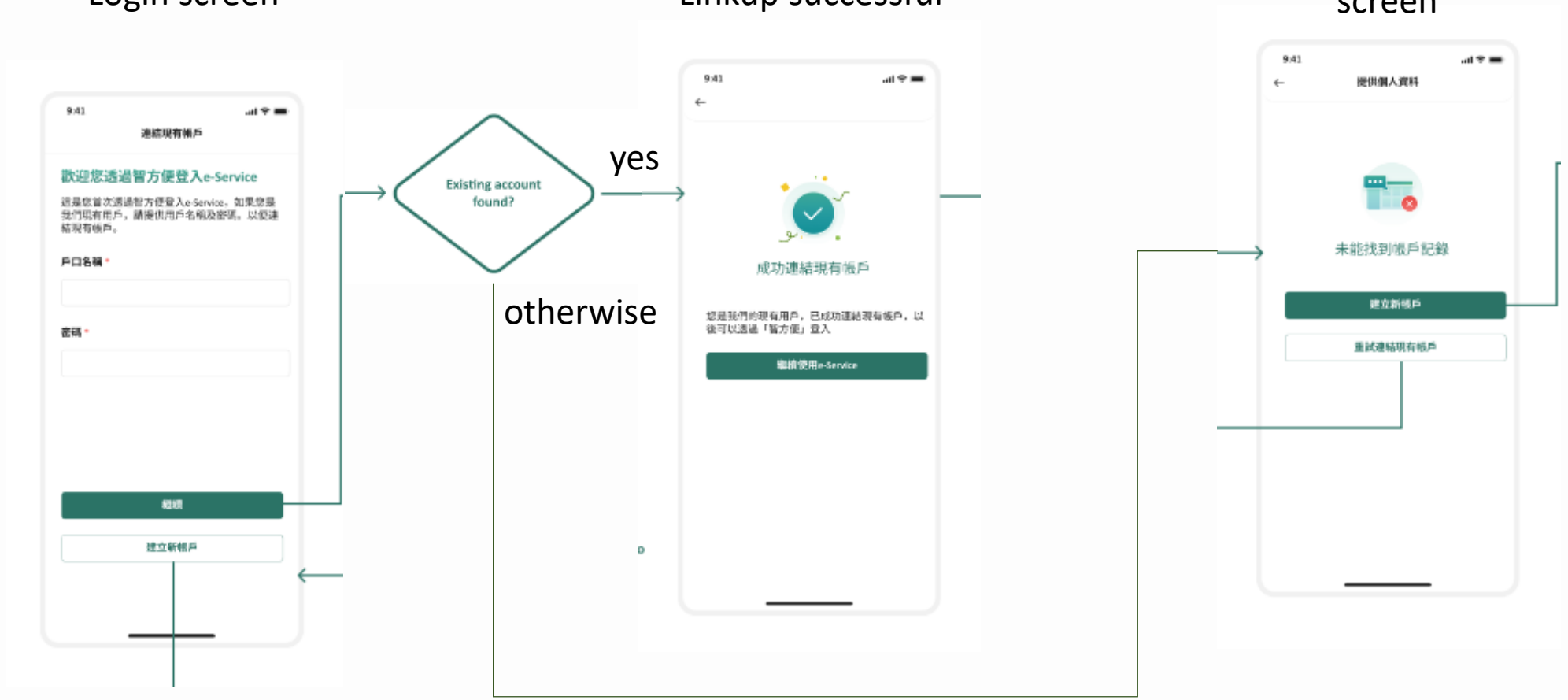


UI Requirements

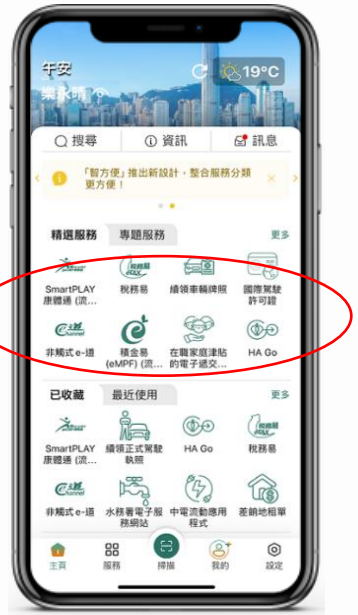
Login screen

Linkup successful

Creation linking screen



Authentication API - Direct Login (Trigger from "iAM Smart" App)



"iAM Smart" homepage

Select the service from "iAM Smart"

Open ID (tokenised id) found?

Yes

No

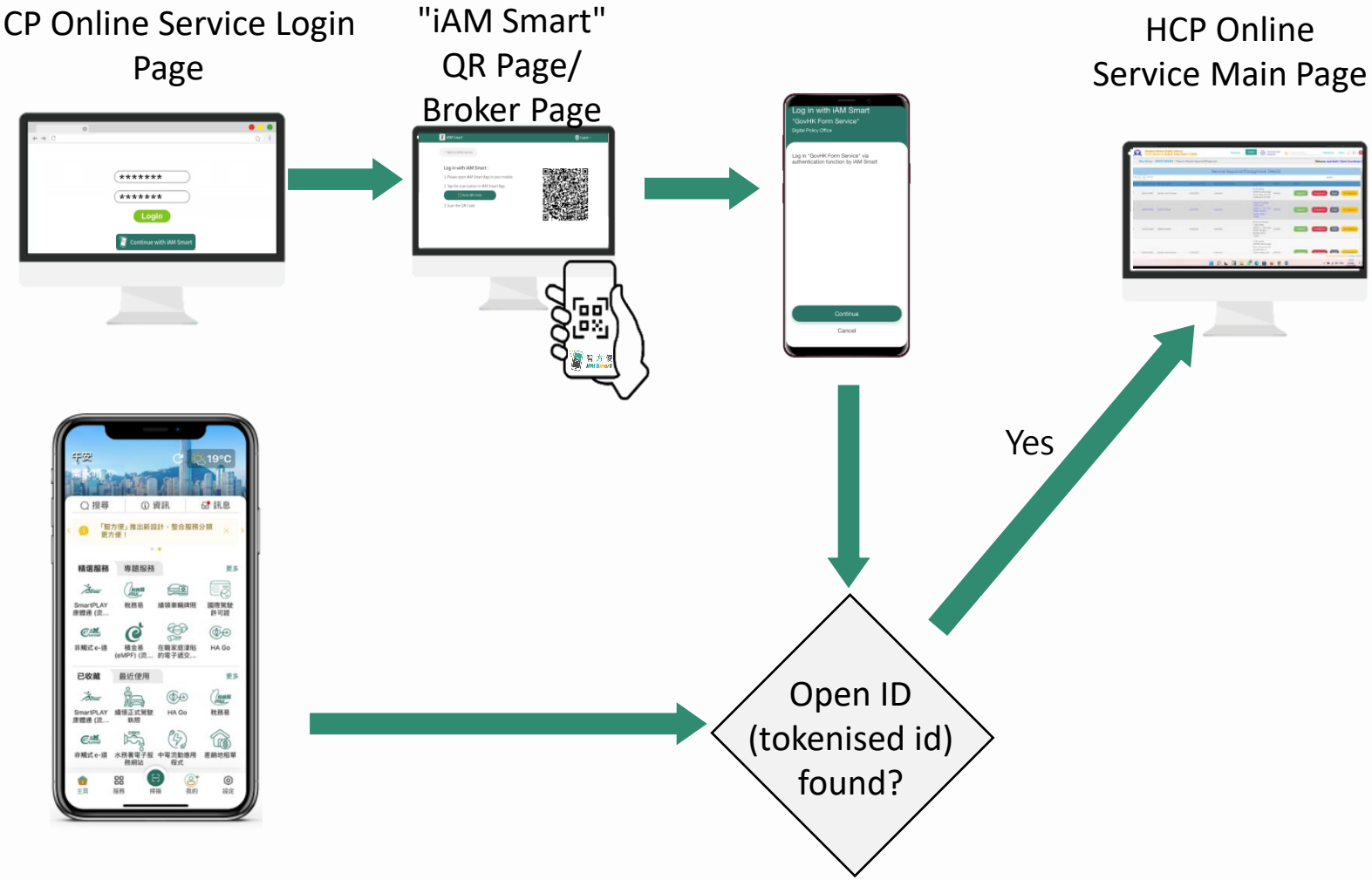
Redirect to the account linkup/creation

Service Main Page

Direct Login

- Trigger from "iAM Smart"
- Situation after login to "iAM Smart"
- Select the online service and redirect with tokenized ID
- Authenticated by the online service

Authentication API - Login with "iAM Smart"



Login with “iAM Smart”

- Trigger from the online service website/application
- Online service authenticate “iAM Smart” user with the tokenized ID
- The user linked up the account with the tokenized ID before
- Otherwise, it redirects to account linkup/creation

Account Linkup and Creation

Account linkup/ creation with Form Filling APIs

- Profile fields are required for the account linkup/creation
- Generally for user identity verification purpose
- Click the button in the consent page to call the form filling API

Profile Fields

profileFields	Description
idNo	ID number
enName	English name
chName	Chinese name
birthDate	Date of birth
gender	gender

Account creation with Form Filling APIs

- Profile fields are necessary for the account creation
- e-ME fields could be voluntarily input for better form filling experience
- For form filling usage

Profile Fields
(e-ME Fields)

eMEFields	Description
idNo	ID number
prefix	prefix
enName	English name
chName	Chinese name
birthDate	Date of birth
gender	gender
maritalStatus	marital status
homeTelNumber	home telephone number
officeTelNumber	office telephone number
mobileNumber	mobile number
emailAddress	email address
residentialAddress	residential address
postalAddress	postal address
educationLevel	education level
addressDocInfo	provider name, retrieval date, owner name and address information related to an e-bill
addressDocFile	e-bill from an address data provider

API - GetQR

GetQR would be a usual API for authenticate user by calling out a broker page or QR page. After authorization by user, the page will be redirected to the **redirectURI** with **authCode** and **state** parameters.
Location: GetQrUriController.java (getQrUri)

state

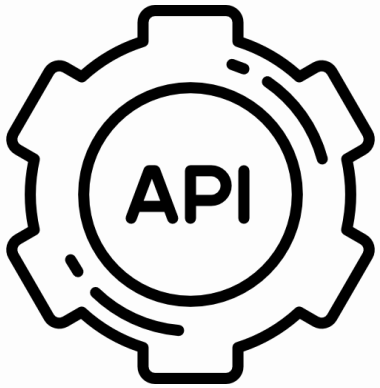
redirectURI

redirectURI with authCode and state

```
String state = UUID.randomUUID().toString().replace("-", "");
boolean isStoredState = storeState(state);
while (!isStoredState) {
    state = UUID.randomUUID().toString().replace("-", "");
    isStoredState = storeState(state);
}

String platform = source.split("_")[0];
boolean isBrokerPage = platform.equals(Constants.SOURCE_PLATFORM_ANDROID) ||
    platform.equals(Constants.SOURCE_PLATFORM_IOS);

String prefix = platform.equals(Constants.SOURCE_PLATFORM_APP) ?
    (ticketID.isEmpty() ? iamUrlScheme + Constants.API_GET_QR_APP :
    iamUrlScheme + Constants.API_ANONYMOUS_FORM_FILLING_REQUEST_APP) :
    iamDomain + Constants.API_GET_QR_WEB;
String scope = URLEncoder.encode(ticketID.isEmpty() ? Constants.TOKEN_SCOPE :
    Constants.TOKEN_SCOPE_ANON, StandardCharsets.UTF_8);
String redirectUri = URLEncoder.encode(redirect, StandardCharsets.UTF_8);

String uri = prefix + "responseType=code&scope=" + scope + "&clientID=" +
    clientId + "&source=" + source + "&redirectURI=" + redirectUri +
    "&state=" + state;
```

API - RequestToken - 1

Online service use this API to retrieve accessToken and openID.
Location: RequestTokenController.java(requestToken)



CEK

jsonObj

Constructing those parameters for calling the get token api.

```java
@GetMapping("/token/{state}/{code}")
public ResponseEntity<Result<ApiRespBase>> requestToken(
        @PathVariable String state,
        @PathVariable String code
) {
    System.out.println("get token, state=" + state);
    StateToken stateToken = findStateToken(state);
    if (stateToken == null) {···
    System.out.println("state ok, " + state);

    if (stateToken.getOpenId() != null) {···
    System.out.println("state not used, " + state);

    ObjectMapper objectMapper = new ObjectMapper();

    byte[] cek = cekObj.getPrivateKey();

    ObjectNode jsonObj = objectMapper.createObjectNode();
    jsonObj.put("code", code);
    jsonObj.put("grantType", Constants.GRANT_TYPE);

    String respJson = callApi(Constants.API_GET_TOKEN, jsonObj, cek);

    System.out.println("call api ok" + state);

    ApiRespDecrypted<ApiContentToken> resp = new ApiRespDecrypted<>(respJson, cek);
```

API - RequestToken - 2

Online service use this API to retrieve accessToken and openID.
Location: RequestTokenController.java(requestToken)

Convert into different format

Get the accessToken and openID

Try to store the accessToken for reuse or re-authentication

Match the openID to find isLinked. If not, require for the profile

```
System.out.println(resp);

// Avoid from java.lang.ClassCastException: java.util.LinkedHashMap cannot be cast to X
JsonNode respNode = objectMapper.convertValue(resp, JsonNode.class);
JsonNode respNodeContent = respNode.get("content");

if (respNodeContent == null || respNodeContent.isEmpty()) {···

System.out.println("No error from iAM Smart");

ApiContentToken content = objectMapper.convertValue(respNodeContent, ApiContentToken.class);
System.out.println(content.getScope());
String accessToken = content.getAccessToken();
String openId = content.getOpenId();

boolean storeSuccess = false;
if (accessToken != null && openId != null && !accessToken.isEmpty() && !openId.isEmpty()) {
    storeSuccess = storeStateToken(content, state);
}

System.out.println("No error to storeSuccess");

if (!storeSuccess) {···

String respCode = resp.getCode();
if (respCode == null || !respCode.equals(Constants.SUCCESS_CODE)) {···

boolean isLinked = findUserByOpenId(openId);
if (!isLinked) {···

System.out.println("End");

return ResponseEntity.ok(new Result<>("SUCCESS", new ApiRespBase(resp)));
```

Form Filling (with Service Login and Anonymous)

API - Form Filling with Service Login

Calling requestEme method require for profileFields and eMeFields at the same time
Location: RequestFormFillingController.java (requestEme)

Asking for accessToken and openID to confirm the service login status

```
IamResponseToken iamResponseToken = findTokenByState(state);
if (iamResponseToken == null) {
    return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new Result<>("state invalid", null));
}

String accessToken = iamResponseToken.getAccessToken();
String openId = iamResponseToken.getOpenId();

if (accessToken == null || openId == null || accessToken.isEmpty() || openId.isEmpty()) {
    return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(new Result<>("state invalid", null));
}
```

Constructing the json as a parameter to call the form filling request api

```
ObjectMapper objectMapper = new ObjectMapper();

byte[] cek = cekObj.getPrivateKey();

ObjectNode jsonObj = objectMapper.createObjectNode();
jsonObj.put("state", state);
jsonObj.put("accessToken", accessToken);
jsonObj.put("openID", openId);
jsonObj.put("businessID", UUID.randomUUID().toString().replace("-", ""));
jsonObj.put("redirectURI", iamCallbackEndpoint + Constants.API_FORM_FILLING_CALLBACK);
jsonObj.put("source", source);
jsonObj.putPOJO("profileFields", profileFields);
jsonObj.putPOJO("eMEFields", eMeFields);

String respJson = callApi(Constants.API_FORM_FILLING_REQUEST, jsonObj, cek);
ApiRespDecrypted<ApiContentRequest> resp = new ApiRespDecrypted<>(respJson, cek);
```

API – Open "iAM Smart" For Form Filling

Online Service App invokes “iAM Smart” Mobile App using URL Scheme with request parameters.

The context links deeply to redirect users to the “iAM Smart” mobile app.

Using the same device as an example

Online service app

<context>

iAM Smart mobile app

API - Form Filling Callback from "iAM Smart" Server

Calling responseEme method for form filling callback from server
Location: ResponseFormFillingController.java (responseEme)

Getting content and key for decryption from the body

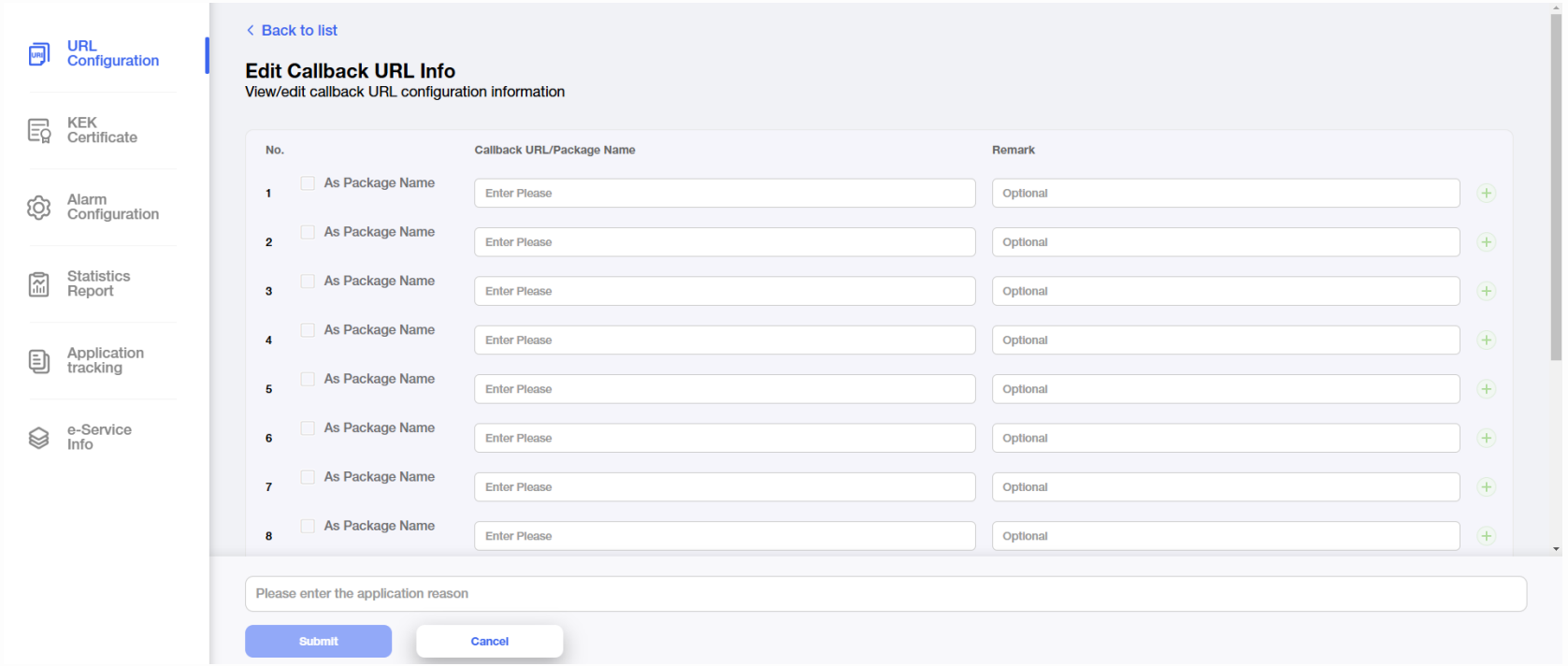
Try to decryption the content

Read the value from the content and store it

```java
@PostMapping(Constants.API_FORM_FILLING_CALLBACK)
public ResponseEntity<Object> responseEme(
        @RequestBody CallbackRespBase<String> body
) {
    String secretKey = body.getSecretKey();
    String content = body.getContent();

    if (secretKey == null || content == null || secretKey.isEmpty() || content.isEmpty()) {…

    byte[] cek = Security.decryptCek(secretKey);
    String respJson = Security.decrypt(content, cek);
    if (respJson == null || respJson.isEmpty()) {…

    ObjectMapper objectMapper = new ObjectMapper();
```

```java
ApiRespSuccess<CallbackContentEme> resp = null;
try {
    resp = objectMapper.readValue(respJson, new TypeReference<>() {
    });
} catch (JsonProcessingException e) {
    e.printStackTrace();
}

if (resp == null) {…

boolean storeSuccess = storeEme(respJson, resp);
```

Self service portal registration - redirectURI



The redirectURI should be registered in self-service portal

Anonymous Form Filling API

Anonymous Form Filling Link

Request "Profile Field" and "e-ME Field"

Device Type

Different Device

"iAM Smart" QR Page/ Broker Page

Browser

Same Device

"iAM Smart" Authorisation Page

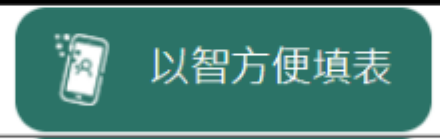
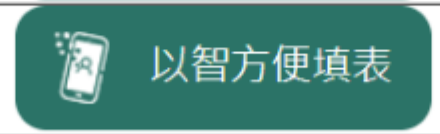
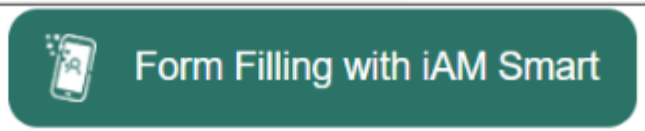
Device Type

Online Service App

UI Requirements

- Button

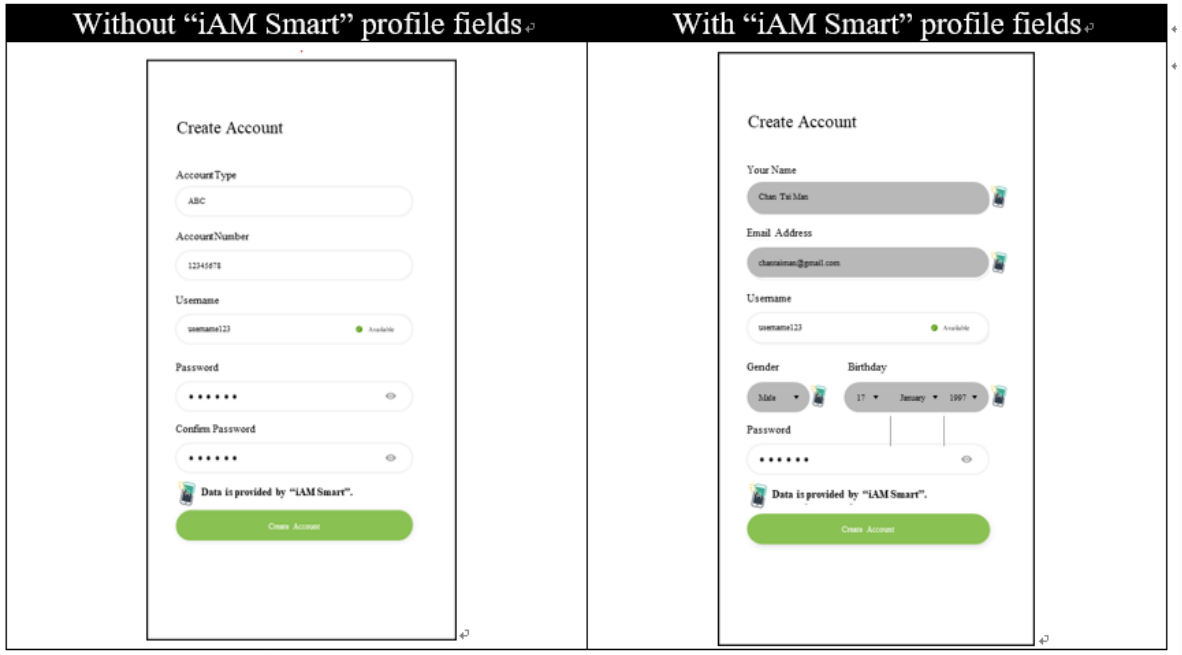
Online services shall use the text provided below for different functions according to the request for retrieving data from "iAM Smart e-ME" or "iAM Smart" profile.

Button Details	
Usage	The button for: - Open the iAM Smart broker page - Form filling with iAM Smart
Traditional Chinese	以智方便填表
Simplified Chinese	以智方便填表
English	Form Filling with iAM Smart
Sample Button	
Traditional Chinese	
Simplified Chinese	
English	

UI Requirements

- “iAM Smart” Tag

For online services using the “iAM Smart” form filling function, it is required to add the below legend in the form to remind users the information is provided by “iAM Smart”.



English	Data is provided by “iAM Smart”.
Traditional Chinese	該項資料由「智方便」提供。
Simplified Chinese	该项资料由「智方便」提供。

API – Initiate Anonymous Form Filling

Online service use this API to request anonymous form filling.
Location: RequestFormFillingController.java (requestAnonymousFormFilling)

Adding the businessID into jsonObj

```java
@GetMapping("/anonymous/formFilling/initiate")
public ResponseEntity<Result<ApiRespSuccess<ApiContentRequest>>> requestAnonymousFormFilling(
        @RequestParam(value = "eMeFields", defaultValue = "") String[] eMeFields,
        @RequestParam(value = "profileFields", defaultValue = "") String[] profileFields
) {

    if ((eMeFields == null || eMeFields.length == 0) && (profileFields == null || profileFields.length == 0)) {…

    ObjectMapper objectMapper = new ObjectMapper();

    byte[] cek = cekObj.getPrivateKey();

    ObjectNode jsonObj = objectMapper.createObjectNode();

    // Record current business ID
    String uuid = UUID.randomUUID().toString().replace("-", "");
    Constants.businessIdAnonymousFormFilling.add(uuid);
    jsonObj.put("businessID", uuid);
    System.out.println("Business ID Added: " + Constants.businessIdAnonymousFormFilling.toString());
```

Adding the eMeFields and profileFields into jsonObj

```java
if (eMeFields != null && eMeFields.length > 0){
    jsonObj.putPOJO("eMEFields", eMeFields);
}
if (profileFields != null && profileFields.length > 0) {
    jsonObj.putPOJO("profileFields", profileFields);
}
```

Calling the API with jsonObj and CEK variables

```java
String respJson = callApi(Constants.API_ANONYMOUS_FORM_FILLING_REQUEST, jsonObj, cek);
ApiRespDecrypted<ApiContentRequest> resp = new ApiRespDecrypted<>(respJson, cek);

String respCode = resp.getCode();
if (respCode != null && respCode.equals(Constants.SUCCESS_CODE)) {
    return ResponseEntity.ok(new Result<>("SUCCESS", resp, uuid));
}
```

API – GetQR

GetQR would be a usual API for authenticate user by calling out a broker page or QR page. After authorization by user, the page will be redirected to the **redirectURI** with **authCode** and **state** parameters.
Location: GetQrUriController.java (getQrUri)

state

redirectURI

redirectURI with authCode and state

```
String state = UUID.randomUUID().toString().replace("-", "");
boolean isStoredState = storeState(state);
while (!isStoredState) {
    state = UUID.randomUUID().toString().replace("-", "");
    isStoredState = storeState(state);
}

String platform = source.split("_")[0];
boolean isBrokerPage = platform.equals(Constants.SOURCE_PLATFORM_ANDROID) ||
    platform.equals(Constants.SOURCE_PLATFORM_IOS);

String prefix = platform.equals(Constants.SOURCE_PLATFORM_APP) ?
    (ticketID.isEmpty() ? iamUrlScheme + Constants.API_GET_QR_APP :
    iamUrlScheme + Constants.API_ANONYMOUS_FORM_FILLING_REQUEST_APP) :
    iamDomain + Constants.API_GET_QR_WEB;
String scope = URLEncoder.encode(ticketID.isEmpty() ? Constants.TOKEN_SCOPE :
    Constants.TOKEN_SCOPE_ANON, StandardCharsets.UTF_8);
String redirectUri = URLEncoder.encode(redirect, StandardCharsets.UTF_8);

String uri = prefix + "responseType=code&scope=" + scope + "&clientID=" +
    clientId + "&source=" + source + "&redirectURI=" + redirectUri +
    "&state=" + state;
```

API – RequestToken

Unlike the Form Filling API with Service Login. The Anonymous Form Filling API would only request the access token for **ONCE**.

accessToken and openID request only in form filling with service login

Since it is the one-time request of accessToken, the online service perform the token request in controller.
Location:
RequestFormFillingController.java
(requestAnonymousFormFillingResult)

```java
@GetMapping("/formFilling/initiate/{state}")
public ResponseEntity<Result<ApiRespSuccess<ApiContentRequest>>> requestEme(
        @PathVariable String state,
        @RequestParam(value = "source", defaultValue = "") String source,
        @RequestParam(value = "profileFields", defaultValue = "") String[] profileFields,
        @RequestParam(value = "eMeFields", defaultValue = "") String[] eMeFields
) {
    if (source.isEmpty()) {···

    if ((eMeFields == null || eMeFields.length == 0) && (profileFields == null || profileFiel

    IamResponseToken iamResponseToken = findTokenByState(state);
    if (iamResponseToken == null) {···

    String accessToken = iamResponseToken.getAccessToken();
    String openId = iamResponseToken.getOpenId();

    if (accessToken == null || openId == null || accessToken.isEmpty() || openId.isEmpty()) {
```

RequestFormFillingController.java (requestEme)

accessToken

openId

```java
ApiContentToken content = objectMapper.convertValue(respNodeContent, ApiContentToken.class);
System.out.println(content.getScope());
String accessToken = content.getAccessToken();
String openId = content.getOpenId();
```

API - Request Anonymous Form Filling Result

The whole process of requesting anonymous form filling result
Location: RequestFormFillingController.java (requestAnonymousFormFillingResult)

Obtain CEK and token by calling API

```
byte[] cek = cekObj.getPrivateKey();

ObjectNode jsonObj = objectMapper.createObjectNode();
jsonObj.put("code", code);
jsonObj.put("grantType", Constants.GRANT_TYPE);

String respJson = callApi(Constants.API_GET_TOKEN, jsonObj, cek);
```

Get anonymous form filling result by matching accessToken and openId

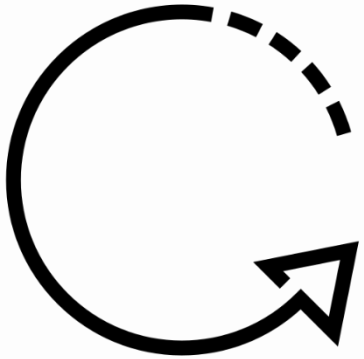
```
// Request Form Filling result
jsonObj = objectMapper.createObjectNode();
jsonObj.put("accessToken", accessToken);
jsonObj.put("openID", openId);

respJson = callApi(Constants.API_ANONYMOUS_FORM_FILLING_RESULT, jsonObj, cek);
ApiRespDecrypted<ApiContentRequest> eMeResp = new ApiRespDecrypted<>(respJson, cek);
```

Practical Tips

Tip 1: Broker Page not applicable for App in same device

Here are some tips for you

- Check App Permissions
- Update the App and browser
- Network Connection
- Clear App Cache
- Review App Configuration
- Contact Support



Tip 2: Account link up with username password

Users are reminded to remember their own usernames and passwords.

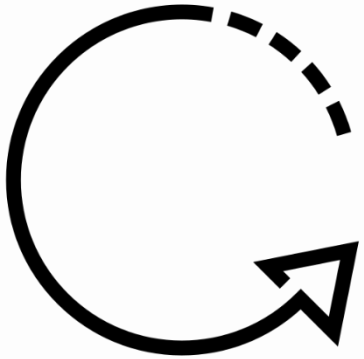
If a user lost his / her username or password unfortunately, user may

- Get help by clicking the “Forget Password” button on the e-Service platform
- Get help by direct login from “iAM Smart”
- Contact Support

Tip 3: Broker Page not applicable for App in same device

Here are some tips for you

- Check App Permissions
- Update the App and browser
- Network Connection
- Clear App Cache
- Review App Configuration
- Contact Support



Quiz

Question:
Can the accessToken in Anonymous Form Filling be re-used?

A. Cannot be re-used
B. Can be re-used once
C. Can be re-used before it is expired
D. Can be re-used permanently

Answer: A

Quiz

Question:
Which statement below is true?

A. Profile fields have more fields than eME fields
B. eME fields are necessary for an “iAM Smart” account
C. eME fields does not relate to profile fields
D. eME fields have more fields than profile fields

Answer: D