

|   |                                                                                              |  |  |
|---|----------------------------------------------------------------------------------------------|--|--|
| 4 | User authentication completes if linkage found in Step 3 and e-Service permits CorpID login. |  |  |
| 5 | e-Service keeps the cAccessToken for subsequent access to CorpID APIs.                       |  |  |

### 1.4.3 Form pre-filling with Service Login

The cAccessToken of the CorpID user that possessed by e-Service should include the authorisation scope of Profiles authorisation. e-Service can request for account information in corpProfileFields and eCorpFields of CorpID user with the following steps:

| Step | Description                                                                                                                                                                      | “iAM Smart” API | Callback API |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|--------------|
| 1a   | e-Service provides a link for form pre-filling in the application.                                                                                                               |                 |              |
| 1b   | Alternatively, CorpID user can login the e-Service via CorpID service catalogue                                                                                                  |                 |              |
| 2a   | e-Service requests for cAuthCode with required authorisation scopes (e.g., CorpID authentication, form pre-filling authorisation, etc.).                                         | 3, 5            | A, B         |
| 2b   | Alternatively, if CorpID user login the e-Service via CorpID service catalogue, e-Service would receive the cAuthCode accordingly.                                               |                 | H            |
| 3    | e-Service gets cAccessToken and Tokenised ID (cOpenID) of CorpID user.                                                                                                           | 4               |              |
| 4    | e-Service requests corpProfileFields, eCorpFields and nonHKResidentFields for the purpose of verification and form pre-filling respectively from CorpID System via Profiles API. | 20              |              |
| 5    | CorpID System notes receipt of e-Service's request.                                                                                                                              |                 |              |
| 6    | e-Service server receives form pre-filling result response from CorpID System                                                                                                    |                 |              |

### 1.4.4 Digital Signing with Service Login

The cAccessToken of the CorpID user that possessed by e-Service should include the authorisation scope of digital signing authorisation. e-Service can request for CorpID digital signing with the following steps:

| Step | Description                                                                                                                                                                                                                                                                                                                                          | CorpID API | Callback API |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------------|
| 1    | e-Service provides digital signing using CorpID in e-Service application.                                                                                                                                                                                                                                                                            |            |              |
| 2    | e-Service requests for cAuthCode with required authorisation scopes (e.g., CorpID authentication, Digital Signing authentication).                                                                                                                                                                                                                   | 3, 5       | A, B         |
| 3    | e-Service gets cAccessToken and Tokenised ID (cOpenID) of CorpID user.                                                                                                                                                                                                                                                                               | 4          |              |
| 4    | e-Service calculates hash of the document (or digest for pdf document) to be signed and requests digital signing from CorpID System.                                                                                                                                                                                                                 | 9,12       |              |
| 5    | CorpID System notes receipt of e-Service's request.                                                                                                                                                                                                                                                                                                  |            |              |
| 6    | e-Service generates a 4-digit identification code using hash of document (or digest for pdf document) and hash of Tokenised ID.                                                                                                                                                                                                                      |            |              |
| 7a   | When e-Service client terminal and CorpID Mini-program are in different device, e-Service client terminal shows the 4-digit identification code with instructions to inform CorpID user to verify the identification code and complete the CorpID digital signing in CorpID Mini-program and polls e-Service server for response from CorpID System. |            |              |
| 7b   | When e-Service client terminal and CorpID Mini-program are in same device, e-Service client terminal shows a 4-digit identification code and invokes CorpID Mini-program and polls e-Service server for response from CorpID System.                                                                                                                 | 6          |              |
| 8    | e-Service server receives signing response from CorpID System via e-Service callback API and computes the signed document.                                                                                                                                                                                                                           |            | E, G         |
| 9    | e-Service server verifies and acknowledges digital signing result to CorpID System.                                                                                                                                                                                                                                                                  | 11         |              |
| 10   | e-Service server informs e-Service client terminal to stop polling.                                                                                                                                                                                                                                                                                  |            |              |
| 11   | e-Service saves the accessToken for subsequent access to CorpID APIs.                                                                                                                                                                                                                                                                                |            |              |

### 1.4.5 Anonymous Form pre-filling

Anonymous form pre-filling describes the situation that user initiates a form pre-filling request without prior authentication to e-Service using CorpID. e-Service can request anonymous form pre-filling with the following steps:

| Step | Description                                                                                                                                                                                                    | CorpID API | Callback API |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------------|
| 1    | e-Service provides a link for anonymous form pre-filling in the application.                                                                                                                                   |            |              |
| 2    | e-Service requests corpProfileFields, eCorpFields and nonHKResidentFields for the purpose of verification and form pre-filling respectively from CorpID System.                                                | 13         |              |
| 3    | CorpID System notes receipt of e-Service's request.                                                                                                                                                            |            |              |
| 4a   | When e-Service client terminal and CorpID Mini-program are in different device, <i>e-Service redirects client terminal to CorpID QR/App broker page.</i>                                                       | 3          |              |
| 4b   | When e-Service client terminal and CorpID Mini-program are in the same device, e-Service client terminal invokes CorpID Mini-program, e-Service App polls e-Service server for response from CorpID System.    | 3, 6       |              |
| 5    | <del>e-Service receives authCode from CorpID System via e-Service callback API.</del>                                                                                                                          |            | A, B         |
| 6    | e-Service gets accessToken and Tokenised ID (cOpenID) of CorpID user.                                                                                                                                          | 4          |              |
| 7    | e-Service server obtains authorised information of corpProfileFields, eCorpFields and nonHKResidentFields from CorpID System using cAccessToken and Tokenised ID (cOpenID) and processes the form pre-filling. | 14         |              |
| 8a   | When e-Service client terminal is browser, e-Service server redirects the browser to view the form pre-filling result.                                                                                         |            |              |
| 8b   | When e-Service client terminal is e-Service App, e-Service server informs e-Service App to stop polling and e-Service App shows the form pre-filling result.                                                   |            |              |

### 1.4.6 Anonymous Digital Signing

Anonymous digital signing describes the situation that user initiates a digital signing request without prior authentication to e-Service using CorpID. e-Service can request anonymous digital signing with the following steps:

| Step | Description                                                                                                                                                                                                                                                                                                                            | “CorpID<br>” API | Callback<br>API |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|-----------------|
| 1    | e-Service provides a link for anonymous digital signing in the application.                                                                                                                                                                                                                                                            |                  |                 |
| 2    | e-Service calculates hash of the document (or digest for pdf document) to be signed and hash of the HKIC number, and requests digital signing from CorpID System.                                                                                                                                                                      | 15, 17           |                 |
| 3    | CorpID System notes receipt of e-Service's request.                                                                                                                                                                                                                                                                                    |                  |                 |
| 4    | e-Service generates a 4-digit identification code using hash of document (or digest for pdf document), hash of the HKIC number and hash of clientID of the e-Service.                                                                                                                                                                  |                  |                 |
| 5a   | When e-Service client terminal and CorpID Mini-program are in different devices, e-Service client terminal shows the 4-digit identification code. For browser, e-Service redirects the browser to CorpID QR/App broker page. For e-Service App, it polls e-Service server for response from CorpID System.                             | 3                |                 |
| 5b   | When e-Service client terminal and CorpID Mini-program are in the same device, e-Service client terminal shows a 4-digit identification code. For browser, e-Service redirects the browser to CorpID QR/App broker page. For e-Service App, it invokes CorpID Mini-program and polls e-Service server for response from CorpID System. | 3, 6             |                 |
| 6    | <del>e-Service receives authCode from CorpID System via e-Service callback API.</del>                                                                                                                                                                                                                                                  |                  | A, B            |
| 7    | e-Service gets cAccessToken and Tokenised ID (cOpenID) of CorpID user.                                                                                                                                                                                                                                                                 | 4                |                 |
| 8    | e-Service server obtains digital signing result from CorpID System using cAccessToken and Tokenised ID (cOpenID) and computes the signed document.                                                                                                                                                                                     | 16, 18           |                 |
| 9    | e-Service server verifies and confirms digital signing result to CorpID System.                                                                                                                                                                                                                                                        | 11               |                 |

|     |                                                                                                                                                             |  |  |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| 10a | When e-Service client terminal is browser, e-Service server redirects the browser to view digital signing result.                                           |  |  |
| 10b | When e-Service client terminal is e-Service App, e-Service server informs e-Service App to stop polling and e-Service App shows the digital signing result. |  |  |

## 2. SYSTEM WORKFLOW USING “CORPID”

There are two types of system workflows using CorpID:

1. “Authentication to e-Services using CorpID” for e-Service to authenticate CorpID user who gives authorisation to e-Service to get the cAccessToken to access the required CorpID APIs (e.g., user authentication).
2. “e-Service business operations using CorpID” for e-Service to access the required CorpID APIs for completing its business operations (e.g., form pre-filling).

For details, please refer to corresponding system workflows described in Section 3.

## 3. IMPLEMENTATION PROCEDURE

The CorpID Platform provides a comprehensive testing environment designed specifically for selected scenarios for web-based applications. The workflow of mobile applications is similar to web-based applications. For details, please refer to the CorpID API Specifications.

It offers four dedicated end-to-end workflows that allow developers to conduct testing before the CorpID system launch. These workflows combine CorpID APIs, a CorpID mini-program, “iAM Smart” APIs, and the “iAM Smart” testing mobile app to simulate the complete user journey with e-Service’s web application.

With these workflows, developers can test the core CorpID functions including CEK/KEK encryption, Corporate Identity Verification, form pre-filling (with and without service login), and electronic signing in the CorpID Platform.

Remarks: The steps outline in the following sections are subject to change. For the most up-to-date information and technical details of each of these APIs, please refer to the latest API specifications.

### 3.1 PREREQUISITE

e-Service should review and fulfill the prerequisites outlined in this section before testing the integration with the CorpID Platform.

- (a) Work out the usage of CorpID for the e-Service application (e.g., form pre-filling, digital signing, etc.) and determine the authorisation scopes (e.g., form pre-filling authorisation for form pre-filling) required.
- (b) Design the lifecycle management of the API data encryption/decryption key (e.g., get, renewal).
- (c) Apply the digital certificates for encipherment as KEK (i.e., for API data encryption/decryption key).
- (d) Study the specifications of relevant CorpID APIs, “iAM Smart” APIs and e-Service callback APIs from CorpID Platform and “iAM Smart” Platform.
- (e) Prepare the redirect URIs and/or URL scheme (or Universal Link/App Link) (for invoking e-Service App to return authorisation code to e-Service server) for each e-Service callback endpoint to be implemented for the e-Service application.
- (f) Work out and enable the bi-directional network connectivity between e-Service server and CorpID System, and between e-Service server and “iAM Smart” System.
- (g) Register e-Service application to CorpID System and get the corresponding CorpID client ID, CorpID client secret and/or URL scheme (or Universal Link/App Link) (for invoking CorpID Mini-program).
- (h) Register e-Service application to “iAM Smart” System and get the corresponding “iAM Smart” client ID, “iAM Smart” client secret and/or URL scheme (or Universal Link/App Link) (for invoking “iAM Smart” Mobile App).
- (i) Configure client authentication for CorpID System in e-Service callback endpoint (optional).
- (j) Configure client authentication for “iAM Smart” System in e-Service callback endpoint (optional).
- (k) Study the relevant system workflows and start implementation.

## 3.2 E-SERVICE AND “CORPID” SYSTEM INTERACTION DESIGN

### 3.2.1 Interaction between e-Service and CorpID Mini-program

The CorpID API is designed to provide better user experience when two forms of e-Service client terminal, e-Service App and e-Service Website (i.e., browser) are interacting with CorpID Mini-program in the same device and different devices. There are three scenarios for the interactions:

(1) ***for authentication using CorpID*** (2) ***for authorisation of anonymous CorpID API access*** (e.g., Anonymous Form pre-filling) and (3) ***for performing e-Service business operations with authentication using CorpID*** (e.g., Form pre-filling by CorpID) as follows:

1. For authentication using CorpID, e-Service should invoke the CorpID API “Request QR Page”. By default, this API returns a webpage with a QR code for scanning by CorpID Mini-program in different device.
  - When e-Service detects that the e-Service client terminal is a browser running in personal computer (“PC”), the API request parameter “brokerPage” can be kept as “false” and QR Code webpage will be shown.

- When e-Service detects that the e-Service client terminal is a browser running in mobile platform, the API request parameter “brokerPage” can be set to “true”. The broker page of CorpID System will show a button to invoke the CorpID Mini-program. If CorpID Mini-program is not installed in the same device, QR Code webpage will be shown.
  - When e-Service client terminal is e-Service App, it should detect the existence of CorpID Mini-program in the device using program code. If the CorpID Mini-program is installed in the same device, e-Service App should invoke CorpID Mini-program using URL Scheme with the API parameters. Otherwise, e-Service App should make use the device browser to invoke the CorpID API with “brokerPage” set to “false” such that QR Code webpage will be shown.
2. For authorisation of anonymous CorpID API access, e-Service will first invoke the relevant anonymous CorpID APIs and get the “ticketID”. e-Service will then invoke the CorpID API “Request QR Page” using the “ticketID” to get the accessToken and Tokenised ID (cOpenID) for obtaining the result of the relevant anonymous CorpID API. Same detection for client terminal by e-Service as in authentication using CorpID, is applied.
  3. To perform e-Service business operations with authentication using CorpID such as “Form pre-filling”, CorpID System will return the immediate API response with parameters “authByQR” and “ticketID” (optional) to e-Service server before the final result is returned to e-Service using e-Service callback API. e-Service should determine the next action based on “authByQR”:
    - When it is “false”, e-Service client terminal (both browser and e-Service App) should invoke the CorpID Mini-program using URL Scheme with the “ticketID”. e-Service client terminal should keep polling e-Service server for the result returned to relevant e-Service callback API.
    - Otherwise, e-Service client terminal should show instructions to inform “iAM Smart” user to complete the request in CorpID Mini-program (CorpID System will also notify CorpID Mini-program using push message) and keep polling e-Service server for the result returned to relevant e-Service callback API.

The table below summarises the scenarios:

| Scenario                                                                 | e-Service App/”CorpID” Mini-program                                                                                                                                                    | e-Service Website/”CorpID” Mini-program                                                                                                                            |
|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Authentication / Authorisation for anonymous CorpID API<br>(Same device) | <ul style="list-style-type: none"> <li>● Determine CorpID Mini-program existence by program code</li> <li>● Invoke CorpID Mini-program using URL scheme with API parameters</li> </ul> | <ul style="list-style-type: none"> <li>● Determine browser platform by program code</li> <li>● Invoke “Request QR Page” with “brokerPage” set to “true”</li> </ul> |

|                                                                                        |                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Authentication /<br>Authorisation for<br>anonymous CorpID<br>API<br>(Different device) | <ul style="list-style-type: none"> <li>• Determine CorpID Mini-program existence by program code</li> <li>• Invoke “Request QR Page” using device browser with “brokerPage” set to “false”</li> </ul>                         | <ul style="list-style-type: none"> <li>• Determine browser platform by program code: <ul style="list-style-type: none"> <li>– If browser in PC, invoke “Request QR Page” with “brokerPage” set to “false”</li> <li>– If browser in mobile, invoke “Request QR Page” with “brokerPage” set to “true”</li> </ul> </li> </ul> |
| Business operations<br>with authentication<br>using CorpID (Same<br>device)            | <ul style="list-style-type: none"> <li>• Determine by “authByQR” is “false”</li> <li>• Invoke CorpID Mini-program using URL scheme with “ticketID”</li> </ul>                                                                 |                                                                                                                                                                                                                                                                                                                            |
| Business operations<br>with authentication<br>using CorpID<br>(Different device)       | <ul style="list-style-type: none"> <li>• Determine by “authByQR” is “true”</li> <li>• Show instructions to inform CorpID user to complete the request in CorpID Mini-program and keep polling e-Service server for</li> </ul> |                                                                                                                                                                                                                                                                                                                            |
|                                                                                        | returned to relevant e-Service callback API                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                            |

Specific for the situation when CorpID Mini-program and e-Service App are in the same device,

- (a) In the scenario (2) when invoking CorpID Mini-program (i.e., CorpID API: Open CorpID Mini-program for Getting Context), e-Service App is required to provide “source” and “redirectURI” parameters. e-Service can specify “source” parameter as “App\_Scheme” (for URL Scheme) or “App\_Link” (for iOS Universal Link/Android App Link) depending on its implementation method and the information submitted at e-Service registration. The “source” provides information for how CorpID Mini-program can invoke e-Service App to return e-Service the authCode for authorisation of anonymous CorpID API access (i.e., CorpID API: Callback with authCode to e-Service App):

- If e-Service specify “source” = “App\_Scheme”, then CorpID System will consider it as e-Service App and using URL Scheme to invoke the e-Service App. e-Service should specify the corresponding URL Scheme in “redirectURI” and CorpID System will verify it with the URL Scheme submitted at e-Service registration.
- If e-Service specify “source” = “App\_Link”, then CorpID System will consider it as e-Service App and using Universal Link/App Link to invoke the e-Service App. e-Service should specify the corresponding Universal Link/App Link in “redirectURI” and CorpID System will verify it with the Universal Link/App Link submitted at e-Service registration.

- (b) In the scenario (3) when e-Service backend invokes the required CorpID API (e.g., CorpID API: Request Form pre-filling), e-Service backend is required to provide

“source” parameter. e-Service can specify “source” parameter as “App\_Scheme” (for URL Scheme) or “App\_Link” (for iOS Universal Link/Android App Link) depending on its implementation method and the corresponding scheme or link information submitted to CorpID System during e-Service registration. CorpID System will query such scheme or link information using the “source” provided for CorpID Mini-program to invoke e-Service App.

### 3.2.2 Process of Common API Request and Response Parameters

There are common parameters defined in the HTTP request header, HTTP request body of CorpID POST APIs. Common parameters are also defined in the CorpID POST API response body and e-Service callback API request body. This section describes how these common parameters should be prepared and processed by e-Service.

Relevant information can be found in the CorpID API Specification.

Common parameters in the HTTP request header and body of CorpID POST APIs:

| Request        | Parameters      | Description                                                                                                                                                                                                                                                                                                                                                                        |
|----------------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Request Header | clientID        | The Client Identity provided by CorpID System during e-Service registration.                                                                                                                                                                                                                                                                                                       |
|                | signatureMethod | Name of signature algorithm used to compute the POST request signature. Currently only support HmacSHA256 and its value must be set to “HmacSHA256”, case sensitive.                                                                                                                                                                                                               |
|                | timestamp       | The timestamp is a number expressed in the number of milliseconds since January 1, 1970 00:00:00 GMT. It must be greater than or equal to the timestamp of previous request submitted by e-Service. CorpID System compares the timestamp with system time and treats it as valid when the difference is within 60 seconds.                                                         |
|                | nonce           | A unique, non-repetitive random string for each request submitted by e-Service. It is used with timestamp to prevent Replay Attack. Its value cannot be repeated within 60 seconds.                                                                                                                                                                                                |
|                | signature       | It is a HmacSHA256 digital signature signed by e-Service using its client secret (paired with the clientID) provided by CorpID System during e-Service registration. The object to be signed includes the parameters concatenating in the following sequence: <i>clientID + signatureMethod + timestamp + nonce + encrypted request body</i><br>All parameters are case sensitive. |

|              |         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Request Body | content | <p>Its value is created using API data encryption. It is BASE64 encoded with the following components concatenating in the following sequence: Content value: <i>4 bytes IV length + IV + ciphertext</i></p> <p>IV is the initialisation vector of the encryption algorithm “AES/GCM/NoPadding” used in API data encryption. The ciphertext is the encrypted textual content of request body of the CorpID POST API. Details of API data encryption can be referred to Section 3.3.</p> |
|--------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Common parameters in the CorpID POST API response body and e-Service callback API request body:

Common parameters in JSON data of the CorpID APIs request body:

| Request/ Callback | Parameters  | Description                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Request Body      | businessID  | It is a unique identifier generated by e-Service for identifying the business request of e-Service for invoking the CorpID API. It will be returned to e-Service via the corresponding e-Service callback API. For e-Service, it is used to match the result returned via e-Service callback API and thus should not be repeated. It is recommended to use 36-byte UUID number (ASCII character set) for this parameter.           |
|                   | accessToken | It is the accessToken e-Service retained after authentication of the CorpID user and must be still valid. accessToken can be used multiple of times before expiry except that requested through the anonymous CorpID APIs which can only be used once.                                                                                                                                                                             |
|                   | openID      | It is the Tokenised ID (cOpenID) of the CorpID user. e-Service should ensure the accessToken and Tokenised ID (cOpenID) are in pair and belongs to the target CorpID user for the request.                                                                                                                                                                                                                                         |
|                   | source      | It can be “App_Scheme” (for URL Scheme), “App_Link” (for iOS Universal Link/Android App Link), or the user agent’s name value of the browser. Please refer to Appendix B of the CorpID API Specification for the list of source values. It is used by CorpID System to determine how to process the interaction with the e-Service client terminal after receiving the CorpID API call. Please refer to Section 3.2.1 for details. |
|                   | redirectURI | It is the callback URI of e-Service. It is used by CorpID System to return the result via e-Service callback API and its value must match with the value provided by e-Service during registration to CorpID System.                                                                                                                                                                                                               |
|                   | state       | It is an optional parameter to prevent CSRF attack. The value of “state” is defined and managed by e-Service. It is recommended to use 36-byte UUID number (ASCII character set) or less with random value. It will be returned to e-Service via the corresponding callback API and e-Service is suggested to verify this value to prevent CSRF attack.                                                                            |

### 3.2.3 Process of Common Errors Returned from CorpID APIs

There are two tiers of errors returned from CorpID System: HTTP error and application error returned from CorpID APIs (given in parameter “code” or “error\_code” in the POST response, e-Service callback API and URL Scheme). For application error, the parameter “message” presents the error description.

Common application errors are listed below:

| Response Code                           | Message/ Description                                                                                                                    |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| M00000                                  | SUCCESS                                                                                                                                 |
| <b>Common Error Code</b>                |                                                                                                                                         |
| M10001                                  | The application is suspended. System should redirect to the “Service suspended” page.                                                   |
| M99990                                  | Unhandled exception. System should redirect to the “Unhandled exception” page.                                                          |
| M99991                                  | Cannot send the email notification. Please try again.                                                                                   |
| M99992                                  | Authorization failed. Applicant is not permitted to submit the request.                                                                 |
| S00001                                  | Database error                                                                                                                          |
| C00001                                  | Connection refused by backend API server (core)                                                                                         |
| M20000                                  | unknown exception                                                                                                                       |
| M20001                                  | parameter {%s} is missing                                                                                                               |
| M20002                                  | empty parameter {%s}                                                                                                                    |
| M20003                                  | invalid parameter {%s}                                                                                                                  |
| M20004                                  | duplicated request                                                                                                                      |
| M20005                                  | unsupported signature method                                                                                                            |
| M20006                                  | signature verification failed                                                                                                           |
| M20007                                  | unsupported source                                                                                                                      |
| M20008                                  | invalid e-service URL<br>The callback URL provided by e-service is not registered in the ESP portal                                     |
| M20009                                  | accessToken not exist or expired                                                                                                        |
| M20010                                  | cOpenID not exist                                                                                                                       |
| M20011                                  | duplicated businessID                                                                                                                   |
| M20012                                  | insufficient permission for e-service / Forbidden<br>e-service shall check the scope or API access control in the ESP portal            |
| M20015                                  | scope mismatch<br>e-service shall verify the scopes approved for the client id in ESP and the scopes configured in e-service catalogue. |
| <b>Encryption/Decryption Error Code</b> |                                                                                                                                         |
| M30001                                  | key encryption key not exist or expired                                                                                                 |
| M30002                                  | content encryption key not exist or expired                                                                                             |
| M30003                                  | encryption exception                                                                                                                    |
| M30004                                  | decryption exception                                                                                                                    |
| <b>Authentication Error Code</b>        |                                                                                                                                         |
| M40000                                  | user cancelled authentication request                                                                                                   |
| M40001                                  | user rejected authentication request                                                                                                    |
| M40002                                  | failed to authenticate                                                                                                                  |
| M40003                                  | authentication request timeout                                                                                                          |
| M40004                                  | cAuthCode not exist or expired                                                                                                          |

| Profile Request Error Code          |                                           |
|-------------------------------------|-------------------------------------------|
| M50001                              | user rejected profile request             |
| M50002                              | failed to request profile                 |
| M50003                              | profile request timeout                   |
| Form Pre-filling Request Error Code |                                           |
| M60000                              | user cancelled Form Pre-filling request   |
| M60001                              | user rejected Form Pre-filling request    |
| M60002                              | failed to request Form Pre-Filling        |
| M60003                              | Form Pre-filling request timeout          |
| Signing Request Error Code          |                                           |
| M70000                              | user cancelled signing request            |
| M70001                              | user rejected signing request             |
| M70002                              | failed to request signing                 |
| M70003                              | signing request timeout                   |
| M70004                              | user not allowed to sign                  |
| M70005                              | inconsistent HKIC number                  |
| M70006                              | failed to process signing acknowledgement |
| M77000                              | failed to sign                            |
| M77001                              | failed to sign with FR                    |
| M77002                              | failed to sign with NFC                   |
| M77003                              | device not supported for NFC              |
| M77004                              | user is still in waiting period           |
| Document Wallet Error Code          |                                           |
| M80000                              | user cancelled sharing request            |
| M80001                              | user rejected signing request             |
| M80002                              | failed to request sharing                 |
| M80003                              | Document sharing request timeout          |
| M88001                              | failed to share                           |

A list of common HTTP errors which may be returned from CorpID System:

| Status Code      | Status description                                                                                          |
|------------------|-------------------------------------------------------------------------------------------------------------|
| 200 OK           | Request successful                                                                                          |
| 302 Found        | The requested resource resides temporarily under a different URI.                                           |
| 400 Bad Request  | The request could not be understood by the server due to malformed syntax.                                  |
| 401 Unauthorised | The request has not been applied because it lacks valid authentication credentials for the target resource. |
| 403 Forbidden    | The server understood the request but is refusing to fulfil it.                                             |

|                           |                                                                                                                   |
|---------------------------|-------------------------------------------------------------------------------------------------------------------|
| 404 Not Found             | The server has not found anything matching the Request-URI.                                                       |
| 413 Payload Too Large     | The uploaded files are larger than 5MB or the request entity is larger than 10MB.                                 |
| 429 Too Many Requests     | The user has sent too many requests in a given amount of time.                                                    |
| 500 Internal Server Error | The server encountered an unexpected condition which prevented it from fulfilling the request.                    |
| 503 Service Unavailable   | The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. |

### 3.3 API DATA ENCRYPTION AND DECRYPTION

HTTPS will be used to secure communication channels between e-Services and CorpID System. However, there are chances that the data transferred by HTTPS could be eavesdropped or manipulated by malicious service providers in some special situations. To protect the data in transit, an additional layer of data encryption will be applied to all APIs POST request and response body (except the API that e-Service request for getting the data encryption key). The response body of all e-Service callback POST APIs invoked by CorpID System will also be encrypted using the same key and method as described in this section. The process is as below:

1. e-Service requests for data encryption key from CorpID System;
2. CorpID System generates an AES256 symmetric data encryption key;
3. CorpID System stores the generated key and returns to the e-Service who initiated the request in secure manner;
4. For subsequent interactions between e-Service and CorpID System, business data will be encrypted by the generated key;
5. Both CorpID System and e-Service will use the same key to decrypt the API data;
6. (Note: For the callback to e-Service, considering that CEK may time out when CorpID System received the request and consequently be regenerated, the encrypted CEK (probably new) will be returned to e-Service in the e-Service callback API. e-Service should use the CEK in callback body for data decryption, but does not retain this CEK.)
7. If the generated key is expired, e-Service has to request a new key and repeat the above process.

#### 3.3.1 Workflow for Encryption and Decryption

The following diagram describes the detailed workflow how e-Service can request data encryption key, perform encryption and decryption with the key, and request for another new encryption key if the existing key is expired:

- Step 1. e-Service requests for data encryption key from CorpID System through invoking CorpID API;  
*CorpID API (POST: Request Symmetric Content Encryption Key);*
- Step 2. CorpID System generates an AES256 content encryption key (“CEK”) for the e-Service and saves it in CorpID System;
- Step 3. CorpID System encrypts the CEK using the public key certificate A (“KEK”) provided by the e-Service in registration to CorpID System based on RSA algorithm, and returns the encrypted CEK, key expiry time to the e-Service;
- Step 4. e-Service decrypts the encrypted CEK using the private key of its KEK and saves it and its key expiry time;
- Step 5. Before e-Service calling APIs of CorpID System, e-Service should check whether its CEK is still valid and use it to encrypt all the JSON data of the POST request body as ciphertext using encryption algorithm “AES/GCM/NoPadding”. This algorithm requires an initialisation vector (“IV”) which is provided by e-Service. The ciphertext, IV and length of IV are then concatenated in the following sequence and BASE64 encoded to formulate the value of JSON name/value pair called “content”:  
*Content value: 4 bytes IV length + IV + ciphertext*
- Step 6-7. After receiving the e-Service request, CorpID System first checks whether the KEK, CEK of the e-Service exist or expired and returns with corresponding error code if required;
- Step 8. CorpID System extracts the initialisation vector IV from the e-Service request and decrypts the JSON data of the POST request body using e-Service's CEK and IV. CorpID System then validates the request parameters and returns error response to e-Service if required;
- Step 9. After CorpID System has processed the request, it generates its initialisation vector IV and encrypts all the JSON data of POST response (except “txID”, “code” and “message”) as ciphertext using the valid e-Service's CEK (i.e., CEK is not expired) and IV by the same encryption algorithm (“AES/GCM/NoPadding”):
- For CorpID POST API response, if CEK does not exist or expires, corresponding error code will be returned.
  - For e-Service callback, if CEK is expired, CorpID System will re-generate a new CEK for the e-Service to perform the encryption.

The ciphertext of encrypted POST response / callback JSON data, IV and length of IV are then concatenated in the following sequence and BASE64 encoded to formulate the value of JSON name/value pair called “content”:

*Content value: 4 bytes IV length + IV + ciphertext*

For e-Service callback, the CEK used for this encryption (e.g., new CEK) will also be encrypted using e-Service's KEK and BASE64 encoded, and will be returned to e-

Service as part of the POST callback body as JSON name/value pair called “secretKey”;

Step 10-11. For the response received, the e-Service should first check whether the return code contains any key-related error code such as “key does not exist or has been expired”. For CorpID POST API response, if there are no such key-related errors, e-Service should use its CEK and IV (extracted from “content”) to decrypt the response data. For e-Service callback, e-Service should decrypt the CEK from “secretKey” parameter in the callback body using its KEK’s private key and then use this CEK and IV (extracted from “content”) to decrypt the callback data. e-Service should request for a new CEK when it is expired.

### **3.3.1.1 Implementing (1) POST: Request Symmetric Content Encryption Key**

#### **Pre-conditions**

- e-Service can access the private key of its KEK.
- e-Service has registered to CorpID System and uploaded its KEK.

#### **Post-conditions**

- e-Service decrypts CEK received from CorpID System using the private key of its KEK and saves the CEK and its key expiry time.

#### **Error conditions**

- For common errors, please refer to Section 3.2.3.

#### **Request and Response Parameters**

- Please refer to CorpID API Specification.

#### **Notes**

- e-Service should ensure the KEK registered to CorpID System is valid and keeps the private key of KEK. The KEK used for this CorpID API should be bound to the same clientID allocated to e-Service by CorpID System.
- If the current CEK for e-Service is not expired, CorpID System will return the same CEK to e-Service with “issueAt” and “expiresIn” unchanged.
- For common parameters in request and response, please refer to Section 3.2.2.
- Response parameter “secretKey”: It is a CEK generated by CorpID System for the e-Service and is encrypted using the RSA algorithm and the latest e-Service's KEK. e-Service should use the private key of its KEK to decrypt “secretKey” to get the CEK.
- Response parameter “pubKey”: It is the latest e-Service’s KEK that is used to encrypt the CEK to form value of “secretKey”. e-Service may make use “pubKey” to locate the corresponding private key.
- Response parameter “issueAt”: It is the issue time of CEK expressed in the number of milliseconds since January 1, 1970 00:00:00 GMT.

- Response parameter “expiresIn”: It is the validity period of CEK expressed in milliseconds.

### **3.3.1.2 Implementing (2) POST: Revoke Symmetric Content Encryption Key**

This API allows e-Service to revoke symmetric Content Encryption Key when necessary.

#### **Pre-conditions**

- Nil

#### **Post-conditions**

- Nil

#### **Error conditions**

- For common errors, please refer to Section 3.2.3.

#### **Request and Response Parameters**

- Please refer to CorpID API Specification.

#### **Notes**

- Each e-Service can only request for one CEK. When CEK was revoked, e-Service is required to request for a new CEK.

## **3.3.2 Encryption & Decryption Scope and Examples**

Please refer to CorpID API Specification.

## **3.3.3 Proper Handling of Encryption and Decryption**

e-Service shall handle the following scenarios properly in their system design:

- Due to unexpected reasons, e-Service may receive Encryption/Decryption Error Code.

e-Service shall perform retry of the “Request Symmetric Content Encryption Key” API specified in section 6.3.1.1 of CorpID API Specification to obtain the latest CEK and try to decrypt content with such CEK when it is received.

- Due to operational needs, the “expiresIn” response parameter of the “Request Symmetric Content Encryption Key” API specified in section 6.3.1.1 of CorpID API Specification may dynamically change, e-Service shall calculate the expiry time of the CEK with “issueAt” and “expiresIn” response parameters.

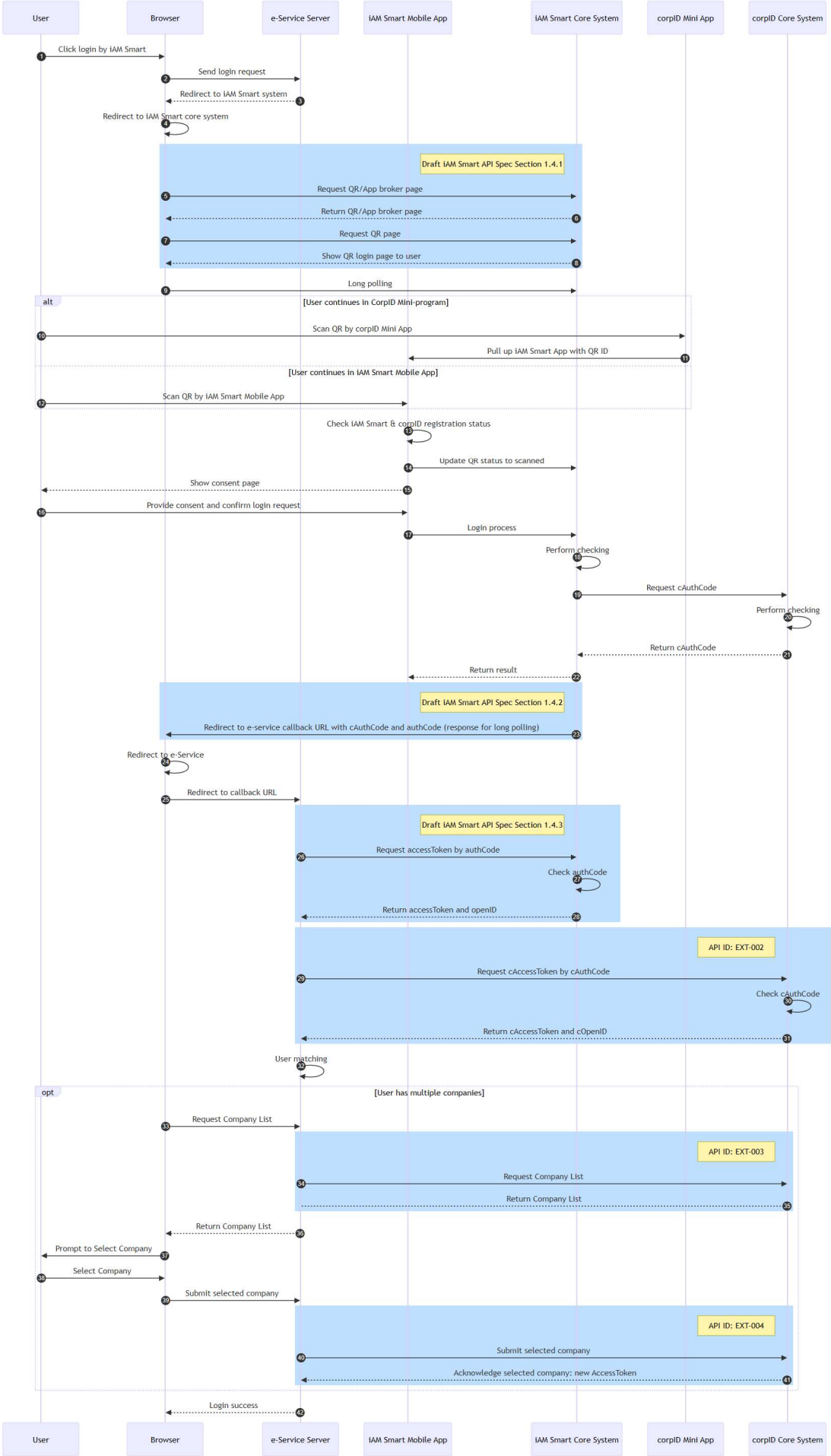
- Under certain circumstances, CorpID API service may be unavailable for a certain period of time. The e-Service shall prepare the fallback procedures (e.g., switch to paper mode, prevent users from accessing CorpID functions, etc.)
- When there are too frequent API requests from an e-Service, the e-Service may receive HTTPS status Code 429, “Too Many Requests”. e-Service shall store the CEK to avoid unnecessary “Request Symmetric Content Encryption Key” API request (see section 6.3.1.1 of CorpID API Specification)
- CorpID API e-Service has to use the old private key to decrypt the CEK until the expiry time. e-Service shall match the public key returned from the “Request Symmetric Content Encryption Key” API response to determine which private key should be used for the decryption
- Even a new KEK is configured for effective in a specified period, the existing CEK will still be returned through the “Request Symmetric Content Encryption Key” API specified in section 6.3.1.1 of CorpID API Specification and encrypted by the old KEK until the expiry of the existing CEK. To use the new KEK, e-Service shall request a new CEK through the “Revoke Symmetric Content Encryption Key” and “Request Symmetric Content Encryption Key” APIs specified in sections 6.3.1.2 and 6.3.1.1 respectively of CorpID API Specification or to request a new CEK after its expiry through the same APIs.

### **3.4 WORKFLOWS FOR AUTHENTICATION**

#### **3.4.1 Scenario 1: Authentication with iAM Smart (e-Service Website in Different Device)**

The sequence diagram below shows how e-Service website leverages the CorpID and “iAM Smart” System to perform user authentication when the “iAM Smart” Mobile App is installed in a separated mobile device.

CorpID Authentication (e-Service Website in Different Device)



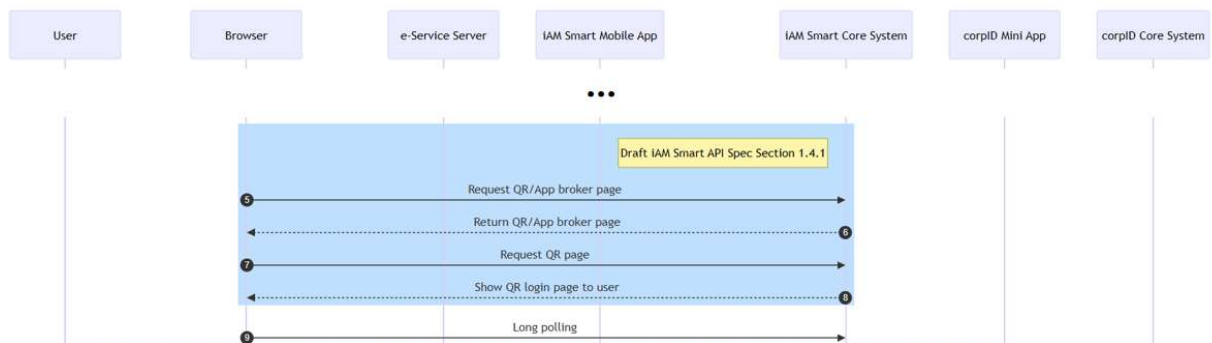
- Step 1. The user clicks Login by iAM Smart button in the browser;
- Step 2. The browser sends the login request to the e-Service Server with the browser's user agent value (for use as value for request parameter "source");
- Step 3-5. The e-Service Server prepares the request parameters such as "clientID", "redirectURI", "source", "scope", etc. and constructs the GET request to invoke the "iAM Smart" API using browser redirection;  
*Draft "iAM Smart" API for CorpID (GET: Request QR page)*
- Step 6-8. "iAM Smart" System returns the broker page (if e-Service has set "brokerPage" to "true") and after the broker page fails to find the "iAM Smart" Mobile App, it will request QR page to be displayed in browser. If e-Service does not request for broker page (i.e., no broker page will be returned), the browser will directly display QR Code page;
- Step 9. QR Code page of "iAM Smart" System polls "iAM Smart" System for the QR Code status;
- Step 10-17. CorpID user logs in "iAM Smart" Mobile App or launch CorpID Mini-program to scan the QR Code and confirms e-Service's login request. CorpID Mini-program check "iAM Smart" & CorpID Registration status on the mobile;
- Step 18-22. "iAM Smart" System verifies the validity of QR code and other necessary information, and responses the login result to "iAM Smart" Mobile App / CorpID Mini-program (e.g., login success and inform "iAM Smart" user to proceed in e-Service, invalid / expired QR code, etc.). AuthCode (of "iAM Smart") and cAuthCode (of CorpID) will be returned;
- Step 23-25. QR Code page of "iAM Smart" System receives the polling result returned by "iAM Smart" System (i.e., response for the polling in Step 9) which includes authCode (of "iAM Smart" Server), cAuthCode (of CorpID Server) or any error code (e.g., CorpID user rejects the login request), assembles and invokes the e-Service callback API given in "redirectURI" using browser redirection;  
*Draft "iAM Smart" API for CorpID (GET: Callback with authCode to e-Service Server)*
- Step 26-28. When e-Service Server gets the authCode, it should invoke the "iAM Smart" API to obtain the accessToken and the Tokenised ID (OpenID) of the "iAM Smart" user. API data encryption is required;  
*Draft "iAM Smart" API for CorpID (POST: Request accessToken & Tokenised ID)*
- Step 29-32. When e-Service Server gets the cAuthCode, it should invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user, user permission, role in the corporation, and other parameters. API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*

- Step 32. The e-Service Server performs user matching;
- Step 33-37. If corpcount return from CorpID “Get Access Token” API is larger than 1, the user has more than 1 corporate associated. The e-Service should call “Get Corporate List” API with cAccessToken and cOpenID. This API gets the list of corporation profiles associated with the logged-in CorpID user. The e-Service uses the result to show selectable corporation options and let CorpID user to choose which corporate profile to use in this session;  
*CorpID API (EXT-003: Get Corporation List)*
- Step 38-41. The user selects a corporate profile in the browser. The browser submits the selected company to the e-Service Server. The e-Service Server submits the selected corporate “ubi” to the corpID Core System and retrieve the updated cAccessToken and cOpenID, new user permission, new role, and other parameters of the corporation selected.  
*CorpID API (EXT-004, Request Token Exchange for Switching Corporation with UBI)*
- Step 42. The e-Service Website shows the login result (e.g., successful when Tokenised ID (cOpenID) matched with a user account of e-Service).

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found in the following sections.

| No | API Name                                                    | API Reference (Section)                |
|----|-------------------------------------------------------------|----------------------------------------|
| 1  | Request QR/App Broker Page                                  | Draft iAM Smart Document Section 1.4.1 |
| 2  | Redirect to e-Service callback URL                          | Draft iAM Smart Document Section 1.4.2 |
| 3  | Get iAM Smart Access Token by authCode                      | Draft iAM Smart Document Section 1.4.3 |
| 4  | Get CorpID cAccessToken by cAuthCode                        | CorpID API Specification Section 4.2   |
| 5  | [Optional] Request Corporation List                         | CorpID API Specification Section 4.3   |
| 6  | [Optional] Request Token Exchange for Switching Corporation | CorpID API Specification Section 4.4   |

### 3.4.1.1 Implementing (1): “iAM Smart” API – GET: Request QR/App Broker Page



#### Pre-conditions

- e-Service should determine all the authorisation scopes of CorpID and “iAM Smart” required for this authentication request.
- e-Service should detect the browser's user agent name.
- e-Service Website can use the optional “brokerPage” parameter, or determine whether the browser and “iAM Smart” Mobile App are on different devices or not (e.g., the browser's agent name, client configure check). For browser type supported by CorpID System, please refer to Appendix B of CorpID API Specification.
- e-Service should determine if it would generate the optional “state” request parameter to prevent CSRF attack. The “state” will be returned to e-Service for checking through the e-Service callback API for receiving the authorisation code from CorpID and “iAM Smart” System in Step 23.

#### Post-conditions

- The browser will show the QR Code page of CorpID and “iAM Smart” System with a “Return” button to allow user to return to e-Service.

#### Error conditions

- This CorpID and “iAM Smart” API does not return JSON response (i.e., no application error will return).

#### Request Parameters

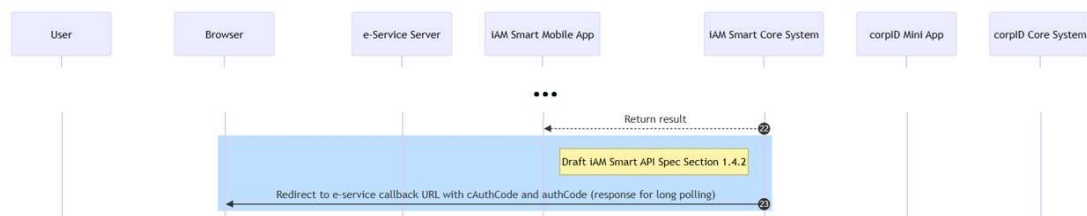
- Please refer to “iAM Smart” API for CorpID Specification.

#### Notes

- For common parameters, please refer to “iAM Smart” Developer Guide.
- Request parameter “responseType”: Value must be 'code'.
- Request parameter “source”: Value should be the browser's user agent value.
- Request parameter “redirectURI”: This is e-Service URL for receiving the authorisation code. The value should be URL encoded. For details, please referred to “iAM Smart” API for CorpID Specification “Callback with cAuthCode to e-Service Server”. The value must be the same as provided during e-Service registration.

- Request parameter “scope”: All “iAM Smart” authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., eidapi\_auth ediapi\_eMe eidapi\_sign). The value should be URL encoded.
- Request parameter “cCscope”: All CorpID authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., corpidapi\_auth corpidapi\_formfilling corpidapi\_sign). The value should be URL encoded.
- Request parameter “ticketID”: Not required in this scenario.
- Request parameter “lang”: Display language of the QR code page.
- Request parameter “state”: The value should be URL encoded.
- Request parameter “brokerPage”: Value should be set to “false” in this scenario.

### 3.4.1.2 Implementing (2): “iAM Smart” API – GET: Redirect to e-Service callback URL



#### Pre-conditions

- CorpID user has scanned the QR Code using “iAM Smart” Mobile App and authorized e-Service's authentication request.
- CorpID user can also reject the authentication request in “iAM Smart” Mobile App.

#### Post-conditions

- cAuthCode will be expired in 1 minute and e-Service can only use the cAuthCode once for requesting the cAccessToken from CorpID System.
- authCode will be expired in 1 minute and e-Service can only use the authCode once for requesting the accessToken from “iAM Smart” System.

#### Error conditions

- This e-Service callback API is a HTTP GET request. If parameter “error\_code” is returned, it means the authentication request is failed.
- For common errors, please refer to “iAM Smart” Developer Guide. Other error for this “iAM Smart” API includes:

| Error Code                   | Error Description                     | Suggested Action                                    |
|------------------------------|---------------------------------------|-----------------------------------------------------|
| error_code - D40000 / M40000 | User cancelled authentication request | Inform user the authentication request is cancelled |

|                              |                                      |                                                                  |
|------------------------------|--------------------------------------|------------------------------------------------------------------|
| error_code - D40001 / M40001 | User rejected authentication request | Inform user the authentication request is rejected               |
| error_code - D40002 / M40002 | Failed to authenticate               | Inform user the authentication request is failed and retry later |

The error\_code D40003 / M40003 (authentication request timeout) does not appear in this scenario. For the QR code timeout or request confirmation timeout in the “iAM Smart” Mobile App, message will be prompted in the corresponding user interface.

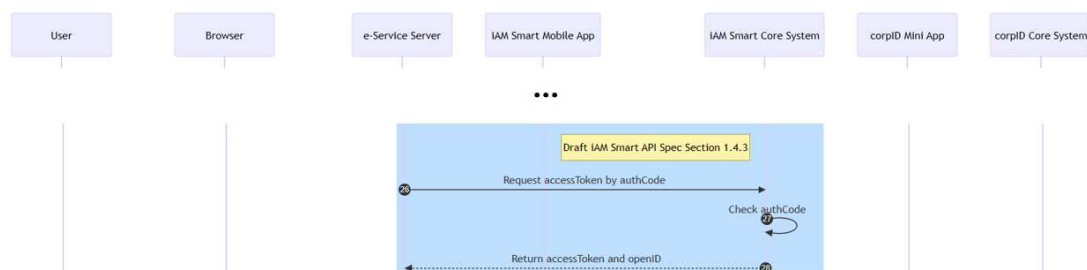
### Callback Parameters

- Please refer to “iAM Smart” API for CorpID Specification.

### Notes

- Callback parameter “businessID”: Not required in this scenario.
- Callback parameter “cCode”: It is the authCode for requesting the cAccessToken from CorpID System. Its validity is 1 minute and can only be used once.
- Callback parameter “code”: It is the authCode for requesting the accessToken from “iAM Smart” System. Its validity is 1 minute and can only be used once.
- If CorpID user rejects the authentication request or other errors occur, “error\_code” instead of “code” will be returned.

### 3.4.1.3 Implementing (3): “iAM Smart” API – POST: Request accessToken & Tokenised ID



### Pre-conditions

- e-Service Service should request the accessToken using a valid authCode. authCode is valid for 1 minute and should not be used more than once.
- API data encryption is required (with “iAM Smart” CEK).

### Post-conditions

- API data decryption is required (with “iAM Smart” CEK).

- e-Service should save the accessToken and Tokenised ID (cOpenID) for subsequent access to other “iAM Smart” APIs, or Integrated Digital Signing / Integrated Form Pre-Filling APIs.
- The validity period of accessToken is given in the response parameter “expiresIn”.
- The accessToken can be used multiple of times before expiry.
- Depending on its business needs, e-Service may also keep and check other response parameters such as “userType”, “scope” to determine subsequent actions to access to other “iAM Smart” APIs, or Integrated Digital Signing / Integrated Form Pre-Filling APIs.

### Error conditions

- For common errors, please refer to “iAM Smart” Developer Guide. Other error for this “iAM Smart” API includes:

| Error Code    | Error Description             | Suggested Action                       |
|---------------|-------------------------------|----------------------------------------|
| code - D40004 | authCode not exist or expired | Check authCode, or redo authentication |

### Request and Response Parameters

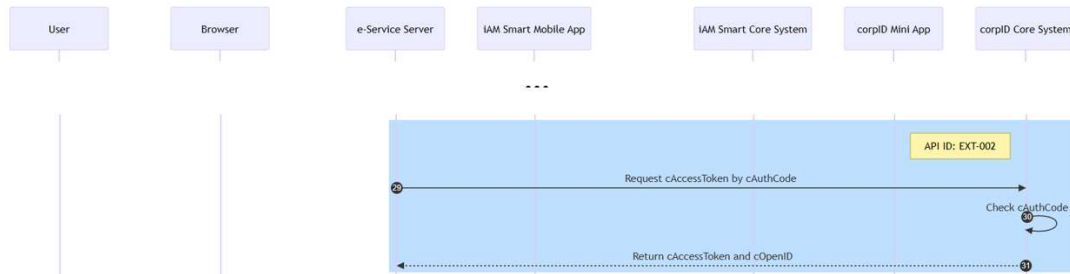
- Please refer to “iAM Smart” API Specification.

### Notes

- For common parameters in request and response, please refer to “iAM Smart” Developer Guide.
- Request parameter “code”: It is the authCode for requesting the accessToken from “iAM Smart” System. Its validity is 1 minute and can only be used once.
- Request parameter “grantType”: Value must be “authorization\_code”.
- Response Parameters “accessToken”, “tokenType”, “issueAt”, “expiresIn”: These are the accessToken information of the “iAM Smart” user. The accessToken is issued by “iAM Smart” System at time specified by “issueAt” and valid for the period specified in “expiresIn”. The “tokenType” must be “Bearer”.
- Response parameter “openID”: It is the Tokenised ID of the “iAM Smart” user for the e-Service. The Tokenised ID of an “iAM Smart” user for different e-Services are different. e-Service can use it to verify if the “iAM Smart” user has linked with any user account in e-Service user repository.
- Response parameter “lastModifiedDate”: It is the last update timestamp of “iAM Smart” user's CFD information in his/her “iAM Smart” Profile.
- Response parameter “userType”: It is the type of the “iAM Smart” account owned by the “iAM Smart” user. It is either “default” (i.e., default “iAM Smart”) or “sign” (i.e., “iAM Smart” with digital signing function) and only “userType” with value “sign” is given an “iAM Smart” e-Cert to carry out digital signing operation.

- Response parameter “scope”: It is the authorisation scope of the accessToken and it should match with the authorisation scope requested by e-Service in Step 5.

#### 3.4.1.4 Implementing (4): EXT-002: Get Access Token



#### Pre-conditions

- e-Service Service should request the cAccessToken using a valid cAuthCode. cAuthCode is valid for 1 minute and should not be used more than once.
- code\_verifier and isDirectLogin are not required.
- API data encryption is required.

#### Post-conditions

- API data decryption is required.
- e-Service should save the cAccessToken and Tokenised ID (cOpenID) for subsequent access to other CorpID APIs.
- The validity period of cAccessToken is given in the response parameter “expiresIn”.
- The cAccessToken can be used multiple of times before expiry.
- Depending on its business needs, e-Service may also keep and check other response parameters such as “cScope”, “userRole”, “userPrivilege”, “eServiceRole”, “ubi”, etc. to determine subsequent actions to access to other CorpID APIs.
- Depending on its business needs, e-Service may also keep and check other response parameters such as “corpNameEN”, “corpNameCH”, “corpLstUpdTs”, “orgCorpUbi”, “orgCorpNameEN”, “orgCorpNameCH”, etc. to display in the e-Service API.
- If “corpCount” is larger than 1, the user has more than 1 Corp Profile, and e-Service should further call the Get Corp List API and Switch Corporation API to display and select a Corp Profile.
- If “isSyncPending” is true, e-Service should be aware the status and information of the Corp Profile may not be latest.  
If “isSyncPending” is false, the Corp Profile “corpProfileFields” (not “eCorpFields”, which is user self-enter) had been validated with trust source.

#### Error conditions

- For common errors, please refer to CorpID Developer Guide. Other error for this CorpID API includes:

| Error Code    | Error Description              | Suggested Action                        |
|---------------|--------------------------------|-----------------------------------------|
| code - M40004 | cAuthCode not exist or expired | Check cAuthCode, or redo authentication |

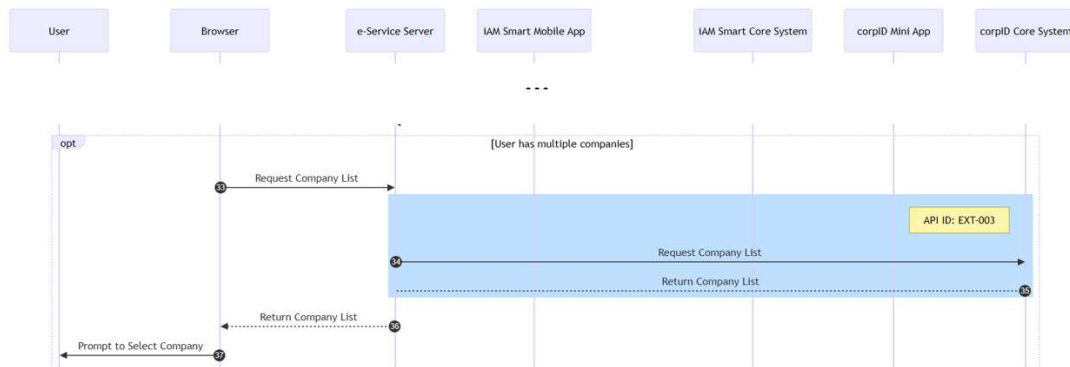
## Request and Response Parameters

- Please refer to CorpID API Specification.

## Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Request parameter “cAuthCode”: It is the cAuthCode for requesting the cAccessToken from CorpID System. Its validity is 1 minute and can only be used once.
- Request parameter “grantType”: Value must be “authorization\_code”.
- Response Parameters “cAccessToken”, “tokenType”, “issueAt”, “expiresIn”: These are the cAccessToken information of the CorpID user. The cAccessToken is issued by CorpID System at time specified by “issueAt” and valid for the period specified in “expiresIn”. The “tokenType” must be “Bearer”.
- Response parameter “cOpenID”: It is the Tokenised ID of the CorpID user for the e-Service. The Tokenised ID of a CorpID user for different e-Services are different. e-Service can use it to verify if the CorpID user has linked with any user account in e-Service user repository.
- Response parameter “corpLstUpdTsts”: It is the last update timestamp of CorpID user's CFD information in his/her “iAM Smart” Profile.
- Response parameter “userRole”: It is the role of the CorpID account owned by the CorpID user. It is either “RP” (i.e., Representative Person, who is Owner or Director of the Corporation), “AR” (i.e., Authorised Representative assigned by RP), “ADM” (i.e., Administrator assigned by RP or AR), and “USR” (i.e. User assigned by RP, AR or ADM). The identity of an “RP” is verified during CorpID onboarding process.
- Response parameter “userPrivilege”: Privileges of CorpID user at e-Service: “user\_access” (i.e. Authentication), “user\_submit” (i.e., Document Wallet), “user\_companychop” (i.e., Digital Signing), “user\_formfilling” (i.e., Form Pre-filling). The value of the parameter is expressed as a list of space-delimited, case-sensitive string.
- Response parameter “eServiceRole”: e-Service self-defined role in integrated portal. A Corporate Administrator may assign the user to the e-Service with this pre-defined role for e-Service to determine the user role in this e-Service.
- Response parameter “cScope”: It is the CorpID authorisation scope of the cAccessToken and it should match with the CorpID authorisation scope requested by e-Service in Step 5.

### 3.4.1.5 Implementing (5): EXT-003: Get Corporation List



#### Pre-conditions

- e-Service Service should already get CorpID “cAccessToken”, “cOpenID”, and valid for the period specified in “expiresIn”.
- API data encryption is required.

#### Post-conditions

- API data decryption is required.
- e-Service can use the response parameters in “corpList” (array list of CorpID Profile), such as “ubi”, “corpNameEN”, “corpNameCH”, “corpLstUpdT”, “orgCorpUbi”, “orgCorpNameEN”, “orgCorpNameCH”, etc. to render switch corporation list display in the e-Service Web / App.

#### Error conditions

- For common errors, please refer to CorpID Developer Guide.

#### Request and Response Parameters

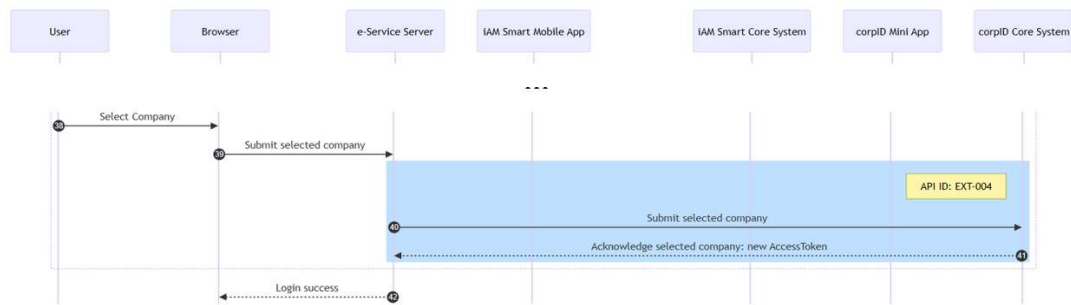
- Please refer to CorpID API Specification.

#### Notes

- For common parameters in request and response, please refer to CorpID API Specification.

### 3.4.1.6 Implementing (6): EXT-004, Request Token Exchange for Switching Corporation with

## UBI



### Pre-conditions

- e-Service Service should already get CorpID “cAccessToken”, “cOpenID”, and valid for the period specified in “expiresIn”.
- “corpCount” is larger than 1.
- “ubi” as requested in Get Corporation List API should exist for this CorpID User.
- API data encryption is required.

### Post-conditions

- API data decryption is required.
- e-Service should save and replace the cAccessToken and Tokenised ID (cOpenID) for subsequent access to other CorpID APIs.
- The validity period of cAccessToken is given in the response parameter “expiresIn”.
- The cAccessToken can be used multiple of times before expiry.
- Depending on its business needs, e-Service may also keep and check other response parameters such as “cScope”, “userRole”, “userPrivilege”, “eServiceRole”, “ubi”, etc. to determine subsequent actions to access to other CorpID APIs.
- Depending on its business needs, e-Service may also keep and check other response parameters such as “corpNameEN”, “corpNameCH”, “corpLstUpdTs”, “orgCorpUbi”, “orgCorpNameEN”, “orgCorpNameCH”, etc. to display in the e-Service API.
- If “isSyncPending” is true, e-Service should be aware the status and information of the Corp Profile may not be latest.  
If “isSyncPending” is false, the Corp Profile “corpProfileFields” (not “eCorpFields”, which is user self-enter) had been validated with trust source.

### Error conditions

- For common errors, please refer to CorpID Developer Guide.

### Request and Response Parameters

- Please refer to CorpID API Specification.

### Notes

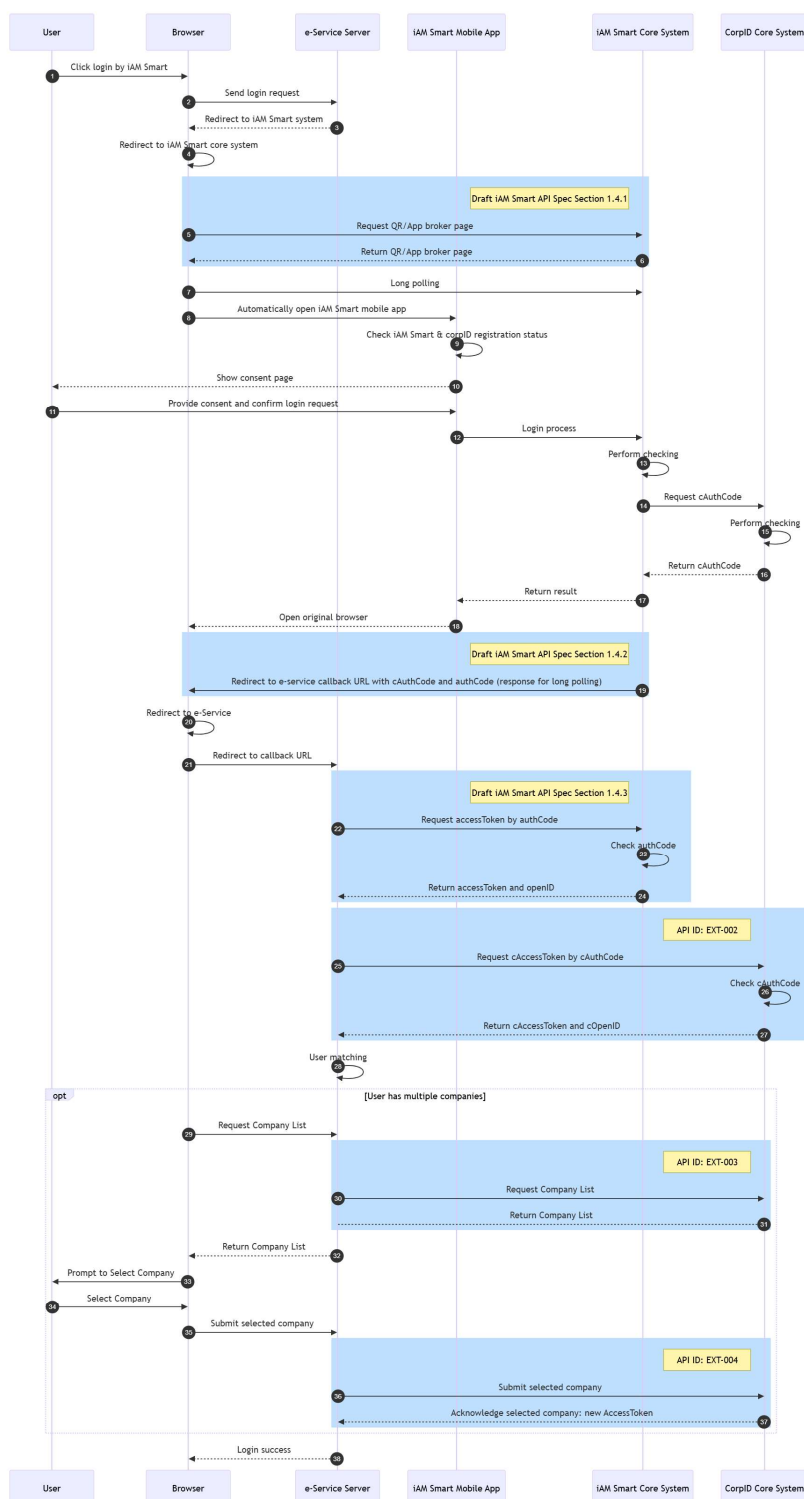
- For common parameters in request and response, please refer to CorpID API Specification.

- Request Parameters “cAccessToken”, “cOpenID”: The AccessToken and Tokenised ID (cOpenID) of the CorpID user for the e-Service from last get Access Token / Request Token Exchange for Switching Corporation with UBI.
- Request parameter “ubi”: “ubi” value of this CorpID user from Get Corporation List array.
- Response parameter “corpLstUpdTs”: It is the last update timestamp of CorpID user's CFD information in his/her “iAM Smart” Profile.
- Response parameter “userRole”: It is the role of the CorpID account owned by the CorpID user. It is either “RP” (i.e., Representative Person, who is Owner or Director of the Corporation), “AR” (i.e., Authorised Representative assigned by RP), “ADM” (i.e., Administrator assigned by RP or AR), and “USR” (i.e. User assigned by RP, AR or ADM). The identity of an “RP” is verified during CorpID onboarding process.
- Response parameter “userPrivilege”: Privileges of CorpID user at e-Service: “user\_access” (i.e. Authentication), “user\_submit” (i.e., Document Wallet), “user\_companychop” (i.e., Digital Signing), “user\_formfilling” (i.e., Form Pre-filling). The value of the parameter is expressed as a list of space-delimited, case-sensitive string.
- Response parameter “eServiceRole”: e-Service self-defined role in integrated portal. A Corporate Administrator may assign the user to the e-Service with this pre-defined role for e-Service to determine the user role in this e-Service.
- Response parameter “cScope”: It is the CorpID authorisation scope of the cAccessToken and it should match with the CorpID authorisation scope requested by e-Service in Step 5.

### **3.4.2 Scenario 2: Authentication with iAM Smart (e-Service Website in Same Device)**

The sequence diagram below shows how e-Service Web leverages the CorpID and “iAM Smart” System to perform user authentication when the “iAM Smart” Mobile App is installed in the same mobile device.

## CorpID Authentication (e-Service Website in Same Device)



- Step 1. The user clicks Login by CorpID in the e-Service Web;
- Step 2-4. The Web sends the login request to the e-Service Server and Server prepare the Broker Page URL;
- Step 5. The Web browser prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as browser’s user agent value), “scope”, “brokerPage” (set to “true”),

etc. constructs the GET request to invoke the “iAM Smart” API;

*Draft “iAM Smart” API for CorpID (GET: Request QR page)*

- Step 6-7. “iAM Smart” System invokes browser (Safari for iOS, Chrome for Android) to submit the GET request and keeps synchronise with e-Service server for the status of the login request;.
- Step 8. Broker page of “iAM Smart” System polls “iAM Smart” System for further action;
- Step 9. Broker page invokes the “iAM Smart” Mobile App in the same device automatically and sends it the relevant parameters. CorpID Mini-program check “iAM Smart” & CorpID Registration status on the mobile;
- Step 10-18. “iAM Smart” System verifies necessary information, and responses the login result to “iAM Smart” Mobile App / CorpID Mini-program (e.g., login success and inform “iAM Smart” user to proceed in e-Service, invalid / expired QR code, etc.). AuthCode (iAM Smart) and cAuthCode (CorpID) will be returned, and trigger to launch original browser;
- Step 19-21. Broker page of “iAM Smart” System receives the polling result returned by “iAM Smart” System (i.e., response for the polling in Step 7) which includes authCode (iAM Smart), cAuthCode (CorpID) or any error code (e.g., CorpID user rejects the login request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection;  
*Draft “iAM Smart” API for CorpID (GET: Callback with authCode to e-Service Server)*
- Step 22-24. When e-Service Server gets the authCode, it should invoke the “iAM Smart” API to obtain the accessToken and the Tokenised ID (i.e., openID) of the “iAM Smart” user. API data encryption is required;  
*Draft “iAM Smart” API for CorpID (POST: Request accessToken & Tokenised ID)*
- Step 25-27. When e-Service Server gets the cAuthCode, it should invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user, user permission, role in the corporation, and other parameters. API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*
- Step 28. The e-Service Server performs user matching;
- Step 29-33. If corpcount return from CorpID “Get Access Token” API is larger than 1, the user has more than 1 corporate associated. The e-Service should call “Get Corporate List” API with cAccessToken and cOpenID. This API gets the list of corporation profiles associated with the logged-in CorpID user. The e-Service uses the result to show selectable corporation options and let CorpID user to choose which corporate profile to use in this session;  
*CorpID API (EXT-003: Get Corporation List)*

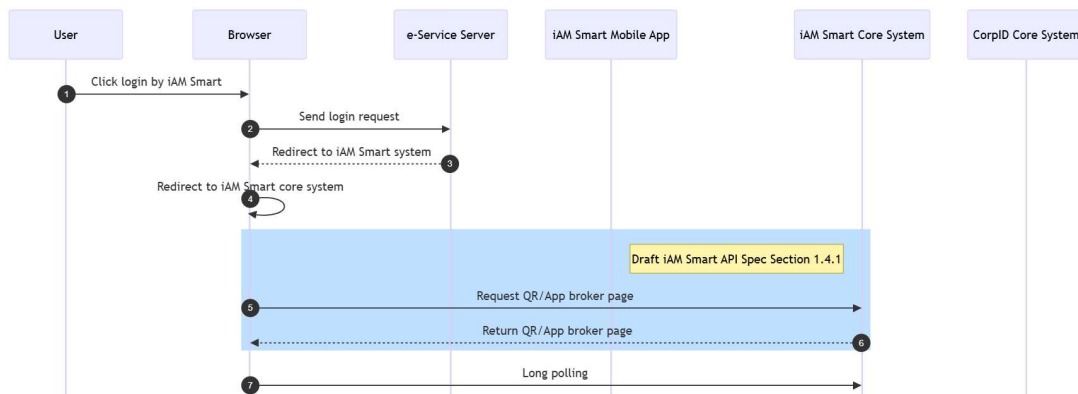
- Step 34-37. The user selects a corporate profile in the e-Service Web. The e-Service Web submits the selected company to the e-Service Server. The e-Service Server submits the selected corporate “ubi” to the corpID Core System and retrieve the updated cAccessToken and cOpenID, new user permission, new role, and other parameters of the corporation selected.  
*CorpID API (EXT-004, Request Token Exchange for Switching Corporation with UBI)*
- Step 38. The e-Service Web shows the login result (e.g., successful when Tokenised ID (cOpenID) matched with a user account of e-Service).

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found in the following sections.

| No | API Name                                                    | API Reference (Section)                |
|----|-------------------------------------------------------------|----------------------------------------|
| 1  | Request QR/App Broker Page                                  | Draft iAM Smart Document Section 1.4.1 |
| 2  | Redirect to e-Service callback URL                          | Draft iAM Smart Document Section 1.4.2 |
| 3  | Get iAM Smart Access Token by authCode                      | Draft iAM Smart Document Section 1.4.3 |
| 4  | Get CorpID cAccessToken by cAuthCode                        | CorpID API Specification Section 4.2   |
| 5  | [Optional] Request Corporation List                         | CorpID API Specification Section 4.3   |
| 6  | [Optional] Request Token Exchange for Switching Corporation | CorpID API Specification Section 4.4   |

### 3.4.2.1 Implementing (1): “iAM Smart” API – GET: Request QR/App Broker Page

#### CorpID Authentication (e-Service Website in Same Device)



#### Pre-conditions

- e-Service should determine all the authorisation scopes of CorpID and “iAM Smart” required for this authentication request.
- e-Service should detect the browser's user agent name.
- e-Service App can use the optional “brokerPage” parameter, or determine whether the browser and “iAM Smart” Mobile App are on different devices or not (e.g., the browser's agent name, client configure check). For browser type supported by CorpID System, please refer to Appendix B of CorpID API Specification.
- e-Service should determine if it would generate the optional “state” request parameter to prevent CSRF attack. The “state” will be returned to e-Service for checking through the e-Service callback API for receiving the authorisation code from CorpID and “iAM Smart” System in Step 21.

#### Post-conditions

- The browser will show the Broker page of CorpID and “iAM Smart” System with a “Return” button to allow user to return to e-Service.

#### Error conditions

- This CorpID and “iAM Smart” API does not return JSON response (i.e., no application error will return).

#### Request Parameters

- Please refer to “iAM Smart” API for CorpID Specification.

#### Notes

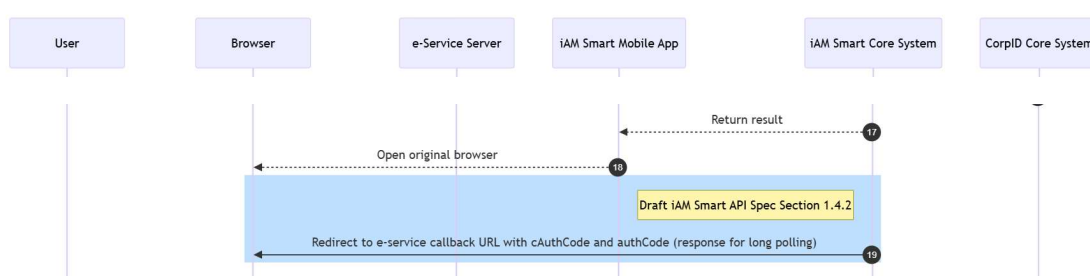
- For common parameters, please refer to “iAM Smart” Developer Guide.
- Request parameter “responseType”: Value must be 'code'.
- Request parameter “source”: Value should be the browser's user agent value.
- Request parameter “redirectURI”: This is e-Service URL for receiving the authorisation code. The value should be URL encoded. For details, please referred to “iAM Smart”

API for CorpID Specification “Callback with cAuthCode to e-Service Server”. The value must be the same as provided during e-Service registration.

- Request parameter “scope”: All “iAM Smart” authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., eidapi\_auth ediapi\_eMe eidapi\_sign). The value should be URL encoded.
- Request parameter “cScope”: All CorpID authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., corpidapi\_auth corpidapi\_formfilling corpidapi\_sign). The value should be URL encoded.
- Request parameter “ticketID”: Not required in this scenario.
- Request parameter “lang”: Display language of the Broker page.
- Request parameter “state”: The value should be URL encoded.
- Request parameter “brokerPage”: Value should be set to “true” in this scenario.

### 3.4.2.2 Implementing (2): “iAM Smart” API – GET: Redirect to e-Service callback URL

#### CorpID Authentication (e-Service Website in Same Device)



#### Pre-conditions

- CorpID user use same device “iAM Smart” Mobile App and authorized e-Service's authentication request.
- CorpID user can also reject the authentication request in “iAM Smart” Mobile App.

#### Post-conditions

- cAuthCode will be expired in 1 minute and e-Service can only use the cAuthCode once for requesting the cAccessToken from CorpID System.
- authCode will be expired in 1 minute and e-Service can only use the authCode once for requesting the accessToken from “iAM Smart” System.

#### Error conditions

- This e-Service callback API is a app-to-app callback request. If parameter “error\_code” is returned, it means the authentication request is failed.
- For common errors, please refer to “iAM Smart” Developer Guide. Other error for this “iAM Smart” API includes:

| Error Code | Error Description | Suggested Action |
|------------|-------------------|------------------|
|------------|-------------------|------------------|

|                              |                                       |                                                                  |
|------------------------------|---------------------------------------|------------------------------------------------------------------|
| error_code - D40000 / M40000 | User cancelled authentication request | Inform user the authentication request is cancelled              |
| error_code - D40001 / M40001 | User rejected authentication request  | Inform user the authentication request is rejected               |
| error_code - D40002 / M40002 | Failed to authenticate                | Inform user the authentication request is failed and retry later |

The error\_code D40003 / M40003 (authentication request timeout) does not appear in this scenario. For the QR code timeout or request confirmation timeout in the “iAM Smart” Mobile App, message will be prompted in the corresponding user interface.

### Callback Parameters

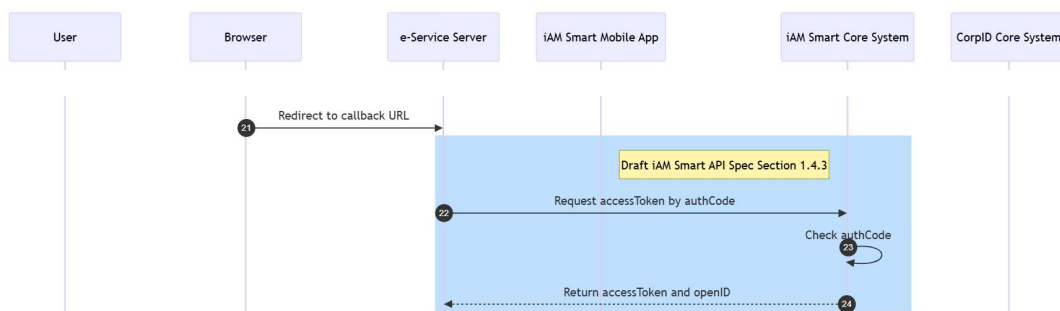
- Please refer to “iAM Smart” API for CorpID Specification.

### Notes

- Callback parameter “businessID”: Not required in this scenario.
- Callback parameter “cCode”: It is the authCode for requesting the cAccessToken from CorpID System. Its validity is 1 minute and can only be used once.
- Callback parameter “code”: It is the authCode for requesting the accessToken from “iAM Smart” System. Its validity is 1 minute and can only be used once.
- If CorpID user rejects the authentication request or other errors occur, “error\_code” instead of “code” will be returned.

### 3.4.2.3 Implementing (3): “iAM Smart” API – POST: Request accessToken & Tokenised ID

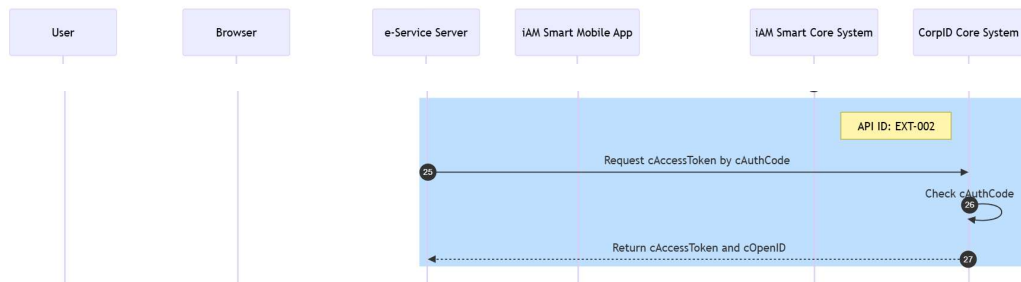
#### CorpID Authentication (e-Service Website in Same Device)



Please refer to process 3.4.1.3.

### 3.4.2.4 Implementing (4): EXT-002: Get Access Token

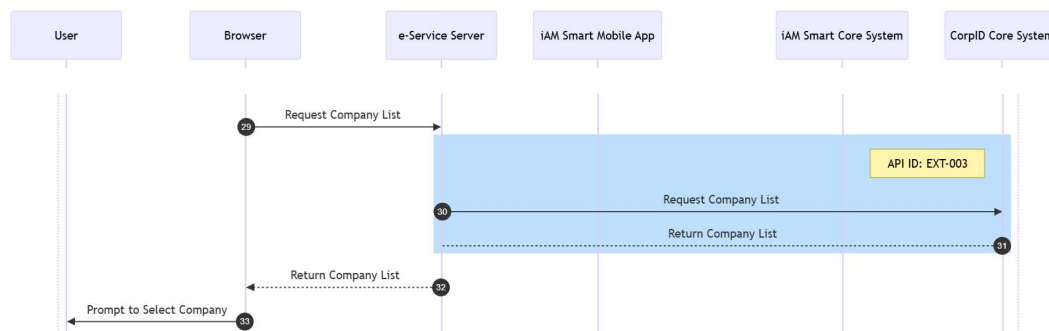
#### CorpID Authentication (e-Service Website in Same Device)



Please refer to Section 3.4.1.4.

### 3.4.2.5 Implementing (5): EXT-003: Get Corporation List

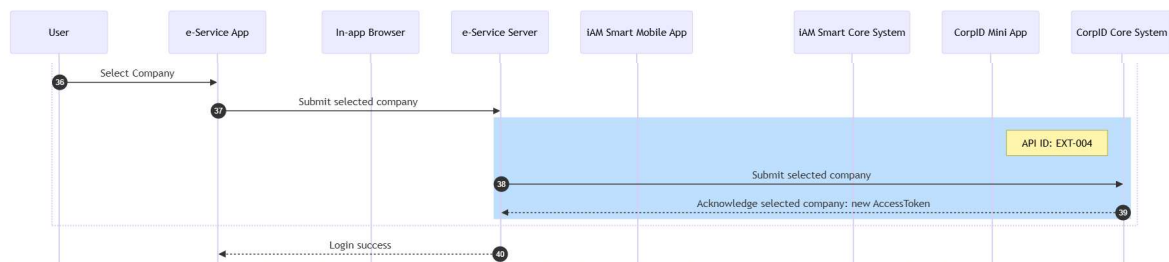
#### CorpID Authentication (e-Service Website in Same Device)



Please refer to Section 3.4.1.5.

### 3.4.2.6 Implementing (6): EXT-004, Request Token Exchange for Switching Corporation with UBI

#### CorpID Authentication (e-Service App in Different Device)

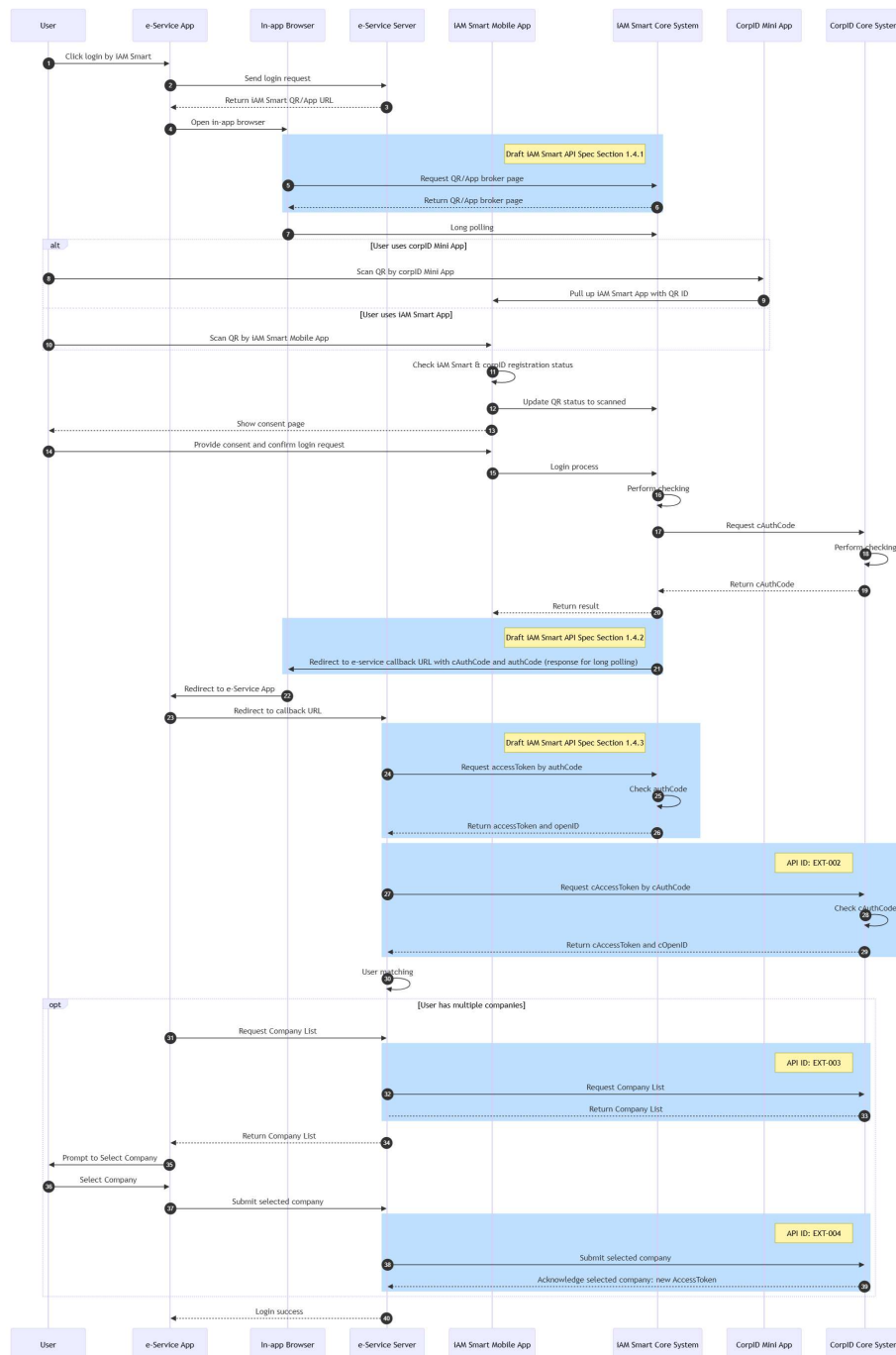


Please refer to Section 3.4.1.6.

### 3.4.3 Scenario 3: Authentication with iAM Smart (e-Service App in Different Device)

The sequence diagram below shows how e-Service App leverages the CorpID and “iAM Smart” System to perform user authentication when the “iAM Smart” Mobile App is installed in a separated mobile device.

CorpID Authentication (e-Service App in Different Device)



Step 1. The user clicks Login by CorpID in the e-Service App;

Step 2-3. The App sends the login request to the e-Service Server and Server prepare the Broker Page URL;

- Step 4-5. The App Open a In-app browser and prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as in-app browser’s user agent value), “scope”, “brokerPage” (set to “false”), etc. constructs the GET request to invoke the “iAM Smart” API;  
*Draft “iAM Smart” API for CorpID (GET: Request QR page)*
- Step 6-8. “iAM Smart” System invokes in-app browser (Safari for iOS, Chrome for Android) to submit the GET request and keeps synchronise with e-Service server for the status of the login request;.
- Step 7. QR Code page of “iAM Smart” System polls “iAM Smart” System for the QR Code status;
- Step 8-11. CorpID user logs in “iAM Smart” Mobile App or launch CorpID Mini-program to scan the QR Code and confirms e-Service's login request. CorpID Mini-program check “iAM Smart” & CorpID Registration status on the mobile;
- Step 12-20. “iAM Smart” System verifies the validity of QR code and other necessary information, and responses the login result to “iAM Smart” Mobile App / CorpID Mini-program (e.g., login success and inform “iAM Smart” user to proceed in e-Service, invalid / expired QR code, etc.). AuthCode (iAM Smart) and cAuthCode (CorpID) will be returned;
- Step 21-23. QR Code page of “iAM Smart” System receives the polling result returned (i.e., response for the polling in Step 7), which includes authCode ( “iAM Smart” Server), cAuthCode (CorpID Server) or any error code (e.g., CorpID user rejects the login request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection;  
*Draft “iAM Smart” API for CorpID (GET: Callback with authCode to e-Service Server)*
- Step 24-26. When e-Service Server gets the authCode, it should invoke the “iAM Smart” API to obtain the accessToken and the Tokenised ID (i.e., openID) of the “iAM Smart” user. API data encryption is required;  
*Draft “iAM Smart” API for CorpID (POST: Request accessToken & Tokenised ID)*
- Step 27-29. When e-Service Server gets the cAuthCode, it should invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user, user permission, role in the corporation, and other parameters. API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*
- Step 30. The e-Service Server performs user matching;
- Step 31-35. If corpcount return from CorpID “Get Access Token” API is larger than 1, the user has more than 1 corporate associated. The e-Service should call “Get Corporate List” API with cAccessToken and cOpenID. This API gets the list of corporation profiles associated with the logged-in CorpID user. The e-Service uses the result to show

selectable corporation options and let CorpID user to choose which corporate profile to use in this session;

*CorpID API (EXT-003: Get Corporation List)*

Step 36-39. The user selects a corporate profile in the e-Service App. The e-Service App submits the selected company to the e-Service Server. The e-Service Server submits the selected corporate “ubi” to the corpID Core System and retrieve the updated cAccessToken and cOpenID, new user permission, new role, and other parameters of the corporation selected.

*CorpID API (EXT-004, Request Token Exchange for Switching Corporation with UBI)*

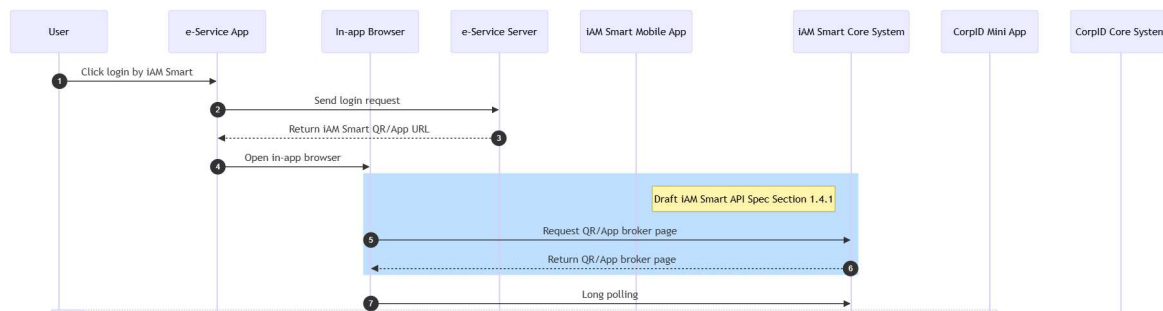
Step 40. The e-Service App shows the login result (e.g., successful when Tokenised ID (cOpenID) matched with a user account of e-Service).

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found in the following sections.

| No. | API Name                                                    | API Reference (Section)                |
|-----|-------------------------------------------------------------|----------------------------------------|
| 1   | Request QR/App Broker Page                                  | Draft iAM Smart Document Section 1.4.1 |
| 2   | Redirect to e-Service callback URL                          | Draft iAM Smart Document Section 1.4.2 |
| 3   | Get iAM Smart Access Token by authCode                      | Draft iAM Smart Document Section 1.4.3 |
| 4   | Get CorpID cAccessToken by cAuthCode                        | CorpID API Specification Section 4.2   |
| 5   | [Optional] Request Corporation List                         | CorpID API Specification Section 4.3   |
| 6   | [Optional] Request Token Exchange for Switching Corporation | CorpID API Specification Section 4.4   |

### 3.4.3.1 Implementing (1): “iAM Smart” API – GET: Request QR/App Broker Page

#### CorpID Authentication (e-Service App in Different Device)



#### Pre-conditions

- e-Service should determine all the authorisation scopes of CorpID and “iAM Smart” required for this authentication request.
- e-Service should detect the in-app browser's user agent name.
- e-Service App can use the optional “brokerPage” parameter, or determine whether the browser and “iAM Smart” Mobile App are on different devices or not (e.g., the in-app browser's agent name, client configure check). For browser type supported by CorpID System, please refer to Appendix B of CorpID API Specification.
- e-Service should determine if it would generate the optional “state” request parameter to prevent CSRF attack. The “state” will be returned to e-Service for checking through the e-Service callback API for receiving the authorisation code from CorpID and “iAM Smart” System in Step 21.

#### Post-conditions

- The browser will show the QR Code page of CorpID and “iAM Smart” System with a “Return” button to allow user to return to e-Service.

#### Error conditions

- This CorpID and “iAM Smart” API does not return JSON response (i.e., no application error will return).

#### Request Parameters

- Please refer to “iAM Smart” API for CorpID Specification.

#### Notes

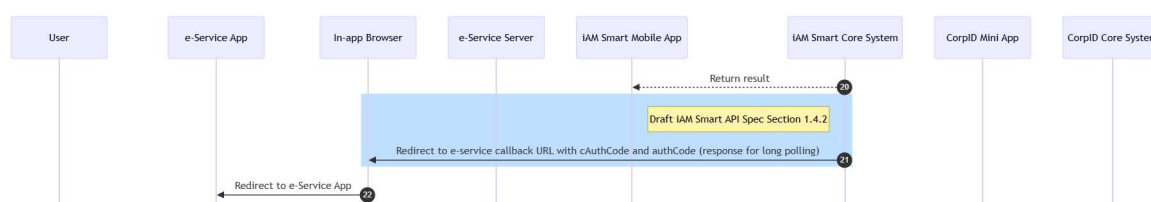
- For common parameters, please refer to “iAM Smart” Developer Guide.
- Request parameter “responseType”: Value must be 'code'.
- Request parameter “source”: Value should be the browser's user agent value.
- Request parameter “redirectURI”: This is e-Service URL for receiving the authorisation code. The value should be URL encoded. For details, please referred to “iAM Smart”

API for CorpID Specification “Callback with cAuthCode to e-Service Server”. The value must be the same as provided during e-Service registration.

- Request parameter “scope”: All “iAM Smart” authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., eidapi\_auth ediapi\_eMe eidapi\_sign). The value should be URL encoded.
- Request parameter “cScope”: All CorpID authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., corpidapi\_auth corpidapi\_formfilling corpidapi\_sign). The value should be URL encoded.
- Request parameter “ticketID”: Not required in this scenario.
- Request parameter “lang”: Display language of the QR code page.
- Request parameter “state”: The value should be URL encoded.
- Request parameter “brokerPage”: Value should be set to “false” in this scenario.

### 3.4.3.2 Implementing (2): “iAM Smart” API – GET: Redirect to e-Service callback URL

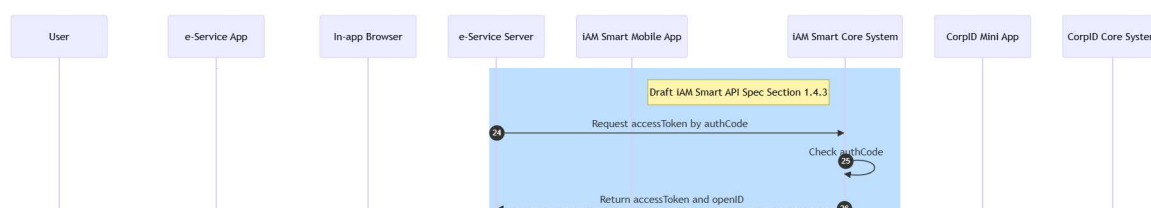
CorpID Authentication (e-Service App in Different Device)



Please refer to Section 3.4.1.2

### 3.4.3.3 Implementing (3): “iAM Smart” API – POST: Request accessToken & Tokenised ID

CorpID Authentication (e-Service App in Different Device)

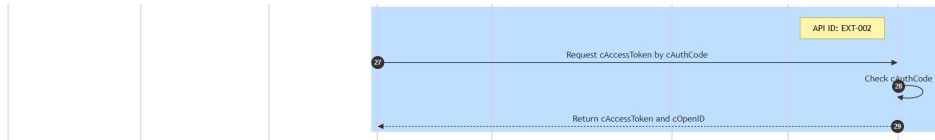


Please refer to process 3.4.1.3.

### 3.4.3.4 Implementing (4): EXT-002: Get Access Token

CorpID Authentication (e-Service App in Different Device)

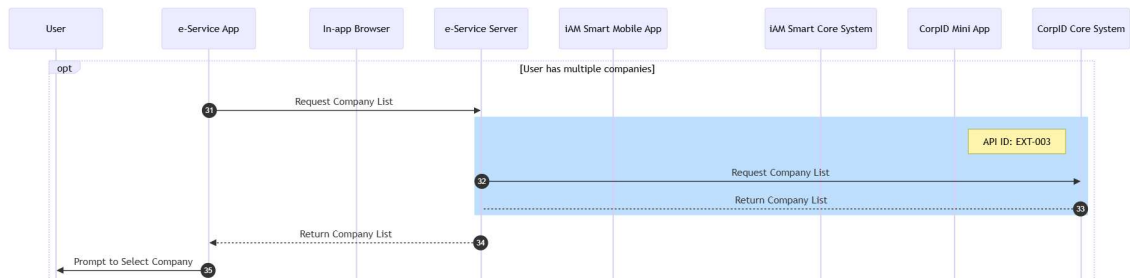




Please refer to Section 3.4.1.4.

### 3.4.3.5 Implementing (5): EXT-003: Get Corporation List

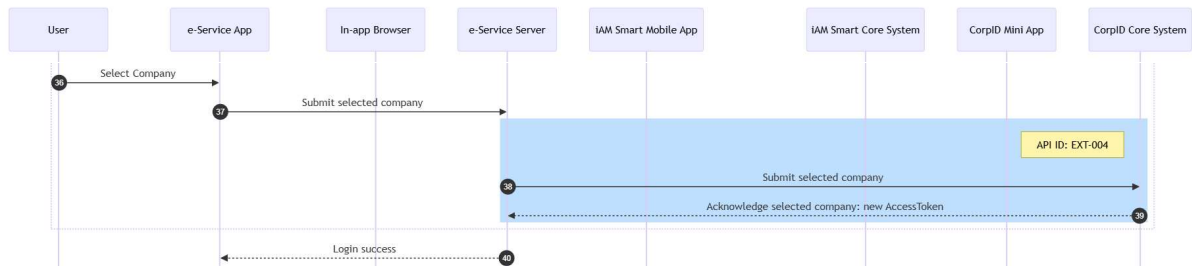
#### CorpID Authentication (e-Service App in Different Device)



Please refer to Section 3.4.1.5.

### 3.4.3.6 Implementing (6): EXT-004, Request Token Exchange for Switching Corporation with UBI

#### CorpID Authentication (e-Service App in Different Device)

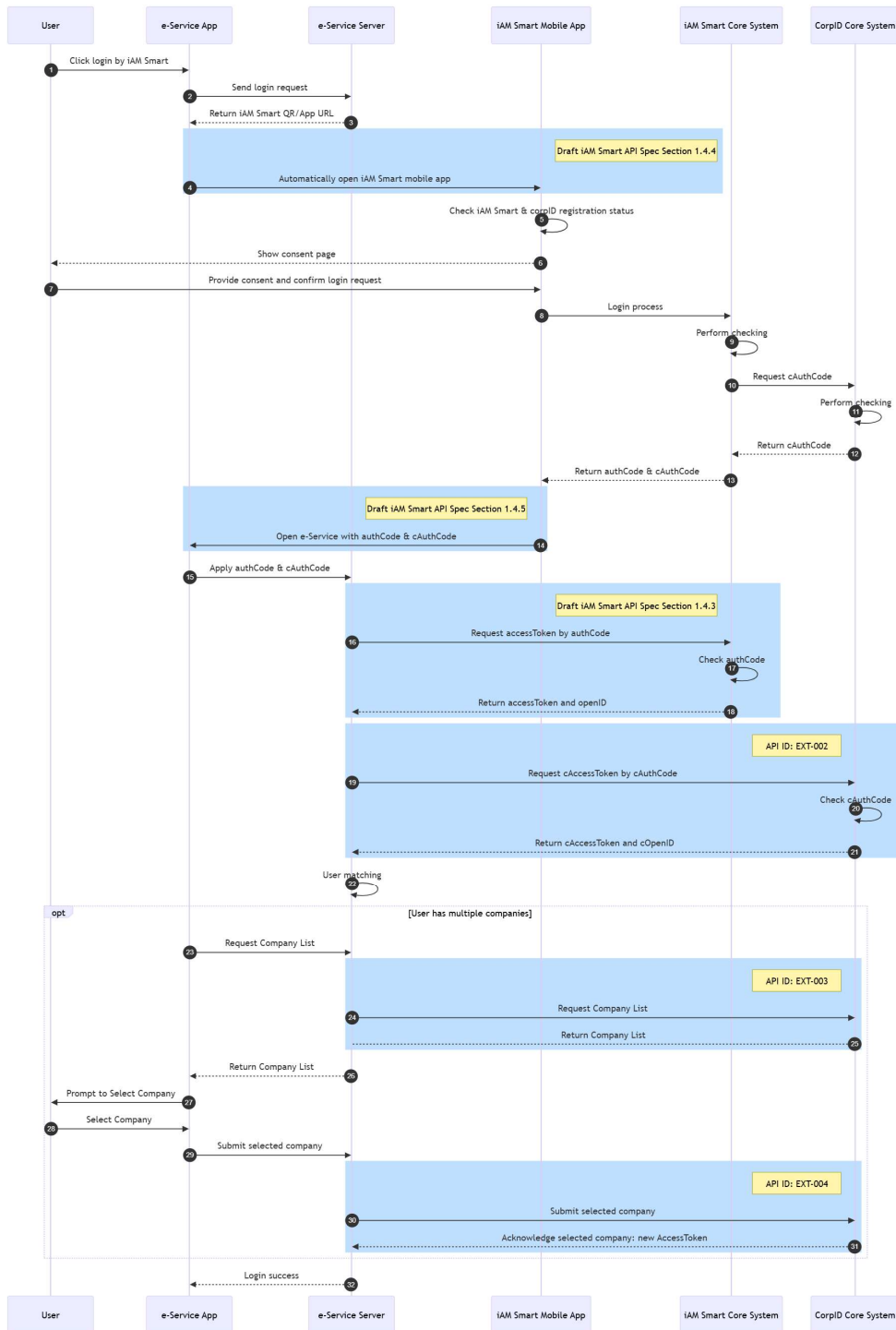


Please refer to Section 3.4.1.6.

### 3.4.4 Scenario 4: Authentication (e-Service App in Same Device)

The sequence diagram below shows how e-Service App leverages the CorpID and “iAM Smart” System to perform user authentication when the “iAM Smart” Mobile App is installed in the same mobile device.

CorpID Authentication (e-Service App in Same Device)



- Step 1. The user clicks Login by CorpID in the e-Service App;
- Step 2. The App sends the login request to the e-Service Server and Server prepare the required parameters;
- Step 3. The App prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as in-app browser’s user agent value), “scope”, “brokerPage” (set to “true”), etc. constructs the URL Schema to invoke the “iAM Smart” API;  
Draft “iAM Smart” API for CorpID (GET: Open iAM Smart Mobile App for Authentication (appv2))
- Step 4. e-Service App launch the “iAM Smart” Mobile App in the same device automatically and sends it the relevant parameters. CorpID Mini-program check “iAM Smart” & CorpID Registration status on the mobile;
- Step 5-13. “iAM Smart” System verifies necessary information, and responses the login result to “iAM Smart” Mobile App / CorpID Mini-program (e.g., login success and inform “iAM Smart” user to proceed in e-Service, invalid / expired QR code, etc.). AuthCode (of “iAM Smart”) and cAuthCode (of CorpID) will be returned, and trigger to launch original e-Service App;
- Step 14-15. e-Service App receives the result returned by “iAM Smart” System which includes authCode (“iAM Smart” Server), cAuthCode (CorpID Server) or any error code (e.g., CorpID user rejects the login request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection;  
*Draft “iAM Smart” API for CorpID (GET: Callback with cAuthCode to e-Service App)*
- Step 16-18. When e-Service Server gets the authCode, it should invoke the “iAM Smart” API to obtain the accessToken and the Tokenised ID (openID) of the “iAM Smart” user. API data encryption is required;  
*Draft “iAM Smart” API for CorpID (POST: Request accessToken & Tokenised ID)*
- Step 19-21. When e-Service Server gets the cAuthCode, it should invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user, user permission, role in the corporation, and other parameters. API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*
- Step 22. The e-Service Server performs user matching;
- Step 23-27. If corpcount return from CorpID “Get Access Token” API is larger than 1, the user has more than 1 corporate associated. The e-Service should call “Get Corporate List” API with cAccessToken and cOpenID. This API gets the list of corporation profiles associated with the logged-in CorpID user. The e-Service uses the result to show selectable corporation options and let CorpID user to choose which corporate profile to use in this session;  
*CorpID API (EXT-003: Get Corporation List)*

Step 28-31. The user selects a corporate profile in the e-Service App. The e-Service App submits the selected company to the e-Service Server. The e-Service Server submits the selected corporate “ubi” to the corpID Core System and retrieve the updated cAccessToken and cOpenID, new user permission, new role, and other parameters of the corporation selected.

*CorpID API (EXT-004, Request Token Exchange for Switching Corporation with UBI)*

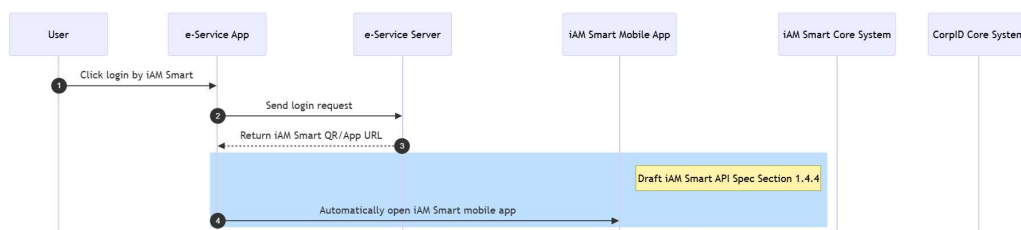
Step 32. The e-Service App shows the login result (e.g., successful when Tokenised ID (cOpenID) matched with a user account of e-Service).

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found in the following sections.

| No. | API Name                                                    | API Reference (Section)                |
|-----|-------------------------------------------------------------|----------------------------------------|
| 1   | Request QR/App Broker Page                                  | Draft iAM Smart Document Section 1.4.1 |
| 2   | Redirect to e-Service callback URL                          | Draft iAM Smart Document Section 1.4.2 |
| 3   | Get iAM Smart Access Token by authCode                      | Draft iAM Smart Document Section 1.4.3 |
| 4   | Get CorpID cAccessToken by cAuthCode                        | CorpID API Specification Section 4.2   |
| 5   | [Optional] Request Corporation List                         | CorpID API Specification Section 4.3   |
| 6   | [Optional] Request Token Exchange for Switching Corporation | CorpID API Specification Section 4.4   |

### 3.4.4.1 Implementing (1): “iAM Smart” API – GET: Open iAM Smart Mobile App for Authentication (appv2)

#### CorpID Authentication (e-Service App in Same Device)



#### Pre-conditions

- e-Service should determine all the authorisation scopes of CorpID and “iAM Smart” required for this authentication request.
- e-Service should determine if it would generate the optional “state” request parameter to prevent CSRF attack. The “state” will be returned to e-Service for checking through the e-Service callback API for receiving the authorisation code from CorpID and “iAM Smart” System in Step 21.

#### **Post-conditions**

- The e-Service App should trigger the “iAM Smart” Mobile App to launch

#### **Error conditions**

- This CorpID and “iAM Smart” API does not return JSON response (i.e., no application error will return).
- “iAM Smart” Mobile App is not installed. e-Service should detect and rollback to different device app approach.

#### **Request Parameters**

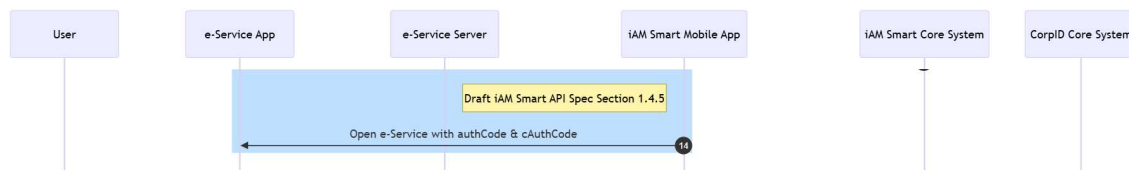
- Please refer to “iAM Smart” API for CorpID Specification.

#### **Notes**

- For common parameters, please refer to “iAM Smart” Developer Guide.
- Request parameter “responseType”: Value must be 'code'.
- Request parameter “source”: Value should be the browser's user agent value.
- Request parameter “redirectURI”: This is e-Service URL for receiving the authorisation code. The value should be URL encoded. For details, please referred to “iAM Smart” API for CorpID Specification “Callback with cAuthCode to e-Service Server”. The value must be the same as provided during e-Service registration.
- Request parameter “scope”: All “iAM Smart” authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., eidapi\_auth ediapi\_eMe eidapi\_sign). The value should be URL encoded.
- Request parameter “cScope”: All CorpID authorisation scopes required for this authentication request should be specified. Multiple values should be separated by space (e.g., corpidapi\_auth corpidapi\_formfilling corpidapi\_sign). The value should be URL encoded.
- Request parameter “ticketID”: Not required in this scenario.
- Request parameter “lang”: Display language of the Broker page.
- Request parameter “state”: The value should be URL encoded.

### 3.4.4.2 Implementing (2): “iAM Smart” API – Callback with cAuthCode to e-Service App

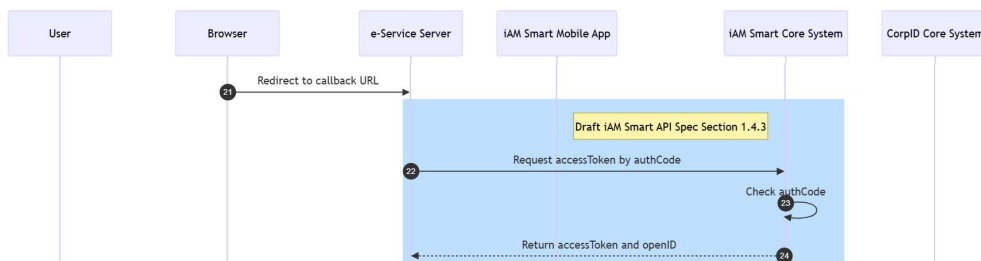
#### CorpID Authentication (e-Service App in Same Device)



Please refer to Section 3.4.1.2

### 3.4.4.3 Implementing (3): “iAM Smart” API – POST: Request accessToken & Tokenised ID

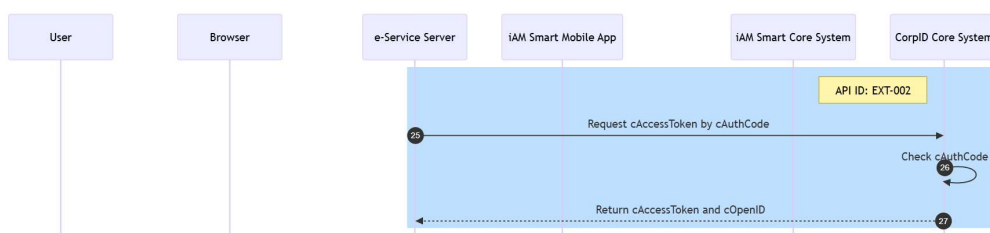
#### CorpID Authentication (e-Service Website in Same Device)



Please refer to process 3.4.1.3.

### 3.4.4.4 Implementing (4): EXT-002: Get Access Token

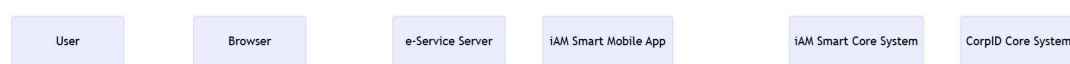
#### CorpID Authentication (e-Service Website in Same Device)

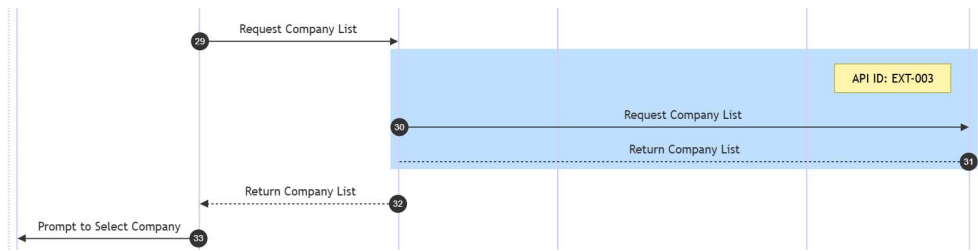


Please refer to Section 3.4.1.4.

### 3.4.4.5 Implementing (5): EXT-003: Get Corporation List

#### CorpID Authentication (e-Service Website in Same Device)

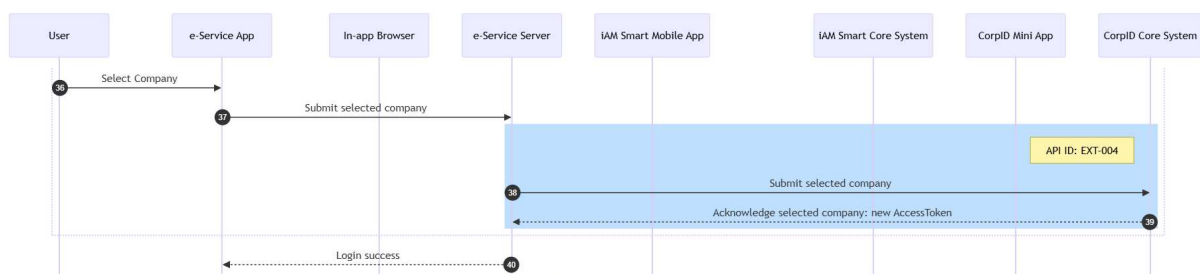




Please refer to Section 3.4.1.5.

### 3.4.4.6 Implementing (6): EXT-004, Request Token Exchange for Switching Corporation with UBI

#### CorpID Authentication (e-Service App in Different Device)



Please refer to Section 3.4.1.6.

#### Post-conditions

- CorpID Mini-program will be launched.
- e-Service App should keep synchronising with e-Service server for the status of the login request.

#### Error conditions

- Nil

#### Request Parameters

- This CorpID API is a URL Scheme. Please refer to CorpID API Specification.

#### Notes

- Please refer to Section 3.4.1.1, except for the following parameters:
  - Request parameter “source”: Value should be “App\_Scheme” (for the e-Service App URL scheme), or “App\_Link” (for the Universal Link/App Link).

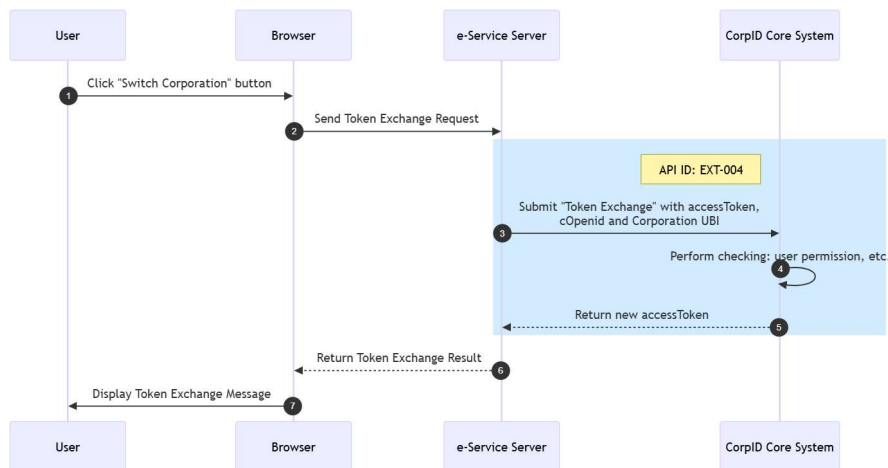
- Request parameter “redirectURI”: This is e-Service App URL scheme or Universal Link/App Link depending on the “source” parameter, which is used for receiving the callback from “iAM Smart” Mobile App to e-Service Web / App.

| Error Code          | Error Description                     | Suggested Action                                                                                    |
|---------------------|---------------------------------------|-----------------------------------------------------------------------------------------------------|
| error_code - D40000 | User cancelled authentication request | Inform user the authentication request is cancelled                                                 |
| error_code - D40001 | User rejected authentication request  | Inform user the authentication request is rejected                                                  |
| error_code - D40002 | Failed to authenticate                | Inform user the authentication request is failed and retry later                                    |
| error_code - D40003 | Authentication request timeout        | Inform user the authentication request is timeout, and provide way for user to do re-authentication |

### 3.4.5 Switching Corporation

The sequence diagram below shows how an e-Service website leverages the CorpID System to perform user token exchange with UBI for switching corporation.

#### CorpID Token Exchange



Pre-conditions: The e-Service App is registered to iAM Smart team with the following details

- CorpID User already login e-Service
- CorpID User getToken return corpcount > 1
- e-Service Server already called “Get Corporation List” during authentication, and cache the list for switch corporation.

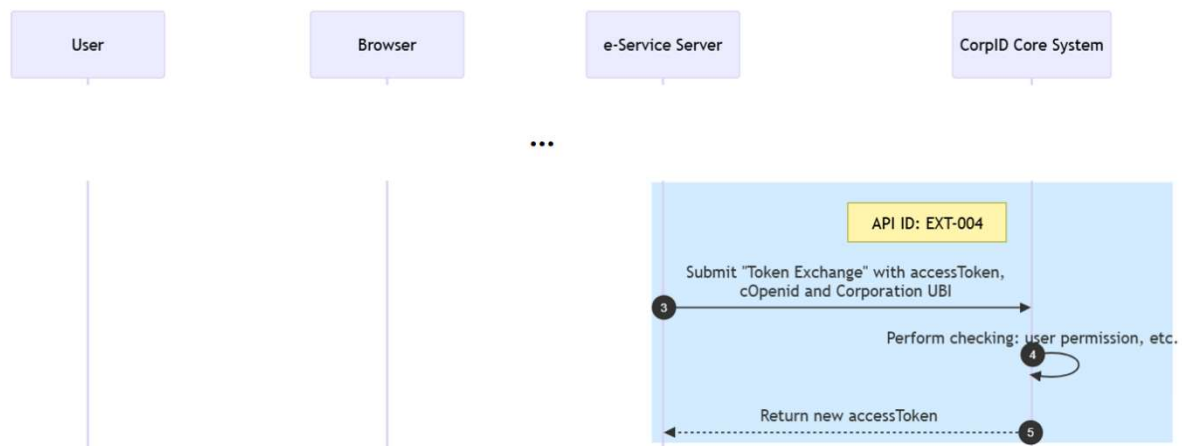
Step 1. The user clicks the “Switch Corporation” button in the browser / e-Service app.

- Step 2-5. The user selects a corporate profile in the browser / app. The browser / app submits the selected company to the e-Service Server. The e-Service Server submits the selected corporate “ubi” to the corpID Core System and retrieve the updated cAccessToken and cOpenID, new user permission, new role, and other parameters of the corporation selected.  
*CorpID API (EXT-004, Request Token Exchange for Switching Corporation with UBI)*
- Step 6-7. The e-Service Server returns the Token Exchange Result to the browser / e-Service App, and the browser /app displays the Token Exchange Message to the user.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found in the following sections.

| No | API Name                                                  | API Reference (Section)              |
|----|-----------------------------------------------------------|--------------------------------------|
| 1  | Request Token Exchange for Switching Corporation with UBI | CorpID API Specification Section 4.4 |

### 3.4.5.1 Implementing (1): EXT-004, Request Token Exchange for Switching Corporation with UBI



#### Pre-conditions

- Please refer to section 3.4.1.6

#### Post-conditions

- Please refer to section 3.4.1.6

#### Error conditions

- Please refer to section 3.4.1.6

#### Request and Response Parameters

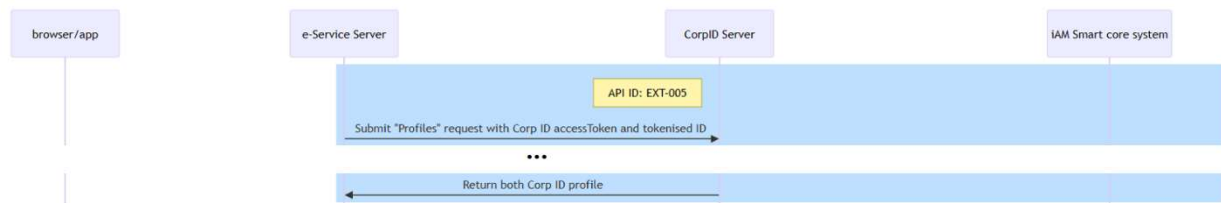
- Please refer to section 3.4.1.6

#### **Notes**

- Please refer to section 3.4.1.6

### 3.5 FORM PRE-FILLING WITH SERVICE LOGIN (PROFILES)

#### 3.5.1 Obtain Profiles Information (EXT-005, FOR B/D AND GOVERNMENT BUREAU ONLY)



##### Pre-conditions

- e-Service Provider must be a Government B/D or Bureau
- e-Service must possess a valid cOpenID, cAccessToken, ubi, corpProfileFields, eCorpFields of the CorpID user with CorpID authorisation scope “Form Pre-Filling authorisation”.
- For “iAM Smart” user, e-Service must possess a valid iAMToken (iAM Smart CEK encrypted token content), eMEFields and profileFields of the CorpID user with “iAM Smart” authorisation scope “Form pre-filling authorisation”.
- e-Service Website/App and iAM Smart Mobile App are in different devices in this scenario.
- API data encryption is required.

##### Post-conditions

- API data decryption is required.
- Response field “corp” shall contain the information of CorpID account
- For “iAM Smart” user,
  - response field “iAMSecretKey” is the Base64 encoded “iAM Smart” CEK encrypted. e-Service Server require to decrypt the field with “iAM Smart” KEK.
  - response field “iAMContent” shall contain the information of “iAM Smart” account, and require decryption with “iAM Smart” CEK.
- e-Service Server should check the requested data fields has authorised for the Form Pre-Filling.

##### Error conditions

- For common errors, please refer to CorpID API Specification. Other error of this API includes:

| Error Code    | Error Description                  | Suggested Action                                                   |
|---------------|------------------------------------|--------------------------------------------------------------------|
| code - M60002 | Failed to request Form Pre-Filling | Inform user the Form Pre-Filling request is failed and retry later |

## Request and Response Parameters

- Please refer to CorpID API Specification.

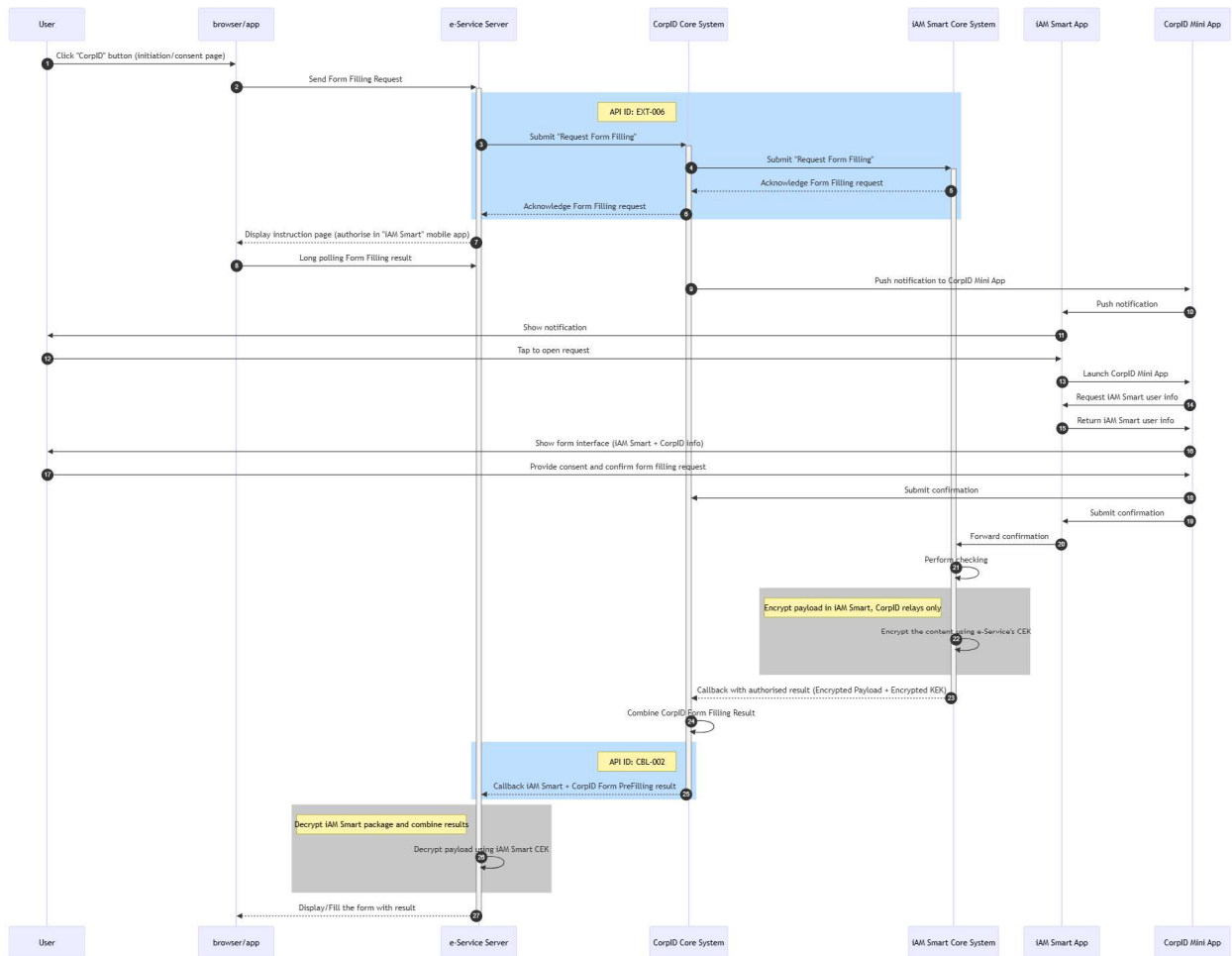
### Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Request parameter “iAMToken”: It contains the “iAM Smart” accessToken and openID encrypted by “iAM Smart” CEK.
- Request parameter “corpProfileFields”: It specifies the corpProfileFields fields required by e-Service. The item name must match with the specification of this CorpID API. Otherwise, error code for the invalid parameter will be returned.
- Request parameter “eCorpFields”: It specifies the eCorpFields fields required by e-Service. The item name must match with the specification of this CorpID API. Otherwise, error code for the invalid parameter will be returned.
- Request parameter “eMEFields”: It specifies the “iAM Smart” e-ME profile fields required by e-Service. The item name must match with the specification of iAM Smart API Specification. Otherwise, error code for the invalid parameter will be returned.
- Request parameter “profileFields”: It specifies the “iAM Smart” profile fields required by e-Service. The item name must match with the specification of iAM Smart API Specification. Otherwise, error code for the invalid parameter will be returned.
- Response parameter “corp”: The corpProfileFields and eCorpFields of CorpID User.
- Response parameter “iAMSecret”: The Base64 encoded CEK encrypted and require “iAM Smart” KEK to decrypt.
- Response parameter “iAMContent”: The eMEFields and profileFields encrypted by “iAM Smart” CEK.

### 3.5.2 Scenario 1: Form pre-filling (e-Service Website/App in Different Device)

The sequence diagram below shows how an authenticated user authorises an e-Service to use his/her CorpID profiles and/or iAM Smart profiles for form pre-filling when e-Service Website / App and the iAM Smart Mobile App are running in different devices.

## CorpID Form Pre-filling (e-Service Website/App in Different Device)



Step 1-2. The user clicks the CorpID button on the initiation or consent page (if CorpID / “iAM Smart” profile fields requested). The e-Service web sends the form pre-filling request to the e-Service Server.

Step 3-6. The e-Service Server submits the Request Form pre-filling request to the CorpID Core System, with “businessID”, “cOpenID”, “cAccessToken”, “iAMToken”, “state”, “source” and other required parameters. The request is processed only when the selected corporation's UBI matches the ubi supplied by the e-Service.

The CorpID Core System check and also submits the Request Form pre-filling request to the “iAM Smart” Core System. The “iAM Smart” Core System acknowledges the form pre-filling request to the CorpID Core System, and the CorpID Core System acknowledges the form pre-filling request to the e-Service Server.

*CorpID API Spec. (EXT-006: Request Form Pre-Filling)*

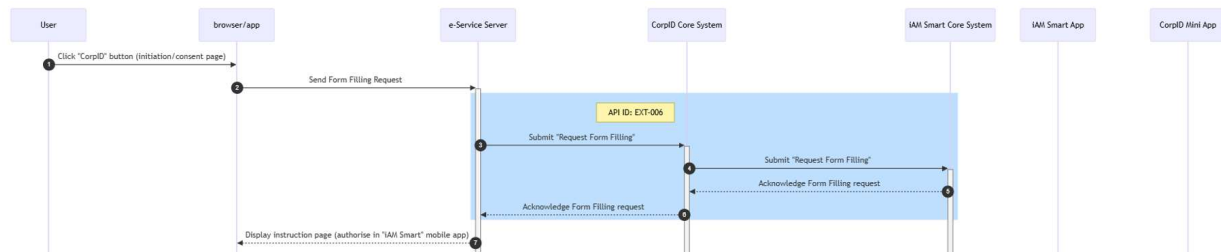
Step 7-8. The e-Service Server displays an instruction page telling the user to authorise in the “iAM Smart” mobile app, and the e-Service web/app starts long polling the e-Service Server for the form pre-filling result.

- Step 9-16. The CorpID Core System fires a notification request to the CorpID Mini-program, by trigger the “iAM Smart” App to show a push notification. The iAM Smart App shows the notification to the user, the user taps the notification to open the request, and trigger the “iAM Smart” App launches the CorpID Mini-program. The CorpID Mini-program requests “iAM Smart” user information from the “iAM Smart” App, and return for display. The CorpID Mini-program shows the form pre-filling consent interface with “iAM Smart” (with data masked) and CorpID information.
- Step 17-24. The user makes consent and confirms the form pre-filling request. The CorpID Mini-program submits the confirmation to the CorpID Core System and the “iAM Smart” Core System. The “iAM Smart” Core System checks and encrypts the “iAM Smart” Profiles using the e-Service “iAM Smart” CEK. The CorpID Core System combines the encrypted “iAM Smart” Profiles and CorpID Profiles as form pre-filling result.
- Step 25-26. The CorpID Core System calls back the iAM Smart and CorpID form pre-filling result to the e-Service Server, with the same “businessID” passed in the request. The e-Service Server decrypts the “iAM Smart” package and combines with the CorpID results.
- For e-Service Website, iAM Smart System instructs iAM Smart Mobile App to invoke the original e-Service browser (i.e. using the browser's user agent value submitted in form pre-filling request in Step 3);
- For e-Service app, iAM Smart System instructs iAM Smart Mobile App to invoke the e-Service App using URL Scheme, or Universal Link/App Link (iAM Smart System queries the information registered at e-Service registration depending on the “source” submitted in Step 3).
- CorpID API Spec. (CRL-002: Form Pre-filling Callback)*
- Step 27. The e-Service Server displays or fills the form with the result in the web / app.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No. | API Name                  | API Reference (Section)              |
|-----|---------------------------|--------------------------------------|
| 1   | Request Form Pre-Filling  | CorpID API Specification Section 4.6 |
| 2   | Form Pre-filling Callback | CorpID API Specification Section 4.7 |

### 3.5.2.1 Implementing (1): EXT-006: Request Form Pre-Filling



#### Pre-conditions

- e-Service must possess a valid cOpenID, cAccessToken, ubi, corpProfileFields, eCorpFields and nonHKResidentFields of the CorpID user with CorpID authorisation scope “Form Pre-Filling authorisation”.
- For “iAM Smart” user, e-Service must possess a valid iAMToken (iAM Smart CEK encrypted token content), eMEFields and profileFields of the CorpID user with “iAM Smart” authorisation scope “Form pre-filling authorisation”.
- The callbackContentType supports “application/json” and “multipart/formdata” values. The default value is “application/json”. If the e-Service endpoint has a request size limit, e-Service may consider using “multipart/form-data” type.
- If state parameter is presented in the request message, the same state value will be returned to e-Service during callback. It is used to prevent the CSRF attack. The value of state is defined by e-Service and it should be a secure random string of any length less than or equal to 36 (UUID string length). Only ASCII letters, numbers, underscore and hyphen are accepted.
- e-Service Website/App and iAM Smart Mobile App are in different devices in this scenario.
- e-Service generates a “businessID” for this Form Pre-Filling request and uses this identifier to match the callback result returned from CorpID System.
- API data encryption is required.

#### Post-conditions

- e-Service server should check the response parameter “authByQR” and “ticketID” and determine the next action. “ticketID” will not be provided by CorpID System in this scenario.
- e-Service Website/App should show instructions to inform CorpID user to process the Form Pre-Filling request in iAM Smart Mobile App when “authByQR” is “true” is returned.
- e-Service Website/App should keep synchronising with e-Service server for the result returned from CorpID Server.
- API data decryption is required.

#### Error conditions

- For common errors, please refer to CorpID API Specification. Other error of this API includes:

| Error Code    | Error Description                  | Suggested Action                                                   |
|---------------|------------------------------------|--------------------------------------------------------------------|
| code - M60002 | Failed to request Form Pre-Filling | Inform user the Form Pre-Filling request is failed and retry later |

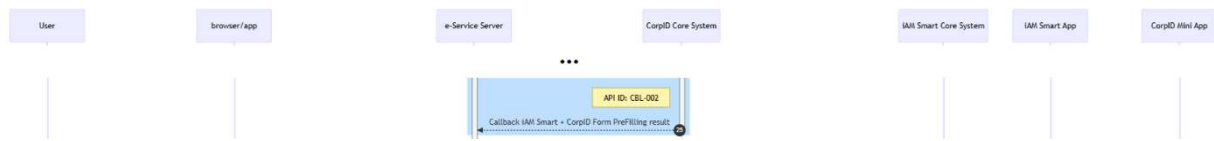
### Request and Response Parameters

- Please refer to CorpID API Specification.

### Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Request parameter “iAMToken”: It contains the “iAM Smart” accessToken and openID encrypted by “iAM Smart” CEK.
- Request parameter “source”: Value should be matched with e-Service client terminal (e.g. “App\_Scheme”/“App\_Link” or browser’s user agent value).
- Request Parameter “redirectURI”: Value should equal to URI of “Callback to Receive Form Pre-Filling Information” e-Service callback API. It must be in the same value as provided during e-Service registration.
- Request parameters “formName”, “formNum”, “formDesc”: They are the information of the e-Service's form and will be shown in CorpID Mini-program for CorpID user's reference. e-Service should determine the language of these items.
- Request parameter “corpProfileFields”: It specifies the corpProfileFields fields required by e-Service. The item name must match with the specification of this CorpID API. Otherwise, error code for the invalid parameter will be returned.
- Request parameter “eCorpFields”: It specifies the eCorpFields fields required by e-Service. The item name must match with the specification of this CorpID API. Otherwise, error code for the invalid parameter will be returned.
- Request parameter “nonHKResidentFields”: It specifies the nonHKResidentFields fields required by e-Service. The item name must match with the specification of this CorpID API. Otherwise, error code for the invalid parameter will be returned.
- Request parameter “eMEFields”: It specifies the e-ME profile fields required by e-Service. The item name must match with the specification of this iAM Smart API. Otherwise, error code for the invalid parameter will be returned.
- Request parameter “profileFields”: It specifies the iAM Smart profile fields required by e-Service. The item name must match with the specification of this iAM Smart API. Otherwise, error code for the invalid parameter will be returned.
- Response parameter “authByQR”: The value returned by iAM Smart System will be “true” in this scenario.
- iAM Smart System will not return the response parameter “ticketID” in this scenario.

### 3.5.2.2 Implementing (2): CRL-002: Form Pre-filling Callback



#### Pre-conditions

- CorpID user can accept the request and authorise data fields to e-Service.
- CorpID user can also reject the form pre-filling request.

#### Post-conditions

- API data decryption is required.
- e-Service Server should match the callback result with corresponding e-Service Website/App using the “businessID”.
- e-Service Server should check the requested data fields has authorised for the Anonymous Form Pre-Filling.

#### Error conditions

- For common errors, please refer to CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description                       | Suggested Action                                                                       |
|---------------|-----------------------------------------|----------------------------------------------------------------------------------------|
| code - M60000 | User cancelled Form Pre-Filling request | Inform user the Form Pre-Filling request is cancelled                                  |
| code - M60001 | User rejected Form Pre-Filling request  | Inform user the Form Pre-Filling request is rejected                                   |
| code - M60002 | Failed to request Form Pre-Filling      | Inform user the Form Pre-Filling request is failed and retry later                     |
| code - M60003 | Form Pre-Filling request timeout        | Inform user the Form Pre-Filling request is timeout, and provide way for user to retry |

#### Callback Parameters

- Please refer to CorpID API Specification.

#### Notes

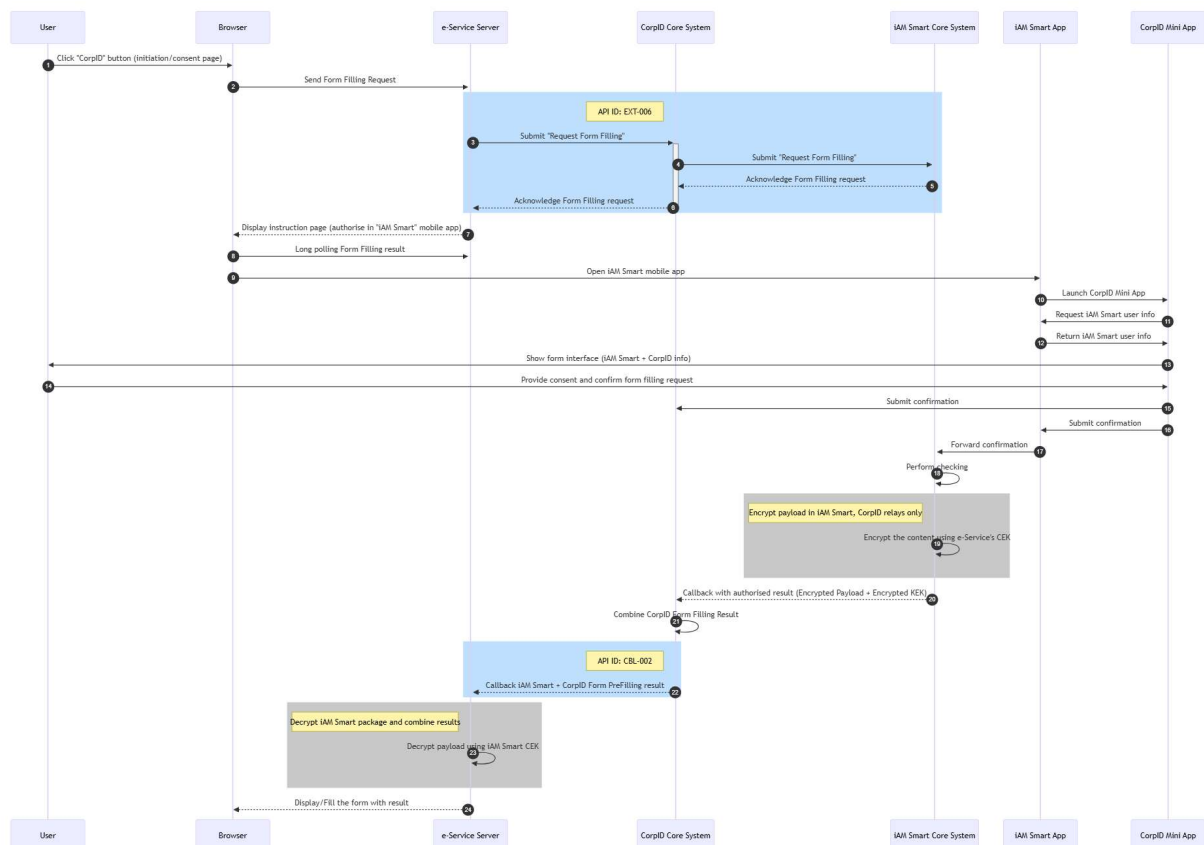
- For common parameters in request and response, please refer to CorpID API Specification.
- “iAMSecret”, the “iAM Smart” CEK, has to decrypted by “iAM Smart” KEK.
- “iAMContent” has to decrypted by “iAM Smart” CEK.

- CorpID or “iAM Smart” profile fields that not authorised by iAM Smart user will not return in result (i.e. JSON name/value of that item will not exist in result).

### 3.5.3 Scenario 2: Form pre-filling (e-Service Website in Same Device)

The sequence diagram below shows how an authenticated user authorises an e-Service to use his/her CorpID profiles and/or iAM Smart profiles for form pre-filling when e-Service Website and the iAM Smart Mobile App are running in the same device.

**CorpID Form Pre-filling (e-Service Website in Same Device)**

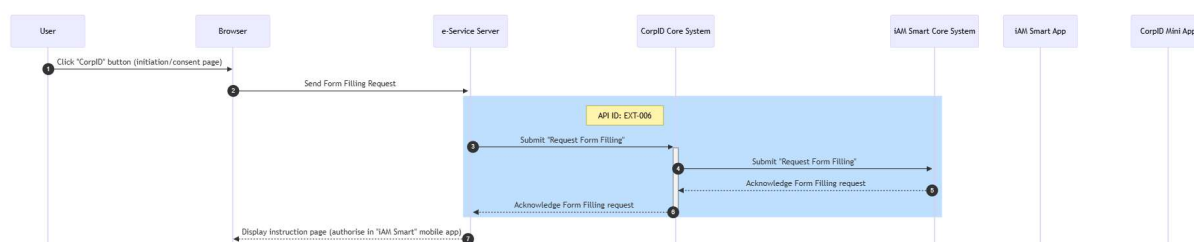


- Step 1-2. The user clicks the CorpID button on the initiation or consent page (if CorpID / “iAM Smart” profile fields requested). The e-Service web sends the form pre-filling request to the e-Service Server.
- Step 3-6. The e-Service Server submits the Request Form pre-filling request to the CorpID Core System, with “businessID”, “cOpenID”, “cAccessToken”, “iAMToken”, “state”, “source” and other required parameters. The request is processed only when the selected corporation's UBI matches the ubi supplied by the e-Service. The CorpID Core System check and also submits the Request Form pre-filling request to the “iAM Smart” Core System. The “iAM Smart” Core System acknowledges the form pre-filling request to the CorpID Core System, and the CorpID Core System

acknowledges the form pre-filling request to the e-Service Server.  
*CorpID API Spec. (EXT-006: Request Form Pre-Filling)*

- Step 7-8. The e-Service Server displays an instruction page telling the user to authorise in the CorpID Mini-program, and the e-Service web/app starts long polling the e-Service Server for the form pre-filling result;
- Step 9. e-Service Website invokes CorpID Mini-program using URL Scheme with “ticketID” (set <Context> as “corpid” in URL Scheme and pass required parameters). The e-Service Website should keep synchronising with the e-Service Server for the request result (e.g. polling).  
*iAM Smart API for CorpID (URL Scheme: Open iAM Smart Mobile App for Form Filling)*
- Step 8-14. CorpID user logs in CorpID Mini-program, reviews the Form Filling authorisation request (e.g. continue or reject the request) and authorises those selected e-ME profile fields and/or CorpID profile fields to e-Services;
- Step 15-22. CorpID System invokes e-Service callback API to return the result with the “businessID” of the Form Filling request. API decryption is required;  
*e-Service callback API (POST: Callback to Receive Form Filling Information)*
- CorpID System instructs CorpID Mini-program to invoke the original e-Service browser (i.e. using the browser's user agent value submitted in form filling request in Step 3);
- Step 23-24. e-Service server processes and matches the result using the “businessID” and shows the result in the corresponding e-Service Website.

### 3.5.3.1 Implementing (1) POST: Request Form Filling



#### Pre-conditions

- Please refer to Section 3.5.2.1 except:
  - e-Service Website and CorpID Mini-program are in the same device in this scenario.

#### Post-conditions

- Please refer to Section 3.5.2.1 except:

- Response “authByQR” is “false”, “ticketID” will be provided by CorpID System in this scenario.

### Error conditions

- Please refer to Section 3.5.2.1.

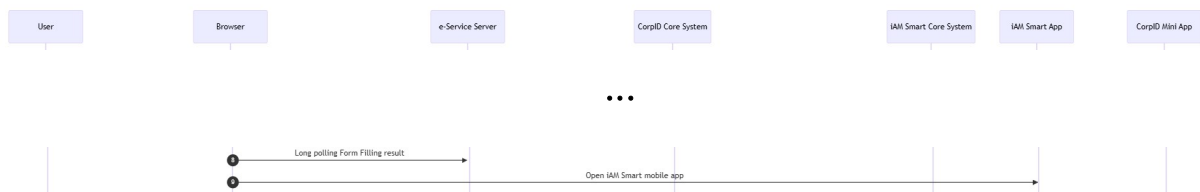
### Request and Response Parameters

- Please refer to CorpID API Specification.

### Notes

- Please refer to Section 3.5.2.1, except for the following parameters:
  - Request parameter “source”: Value should be the browser's user agent value.
  - Response parameter “authByQR”: The value is “false” in this scenario.
  - Response parameter “ticketID”: It will be provided by CorpID System in this scenario. It is a 36-byte (or less) UUID number (ASCII character set).

## 3.5.3.2 Implementing (2) URL Scheme: Open iAM Smart Mobile App for Getting Context



### Pre-conditions

- e-Service Website and CorpID Mini-program are in the same user device.
- e-Service has the “ticketID” provided by CorpID System for the form filling request;
- Other parameters of this API are not used in this scenario.

### Post-conditions

- CorpID Mini-program will be launched.
- e-Service Website should keep synchronising with e-Service server for the callback response of the form filling request from CorpID System.

### Error conditions

- Nil

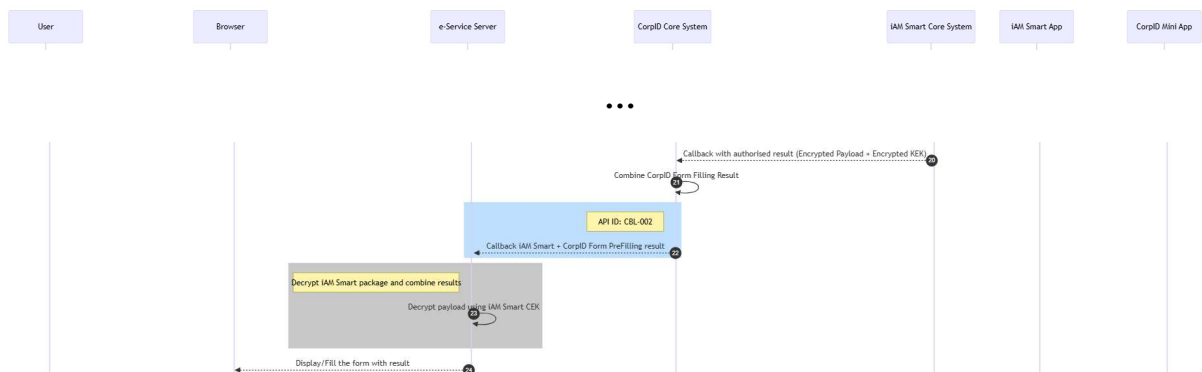
### Request Parameters

- Please refer to CorpID API Specification.

### Notes

- Set <Context> as “corpid” in URL Scheme, and form filling parameters.

### 3.5.3.3 Implementing (3) POST: Callback to Receive Form Filling Information

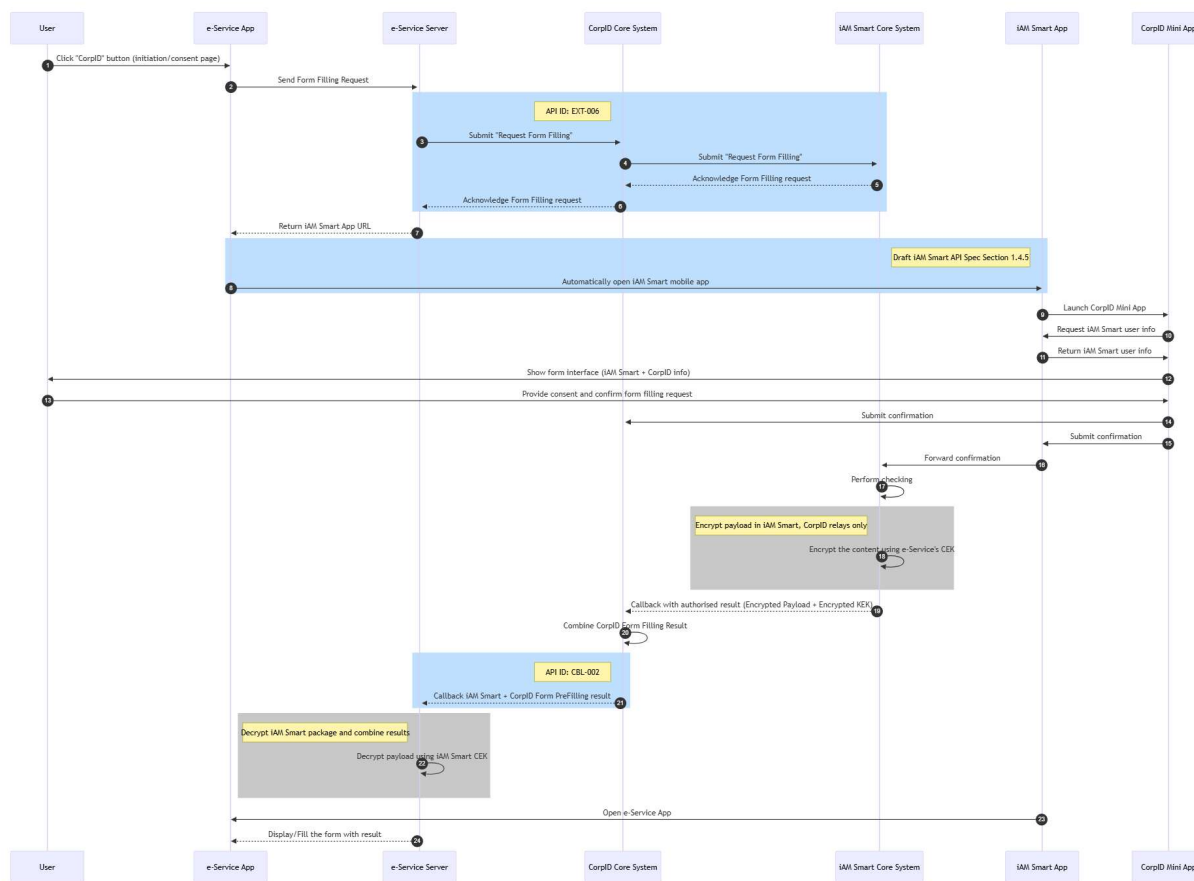


Please refer to Section 3.5.2.2.

### 3.5.4 Scenario 3: Form pre-filling (e-Service App in Same Device)

The sequence diagram below shows how an authenticated user authorises an e-Service to use his/her CorpID profiles and/or iAM Smart profiles for form pre-filling when e-Service App and the iAM Smart Mobile App are running in the same device.

## CorpID Form Pre-filling (e-Service App in Same Device)



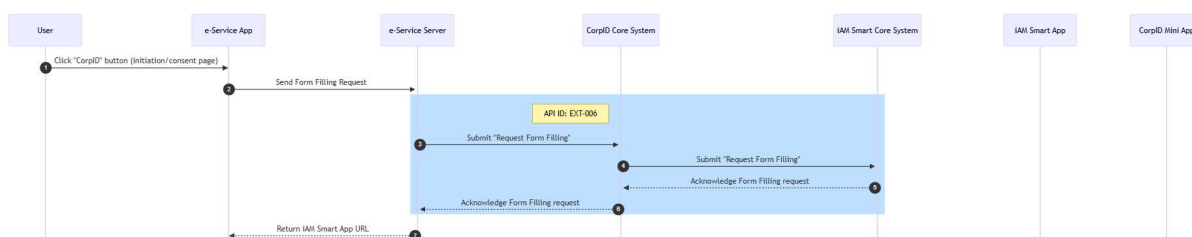
- Step 1. The user clicks the CorpID button on the initiation or consent page (if CorpID / “iAM Smart” profile fields requested).
- Step 2. e-Service App initiates form filling request to e-Service Server with “App\_Scheme” or “App\_Link” (for use as value for request parameter “source”);
- Step 3-6. The e-Service Server submits the Request Form pre-filling request to the CorpID Core System, with “businessID”, “cOpenID”, “cAccessToken”, “iAMToken”, “state”, “source” and other required parameters. The request is processed only when the selected corporation's UBI matches the ubi supplied by the e-Service. CorpID verifies the validity of cAccessToken, Tokenised ID, other parameters and confirms receipt of the Form Filling request to e-Service Server by returning POST response with parameter “authByQR” (set to “false”) and “ticketID” in this scenario; *CorpID API Spec. (EXT-006: Request Form Pre-Filling)*
- Step 7. After receiving the response, e-Service Server prepares necessary parameters such as “authByQR”, “ticketID”, etc. to e-Service App;
- Step 8. e-Service App determines “authByQR” to be false, or uses program code to check if iAM Smart Mobile App exists in the same device, then invokes iAM Smart Mobile App using URL Scheme with “ticketID” (set <Context> as “corpId” in URL Scheme,

and parameters for form pre-filling). The e-Service App should keep synchronising with the e-Service Server for the request result (e.g. polling);

*iAM Smart API for CorpID (URL Scheme: Open iAM Smart Mobile App for Form Filling)*

- Step 9-13. CorpID user logs in CorpID Mini-program, reviews the Form Filling authorisation request (e.g. continue or reject the request) and authorises those selected e-ME profile fields and/or CorpID profile fields to e-Services;
- Step 14-21. CorpID System invokes e-Service callback API to return the result with the “businessID” of the Form Filling request. API decryption is required;  
*CorpID API (POST: Form Pre-Filling Callback)*
- Step 22-24. CorpID System instructs CorpID Mini-program to invoke the e-Service App using URL Scheme, or Universal Link/App Link (iAM Smart System queries the information registered at e-Service registration depending on the “source” submitted in Step 3).

### 3.5.4.1 Implementing (1) POST: Request Form Filling



#### Pre-conditions

- Please refer to Section 3.5.2.1 except:
  - e-Service App and CorpID Mini-program are in the same device in this scenario.

#### Post-conditions

- Please refer to Section 3.5.2.1.

#### Error conditions

- Please refer to Section 3.5.2.1.

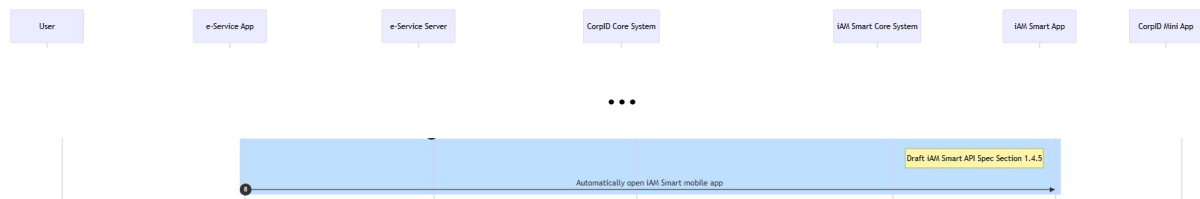
#### Request and Response Parameters

- Please refer to CorpID API Specification.

#### Notes

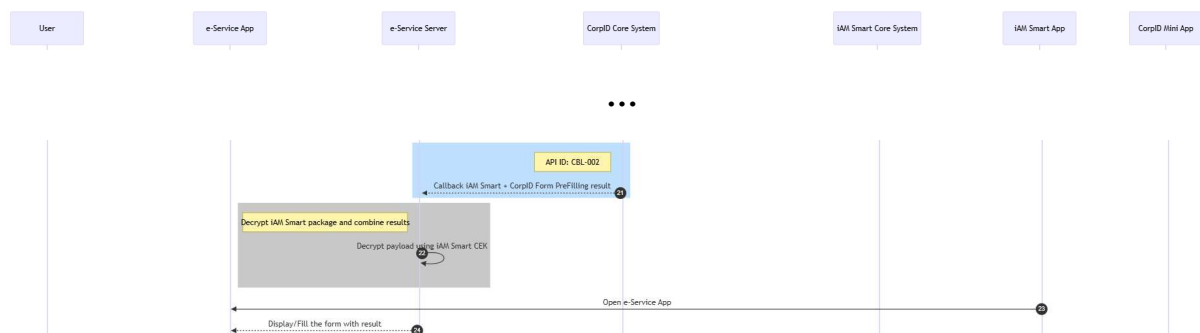
- Please refer to Section 3.5.3.1, except for the following parameter:
  - Request parameter “source”: Value should be “App\_Scheme” (for the e-Service App URL scheme), or “App\_Link” (for the Universal Link/App Link).

### 3.5.4.2 Implementing (2) URL Scheme: Open iAM Smart Mobile App for Getting Context



Please refer to Section 3.5.3.2.

### 3.5.4.3 Implementing (3) POST: Callback to Receive Form Filling Information



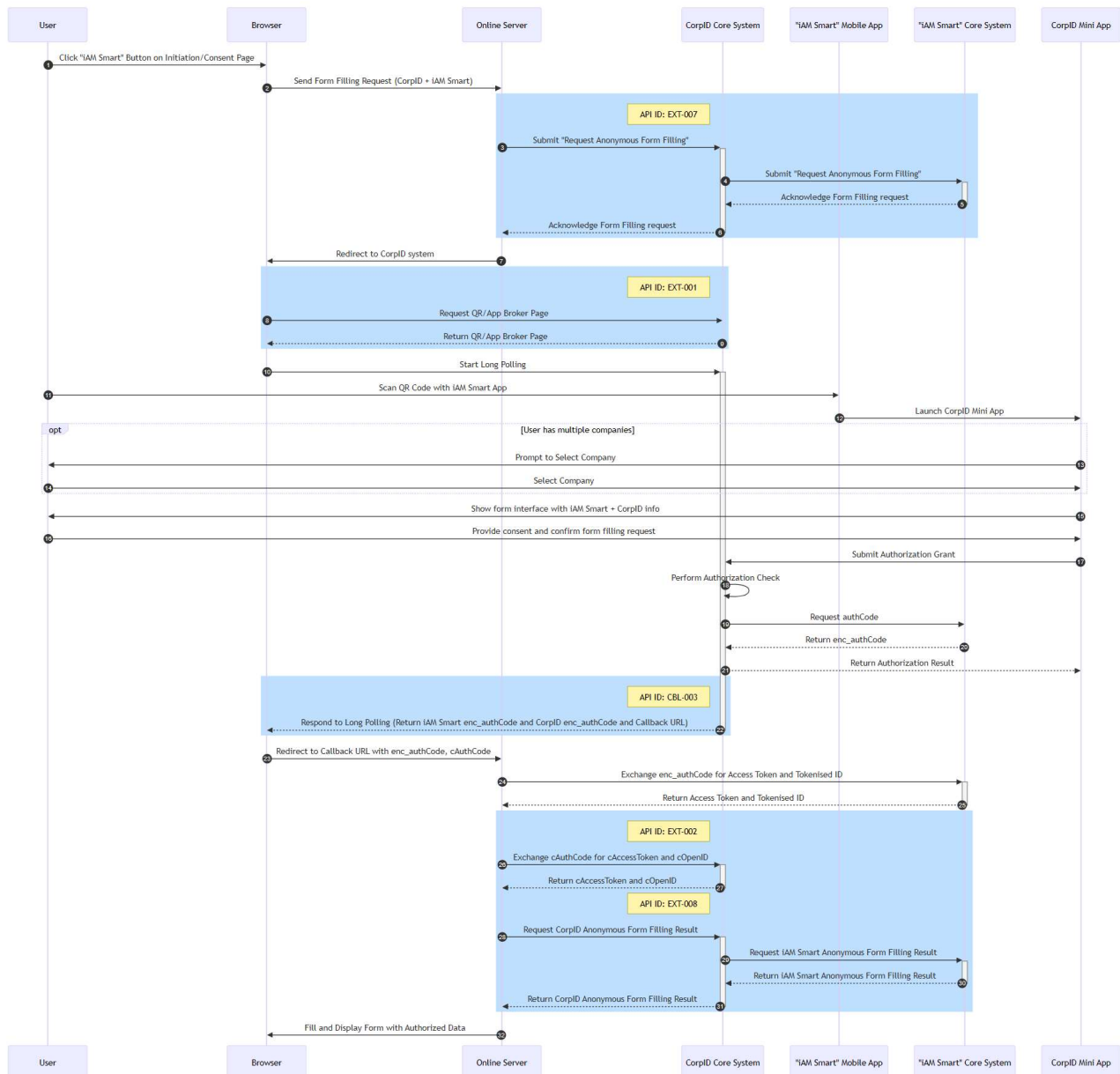
Please refer to Section 3.5.2.2.

## 3.6 FORM PRE-FILLING WITHOUT SERVICE LOGIN (ANONYMOUS FORM PRE-FILLING)

### 3.6.1 Scenario 1: Anonymous Form Pre-filling (e-Service Website in Different Device)

The sequence diagram below shows how an anonymous user authorises an e-Service to use his/her CorpID Fields and “iAM Smart” Profile Fields for form pre-filling when e-Service website and the “iAM Smart” Mobile App are running in different devices.

## CorpID Anonymous Form Filling (e-Service Website in Different Device)



- Step 1. User reads the consent page (if CorpID / “iAM Smart” profile fields requested) and clicks the button in e-Service Website;
- Step 2. e-Service Website initiates anonymous form pre-filling request to e-Service Server with browser's user agent name (use as value for request parameter “source”);
- Step 3. The e-Service Server initiates an Anonymous Form Pre-Filling Request to the CorpID Core System with the “businessID” of this request, required profile fields from CorpID and “iAM Smart”, etc.  
*CorpID API Spec. (EXT-007: Request Anonymous Form Pre-Filling)*

- Step 4-6. The CorpID Server and “iAM Smart” Server verifies and confirms receipt of the anonymous Form Pre-Filling request by returning POST response with parameter “ticketID”.
- Step 7-8. e-Service Server prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as browser’s user agent value), “scope”, “ticketID”, etc. and constructs the GET request to invoke the CorpID API using browser redirection;  
*CorpID API Spec. (EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request)*
- Step 9. CorpID System returns the broker page (if e-Service has set “brokerPage” to “true”) and after the broker page fails to find the “iAM Smart” Mobile App, it will request QR page to be displayed in browser. If e-Service does not request for broker page (i.e., no broker page will be returned), the browser will directly display QR Code page;
- Step 10. QR Code page of CorpID and “iAM Smart” System polls CorpID and “iAM Smart” System for the QR Code status;
- Step 11-16. CorpID user uses “iAM Smart” Mobile App to scan the QR Code, launches the CorpID Mini-program, reviews the form pre-filling authorisation request (e.g., continue or reject the request) and authorises those selected CorpID Profile fields and “eMEFields” to e-Services (e.g., authorise only his HKIC number and English name in account information and residential address in “e-ME profile” (unchecked other “eMEFields”));  
If the user has multiple corporations, the CorpID Mini-program will also allow the user to select a corporation.
- Step 17-21. CorpID and “iAM Smart” System verifies the validity of QR code and other necessary information, and return the authorisation result to CorpID Mini-program;
- Step 22-23. QR Code page of CorpID and “iAM Smart” System receives the polling result returned by CorpID System (i.e., response for the polling in Step 10) which includes cAuthCode (of CorpID), authCode (of “iAM Smart”) and businessID or any error code (e.g., CorpID user rejects the request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection;  
*CorpID API Spec. (CLB-003: Callback with authCode for Anonymous Form Pre-Filling)*
- Step 24-25. When e-Service Server gets the authCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the “iAM Smart” API to obtain the accessToken and the Tokenised ID (openID) of the “iAM Smart” user. API data encryption is required;  
*“iAM Smart” API for CorpID (POST: Request accessToken & Tokenised ID)*
- Step 26-27. When e-Service Server gets the cAuthCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user. API data

encryption is required;

*CorpID API Spec (EXT-002: Get Access Token)*

Step 28-31. e-Service Server invokes the CorpID API with the cAccessToken, cOpenID, “iAMToken” (“iAM Smart” CEK encrypted AccessToken and OpenID) to obtain the result of anonymous form pre-filling request. API data encryption is required;

“iAM Smart” profile is returned in CEK encrypted “iAMContent”;

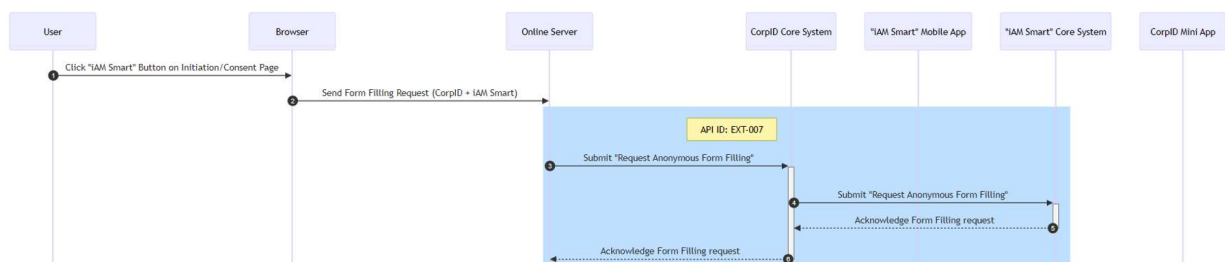
*CorpID API Spec (EXT-008: Get Anonymous Form Pre-filling Result)*

Step 32. e-Service Server processes and shows the result in the corresponding e-Service Website.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No | API Name                                          | API Reference (Section)               |
|----|---------------------------------------------------|---------------------------------------|
| 1  | Request Anonymous Form Pre-Filling                | CorpID API Specification Section 4.8  |
| 2  | Authentication by requesting QR Page/ Broker Page | CorpID API Specification Section 4.1  |
| 3  | Callback with authCode to e-Service Server        | CorpID API Specification Section 4.9  |
| 4  | Get Access Token                                  | CorpID API Specification Section 4.2  |
| 5  | Get Anonymous Form Pre-filling Result             | CorpID API Specification Section 4.10 |

### 3.6.1.1 Implementing (1): EXT-007: Request Anonymous Form Pre-Filling



#### Pre-conditions

- e-Service Website and “iAM Smart” Mobile App are in different devices in this scenario.
- e-Service generates a “businessID” for this request and uses this identifier to match the cAuthCode returned from “iAM Smart” System.
- API data encryption is required.

#### Post-conditions

- API data decryption is required.

#### Error conditions

- For common errors, please refer to CorpID API Specification.

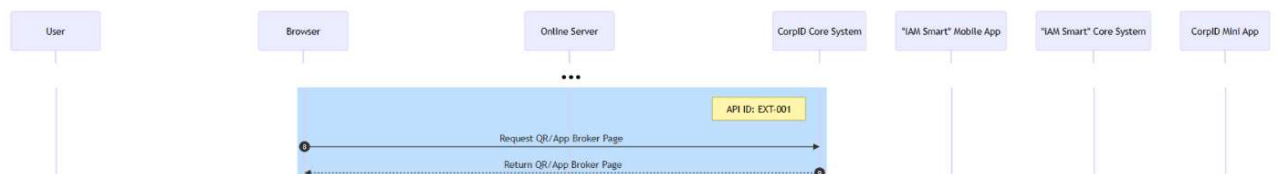
#### Request and Response Parameters

- Please refer to CorpID API Specification.

#### Notes

- Please refer to Section 3.5.2.1, except for the following parameters:
  - No request parameter “cAccessToken”, “cOpenID”, “iAMToken”, “source”, “redirectURI” and “state”.
  - No response parameter “authByQR”.
  - Response parameter “ticketID”: It is returned from CorpID System for CorpID and “iAM Smart” APIs for anonymous request. It will be used for invoking subsequent CorpID APIs depending on the scenario. It is a 36-byte (or less) UUID number (ASCII character set).

### 3.6.1.2 Implementing (2): EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request



#### Pre-conditions

- Please refer to Section 3.4.1.1 except:
  - This one is CorpID API, for detail please refer to CorpID API Specification.
  - e-Service has the “ticketID” provided by “iAM Smart” System for the anonymous request in step 6.
  - Authorisation scope submitted should be “corpidapi\_formfilling”.

#### Post-conditions

- Please refer to CorpID API Specification.

## Error conditions

- This CorpID API does not return JSON response (i.e., no application error will return).

## Request Parameters

- Please refer to CorpID API Specification.

## Notes

- Please refer to Section 3.4.1.1 except the following parameter:
  - Request parameter “ticketID”: Value provided by CorpID Server for the anonymous request.
  - Request parameter “scope”: It should be “corpidapi\_formfilling”.

### 3.6.1.3 Implementing (3): CLB-003: Callback with authCode for Anonymous Form Pre-Filling



## Pre-conditions

- Please refer to Section 3.4.1.2. except:
  - This one is CorpID API, for detail please refer to CorpID API Specification.

## Post-conditions

- Please refer to Section 3.4.1.2 except:
  - This one is CorpID API, for detail please refer to CorpID API Specification.
  - e-Service Server should match the callback result with corresponding e-Service client terminal using the “businessID”.

## Error conditions

- This e-Service callback API is a HTTP GET request. If parameter “cError\_code” is returned, it means the request is failed.

| Error Code           | Error Description                       | Suggested Action                                      |
|----------------------|-----------------------------------------|-------------------------------------------------------|
| cError_code - M60000 | User cancelled form pre-filling request | Inform user the Form Pre-Filling request is cancelled |
| cError_code - M60001 | User rejected form pre-filling request  | Inform user the Form Pre-Filling request is rejected  |

|                      |                                    |                                                                    |
|----------------------|------------------------------------|--------------------------------------------------------------------|
| cError_code - M60002 | Failed to request form pre-filling | Inform user the Form Pre-Filling request is failed and retry later |
|----------------------|------------------------------------|--------------------------------------------------------------------|

The error\_code M60003 (Form pre-filling request timeout) does not appear in this scenario. For the QR code timeout or request confirmation timeout in the “iAM Smart” Mobile App, message will be prompted in the corresponding user interface.

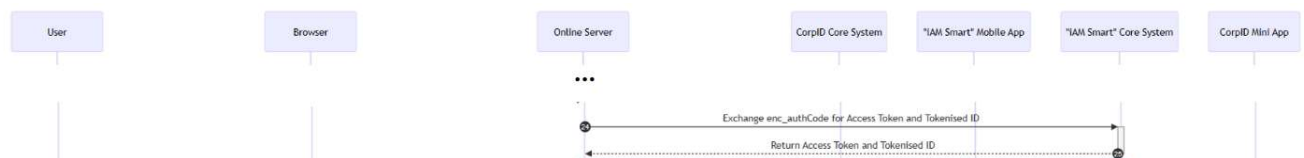
### Callback Parameters

- Please refer to Section 3.4.1.2.

### Notes

- Please refer to Section 3.4.1.2 except the following parameter:
  - Callback parameter “businessID”: Value returned by CorpID System should be the same one as e-Service submitted in the anonymous request.

### 3.6.1.4 Implementing (4): “iAM Smart” API for CorpID - POST: Request accessToken & Tokenised ID



### Pre-conditions

- Please refer to Section 3.4.1.3.

### Post-conditions

- Please refer to Section 3.4.1.3. except:
  - The accessToken can only be used once before expiry.

### Error conditions

- Please refer to Section 3.4.1.3.

### Request and Response Parameters

- Please refer to Section 3.4.1.3.

### Notes

- Please refer to Section 3.4.1.3.

### 3.6.1.5 Implementing (5): EXT-002: Get Access Token



#### Pre-conditions

- Please refer to Section 3.4.1.4.

#### Post-conditions

- Please refer to Section 3.4.1.4. except:
  - The cAccessToken can only be used once before expiry.
  - The corpCount and corpInfo are not used. CorpID User will select the corporation in CorpID Mini-program.

#### Error conditions

- Please refer to Section 3.4.1.4.

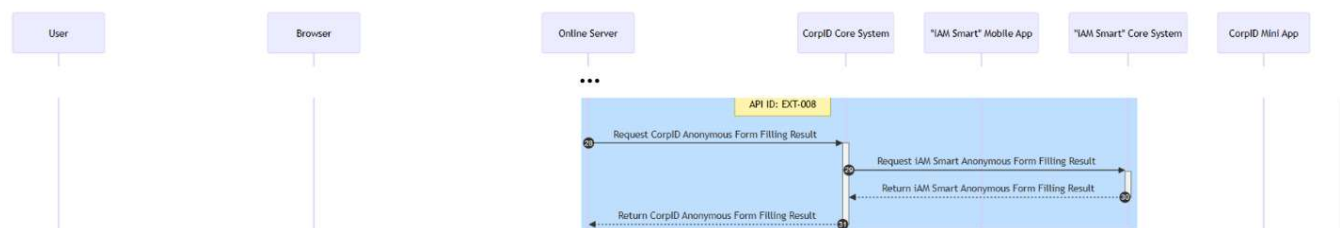
#### Request and Response Parameters

- Please refer to Section 3.4.1.4.

#### Notes

- Please refer to Section 3.4.1.4. except:
  - The cAccessToken can only be used once before expiry.
  - CorpID User will select the corporation in CorpID Mini-program. e-Service does not need to implement select corporation UI in e-Service Web or App.

### 3.6.1.6 Implementing (6): EXT-008: Get Anonymous Form Pre-filling Result



#### Pre-conditions

- e-Service must possess a valid cAccessToken of the CorpID user with authorisation scope “form pre-filling authorisation”.
- e-Service must possess a valid accessToken of the “iAM Smart” user with authorisation scope “form pre-filling authorisation”. The openID and accessToken of the “iAM Smart”, as parameter “iAMToken”, should be encrypted with “iAM Smart” CEK

#### Post-conditions

- API data decryption is required.
- e-Service Server should check the requested data fields has authorised for the Anonymous Form pre-filling.
- e-Service should discard the cAccessToken and accessToken as they can only be used once.

#### Error conditions

- For common errors, please refer CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description                  | Suggested Action                                                                       |
|---------------|------------------------------------|----------------------------------------------------------------------------------------|
| code - M60002 | Failed to request Form Pre-Filling | Inform user the Form Pre-Filling request is failed and retry later                     |
| code - M60003 | Form Pre-Filling request timeout   | Inform user the Form Pre-Filling request is timeout, and provide way for user to retry |

#### Request and Response Parameters

- Please refer to CorpID API Specification.

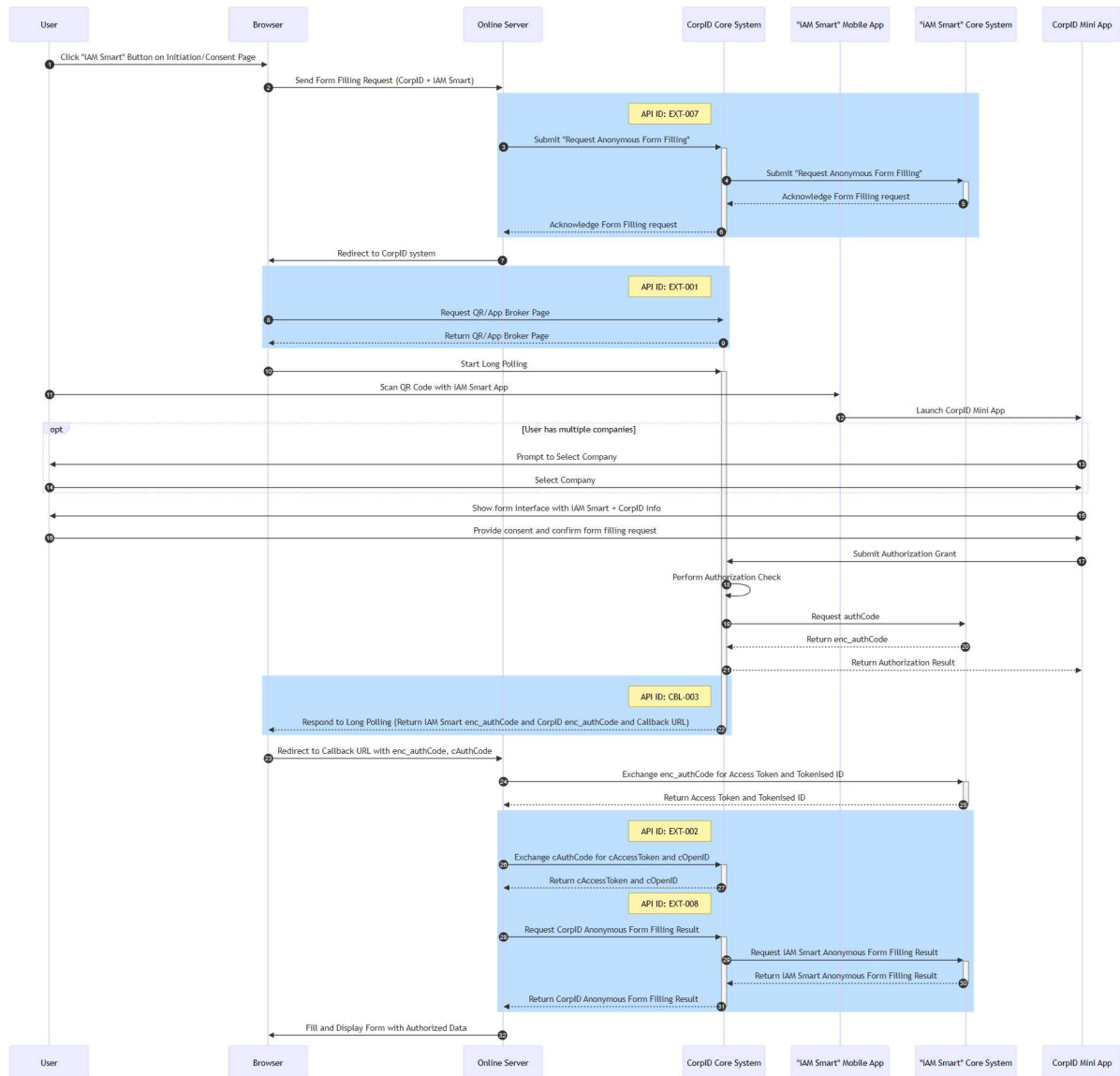
#### Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- CorpID profile field items or “iAM Smart” profile field items not authorised by CorpID user will not return in result (i.e., JSON name/value of that item will not exist in result).
- “iAMSecret”, the “iAM Smart”
- CEK, has to decrypted by “iAM Smart” KEK.
- “iAMContent” has to decrypted by “iAM Smart” CEK.

### 3.6.2 Scenario 2: Anonymous Form pre-filling (e-Service Website in Same Device)

The sequence diagram below shows how an anonymous user authorises an e-Service to use his/her CorpID Fields and “iAM Smart” Profile Fields for form pre-filling when e-Service website and the “iAM Smart” Mobile App are running in the same device.

## CorpID Anonymous Form Filling (e-Service Website in Same Device)



- Step 1. User reads the consent page (if CorpID / “iAM Smart” profile fields requested) and clicks the “IAM Smart” button in e-Service Website;
- Step 2. e-Service Website initiates anonymous Form Pre-Filling request to e-Service Server with browser's user agent name (use as value for request parameter “source”);
- Step 3. The e-Service Server initiates an Anonymous Form Pre-Filling Request to the CorpID Core System with the “businessID” of this request, required data fields, etc. API data encryption is required;  
*CorpID API Spec. (EXT-007: Request Anonymous Form Pre-Filling)*
- Step 4-5. The CorpID Server and “iAM Smart” Server verifies and confirms receipt of the anonymous Form Pre-Filling request to e-Service Server by returning POST response with parameter “ticketID”;

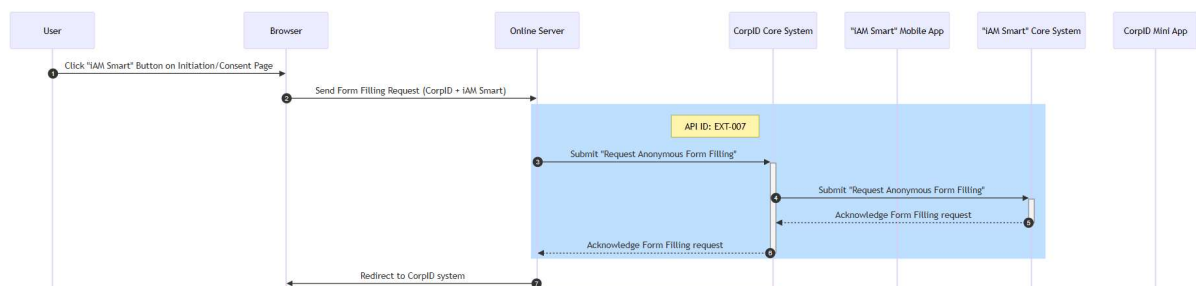
- Step 6-7. e-Service Server prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as browser’s user agent value), “scope”, “brokerPage” (set as “true”), “ticketID”, etc. and constructs the GET request to invoke the CorpID API using browser redirection;  
*CorpID API Spec. (EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request)*
- Step 8. CorpID System returns the broker page (if e-Service has set “brokerPage” to “true”) and after the broker page fails to find the “iAM Smart” Mobile App, it will request QR page to be displayed in browser. If e-Service does not request for broker page (i.e., no broker page will be returned), the browser will directly display QR Code page;
- Step 9-12. Broker page invokes the CorpID Mini-program in the same device automatically and sends it the relevant parameters. CorpID user logs in CorpID Mini-program, reviews the Form Pre-Filling authorisation request (e.g., continue or reject the request) and authorises those selected CorpID Profile fields and “eMEFields” to e-Services (e.g., authorise only his HKIC number and English name in account information and residential address in “e-ME profile” (unchecked other “eMEFields”) );
- Step 13-14. If the user has multiple corporations, the CorpID Mini-program will also allow the user to select a corporation.
- Step 15-21. CorpID and “iAM Smart” System verifies the validity of necessary information, and return the authorisation result to “iAM Smart” Mobile App.  
CorpID Mini-program invokes the original e-Service browser (i.e., using the browser's user agent value submitted);
- Step 22. CorpID and “iAM Smart” System receives the polling result returned by CorpID System which includes cAuthCode (of CorpID), authCode (of “iAM Smart”) and businessID or any error code (e.g., CorpID user rejects the request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection;  
*CorpID API Spec. (CLB-003: Callback with authCode for Anonymous Form Pre-Filling)*
- Step 23-25. When e-Service Server gets the authCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the “iAM Smart” API to obtain the accessToken and the Tokenised ID (i.e., openID) of the “iAM Smart” user. API data encryption is required;  
*“iAM Smart” API for CorpID (POST: Request accessToken & Tokenised ID)*
- Step 26-27. When e-Service Server gets the cAuthCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user. API data encryption is required;  
*CorpID API Spec (EXT-002: Get Access Token)*

Step 28-31. e-Service Server invokes the CorpID API with the cAccessToken, cOpenID, “iAMToken” (“iAM Smart” CEK encrypted AccessToken and OpenID) to obtain the result of anonymous form pre-filling request. API data encryption is required; “iAM Smart” profile is returned in CEK encrypted “iAMContent”;

*CorpID API Spec (EXT-008: Get Anonymous Form Pre-filling Result)*

Step 32. e-Service Server processes and shows the result in the corresponding e-Service Website.

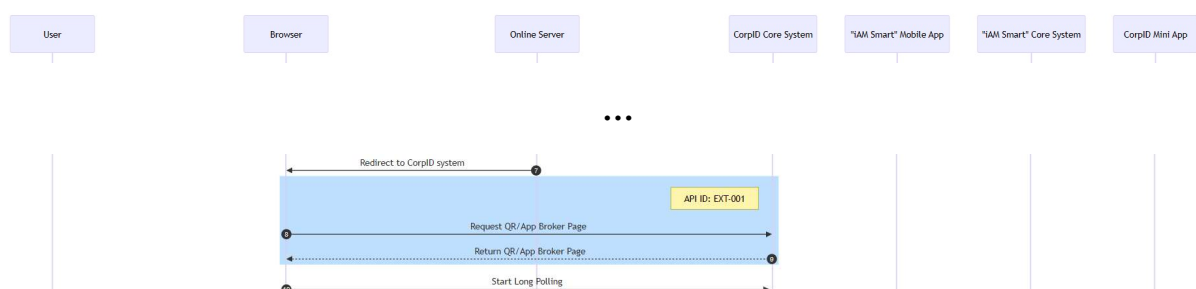
### 3.6.2.1 Implementing (1) POST: Request Anonymous Form Pre-Filling



Please refer to Section 3.6.1.1 except:

- e-Service Website and “iAM Smart” Mobile App are in the same device in this scenario.

### 3.6.2.2 Implementing (2) GET: Request QR Page



#### Pre-conditions

- Please refer to Section 3.4.2.1 except:
  - e-Service has the “ticketID” provided by “iAM Smart” System for the anonymous request in step 4.
  - Authorisation scope submitted should be “eidapi\_formFilling”.

#### Post-conditions

- Please refer to Section 3.4.2.1.

#### Error conditions

- Please refer to Section 3.4.2.1.

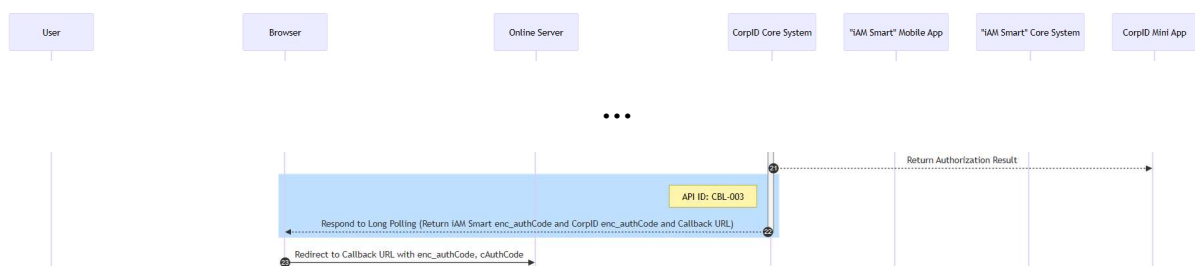
#### Request Parameters

- Please refer to Section 3.4.2.1.

#### Notes

- Please refer to Section 3.4.2.1 except the following parameter:
  - Request parameter “ticketID”: provided by “iAM Smart” System for the anonymous request.
  - Request parameter “scope”: It should be “eidapi\_formFilling”.

### 3.6.2.3 Implementing (3) GET: Callback with authCode to e-Service Server



#### Pre-conditions

- Please refer to Section 3.4.2.2.

#### Post-conditions

- Please refer to Section 3.4.2.2 except:
  - e-Service Server should match the callback result with corresponding e-Service Website using the “businessID”.

#### Error conditions

- Please refer to Section 3.4.2.2

#### Callback Parameters

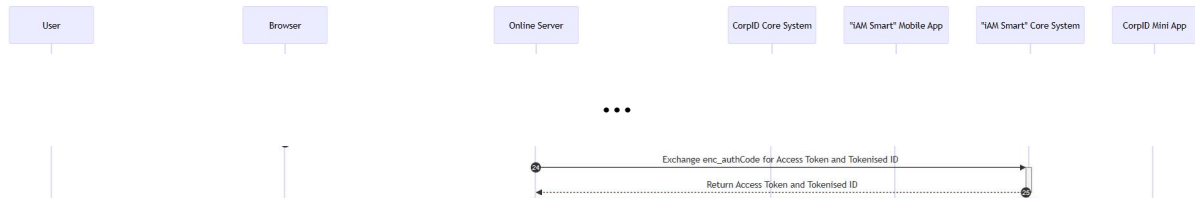
- Please refer to Section 3.4.2.2.

#### Notes

- Please refer to Section 3.4.2.2 except the following parameter:

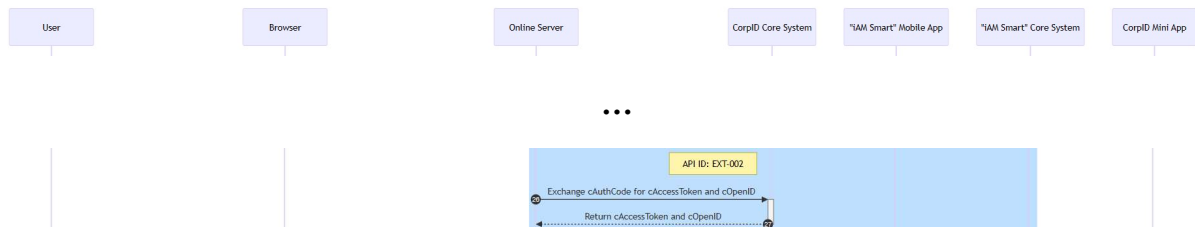
- Callback parameter “businessID”: Value returned by “iAM Smart” System should be the same one as e-Service submitted in the anonymous request.

#### 3.6.2.4 Implementing (4) POST: Request accessToken & Tokenised ID



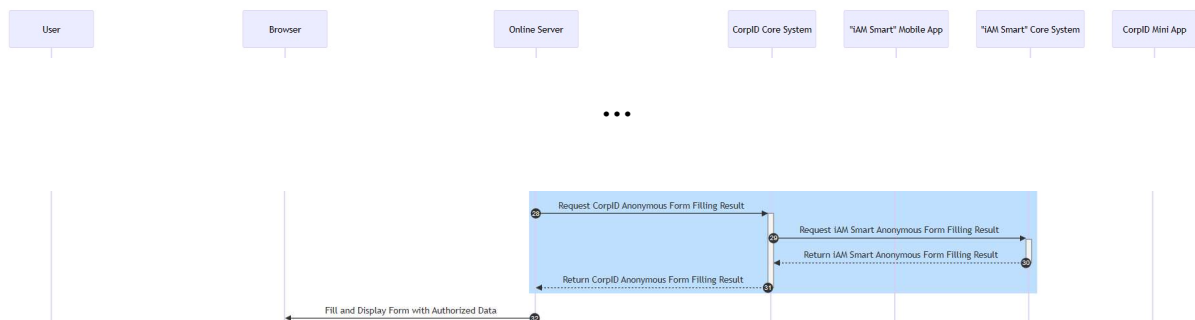
Please refer to Section 3.4.1.3.

#### 3.6.2.5 Implementing (5): ext-002: Get Access Token



Please refer to Section 3.4.1.4.

#### 3.6.2.6 Implementing (5) POST: Obtain Anonymous Form Pre-Filling Result

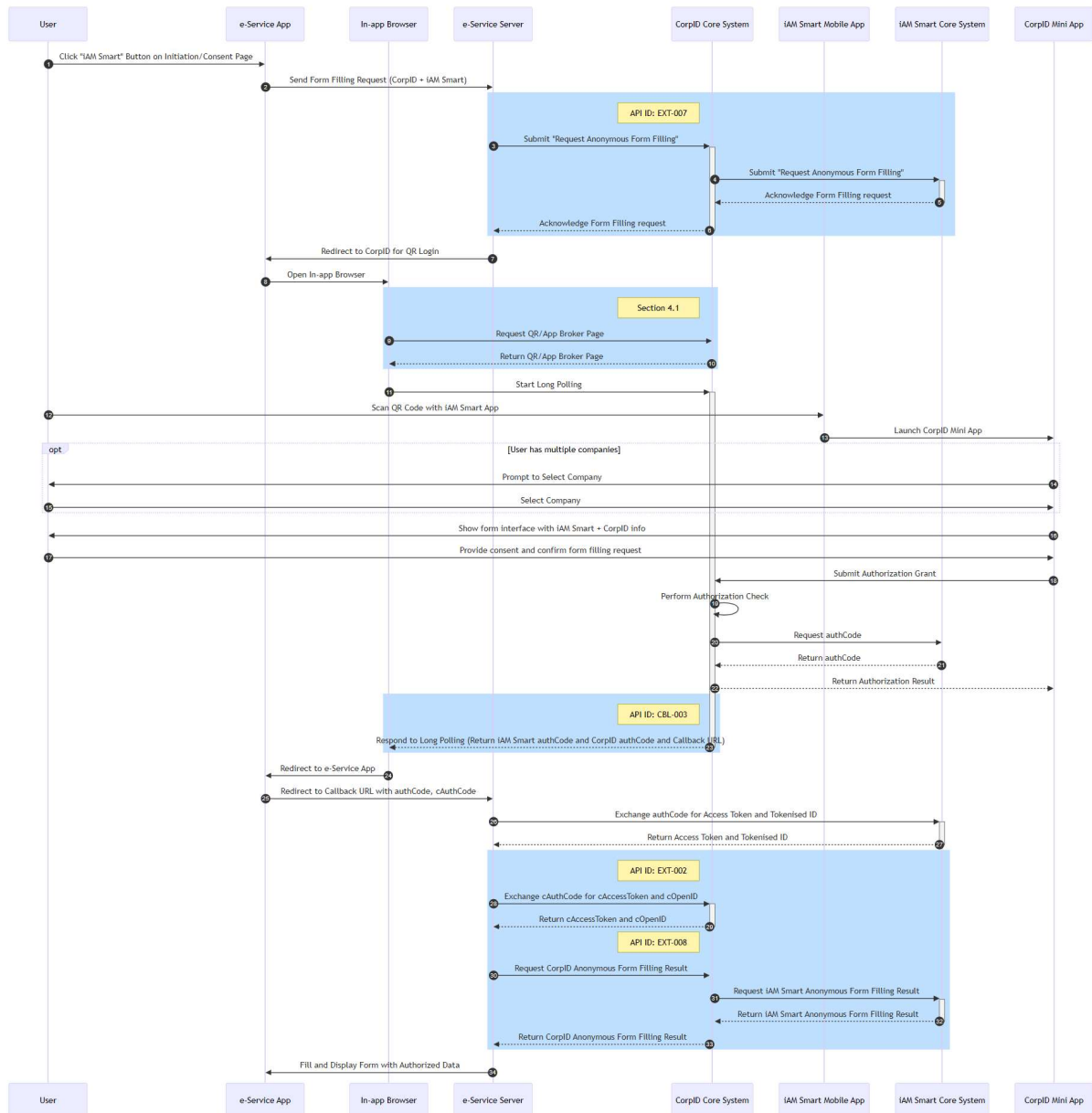


Please refer to Section 3.8.1.6.

### 3.6.3 Scenario 3: Anonymous Form pre-filling (e-Service App in Different Device)

The sequence diagram below shows how an anonymous user authorises an e-Service to use his/her CorpID Fields and “iAM Smart” Profile Fields for form pre-filling when e-Service App and the “iAM Smart” Mobile App are running in different devices.

## CorpID Anonymous Form Filling (e-Service App in Different Device)



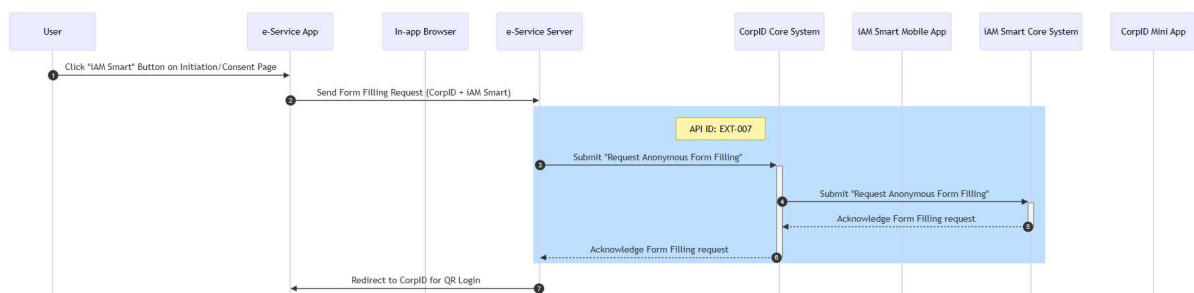
- Step 1. User reads the consent page (if CorpID / “iAM Smart” profile fields requested) and clicks the “IAM Smart” button in e-Service App;
- Step 2. e-Service App determines there is no “iAM Smart” Mobile App in the device using program code, initiates anonymous Form Pre-Filling request to e-Service Server with the in-app browser’s user agent value (use as value for request parameter “source”);
- Step 3. e-Service Server initiates an anonymous Form Pre-Filling request to the CorpID Core System with the “businessID” of this request, required data fields, etc. API data encryption is required;  
*CorpID API Spec. (EXT-007: Request Anonymous Form Pre-Filling)*

- Step 4. The CorpID Server and “iAM Smart” Server verifies and confirms receipt of the anonymous Form Pre-Filling request to e-Service Server by returning POST response with parameter “ticketID”;
- Step 5-7. After receiving the response, e-Service Server prepares necessary parameters such as “clientID”, “redirectURI”, “scope”, “source” (set as in-app browser’s user agent value), “brokerPage” (set as “false”), “ticketID”, etc. and constructs the GET request to invoke the CorpID API using browser redirection;  
*CorpID API Spec. (EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request);*
- Step 8. e-Service App invokes in-app browser (Safari for iOS, Chrome for Android) to submit the GET request;
- Step 8-10. Browser opens the GET request to invoke the “iAM Smart” API and QR Code page will be shown;  
*“iAM Smart” API (GET: Request QR page)*
- Step 11. QR Code page of “iAM Smart” System polls “iAM Smart” System for the QR Code status;
- Step 12-13. CorpID user uses “iAM Smart” Mobile App to scan the QR Code, launches the CorpID Mini-program, reviews the form pre-filling authorisation request (e.g., continue or reject the request) and authorises those selected CorpID Profile fields and “eMEFields” to e-Services (e.g., authorise only his HKIC number and English name in account information and residential address in “e-ME profile” (unchecked other “eMEFields”));  
If the user has multiple corporations, the CorpID Mini-program will also allow the user to select a corporation.
- Step 14-15. CorpID and “iAM Smart” System verifies the validity of QR code and other necessary information, and return the authorisation result to CorpID Mini-program;
- Step 16-24. QR Code page of CorpID and “iAM Smart” System receives the polling result returned by CorpID System (i.e., response for the polling in Step 10) which includes cAuthCode (of CorpID), authCode (of “iAM Smart”) and businessID or any error code (e.g., CorpID user rejects the request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection;  
*CorpID API Spec. (CLB-003: Callback with authCode for Anonymous Form Pre-Filling)*
- Step 25-27. When e-Service Server gets the authCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the “iAM Smart” API to obtain the accessToken and the Tokenised ID (i.e., openID) of the “iAM Smart” user. API data encryption is required;  
*“iAM Smart” API (POST: Request accessToken & Tokenised ID)*

- Step 28-29. When e-Service Server gets the cAuthCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user. API data encryption is required;  
*CorpID API Spec (EXT-002: Get Access Token)*
- Step 30-33. e-Service Server invokes the CorpID API with the cAccessToken, cOpenID, “iAMToken” (“iAM Smart” CEK encrypted AccessToken and OpenID) to obtain the result of anonymous form pre-filling request. API data encryption is required;  
 “iAM Smart” profile is returned in CEK encrypted “iAMContent”;  
*CorpID API Spec (EXT-008: Get Anonymous Form Pre-filling Result)*
- Step 34. e-Service Server processes and shows the result in the corresponding e-Service App.

### 3.6.3.1 Implementing (1) POST: Request Anonymous Form pre-filling

CorpID Anonymous Form Filling (e-Service App in Different Device)



#### Pre-conditions

- Please refer to Section 3.6.1.1 except:
  - e-Service should determine the “iAM Smart” Mobile App is not in the same device of e-Service App using program code.

#### Post-conditions

- Please refer to Section 3.6.1.1.

#### Error conditions

- Please refer to Section 3.6.1.1.

#### Request and Response Parameters

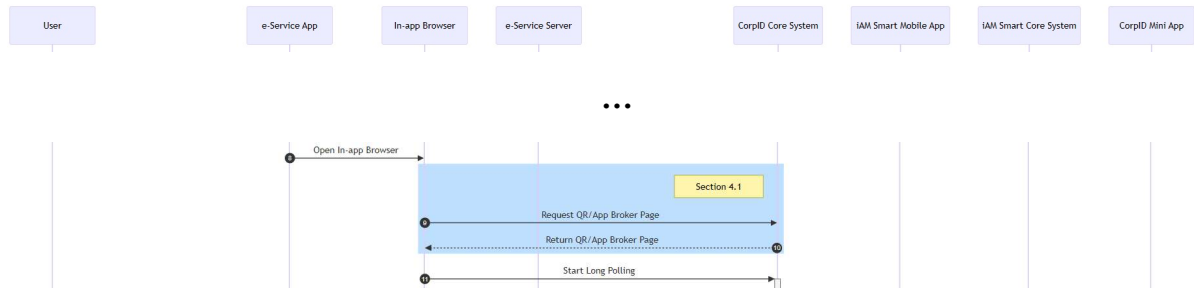
- Please refer to Section 3.6.1.1.

#### Notes

- Please refer to Section 3.6.1.1, except for the following parameter:

- Request parameter “source”: Value should be the in-app browser's user agent value.

### 3.6.3.2 Implementing (2): Authentication by Requesting QR Page / Broker Page for Anonymous Request



#### Pre-conditions

- Please refer to Section 3.4.3.1 except:
  - e-Service has the “ticketID” provided by “iAM Smart” System for the anonymous request in step 4.
  - Authorisation scope submitted should be “eidapi\_formFilling”.

#### Post-conditions

- Please refer to Section 3.4.3.1.

#### Error conditions

- Please refer to Section 3.4.3.1.

#### Request Parameters

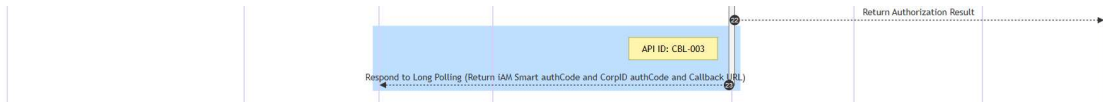
- Please refer to “iAM Smart” API Specification.

#### Notes

- Please refer to Section 3.4.3.1 except the following parameter:
  - Request parameter “ticketID”: Value provided by “iAM Smart” System for the anonymous request.
  - Request parameter “scope”: It should be “eidapi\_formFilling”.

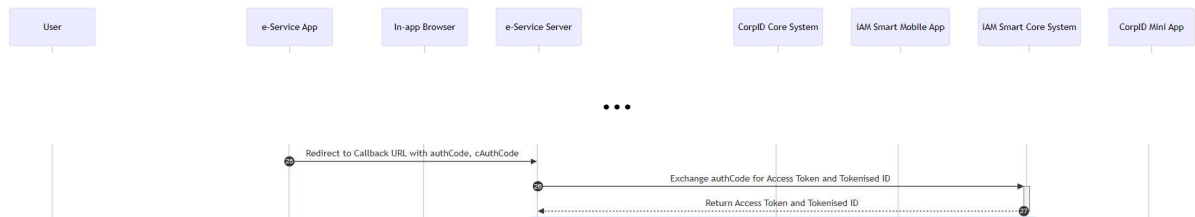
### 3.6.3.3 Implementing (3) GET: Callback with authCode to e-Service Server





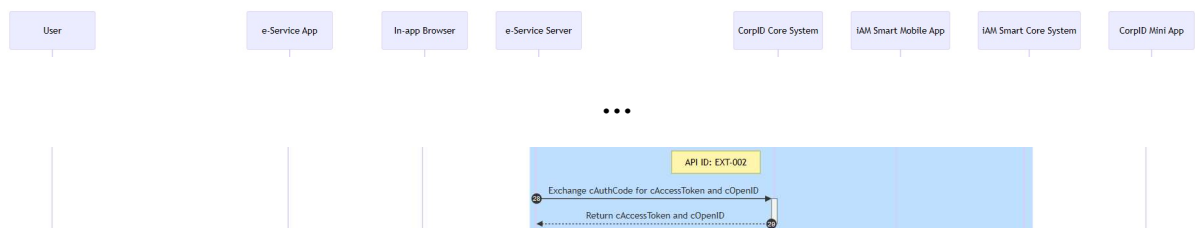
Please refer to Section 3.4.2.2.

### 3.6.3.4 Implementing (4) POST: Request accessToken & Tokenised ID



Please refer to Section 3.4.2.3.

### 3.6.3.5 Implementing (5) POST: Get Access Token

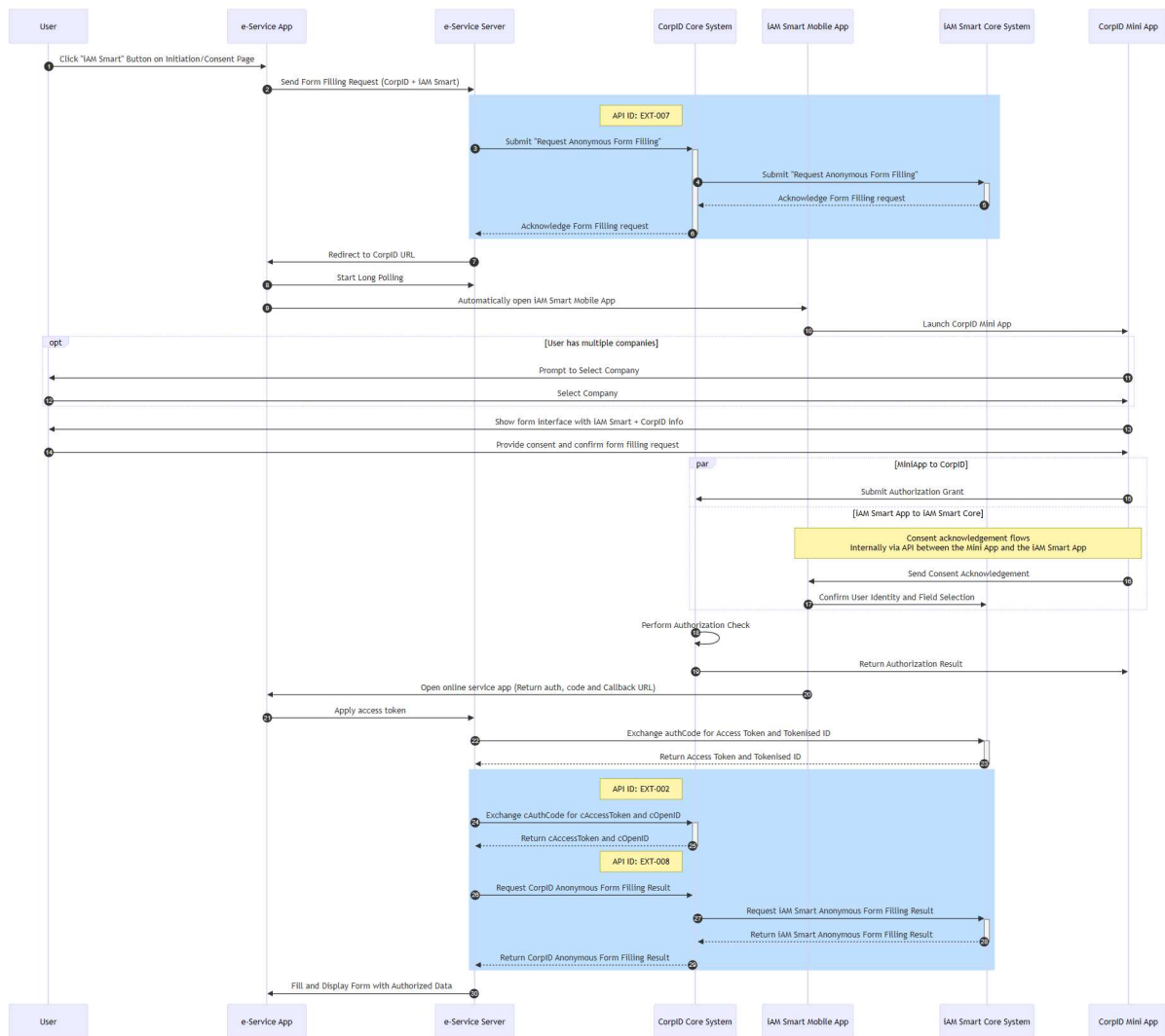


Please refer to Section 3.4.2.4

## 3.6.4 Scenario 4: Anonymous Form pre-filling (e-Service App in Same Device)

The sequence diagram below shows how an anonymous user authorises an e-Service to use his/her CorpID Fields and “iAM Smart” Profile Fields for form pre-filling when e-Service App and the “iAM Smart” Mobile App are running in the same device.

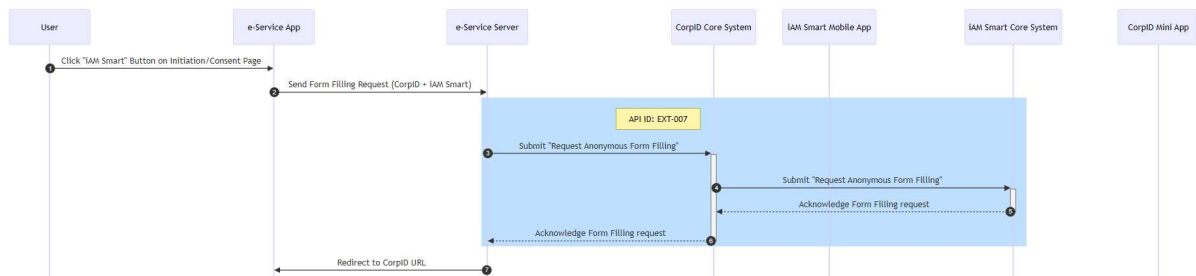
## CorpID Anonymous Form Filling (e-Service App in Different Device)



- Step 1. User reads the consent page (if CorpID / “iAM Smart” profile fields requested) and clicks the button in e-Service App;
- Step 2. e-Service App determines “iAM Smart” Mobile App is installed in the device using program code, initiates anonymous Form Pre-Filling request to e-Service Server with “App\_Scheme” or “App\_Link” (use as value for request parameter “source”);
- Step 3-7. The e-Service Server initiates an Anonymous Form Pre-Filling Request to the CorpID Core System with the “businessID” of this request, required data fields, etc. API data encryption is required;  
*CorpID API Spec. (EXT-007: Request Anonymous Form Pre-Filling)*  
 The CorpID Server and “iAM Smart” Server verifies and confirms receipt of the anonymous Form Pre-Filling request to e-Service Server.
- Step 8. e-Service App polls e-Service Server for the Form Pre-Filling result from “iAM Smart” System;

- Step 9. e-Service App invokes “iAM Smart” Mobile App using URL Scheme with request parameters (set <Context> as “corpId” in URL Scheme, and parameters as anonymous form pre-filling);  
*“iAM Smart” API for CorpID (URL Scheme: Open “iAM Smart” Mobile App for Form Filling)*
- Step 10-17. CorpID user logs in CorpID Mini-program, reviews the Form Pre-Filling authorisation request (e.g., continue or reject the request) and authorises those selected CorpID Profile fields and “eMEFields” to e-Services (e.g., authorise only his HKIC number and English name in account information and residential address in “e-ME profile” (unchecked other “eMEFields”));  
 If the user has multiple corporations, the CorpID Mini-program will also allow the user to select a corporation.;
- Step 18-20. CorpID and “iAM Smart” System verifies the validity of necessary information, and return the authorisation result to “iAM Smart” Mobile App.  
*“iAM Smart” Mobile App invokes e-Service App using URL Scheme or Universal Link/App Link with required parameters such as “authCode”, “businessID”;*  
*CorpID API (Callback with authCode to e-Service App)*
- Step 21. e-Service App sends authCode, businessID, etc. to e-Service Server;
- Step 22-23. When e-Service Server gets the authCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the “iAM Smart” API to obtain the accessToken and the Tokenised ID (i.e., openID) of the “iAM Smart” user. API data encryption is required;  
*“iAM Smart” API for CorpID (POST: Request accessToken & Tokenised ID)*
- Step 24-25. When e-Service Server gets the cAuthCode and businessID, it verifies businessID as generated in Step 3 and should then invoke the CorpID API to obtain the cAccessToken and the Tokenised ID (cOpenID) of the CorpID user. API data encryption is required;  
*CorpID API Spec (EXT-002: Get Access Token)*
- Step 26-29. e-Service Server invokes the CorpID API with the cAccessToken, cOpenID, “iAMToken” (“iAM Smart” CEK encrypted AccessToken and OpenID) to obtain the result of anonymous form pre-filling request. API data encryption is required;  
*“iAM Smart” profile is returned in CEK encrypted “iAMContent”;*  
*CorpID API Spec (EXT-008: Get Anonymous Form Pre-filling Result)*
- Step 30. e-Service Server processes and shows the result in the corresponding e-Service App.

### 3.6.4.1 Implementing (1) POST: Request Anonymous Form pre-filling



#### Pre-conditions

- Please refer to Section 3.6.1.1 except:
  - e-Service should determine the “iAM Smart” Mobile App is on the same device of e-Service App using program code.

#### Post-conditions

- Please refer to Section 3.6.1.1.

#### Error conditions

- Please refer to Section 3.6.1.1.

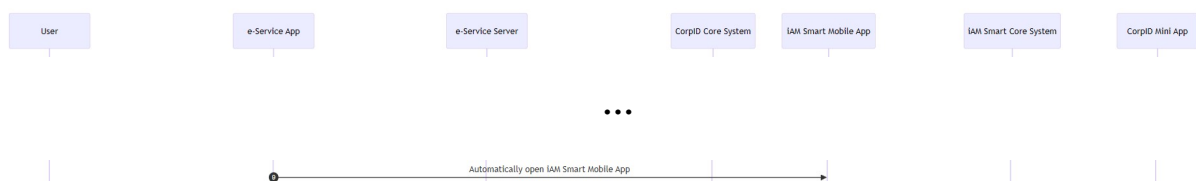
#### Request and Response Parameters

- Please refer to Section 3.6.1.1.

#### Notes

- Please refer to Section 3.6.1.1.

### 3.6.4.2 Implementing (2) URL Scheme: Open CorpID Mini-program for Getting Context



#### Pre-conditions

- e-Service App and “iAM Smart” Mobile App are in the same user device.
- e-Service has the “ticketID” provided by “iAM Smart” System for the anonymous request in step 4.
- e-Service should determine if it would generate the optional “state” request parameter to prevent CSRF attack. The “state” will be returned to e-Service for checking through the

e-Service callback API for receiving the authorisation code from “iAM Smart” System in Step 12.

#### Post-conditions

- “iAM Smart” Mobile App will be launched.
- e-Service App should keep synchronising with e-Service server for the result from “iAM Smart” System.

#### Error conditions

- Nil

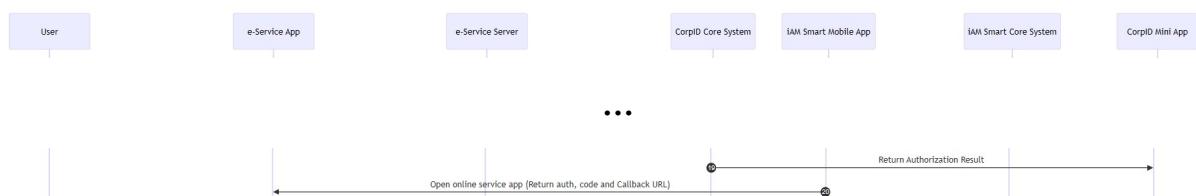
#### Request Parameters

- Please refer to “iAM Smart” API Specification.

#### Notes

- Request parameter “ticketID”: Value provided by “iAM Smart” System for the anonymous request.
- Request parameter “source”: Value should be “App\_Scheme” (for e-Service App URL scheme), or “App\_Link” (for the Universal Link/App Link).
- Request parameter “redirectURI”: This is e-Service App URL scheme or Universal Link/App Link depending on the “source” parameter, which is used for receiving the authorisation code. The value should be URL encoded. For details, please refer to Section 6.4.1 of “iAM Smart” API Specification. The value must be the same as provided during e-Service registration.
- Request parameter “state”: The value should be URL encoded.
- Set <Context> as “anon\_form-filling” in URL Scheme.

### 3.6.4.3 Implementing (3) Callback with cAuthCode to e-Service App



#### Pre-conditions

- Please refer to Section 3.4.4.2.

#### Post-conditions

- Please refer to Section 3.4.4.2 except:

- e-Service Server should match the callback result with corresponding e-Service App using the “businessID”.

### Error conditions

- This e-Service callback API is a URL Scheme, or Universal Link/App Link. If parameter “error\_code” is returned, it means the authentication request is failed.

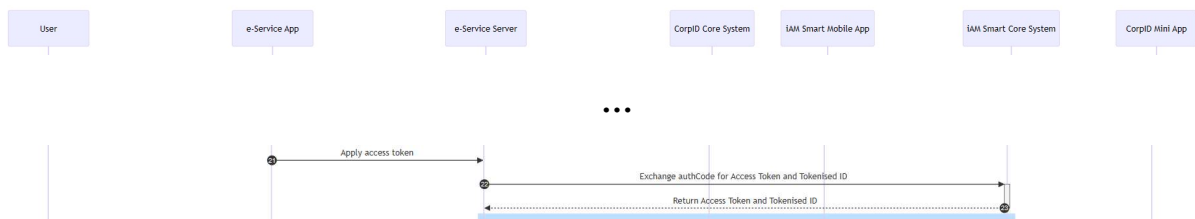
### Callback Parameters

- Please refer to CorpID API Specification.

### Notes

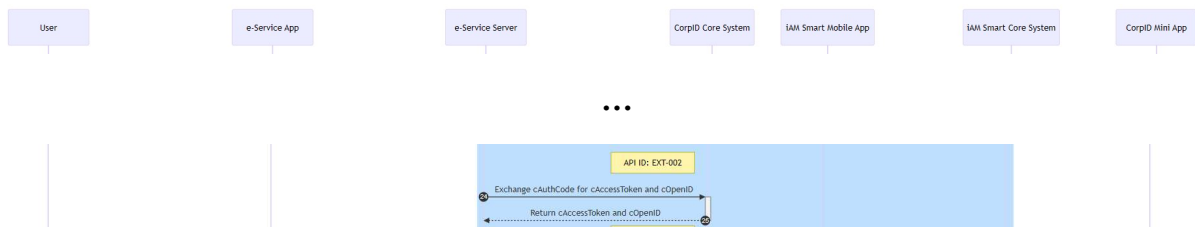
- Please refer to Section 3.4.4.2 except the following parameter:
  - Callback parameter “businessID”: Value returned by “iAM Smart” System should be the same one as e-Service submitted in the anonymous request.

#### 3.6.4.4 Implementing (4) POST: Request accessToken & Tokenised ID



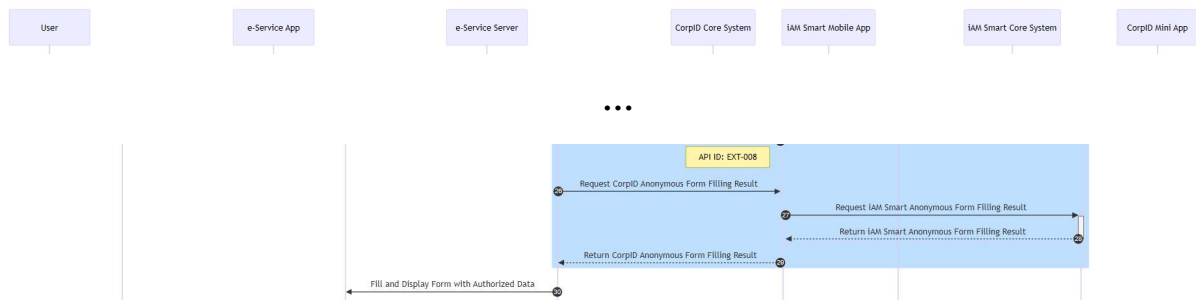
Please refer to Section 3.4.1.3.

#### 3.6.4.5 Implementing (5) POST: Get Access Token



Please refer to Section 3.6.1.5.

### 3.6.4.6 Implementing (6) POST: Obtain Anonymous Form pre-filling Result



Please refer to Section 3.6.1.6.

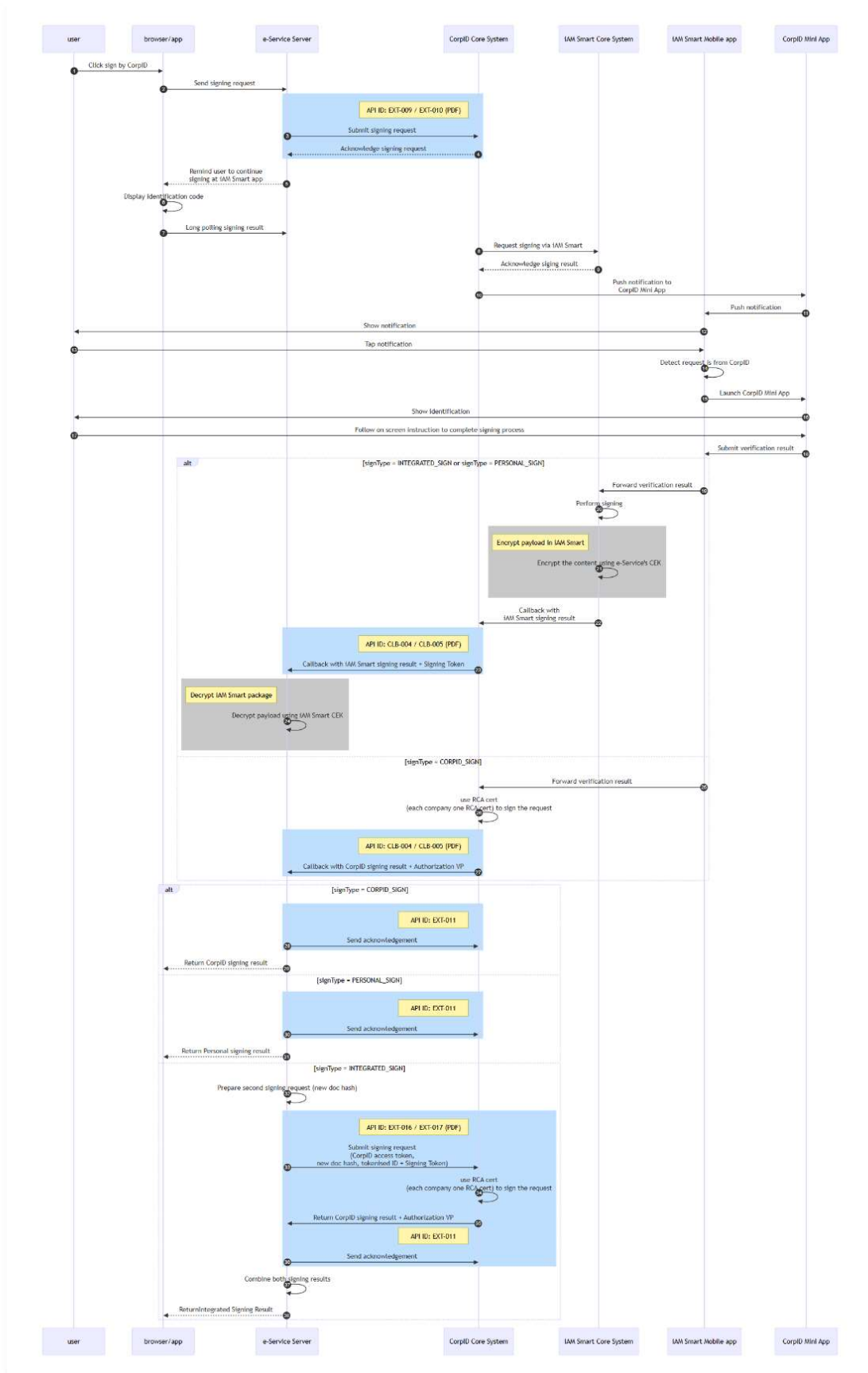
## 3.7 WORKFLOWS FOR DIGITAL SIGNING WITH SERVICE LOGIN

The following process includes non-anonymous hash digital signing and non-anonymous pdf digital signing.

### 3.7.1 Scenario 1: Digital Signing (e-Service Website/App in Different Device)

The sequence diagram below shows how an authenticated user authorises and signs the Document Hash when e-Service website and the “iAM Smart” Mobile App are running in different devices.

## CorpID Digital Signing (e-Service Website/App in Different Device)



- Step 1. User clicks Sign by CorpID and the browser/app sends a signing request to the e-Service Server.
- Step 2-3. The e-Service Server submits the signing request, with “Request Digital Signing” API for binary and general content or “Request PDF Digital Signing” for PDF, with “businessID”, “cOpenID”, “cAccessToken”, “iAMToken”, “state”, “source”, “hashCode”, “signType” and other required parameters, to CorpID Core System, with response with “authByQR” (set to “true”) to indicate different device.
- If “signType” is “INTEGRATED\_SIGN” or “PERSONAL\_SIGN”, the request trigger the personal signing (“iAM Smart” for HKIC Holders, signing partner for passport holders). If “signType” is “CORPID\_SIGN”, the request trigger the CorpID-only signing.
- If “HKICHash” is passed, the 4-digit identification code will be checked.
- CorpID Core System acknowledges the signing request and the e-Service reminds the user to continue in iAM Smart.
- CorpID API Spec. (EXT-009: Request Digital Signing)*  
*CorpID API Spec. (EXT-010: Request PDF Digital Signing)*
- Step 4. The browser/app displays an identification code and starts long polling for the signing result.
- Step 5-18. CorpID Core System requests signing via “iAM Smart” and receives acknowledgement. CorpID Core System pushes a notification to CorpID Mini-program, which forwards a push notification to “iAM Smart” Mobile App. The user sees the notification, taps it, and “iAM Smart” detects that the request is from CorpID. “iAM Smart” Mobile App launches CorpID Mini-program. CorpID Mini-program shows the consent page, identification code and the user follows on-screen instructions to complete signing. CorpID Mini-program submits the verification result to “iAM Smart” Mobile App.
- Step 19-24. For INTEGRATED\_SIGN or PERSONAL\_SIGN,  
 “iAM Smart” forwards verification, performs signing, encrypts the payload with the e-Service CEK, and callbacks with the iAM Smart signing result. CorpID Core System callbacks to the e-Service Server with the “iAM Smart” signing result and Signing Token. The API is encrypted with CorpID CEK.  
 The e-Service Server decrypts the iAM Smart package using the iAM Smart CEK.
- CorpID API Spec. (CLB-004: Digital Signing Callback)*  
*CorpID API Spec. (CLB-005: PDF Digital Signing Callback)*
- Step 25-27. For CORPID\_SIGN,  
 “iAM Smart” forwards verification to CorpID Core System; CorpID signs with the company RCA certificate and returns the CorpID signing result plus Authorization VP.

*CorpID API Spec. (CLB-004: Digital Signing Callback)*

*CorpID API Spec. (CLB-005: PDF Digital Signing Callback)*

Step 28-29. For CORPID\_SIGN, the e-Service process the signature with the original document, send acknowledgement to the CorpID Server and returns the CorpID signing result to the browser/app.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

Step 30-31. For PERSONAL\_SIGN, the e-Service process the signature with the original document, send acknowledgement to the CorpID Server and returns the CorpID signing result to the browser/app.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

Step 32-35. For INTEGRATED\_SIGN, the e-Service process the 1<sup>st</sup> signature with the original document, and prepares a second signing request with a new document hash. The e-Service submits the second signing request to CorpID Core System with access token, tokenised ID, Signing Token and other required parameters. CorpID signs the second request with the company RCA certificate and returns the CorpID signing result plus Authorization VP.

*CorpID API Spec. (EXT-016: Request Corporation Digital Signing)*

*CorpID API Spec. (EXT-017: Request Corporation PDF Digital Signing)*

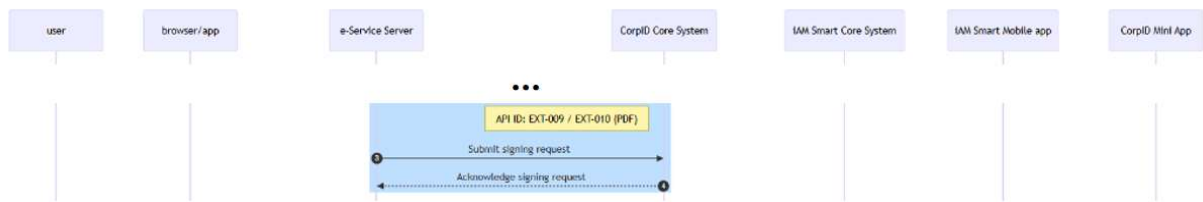
The e-Service Server send acknowledgement to the CorpID Server, combines both signing results and displays the integrated signing result.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No | API Name                                | API Reference (Section)               |
|----|-----------------------------------------|---------------------------------------|
| 1  | Request Digital Signing                 | CorpID API Specification Section 4.11 |
|    | Request PDF Digital Signing             | CorpID API Specification Section 4.13 |
| 2  | Digital Signing Callback                | CorpID API Specification Section 4.12 |
|    | PDF Digital Signing Callback            | CorpID API Specification Section 4.14 |
| 3  | Request Corporation Digital Signing     | CorpID API Specification Section 4.20 |
|    | Request Corporation PDF Digital Signing | CorpID API Specification Section 4.21 |
| 4  | Acknowledges Digital Signing Result     | CorpID API Specification Section 4.15 |

### 3.7.1.1 Implementing (1): EXT-009: Request Digital Signing



#### Pre-conditions

- e-Service must possess a valid cAccessToken of the CorpID user with authorisation scope “Signing authorisation”.
- e-Service must possess a valid accessToken of the “iAM Smart” user with authorisation scope “Signing authorisation”, and with openID encrypted as “iAMToken” by “iAM Smart” CEK.
- e-Service Website/App and “iAM Smart” Mobile App are in different devices in this scenario.
- e-Service should set the “signType” and determine whether the signing request is for:
  - INTEGRATED\_SIGN: For integrated signing (both personal and company chop)
  - CORPID\_SIGN: For signing with company chop only
  - PERSONAL\_SIGN: For signing via iAM Smart or remote signing service provider only
- If integrated signing is selected, CorpID will process the personal signing first, and then the e-Service can proceed with the CorpID signing signType
- Signing request will only be processed when the UBI number of the corporate is matched with the “ubi” provided by e-Service.
- If the iAMToken is not set, the HKICHash should not be passed.  
If the iAMToken is set, and HKIC Hash is not passed, the Signing request will not check the value.  
If the iAMToken is set and HKIC Hash is passed, signing request will only be processed when the hash of the HKIC number of the CorpID user is matched with the HKICHash provided by e-Service. For detail refer to CorpID API Specification.
- The CorpID user who signs the document with CorpID must have a “user\_companychop” in “userPrivilege” during authentication or switching corporation’s response.
- The CorpID user who signs the document with “iAM Smart” must have a sign version “iAM Smart” (i.e., “iAM Smart+”, with “iAM Smart” API Request accessToken & Tokenised ID response “userType” with value “sign”).
- e-Service generates a “businessID” for this Digital Signing request and uses this identifier to match the callback result returned from CorpID System.

- e-Service generates the Document Hash as “hashCode” parameter from the original documents to be signed using a hash algorithm that can suit the specific business need. It is recommended to use a hash algorithm that is at least SHA-256 or equivalent signType.
- “requireAuthProof” default is false. Set to true if the e-Service requires a cryptographic proof (Verifiable Presentation, VP) to verify the signer's identity and authority.
- API data encryption is required.

### Post-conditions

- e-Service server should check the response parameter “authByQR” and “ticketID” and determine the next action. For details, please refer to CorpID API Specification. “ticketID” will not be provided by CorpID System in this scenario.
- e-Service has to calculate a 4-digit identification code using the Document Hash and CorpID Tokenised ID (cOpenID) hash value. e-Service Website/App should show the 4-digit identification code and instruction to inform CorpID user to process the digital signing request in “iAM Smart” Mobile App when “authByQR” is “true”.
- 4-digit identification code generation algorithm refer to Appendix A.
- e-Service Website/App should keep synchronising with e-Service Server for the digital signing result returned from CorpID System.
- API data decryption is required.

### Error conditions

- For common errors, please refer to CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description                 | Suggested Action                                                                                                                                                                                                                                                     |
|---------------|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| code - M70002 | Failed to request digital signing | Inform user the CorpID or “iAM Smart” digital signing request is failed and retry later                                                                                                                                                                              |
| code - M70004 | User not allowed to sign          | Inform user that CorpID or “iAM Smart” digital signing is not allowed.<br>For CorpID, e-Service should suggest user to assign signing permission.<br>For default “iAM Smart”, e-Service should suggest user to upgrade to “iAM Smart” with digital signing function. |
| code - M70005 | Inconsistent HKIC number          | Hash of HKIC number provided is not matched with “iAM Smart” user. Check hash of HKIC number or inform user to authorise the digital signing using the correct “iAM Smart” account                                                                                   |

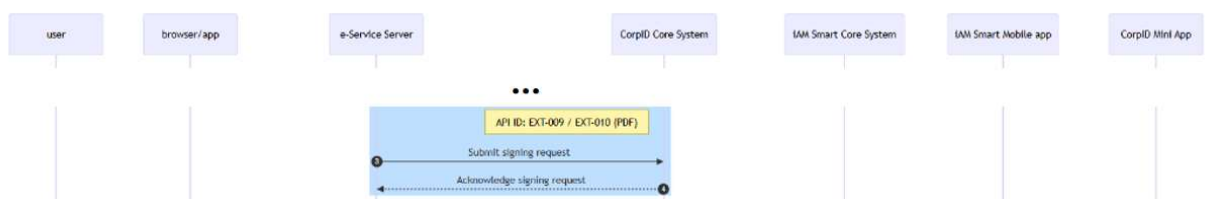
### Request and Response Parameters

- Please refer to CorpID API Specification.

## Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Request parameter “source”: Value should be matched with e-Service client terminal (e.g., “App\_Scheme”/“App\_Link” or browser’s user agent value).
- Request parameter “cOpenID”: It is the CorpID Tokenised ID (cOpenID) of the CorpID user. e-Service should ensure the cAccessToken and CorpID Tokenised ID (cOpenID) are in pair and belongs to the target CorpID user for the request. The target CorpID user must have a privilege “user\_companyhop”.
- Request parameter “openID”: It is the Tokenised ID of the “iAM Smart” user. e-Service should ensure the accessToken and Tokenised ID are in pair and belongs to the target “iAM Smart” user for the request in encrypted “iAMToken”. The target “iAM Smart” user must have a sign version “iAM Smart”.
- Request Parameter “redirectURI”: Value should equal to URI of “Digital Signing Callback” e-Service callback API. It must be in the same value as provided during e-Service registration.
- Request parameter “hashCode”: It is the Document Hash to be signed. SHA-256 or equivalent is recommended.
- Request parameter “sigAlgo”: It is the signature algorithm to be used by CorpID. The default value is "SHA256withRSA". While using "NONEwithRSA", hashCode provided must be hashed with SHA-256.
- Request parameter “HKICHash”: If provided, “iAM Smart” System will verify the HKIC hash of the “iAM Smart” user with the “HKICHash” provided by e-Service, and proceed the digital signing only when they matched. e-Service should convert the HKIC number into hash value using SHA-256 before sending to “iAM Smart” System. Only the identifier of HKIC number will be hashed, no check digit is needed.
- Request parameters “department”, “serviceName”, “documentName”: They are the information of the document to be signed and will be shown in CorpID Mini-program for CorpID user's reference with the 4-digit identification code.
- Response parameter “authByQR”: The value returned by CorpID and “iAM Smart” platform will be “true” in this scenario.
- CorpID Server will not return the response parameter “ticketID” in this scenario.

### 3.7.1.2 Implementing (1): EXT-010: Request PDF Digital Signing



## Pre-conditions

- Please refer to Section 3.7.1.1 except:
  - e-Service generates the digest as the parameter “hashCode” from the pdf document to be signed using the Adobe.PPKLite filter and the adbe.pkcs7.detached subfilter.

#### Post-conditions

- Please refer to Section 3.7.1.1 except:
  - Computing 4-digit identification code with “hashCode” instead of “hashCode”.

#### Error conditions

- Please refer to Section 3.7.1.1.

#### Request and Response Parameters

- Please refer to CorpID API Specification.

#### Notes

- Please refer to Section 3.7.1.1, except “hashCode” request parameter is applied instead of “hashCode”.

### 3.7.1.3 Implementing (2): CLB-004: Digital Signing Callback



#### Pre-conditions

- CorpID user can accept and complete the digital signing request.
- CorpID user can also reject the digital signing request.

#### Post-conditions

- API data decryption is required.
- e-Service Server should match the callback result with corresponding e-Service Website/App using the “businessID”.
- For digital signature and certificate,
  - For PERSONAL\_SIGN or INTEGRATED\_SIGN, the “signature” is passed in CEK encrypted “iAMContent”, encrypted by the CEK “iAMSecret” encrypted by “iAM Smart” KEK. The Certificate passed in “cert” is the “iAM Smart” user certificate. Refer to “iAM Smart API Specification” 6.5.4 Callback to Receive Digital Signing Result” callback parameters, except “businessID” and “state”.

- For CORPID\_SIGN, the signature is passed in “signature”, and the “cert” is the Corporation’s Cert in CorpID.
- For INTEGRATED\_SIGN, e-Service should store “signToken” returned for calling the 2nd API for Corp Signing.
- For PERSONAL\_SIGN or CORPID\_SIGN, e-Service server completes the document digital signing process and acknowledges CorpID System the digital signing result.
- If “requireAuthProof” was set in the Request Digital Signing Request, “authProof” will be returned as the signing request authorization proof in Verifiable Presentation (VP) format.

### Error conditions

- For common errors, please refer to CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description              | Suggested Action                                                                            |
|---------------|--------------------------------|---------------------------------------------------------------------------------------------|
| code - M70000 | User cancelled signing request | Inform user the CorpID digital signing request is cancelled                                 |
| code - M70001 | User rejected signing request  | Inform user the CorpID digital signing request is rejected                                  |
| code - M70002 | Failed to request signing      | Inform user the CorpID digital signing request is failed and retry later                    |
| code - M70003 | Signing request timeout        | Inform user the CorpID digital signing request is timeout and provide way for user to retry |

### Callback Parameters

- Please refer to CorpID API Specification.

### Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Callback parameter “hashCode”: It is the Document Hash provided by e-Service and e-Service may use for checking purpose.
- Callback parameter “timestamp”: It is the timestamp returned by CorpID System when the digital signing operation is successful.
- Callback parameter “signature” in decrypted “iAMContent”: For PERSONAL\_SIGN or INTEGRATED\_SIGN, it is the RSA signature of the Document Hash submitted by e-Service for digital signing signed by “iAM Smart” user certificate. The value is BASE64 encoded.

- Callback parameter “signature”: For CORPID\_SIGN only, it is the RSA signature of the Document Hash submitted by e-Service for digital signing signed by CorpID Corporation Certificate. The value is BASE64 encoded.
- Callback parameter “cert”: For CORPID\_SIGN only, It is the CorpID e-Cert of the Corporation. It is in X.509 format and the value is BASE64 encoded.
- Callback parameter “cert” in decrypted “iAMContent”: For PERSONAL\_SIGN or INTEGRATED\_SIGN, it is the “iAM Smart” e-Cert of the “iAM Smart” user. It is in X.509 format and the value is BASE64 encoded.
- Callback parameter “signToken”: For INTEGRATED\_SIGN, it is used in calling 2<sup>nd</sup> signing API for Corporate Signing.

#### 3.7.1.4 Implementing (2): CLB-005: PDF Digital Signing Callback



##### Pre-conditions

- Please refer to Section 3.7.1.3.

##### Post-conditions

- Please refer to Section 3.7.1.3.

##### Error conditions

- Please refer to Section 3.7.1.3.

##### Callback Parameters

- Please refer to CorpID API Specification.

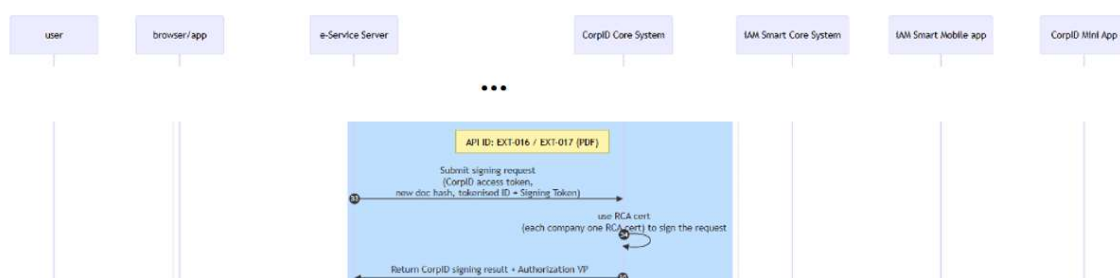
##### Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Please refer to Section 3.7.1.3, except the following for PDF signing.
- Callback parameter “hashCode”: For CORPID\_SIGN only, it is the pdf document digest submitted by e-Service and e-Service may use for checking purpose.
- Callback parameter “hashCode” in encrypted “iAMContent”: For PERSONAL\_SIGN or INTEGRATED\_SIGN, it is the pdf document digest submitted by e-Service and e-Service may use for checking purpose.
- Callback parameter “pdfSignature”: For CORPID\_SIGN only, it is the Base64-encoded PKCS#7 object that is the actual PDF signature value. It contains signer’s certificate as

Corporation, signed hash value, and the digital signing timestamp information. e-Service can embed this value to the PDF document for future verification. CorpID will provide this to e-Service only when the digital signing is successful.

- Callback parameter “pdfSignature” in encrypted “iAMContent”: For PERSONAL\_SIGN or INTEGRATED\_SIGN, it is the Base64-encoded PKCS#7 object that is the actual PDF signature value. It contains signer’s certificate as “iAM Smart” user, signed hash value, and the digital signing timestamp information. e-Service can embed this value to the PDF document for future verification. “iAM Smart” will provide this to e-Service only when the digital signing is successful.

### 3.7.1.5 Implementing (3): EXT-016: Request Corporation Digital Signing



#### Pre-conditions

- e-Service must possess a valid cAccessToken of the CorpID user with authorisation scope “Signing authorisation”.
- e-Service must set “signType” as “INTEGRATED\_SIGN” when initiate the 1st Request Document Signing for personal signing, and possess a valid “signToken” from personal signing callback.
- e-Service generates the Document Hash as “hashCode” parameter from the documents signed with 1st personal signature, and to be signed using a hash algorithm that can suit the specific business need. It is recommended to use a hash algorithm that is at least SHA-256 or equivalent.
- API data encryption is required.

#### Post-conditions

- API data decryption is required.
- The signature is passed in “signature”, and the “cert” is the Corporation’s Cert in CorpID.
- e-Service server completes the document digital signing process combining the 2nd Digital Signature of Corporation and acknowledges CorpID System the digital signing result.
- If “requireAuthProof” was set in the Request Digital Signing Request, “authProof” will be returned as the signing request authorization proof in Verifiable Presentation (VP) format.

## Error conditions

- For common errors, please refer to CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description                 | Suggested Action                                                                                                                                                                                                                                                     |
|---------------|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| code - M70002 | Failed to request digital signing | Inform user the CorpID or “iAM Smart” digital signing request is failed and retry later                                                                                                                                                                              |
| code - M70004 | User not allowed to sign          | Inform user that CorpID or “iAM Smart” digital signing is not allowed.<br>For CorpID, e-Service should suggest user to assign signing permission.<br>For default “iAM Smart”, e-Service should suggest user to upgrade to “iAM Smart” with digital signing function. |

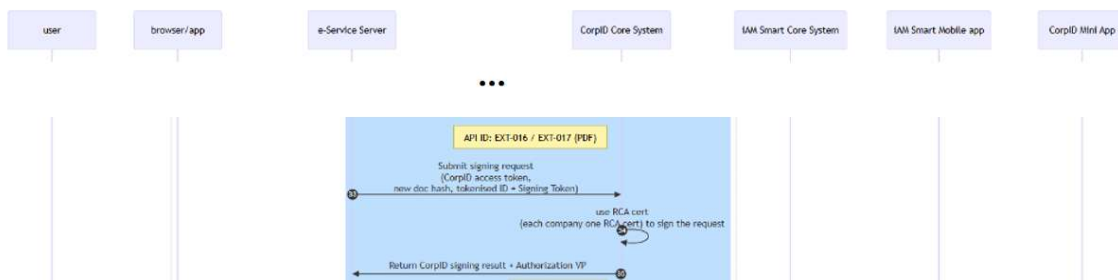
## Request and Response Parameters

- Please refer to CorpID API Specification.

## Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Request parameter “signToken”: Value should be used from the callback for 1st Digital Signing.
- Request parameter “cOpenID”: It is the CorpID Tokenised ID (cOpenID) of the CorpID user. e-Service should ensure the cAccessToken and CorpID Tokenised ID (cOpenID) are in pair and belongs to the target CorpID user for the request. The target CorpID user must have a privilege “user\_companychop”.
- Request parameter “hashCode”: It is the Document Hash to be signed. e-Service should ensure whether the 1st signature signed document is used to generate this hashCode. SHA-256 or equivalent is recommended.
- Request parameter “sigAlgo”: It is the signature algorithm to be used by CorpID. The default value is "SHA256withRSA". While using "NONEwithRSA", hashCode provided must be hashed with SHA-256.

### 3.7.1.6 Implementing (3): EXT-017: Request Corporation PDF Digital Signing



#### Pre-conditions

- Please refer to Section 3.7.1.5.

#### Post-conditions

- Please refer to Section 3.7.1.5.

#### Error conditions

- Please refer to Section 3.7.1.5.

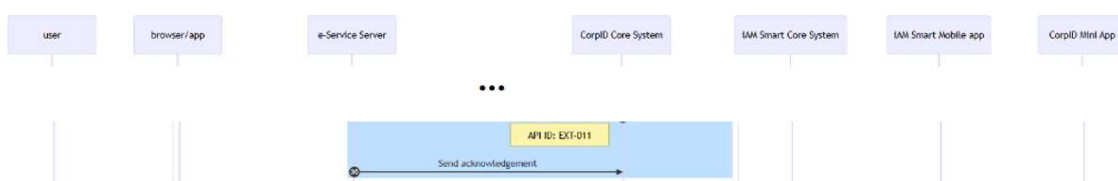
#### Callback Parameters

- Please refer to CorpID API Specification.

#### Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Please refer to Section 3.7.1.5, except the following for PDF signing.
- Callback parameter “hashCode”: It is the pdf document digest submitted by e-Service and e-Service may use for checking purpose.
- Callback parameter “pdfSignature”: It is the Base64-encoded PKCS#7 object that is the actual PDF signature value. It contains signer’s certificate as Corporation, signed hash value, and the digital signing timestamp information. e-Service can embed this value to the PDF document for future verification. CorpID will provide this to e-Service only when the digital signing is successful.

### 3.7.1.7 Implementing (4): EXT-011: Acknowledges Digital Signing Result



#### Pre-conditions

- e-Service receives the CorpID digital signing result and completes its document digital signing process (i.e., verify the result).
- API data encryption is required.

#### **Post-conditions**

- e-Service may record the digital signing acknowledgement result.

#### **Error conditions**

- For common errors, please refer to CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description                         | Suggested Action                                                      |
|---------------|-------------------------------------------|-----------------------------------------------------------------------|
| code - M70006 | Failed to process signing acknowledgement | CorpID digital signing acknowledgement processing failed, retry later |

#### **Request and Response Parameters**

- Please refer to CorpID API Specification.

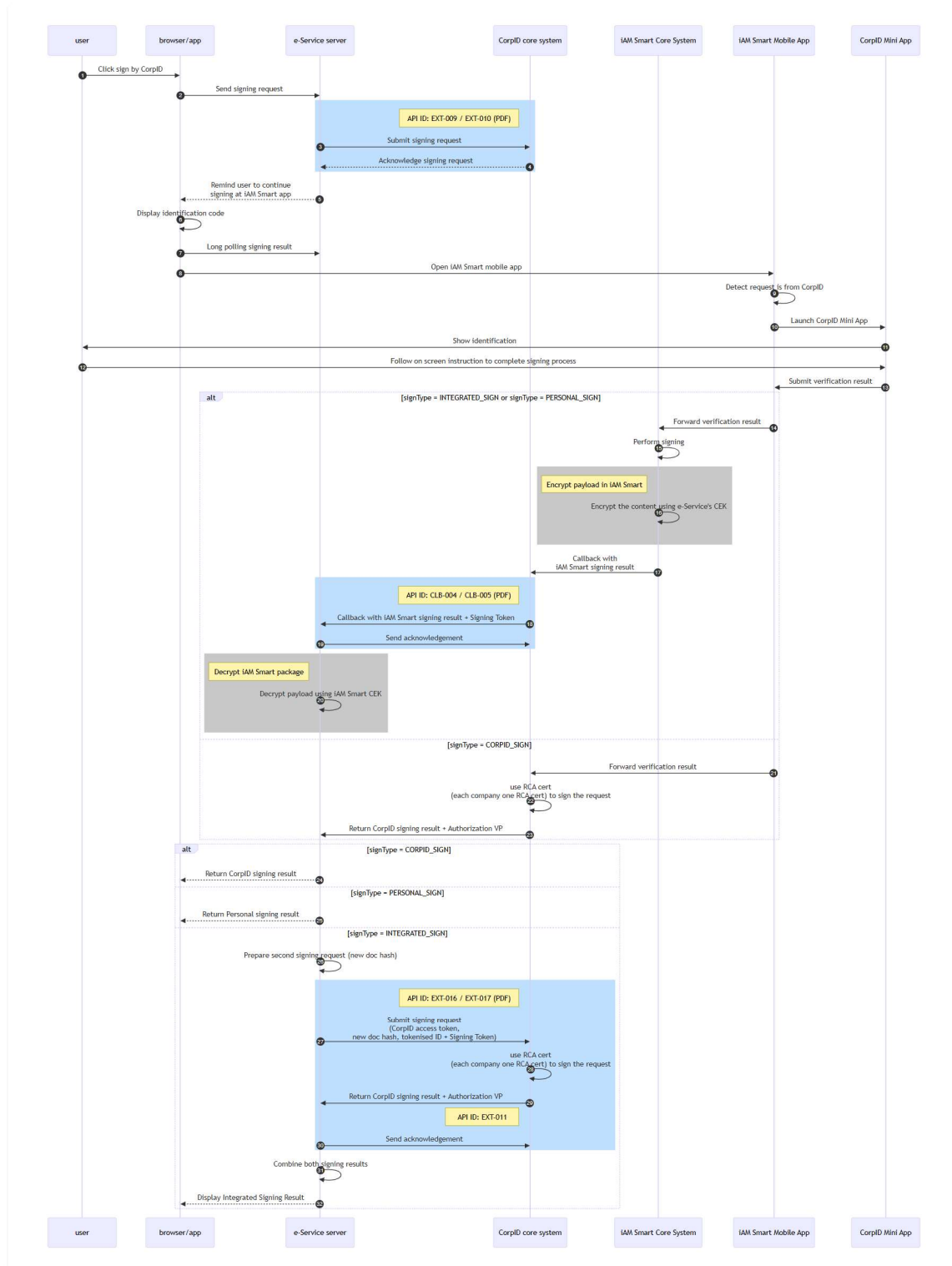
#### **Notes**

- For common parameters in request and response, please refer to CorpID API Specification.
- Request parameter “businessID”: It is the same “businessID” submitted in the digital signing request.
- Request parameter “signingResult”: It is the status of e-Service document digital signing process after receiving the CorpID digital signing result. Its value must be equal to those specified in this API.

### 3.7.2 Scenario 2: Digital Signing (e-Service Website in Same Device)

The sequence diagram below shows how an authenticated user authorises and signs the Document Hash when e-Service website and the “iAM Smart” Mobile App are running in the same device.

CorpID Digital Signing (e-Service Website/App in Same Device)



- Step 1. User clicks Sign by CorpID and the browser sends a signing request to the e-Service Server.
- Step 2-5. The e-Service Server submits the signing request, with “Request Digital Signing” API for binary and general content or “Request PDF Digital Signing” for PDF, with “businessID”, “cOpenID”, “cAccessToken”, “iAMToken”, “state”, “source”, “hashCode”, “signType” and other required parameters, to CorpID Core System, with response with “authByQR” (set to “false”) to indicate same device.
- If “signType” is “INTEGRATED\_SIGN” or “PERSONAL\_SIGN”, the request trigger the personal signing (“iAM Smart” for HKIC Holders, signing partner for passport holders). If “signType” is “CORPID\_SIGN”, the request trigger the CorpID-only signing.
- If “HKICHash” is passed, the 4-digit identification code will be checked.
- CorpID Core System acknowledges the signing request and the e-Service reminds the user to continue in iAM Smart.
- CorpID API Spec. (EXT-009: Request Digital Signing)*  
*CorpID API Spec. (EXT-010: Request PDF Digital Signing)*
- Step 6-7. The browser/app displays an identification code and starts long polling for the signing result.
- Step 8 e-Service website invokes CorpID Mini-program using URL Scheme with “ticketID” (set <Context> as “corpid” in URL Scheme and pass required parameters). The e-Service Website should keep synchronising with the e-Service Server for the request result (e.g. polling).
- iAM Smart API for CorpID (URL Scheme: Open iAM Smart Mobile App for Digital Signing)*
- Step 9-13. The user launch the “iAM Smart” Mobile App, and “iAM Smart” detects that the request is from CorpID. “iAM Smart” Mobile App launches CorpID Mini-program. CorpID Mini-program shows the consent page, identification code and the user follows on-screen instructions to complete signing. CorpID Mini-program submits the verification result to “iAM Smart” Mobile App.
- Step 14-20. For INTEGRATED\_SIGN or PERSONAL\_SIGN, “iAM Smart” forwards verification, performs signing, encrypts the payload with the e-Service CEK, and callbacks with the iAM Smart signing result. CorpID Core System callbacks to the e-Service Server with the “iAM Smart” signing result and Signing Token. The API is encrypted with CorpID CEK.
- The e-Service Server decrypts the iAM Smart package using the iAM Smart CEK.
- CorpID API Spec. (CLB-004: Digital Signing Callback)*  
*CorpID API Spec. (CLB-005: PDF Digital Signing Callback)*

Step 21-23. For CORPID\_SIGN,  
 “iAM Smart” forwards verification to CorpID Core System; CorpID signs with the company RCA certificate and returns the CorpID signing result plus Authorization VP, and trigger the launch of external browser.

*CorpID API Spec. (CLB-004: Digital Signing Callback)*

*CorpID API Spec. (CLB-005: PDF Digital Signing Callback)*

Step 24. For CORPID\_SIGN, the e-Service process the signature with the original document, send acknowledgement to the CorpID Server, and trigger the launch of external browser to show the signing result.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

Step 25. For PERSONAL\_SIGN, the e-Service process the signature with the original document, send acknowledgement to the CorpID Server, and trigger the launch of external browser to show the signing result.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

Step 26-32. For INTEGRATED\_SIGN, the e-Service process the 1<sup>st</sup> signature with the original document, and prepares a second signing request with a new document hash. The e-Service submits the second signing request to CorpID Core System with access token, tokenised ID, Signing Token and other required parameters. CorpID signs the second request with the company RCA certificate and returns the CorpID signing result plus Authorization VP.

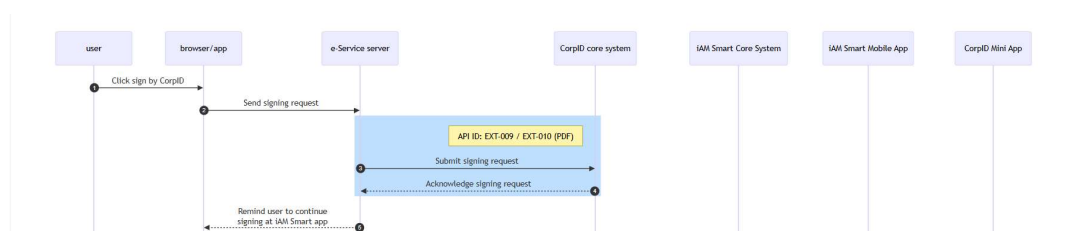
*CorpID API Spec. (EXT-016: Request Corporation Digital Signing)*

*CorpID API Spec. (EXT-017: Request Corporation PDF Digital Signing)*

The e-Service Server send acknowledgement to the CorpID Server, combines both signing results and displays the integrated signing result.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

### 3.7.2.1 Implementing (1) POST: Request Digital Signing



#### Pre-conditions

- Please refer to Section 3.7.1.1 except:
  - e-Service Website and “iAM Smart” Mobile App are in the same device in this scenario.

### Post-conditions

- Please refer to Section 3.7.1.1 except:
  - Response “authByQR” is “false”, “ticketID” will be provided by “iAM Smart” System in this scenario.

### Error conditions

- Please refer to Section 3.7.1.1.

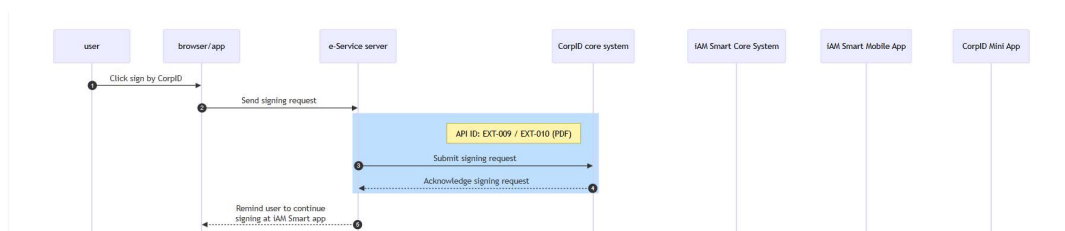
### Request and Response Parameters

- Please refer to “iAM Smart” API Specification.

### Notes

- Please refer to Section 3.7.1.1, except for the following parameters:
  - Request parameter “source”: Value should be the browser's user agent value.
  - Response parameter “authByQR”: The value is “false” in this scenario.
  - Response parameter “ticketID”: It will be provided by “iAM Smart” System in this scenario. It is a 36-byte (or less) UUID number (ASCII character set).

## 3.7.2.2 Implementing (1) POST: Request PDF Digital Signing



### Pre-conditions

- Please refer to Section 3.7.1.2 except:
  - e-Service generates the digest as the parameter “hashCode” from the pdf document to be signed using the Adobe.PPKLite filter and the adbe.pkcs7.detached subfilter.

### Post-conditions

- Please refer to Section 3.7.1.2 except:
  - Computing 4-digit identification code with “docDigest” instead of “hashCode”.

### Error conditions

- Please refer to Section 3.7.1.2.

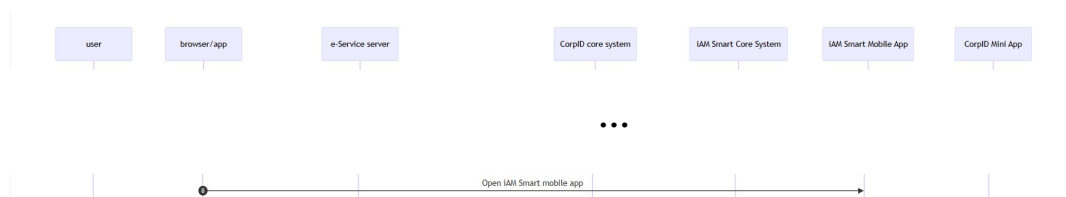
### Request and Response Parameters

- Please refer to “iAM Smart” API Specification.

### Notes

- Please refer to Section 3.7.2.1, except “hashCode” request parameter is applied instead of “hashCode”.

### 3.7.2.3 Implementing (2) URL Scheme: Open iAM Smart Mobile App for CorpID Digital Signing



### Pre-conditions

- e-Service Website and “iAM Smart” Mobile App are in the same device.

### Post-conditions

- “iAM Smart” Mobile App will be launched.
- e-Service Website should keep synchronising with e-Service server for the callback response of the digital signing request from “iAM Smart” System.

### Error conditions

- Nil

### Request Parameters

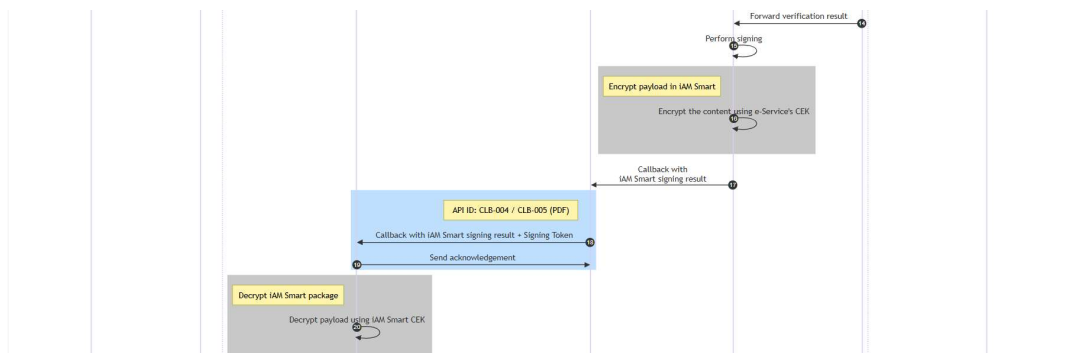
- Please refer to “iAM Smart” for CorpID API Specification.

### Notes

- Set <Context> as “corpId?params=corpapi\_auth%20corpapi\_sign” in URL Scheme.

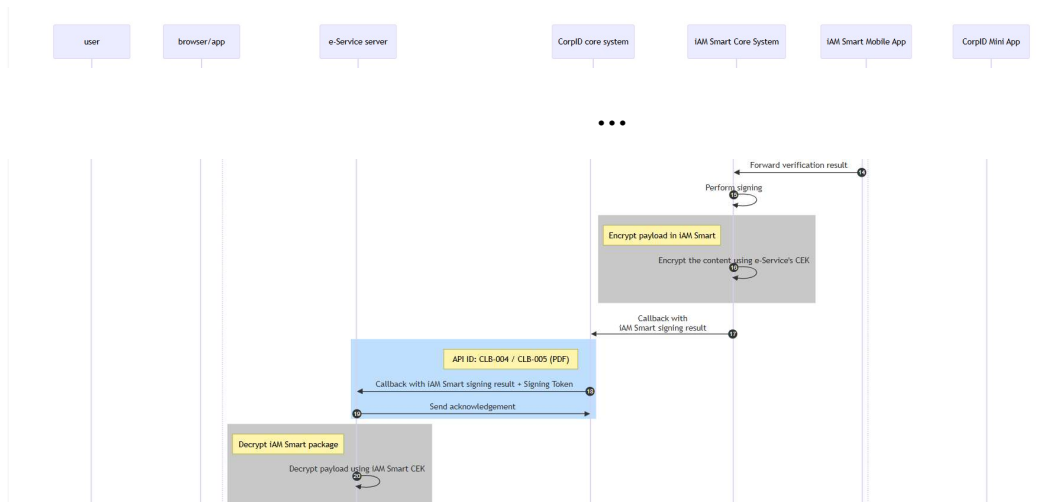
### 3.7.2.4 Implementing (3) POST: Callback to Receive Digital Signing Result





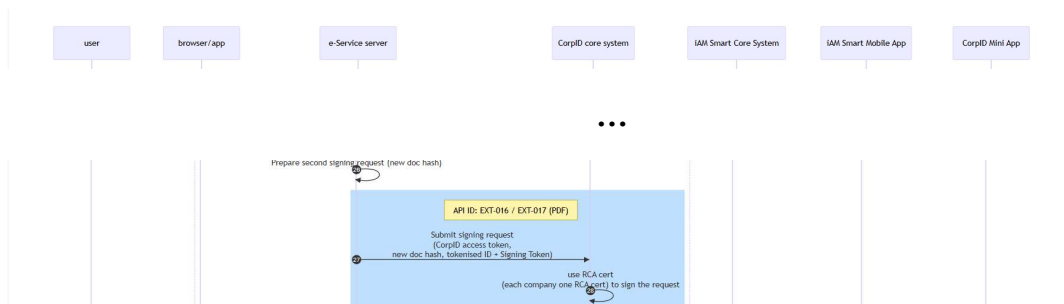
Please refer to Section 3.7.1.3.

### 3.7.2.5 Implementing (3) POST: Callback to Receive PDF Digital Signing Result



Please refer to Section 3.7.1.4.

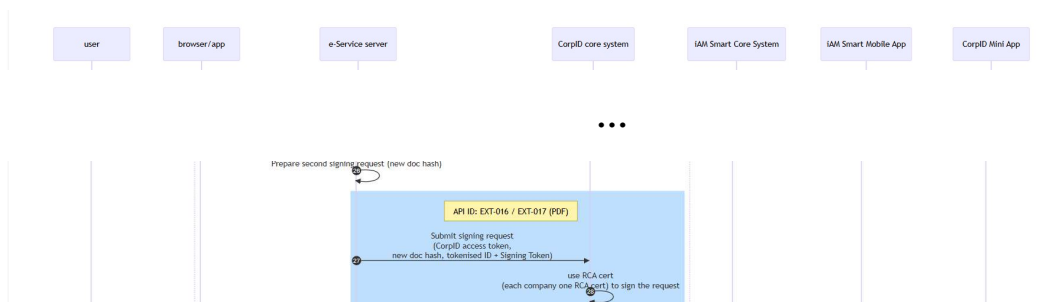
### 3.7.2.6 Implementing (4): EXT-016: Request Corporation Digital Signing



For Integrated Signing, this API is required.

Please refer to Section 3.7.1.6.

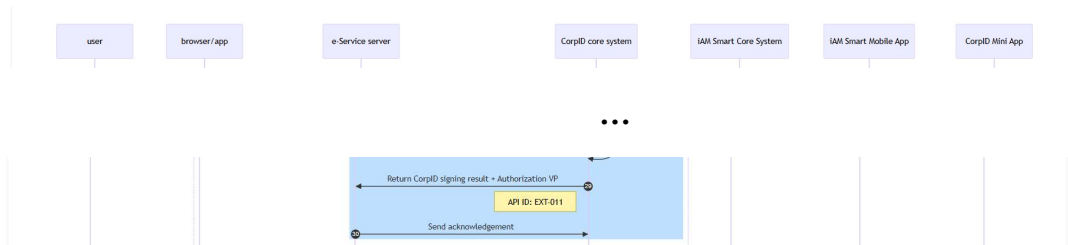
### 3.7.2.7 Implementing (4): EXT-017: Request Corporation PDF Digital Signing



For Integrated Signing, this API is required.

Please refer to Section 3.7.1.6.

### 3.7.2.8 Implementing (5) POST: e-Service Acknowledges Digital Signing Result

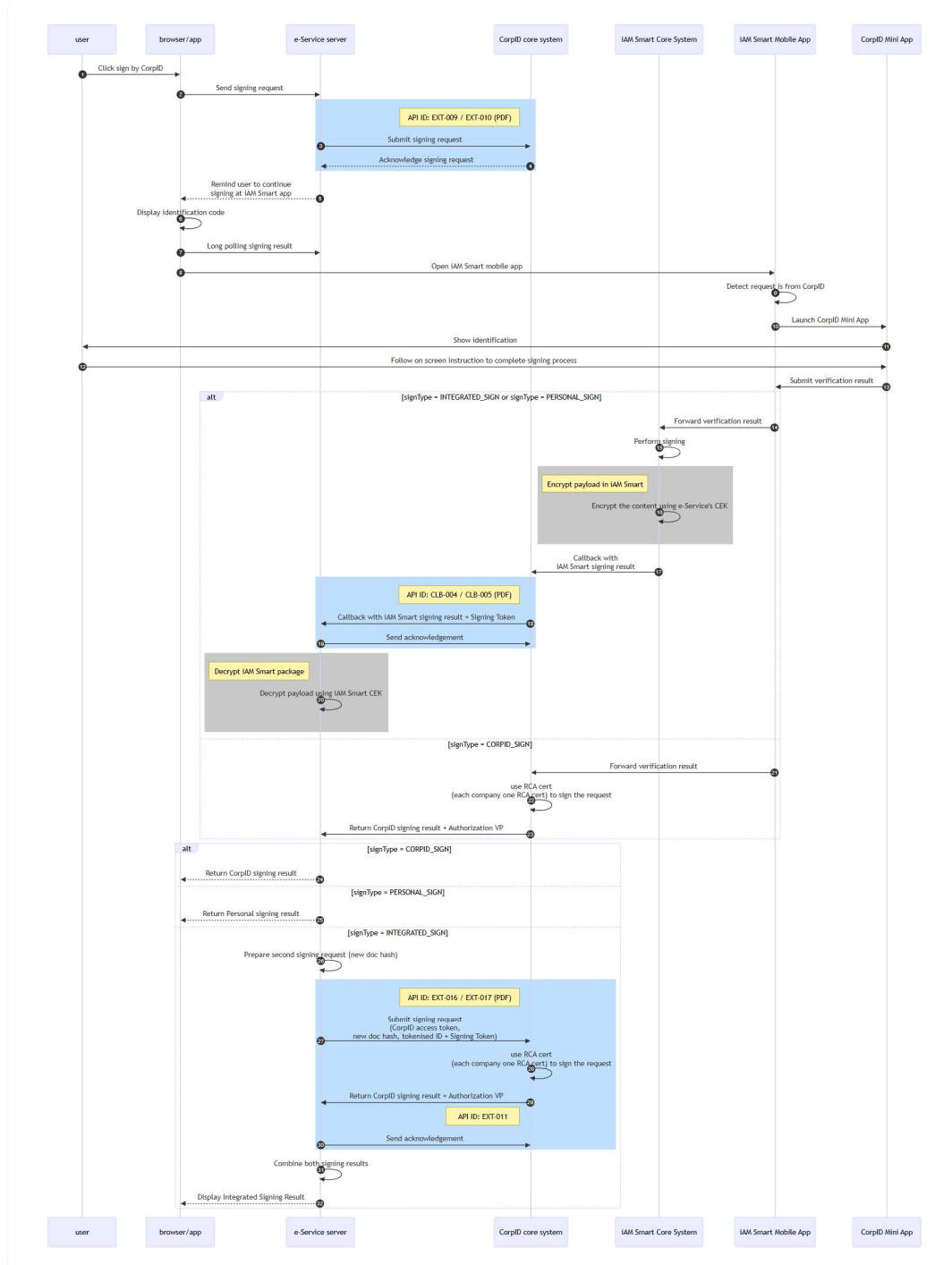


Please refer to Section 3.7.1.7.

### 3.7.3 Scenario 3: Digital Signing (e-Service App in Same Device)

The sequence diagram below shows how an authenticated user authorises and signs the Document Hash when e-Service App and the “iAM Smart” Mobile App are running in the same device.

CorpID Digital Signing (e-Service Website/App in Same Device)



Step 1. User clicks Sign by CorpID and the e-Service App sends a signing request to the e-Service Server.

Step 2-5. The e-Service Server submits the signing request, with “Request Digital Signing” API for binary and general content or “Request PDF Digital Signing” for PDF, with “businessID”, “cOpenID”, “cAccessToken”, “iAMToken”, “state”, “source”, “hashCode”, “signType” and other required parameters, to CorpID Core System, with response with “authByQR” (set to “false”) to indicate same device.

If “signType” is “INTEGRATED\_SIGN” or “PERSONAL\_SIGN”, the request trigger the personal signing (“iAM Smart” for HKIC Holders, signing partner for passport holders). If “signType” is “CORPID\_SIGN”, the request trigger the CorpID-only signing.

If “HKICHash” is passed, the 4-digit identification code will be checked.

CorpID Core System acknowledges the signing request and the e-Service reminds the user to continue in iAM Smart.

*CorpID API Spec. (EXT-020: Request Digital Signing (for App e-Service))*

*CorpID API Spec. (EXT-021: Request PDF Digital Signing (for App e-Service))*

Step 6-7. The app displays an identification code and starts long polling for the signing result.

Step 8 e-Service App invokes CorpID Mini-program using URL Scheme with “ticketID” (set <Context> as “corpid” in URL Scheme and pass required parameters). The e-Service App should keep synchronising with the e-Service Server for the request result (e.g. polling).

*iAM Smart API for CorpID (URL Scheme: Open iAM Smart Mobile App for Digital Signing)*

Step 9-13. The user launch the “iAM Smart” Mobile App, and “iAM Smart” detects that the request is from CorpID. “iAM Smart” Mobile App launches CorpID Mini-program. CorpID Mini-program shows the consent page, identification code and the user follows on-screen instructions to complete signing. CorpID Mini-program submits the verification result to “iAM Smart” Mobile App.

Step 14-20. For INTEGRATED\_SIGN or PERSONAL\_SIGN, “iAM Smart” forwards verification, performs signing, encrypts the payload with the e-Service CEK, and callbacks with the iAM Smart signing result. CorpID Core System callbacks to the e-Service Server with the “iAM Smart” signing result and Signing Token. The API is encrypted with CorpID CEK. The e-Service Server decrypts the iAM Smart package using the iAM Smart CEK.

*CorpID API Spec. (CLB-004: Digital Signing Callback)*

*CorpID API Spec. (CLB-005: PDF Digital Signing Callback)*

Step 21-23. For CORPID\_SIGN, “iAM Smart” forwards verification to CorpID Core System; CorpID signs with the company RCA certificate and returns the CorpID signing result plus Authorization VP, and trigger the launch of e-Service App.

*CorpID API Spec. (CLB-004: Digital Signing Callback)*

*CorpID API Spec. (CLB-005: PDF Digital Signing Callback)*

- Step 24. For CORPID\_SIGN, the e-Service process the signature with the original document, send acknowledgement to the CorpID Server, and trigger the launch of e-Service App to show the signing result.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

- Step 25. For PERSONAL\_SIGN, the e-Service process the signature with the original document, send acknowledgement to the CorpID Server, and trigger the launch of e-Service App to show the signing result.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

- Step 26-32. For INTEGRATED\_SIGN, the e-Service process the 1<sup>st</sup> signature with the original document, and prepares a second signing request with a new document hash. The e-Service submits the second signing request to CorpID Core System with access token, tokenised ID, Signing Token and other required parameters. CorpID signs the second request with the company RCA certificate and returns the CorpID signing result plus Authorization VP.

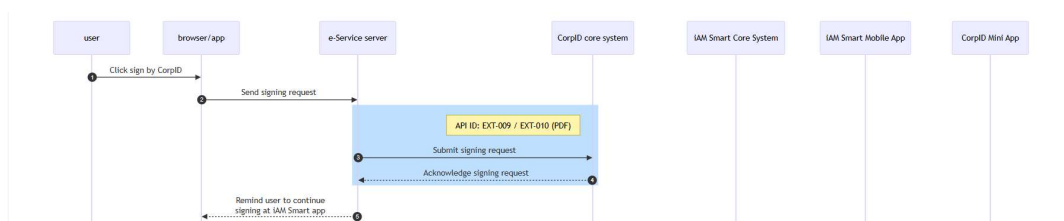
*CorpID API Spec. (EXT-016: Request Corporation Digital Signing)*

*CorpID API Spec. (EXT-017: Request Corporation PDF Digital Signing)*

The e-Service Server send acknowledgement to the CorpID Server, combines both signing results and trigger the launch of e-Service App to displays the integrated signing result.

*CorpID API Spec. (EXT-011: Acknowledges Digital Signing Result)*

### 3.7.3.1 Implementing (1) POST: Request Digital Signing (for App e-Service)



#### Pre-conditions

- Please refer to Section 3.7.2.1 except:
  - e-Service App and “iAM Smart” Mobile App are in the same device in this scenario.

#### Post-conditions

- Please refer to Section 3.7.2.1.

### Error conditions

- Please refer to Section 3.7.2.1.

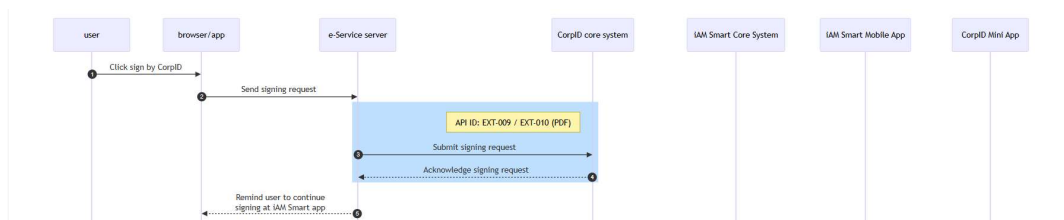
### Request and Response Parameters

- Please refer to “iAM Smart” API Specification.

### Notes

- Please refer to Section 3.7.2.1, except for the following parameter:
  - Request parameter “source”: Value should be “App\_Scheme” (for the e-Service App URL scheme), or “App\_Link” (for the Universal Link/App Link).

### 3.7.3.2 Implementing (1) POST: Request PDF Digital Signing (for App e-Service)



### Pre-conditions

- Please refer to Section 3.7.2.2 except:
  - e-Service generates the digest as the parameter “hashCode” from the pdf document to be signed using the Adobe.PPKLite filter and the adbe.pkcs7.detached subfilter.

### Post-conditions

- Please refer to Section 3.7.2.2 except:
  - Computing 4-digit identification code with “hashCode” instead of “hashCode”.

### Error conditions

- Please refer to Section 3.7.2.2.

### Request and Response Parameters

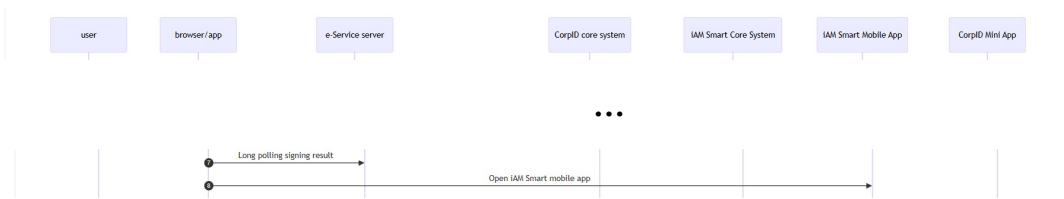
- Please refer to “iAM Smart” API Specification.

### Notes

- Please refer to Section 3.7.2.2, except “hashCode” request parameter is applied instead of “hashCode”.

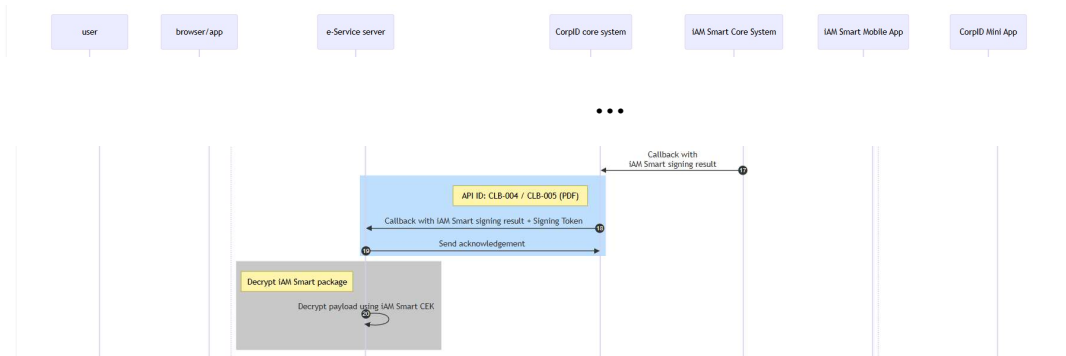
### 3.7.3.3 Implementing (2) URL Scheme: Open iAM Smart Mobile App for CorpID Digital

## Signing



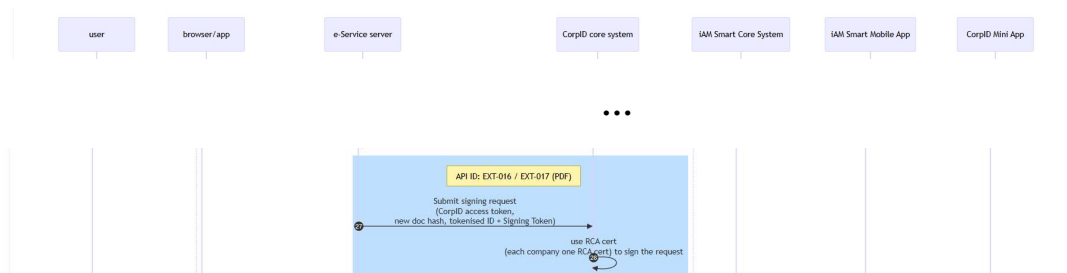
Please refer to Section 3.7.2.3.

### 3.7.3.4 Implementing (3) POST: Callback to Receive Digital Signing Result



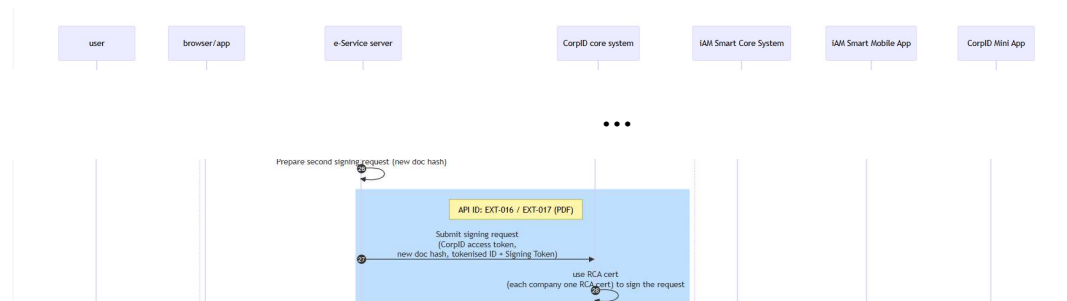
Please refer to Section 3.7.1.3.

### 3.7.3.5 Implementing (3) POST: Callback to Receive PDF Digital Signing Result



Please refer to Section 3.7.1.4.

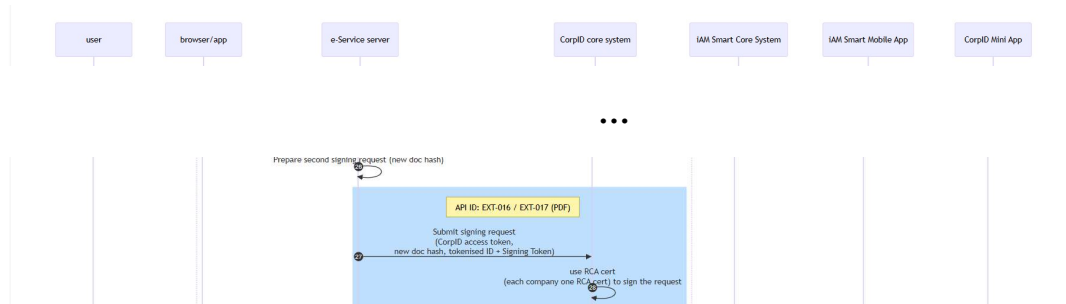
### 3.7.3.6 Implementing (4): EXT-016: Request Corporation Digital Signing



For Integrated Signing, this API is required.

Please refer to Section 3.7.1.6.

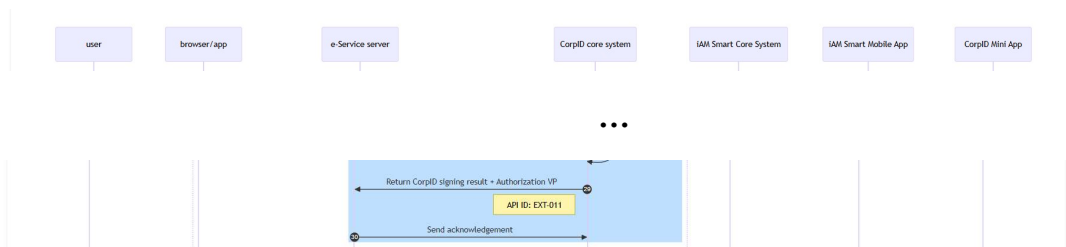
### 3.7.3.7 Implementing (4): EXT-017: Request Corporation PDF Digital Signing



For Integrated Signing, this API is required.

Please refer to Section 3.7.1.6.

### 3.7.3.8 Implementing (5) POST: e-Service Acknowledges Digital Signing Result



Please refer to Section 3.7.1.7.

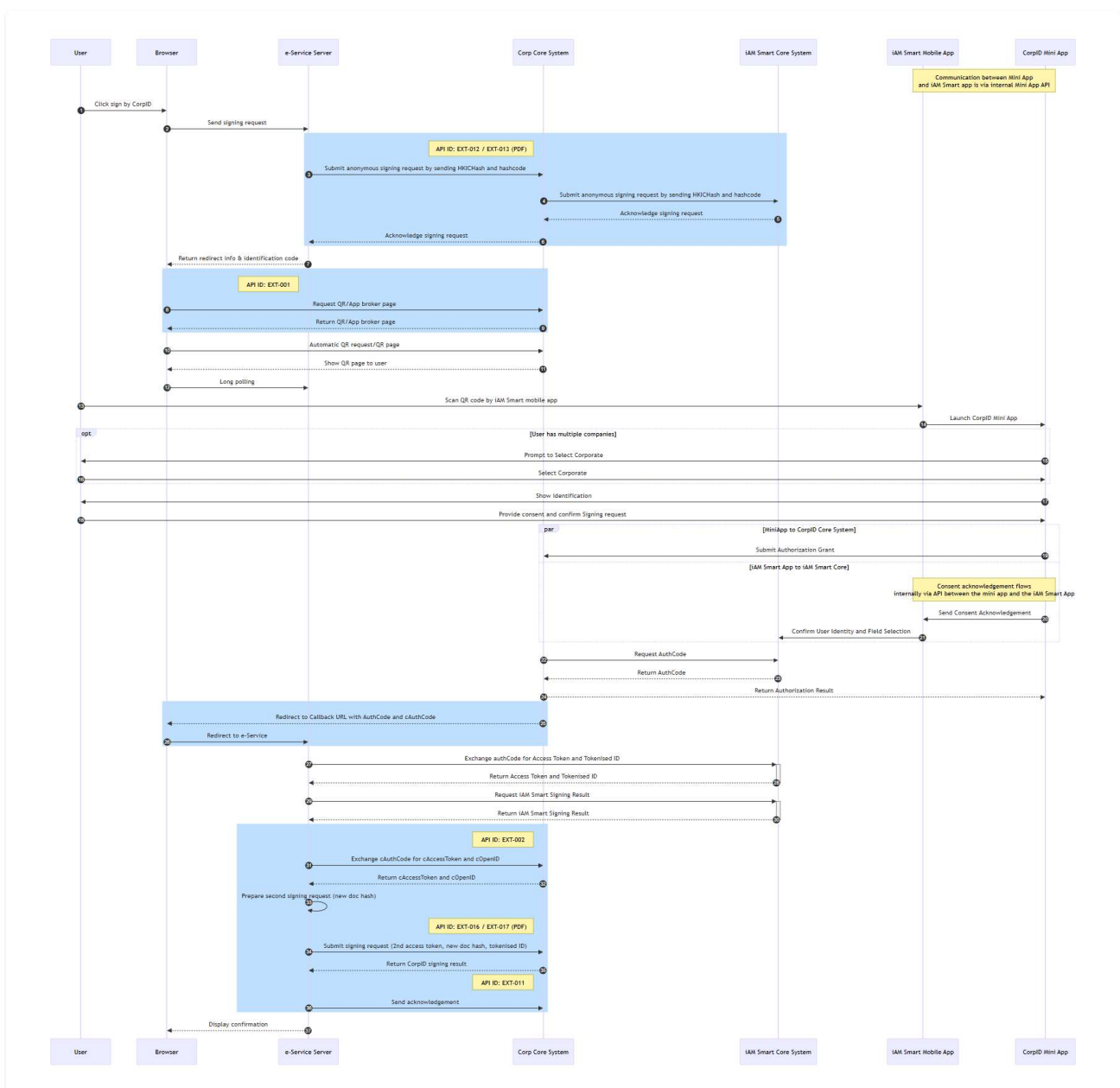
### 3.8 WORKFLOWS FOR DIGITAL SIGNING WITHOUT SERVICE LOGIN (ANONYMOUS DIGITAL SIGNING)

The following process includes anonymous hash digital signing and anonymous pdf digital signing.

#### 3.8.1 Scenario 1: Anonymous Digital Signing (e-Service Website in Different Device)

The sequence diagram below shows how an anonymous user authorises and signs the Document Hash when e-Service website and the “iAM Smart” Mobile App are running in different devices.

##### CorpID Anonymous Digital Signing (e-Service Website in Different Device)



- Step 1-2. The user clicks “Sign by CorpID” in the browser to start digital signing. The browser initiates an anonymous digital signing request to the e-Service Server.
- Step 3. The e-Service Server initiates an anonymous digital signing request to invoke the CorpID API with the “businessID” of this request, Document Hash / PDF digest, HKICHash, signature algorithm (only for “Request Anonymous Digital Signing”), etc. API encryption is required;  
*CorpID API (EXT-012: Request Anonymous Digital Signing)*  
*CorpID API (EXT-013 (PDF): Request Anonymous PDF Digital Signing)*
- Step 4. The CorpID Server verifies and confirm receipt of the anonymous digital signing request to the e-Service server by returning response with parameter “ticketID”.
- Step 5-7. e-Service server uses the Document Hash / PDF Hash, HKICHash and hash of clientID to calculate a 4-digit identification code and instructs e-Service website to display the instructions with a 4-digit identification code to inform CorpID user to process the digital signing authorisation request in CorpID Mini-program.  
e-Service server prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as browser’s user agent value), “scope”, “ticketID”, etc. and constructs the GET request to invoke the CorpID API using browser redirection;  
*CorpID API (EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request)*
- Step 8-11. CorpID System returns the broker page (if e-Service has set “brokerPage” to “true”) and after the broker page fails to find the “iAM Smart” Mobile App, it will request QR page to be displayed in browser. If e-Service does not request for broker page (i.e. no broker page will be returned), the browser will directly display QR Code page.
- Step 12. QR Code page of CorpID and “iAM Smart” System polls “iAM Smart” System for the QR Code status;
- Step 13-14. CorpID user logs in “iAM Smart” Mobile App to scan the QR code, and launch the CorpID Mini-program;
- Step 15-16. If CorpID user has multiple corporations, he / she will be prompted to select a corporate account.
- Step 17-18. CorpID Mini-program requests and displays 4-digit identification code from CorpID System. CorpID user reviews the digital signing authorisation request (e.g. continue or reject the request) and verifies the 4-digit identification code with e-Service Website;
- Step 19-24. CorpID System and “iAM Smart” System verifies the validity of QR code and other necessary information, and return the authorisation result to CorpID Mini-program;
- Step 25-26. QR Code page of CorpID System receives the polling result returned by CorpID and “iAM Smart” System (i.e. response for the polling in step 12) which includes cAuthCode “cCode” of CorpID System, authCode “code” of “iAM Smart System” and businessID or any error code (e.g. CorpID user rejects the request), assembles and

invokes the e-Service callback API given in “redirectURI” using browser redirection. the flow redirects to the callback URL with the AuthCode.

*CorpID API (CBL-003, Callback with AuthCode for Anonymous)*

Step 27-28. When e-Service server gets the cAuthCode, authCode, and businessID, it verifies businessID as generated in Step 3. e-Service then invoke “iAM Smart” API to obtain the “iAM Smart” accessToken and the Tokenised ID (i.e., openID). API data encryption is required;

*“iAM Smart” API (POST: Request accessToken & Tokenised ID)*

Step 29-30. e-Service Server invokes the “iAM Smart” API with the accessToken and openID to obtain the result of anonymous digital signing. API data encryption is required;

*CorpID API (POST: Get Anonymous Digital Signing Result (Personal Signing))*

*CorpID API (POST: Get Anonymous PDF Digital Signing Result (Personal Signing))*

Step 31-32. e-Service then invoke CorpID API to obtain the CorpID cAccessToken and the Tokenised ID (cOpenID). API data encryption is required;

*CorpID API (EXT-002: Get Access Token)*

Step 33-35. e-Service Server use the 1<sup>st</sup> signing result and combine with raw document to create a “iAM Smart”-only signed document. A new document hash is generated and call the CorpID API with required parameters to obtain the result of CorpID signing. Api data encryption is required.

Under API ID: EXT-002, a second signing request is sent with a new document hash.

*CorpID API (EXT-016: Request Corporation Digital Signing (For Integrated Signing))*

*CorpID API (EXT-017: Request Corporation PDF Digital Signing (For Integrated Signing))*

Step 36. e-Service Server completes the document digital signing process, verify the result and acknowledges CorpID System the digital signing result using CorpID API with the same “businessID” of the digital signing request. API data encryption is required;

*CorpID API (EXT-011: Acknowledges Digital Signing result)*

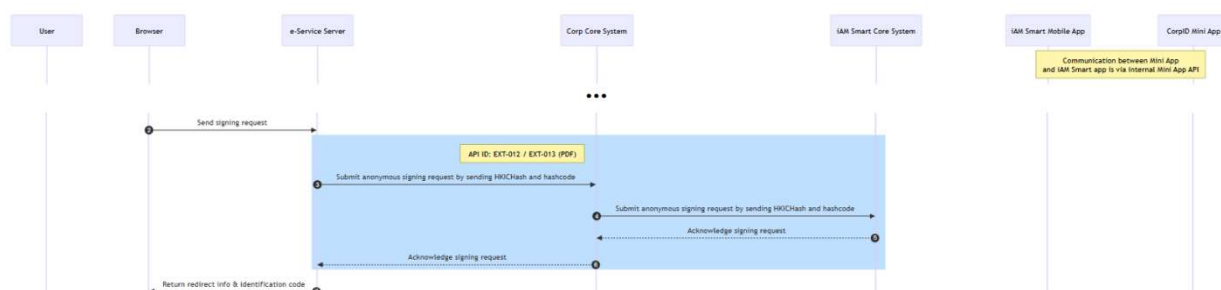
Step 37. The browser displays the digital signing result to the user.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No | API Name                                          | API Reference (Section)               |
|----|---------------------------------------------------|---------------------------------------|
| 1  | Request Anonymous Digital Signing                 | CorpID API Specification Section 4.16 |
|    | Request Anonymous PDF Digital Signing             | CorpID API Specification Section 4.17 |
| 2  | Authentication by requesting QR Page/ Broker Page | CorpID API Specification Section 4.1  |

|   |                                            |                                       |
|---|--------------------------------------------|---------------------------------------|
| 3 | Callback with authCode to e-Service Server | CorpID API Specification Section 4.9  |
| 4 | Get Access Token                           | CorpID API Specification Section 4.2  |
| 5 | Request Corporation Digital Signing        | CorpID API Specification Section 4.20 |
|   | Request Corporation PDF Digital Signing    | CorpID API Specification Section 4.21 |
| 6 | Digital Signing Callback                   | CorpID API Specification Section 4.12 |
|   | PDF Digital Signing Callback               | CorpID API Specification Section 4.14 |
| 7 | Acknowledges Digital Signing Result        | CorpID API Specification Section 4.15 |

### 3.8.1.1 Implementing (1): EXT-012: Request Anonymous Digital Signing



#### Pre-conditions

- Please refer to Section 3.7.1.1 except:
  - No request parameter “accessToken” exists.
  - e-Service generates a “businessID” for this request and uses this identifier to match the authCode and cAuthCode returned from CorpID and “iAM Smart” System.
  - e-Service should convert the HKIC number (no check digit is needed) into hash value using SHA256.

#### Post-conditions

- Please refer to Section 3.7.1.1 except:
  - No response parameter “authByQR” exists.
  - e-Service calculates a 4-digit identification code using: Document Hash + HKICHash + clientID hash value (calculated with SHA-512)

#### Error conditions

- For common errors, please refer to CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description         | Suggested Action                                                                         |
|---------------|---------------------------|------------------------------------------------------------------------------------------|
| code - M70002 | Failed to request signing | Inform user the CorpID and “iAM Smart” digital signing request is failed and retry later |

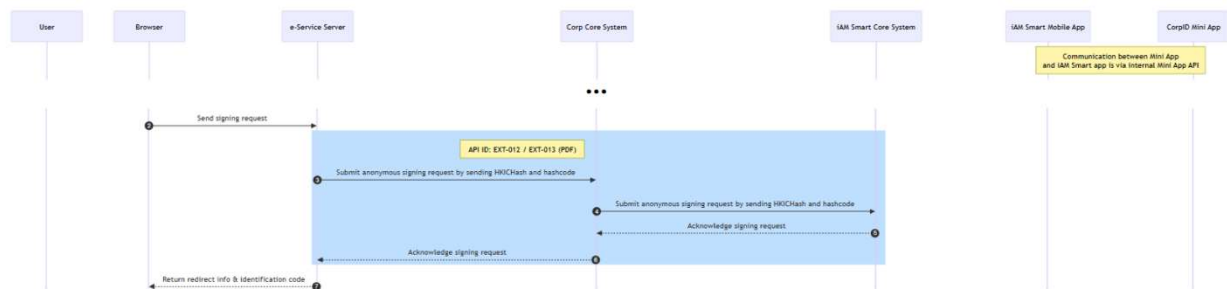
## Request and Response Parameters

- Please refer to CorpID API Specification.

## Notes

- Please refer to Section 3.7.1.1, except for the following parameters:
  - No request parameter “cAccessToken”, “cOpenID”, “accessToken”, “openID”, “source”, “redirectURI” and “state”.
  - No response parameter “authByQR”.
  - Response parameter “ticketID”: It is returned from CorpID and “iAM Smart” System for CorpID and “iAM Smart” APIs for anonymous request. It will be used for invoking subsequent CorpID and “iAM Smart” APIs depending on the scenario. It is a 36-byte (or less) UUID number (ASCII character set).

### 3.8.1.2 Implementing (1): EXT-013 (PDF): Request Anonymous PDF Digital Signing



## Pre-conditions

- Please refer to Section 3.7.1.2.

## Post-conditions

- Please refer to Section 3.7.1.2 except:
  - e-Service calculates a 4-digit identification code using: hashCode of PDF document + HKICHash + clientID hash value (calculated with SHA-512)

## Error conditions

- Please refer to Section 3.7.1.2.

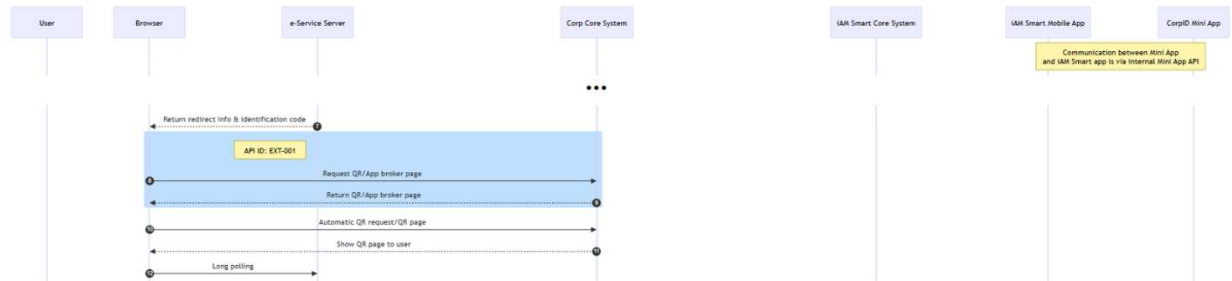
### **Request and Response Parameters**

- Please refer to CorpID API Specification.

### **Notes**

- Please refer to Section 3.7.1.2.

### 3.8.1.3 Implementing (2): EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request



#### Pre-conditions

- Please refer to Section 3.4.1.1 except:
  - e-Service has the “ticketID” provided by “iAM Smart” System for the anonymous request in step 4.

#### Post-conditions

- Please refer to Section 3.4.1.1.

#### Error conditions

- Please refer to Section 3.4.1.1.

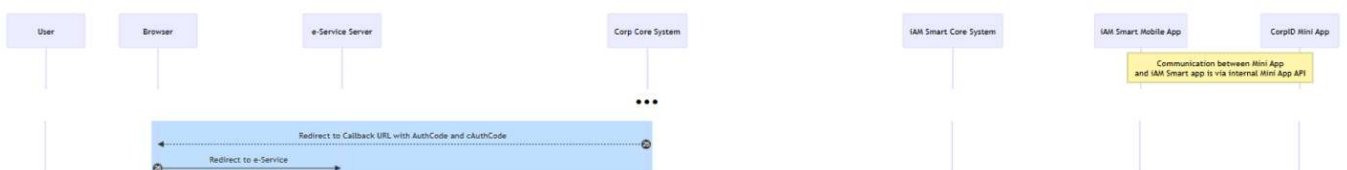
#### Request Parameters

- Please refer to Section 3.4.1.1.

#### Notes

- Please refer to Section 3.4.1.1 except the following parameter:
  - Request parameter “ticketID”: Value provided by CorpID and “iAM Smart” System for the anonymous request.

### 3.8.1.4 Implementing (3): CBL-003, Callback with AuthCode for Anonymous



#### Pre-conditions

- Please refer to Section 3.4.1.2. except:

- This one is CorpID API, for detail please refer to CorpID API Specification.

### Post-conditions

- Please refer to Section 3.4.1.2 except:
  - This one is CorpID API, for detail please refer to CorpID API Specification.
  - e-Service Server should match the callback result with corresponding e-Service client terminal using the “businessID”.

### Error conditions

- This e-Service callback API is a HTTP GET request. If parameter “cError\_code” is returned, it means the request is failed.

| Error Code           | Error Description                       | Suggested Action                                                   |
|----------------------|-----------------------------------------|--------------------------------------------------------------------|
| cError_code - M60000 | User cancelled form pre-filling request | Inform user the Form Pre-Filling request is cancelled              |
| cError_code - M60001 | User rejected form pre-filling request  | Inform user the Form Pre-Filling request is rejected               |
| cError_code - M60002 | Failed to request form pre-filling      | Inform user the Form Pre-Filling request is failed and retry later |

The error\_code M60003 (Form pre-filling request timeout) does not appear in this scenario. For the QR code timeout or request confirmation timeout in the “iAM Smart” Mobile App, message will be prompted in the corresponding user interface.

### Callback Parameters

- Please refer to Section 3.4.1.2.

### Notes

- Please refer to Section 3.4.1.2 except the following parameter:
  - Callback parameter “businessID”: Value returned by CorpID System should be the same one as e-Service submitted in the anonymous request.

### 3.8.1.5 Implementing (4): “iAM Smart” API - POST: Request accessToken & Tokenised ID



### Pre-conditions

- Please refer to Section 3.4.1.3.

### Post-conditions

- Please refer to Section 3.4.1.3. except:
  - The accessToken can only be used once before expiry.

### Error conditions

- Please refer to Section 3.4.1.3.

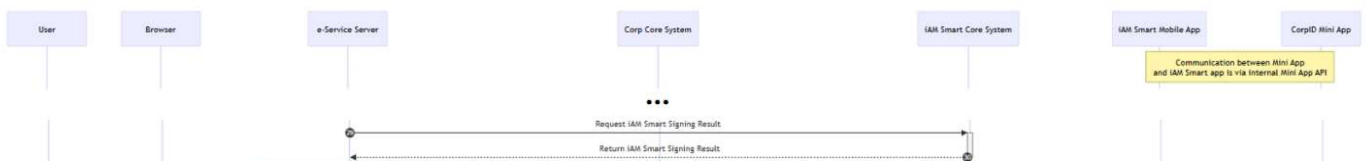
### Request and Response Parameters

- Please refer to Section 3.4.1.3.

### Notes

- Please refer to Section 3.4.1.3.

### 3.8.1.6 Implementing (5): “iAM Smart” API - POST: Obtain Anonymous Digital Signing Result



### Pre-conditions

- e-Service must possess a valid “iAM Smart” accessToken of the CorpID user with authorisation scope “Signing authorisation”.
- API data encryption is required.

### Post-conditions

- API data decryption is required.
- e-Service should discard the accessToken as they can only be used once.

### Error conditions

- For common errors, please refer to “iAM Smart” API Specification. Other errors of this API include:

| Error Code    | Error Description         | Suggested Action                                                              |
|---------------|---------------------------|-------------------------------------------------------------------------------|
| code - D70002 | Failed to request signing | Inform user the “iAM Smart” digital signing request is failed and retry later |

|               |                          |                                                                                                                                                              |
|---------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| code - D70003 | Signing request timeout  | Inform user the “iAM Smart” digital signing request is timeout and provide way for user to retry                                                             |
| code - D70004 | User not allowed to sign | Inform user that “iAM Smart” digital signing is not allowed for default “iAM Smart” and suggest user to upgrade to “iAM Smart” with digital signing function |

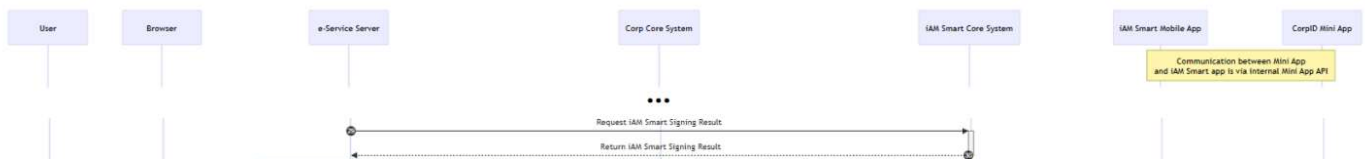
### Request and Response Parameters

- Please refer to “iAM Smart” API Specification.

### Notes

- Please refer to “iAM Smart” API Specification.

### 3.8.1.7 Implementing (5): “iAM Smart” API - POST: Obtain Anonymous PDF Digital Signing Result



### Pre-conditions

- Please refer to Section 3.7.1.6

### Post-conditions

- Please refer to Section 3.7.1.6.

### Error conditions

- Please refer to Section 3.7.1.6.

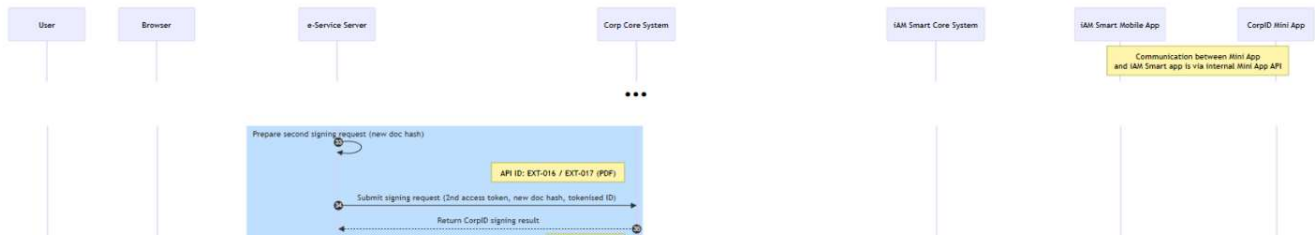
### Request and Response Parameters

- Please refer to “iAM Smart” API Specification.

### Notes

- Please refer to Section 3.7.1.6 except:
  - Response parameter “hashCode” in response, not “hashCode”.
  - Response parameter “pdfSignature” in response, not “signature” and “timestamp”.

### 3.8.1.8 Implementing (6): EXT-016: Request Corporation Digital Signing (For Integrated Signing)



#### Pre-conditions

- Please refer to Section 3.7.1.5, except:
  - No “signToken”
  - e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

#### Post-conditions

- Please refer to Section 3.7.1.5, except:
  - e-Service should discard the cAccessToken as they can only be used once.

#### Error conditions

- Please refer to Section 3.7.1.5.

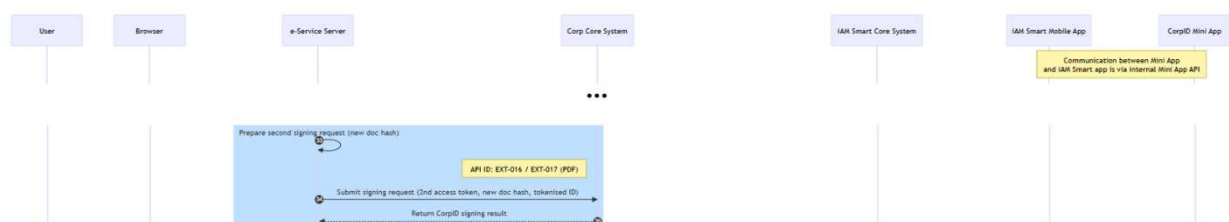
#### Request and Response Parameters

- Please refer to CorpID API Specification.

#### Notes

- Please refer to Section 3.7.1.5 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

### 3.8.1.9 Implementing (6): EXT-017: Request Corporation PDF Digital Signing (For Integrated Signing)



### Pre-conditions

- Please refer to Section 3.7.1.6, except:
  - No “signToken”
  - e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

### Post-conditions

- Please refer to Section 3.7.1.6, except:
  - e-Service should discard the cAccessToken as they can only be used once.

### Error conditions

- Please refer to Section 3.7.1.6.

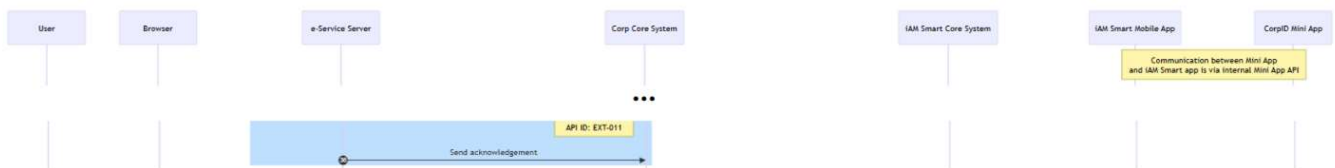
### Request and Response Parameters

- Please refer to CorpID API Specification.

### Notes

- Please refer to Section 3.7.1.6 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

### 3.8.1.10 Implementing (7): EXT-011: Acknowledges Digital Signing result



### Pre-conditions

- Please refer to Section 3.7.1.7.

### Post-conditions

- Please refer to Section 3.7.1.7.

### Error conditions

- Please refer to Section 3.7.1.7.

### Request and Response Parameters

- Please refer to Section 3.7.1.7.

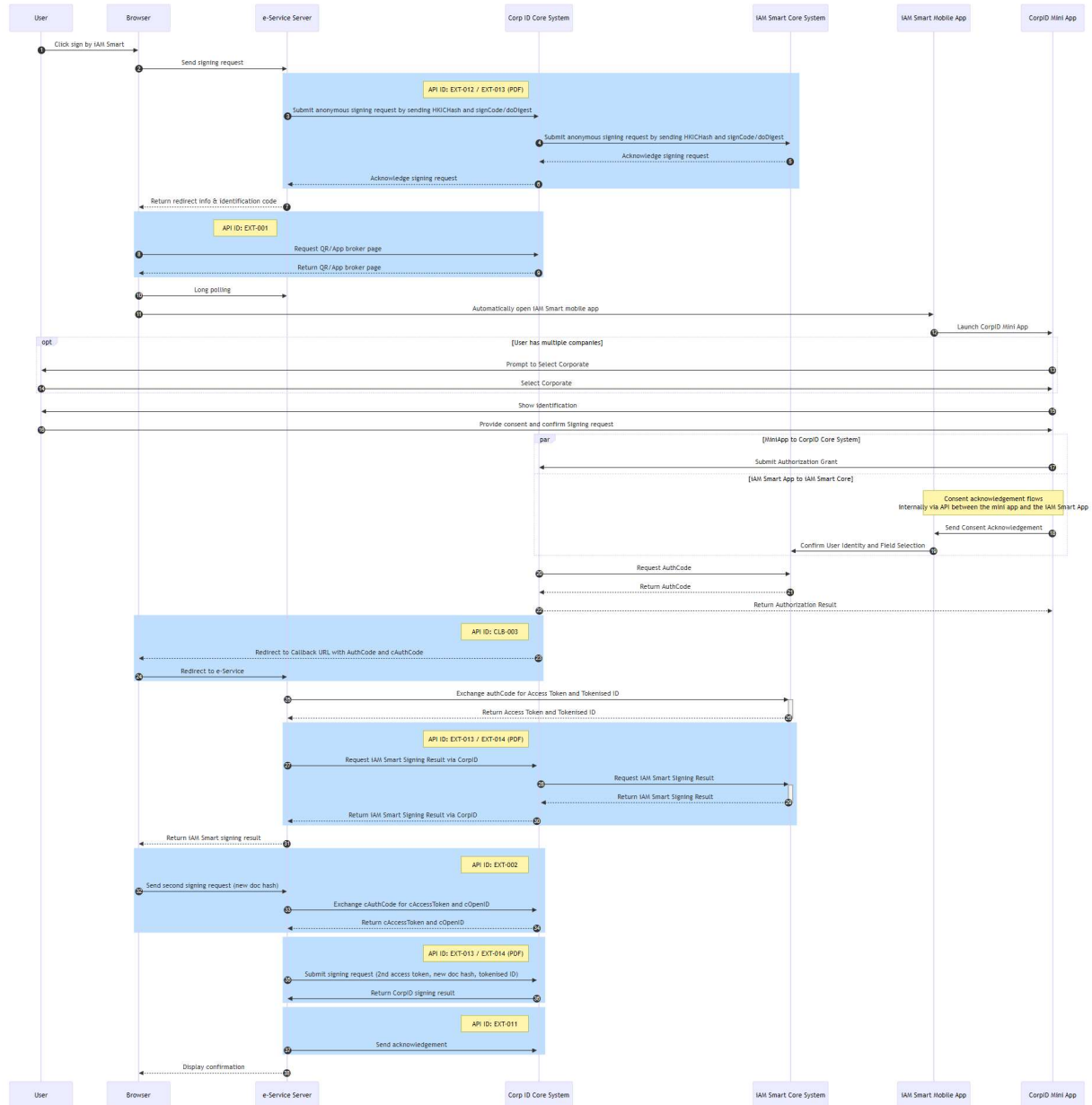
## Notes

- Please refer to Section 3.7.1.7.

### 3.8.2 Scenario 2: Anonymous Digital Signing (e-Service Website in Same Device)

The sequence diagram below shows how an anonymous user authorises and signs the Document Hash when e-Service website and the “iAM Smart” Mobile App are running in the same device.

CorpID Anonymous Digital Signing (e-Service Website in Same Device)



Step 1-2. The user clicks “Sign by CorpID” in the browser to start digital signing. The browser initiates an anonymous digital signing request to the e-Service Server.

Step 3-4. The e-Service Server initiates an anonymous digital signing request to invoke the CorpID API with the “businessID” of this request, Document Hash / PDF digest, HKICHash, signature algorithm (only for “Request Anonymous Digital Signing”), etc. API encryption is required;  
*CorpID API (EXT-012: Request Anonymous Digital Signing)*  
*CorpID API (EXT-013 (PDF): Request Anonymous PDF Digital Signing)*

- Step 5-6. The CorpID Server verifies and confirm receipt of the anonymous digital signing request to the e-Service server by returning response with parameter “ticketID”.
- Step 7-9. e-Service server uses the Document Hash / PDF Hash, HKICHash and hash of clientID to calculate a 4-digit identification code and instructs e-Service website to display the instructions with a 4-digit identification code to inform CorpID user to process the digital signing authorisation request in CorpID Mini-program.  
e-Service server prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as browser’s user agent value), “scope”, “ticketID”, etc. and constructs the GET request to invoke the CorpID API using browser redirection;  
*CorpID API (EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request)*
- Step 8-10. CorpID System returns the broker page (if e-Service has set “brokerPage” to “true”).
- Step 11. QR Code page of CorpID and “iAM Smart” System polls “iAM Smart” System for the QR Code status;
- Step 12. CorpID user logs in “iAM Smart” Mobile App to scan the QR code, and launch the CorpID Mini-program;
- Step 13-14. If CorpID user has multiple corporations, he / she will be prompted to select a corporate account.
- Step 15-16. CorpID Mini-program requests and displays 4-digit identification code from CorpID System. CorpID user reviews the digital signing authorisation request (e.g. continue or reject the request) and verifies the 4-digit identification code with e-Service Website;
- Step 17-19. CorpID System and “iAM Smart” System verifies the validity of QR code and other necessary information, and return the authorisation result to CorpID Mini-program;
- Step 20-23. CorpID System receives the polling result returned by CorpID and “iAM Smart” System (i.e. response for the polling in step 12) which includes cAuthCode “cCode” of CorpID System, authCode “code” of “iAM Smart System” and businessID or any error code (e.g. CorpID user rejects the request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection. the flow redirects to the callback URL with the AuthCode.  
*CorpID API (CBL-003, Callback with AuthCode for Anonymous)*
- Step 24-26. When e-Service server gets the cAuthCode, authCode, and businessID, it verifies businessID as generated in Step 3. e-Service then invoke “iAM Smart” API to obtain the “iAM Smart” accessToken and the Tokenised ID (i.e., openID). API data encryption is required;  
*“iAM Smart” API (POST: Request accessToken & Tokenised ID)*
- Step 27-31. e-Service Server invokes the “iAM Smart” API with the accessToken and openID to obtain the result of anonymous digital signing. API data encryption is required;

*CorpID API (POST: Get Anonymous Digital Signing Result (Personal Signing))*  
*CorpID API (POST: Get Anonymous PDF Digital Signing Result (Personal Signing))*

Step 32-34. e-Service then invoke CorpID API to obtain the CorpID cAccessToken and the Tokenised ID (cOpenID). API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*

Step 35-36. e-Service Server use the 1<sup>st</sup> signing result and combine with raw document to create a “iAM Smart”-only signed document. A new document hash is generated and call the CorpID API with required parameters to obtain the result of CorpID signing. Api data encryption is required.

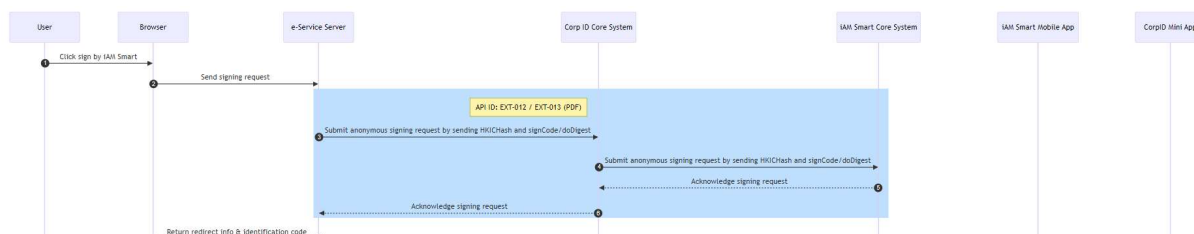
Under API ID: EXT-002, a second signing request is sent with a new document hash.

*CorpID API (EXT-016: Request Corporation Digital Signing (For Integrated Signing)*  
*CorpID API (EXT-017: Request Corporation PDF Digital Signing (For Integrated Signing)*

Step 37. e-Service Server completes the document digital signing process, verify the result and acknowledges CorpID System the digital signing result using CorpID API with the same “businessID” of the digital signing request. API data encryption is required;  
*CorpID API (EXT-011: Acknowledges Digital Signing result)*

Step 38. The browser displays the digital signing result to the user.

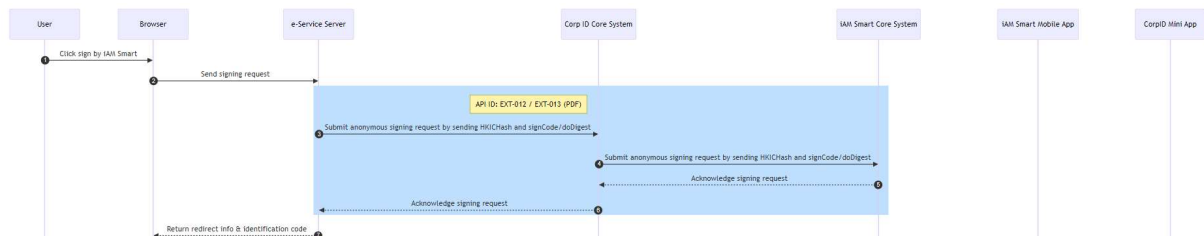
### 3.8.2.1 Implementing (1) POST: Request Anonymous Digital Signing



Please refer to Section 3.8.1.1 except:

- e-Service Website and “iAM Smart” Mobile App are in the same device in this scenario.

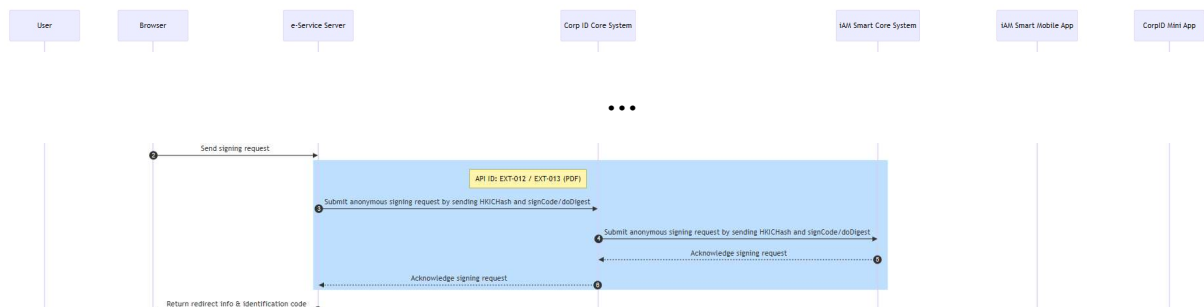
### 3.8.2.2 Implementing (1) POST: Request Anonymous PDF Digital Signing



Please refer to Section 3.8.1.2 except:

- e-Service Website and “iAM Smart” Mobile App are in the same device in this scenario.

### 3.8.2.3 Implementing (2) GET: Request QR Page



#### Pre-conditions

- Please refer to Section 3.4.2.1 except:
  - e-Service has the “ticketID” provided by “iAM Smart” System for the anonymous request in step 4.

#### Post-conditions

- Please refer to Section 3.4.2.1.

#### Error conditions

- Please refer Section 3.4.2.1.

#### Request Parameters

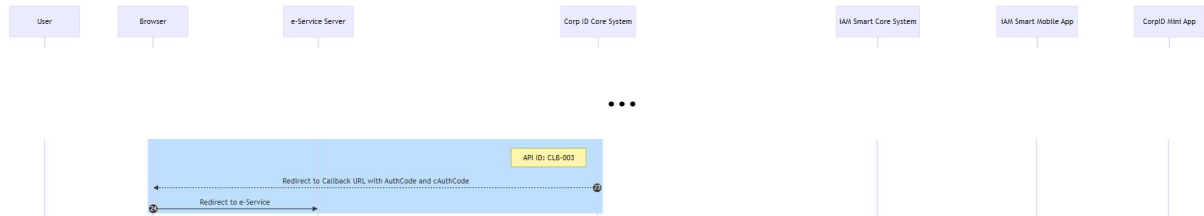
- Please refer to “iAM Smart” API Specification.

#### Notes

- Please refer to Section 3.4.2.1 except the following parameter:

- Request parameter “ticketID”: Value provided by “iAM Smart” System for the anonymous request.

#### 3.8.2.4 Implementing (3) GET: Callback with authCode to e-Service



#### Pre-conditions

- Please refer to Section 3.6.2.3.

#### Post-conditions

- Please refer to Section 3.6.2.3.

#### Error conditions

- Please refer to Section 3.8.1.4.

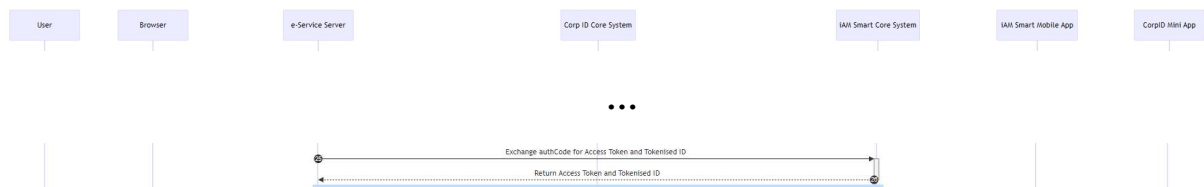
#### Callback Parameters

- Please refer to Section 3.6.2.3.

#### Notes

- Please refer to Section 3.6.2.3.

#### 3.8.2.5 Implementing (4) POST: Request accessToken & Tokenised ID

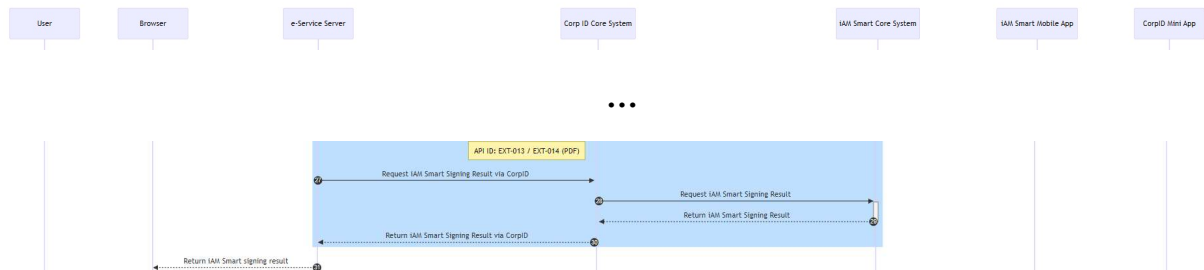


Please refer to Section 3.6.1.4.

### 3.8.2.6 Implementation (5): EXT-002: Get Access Token

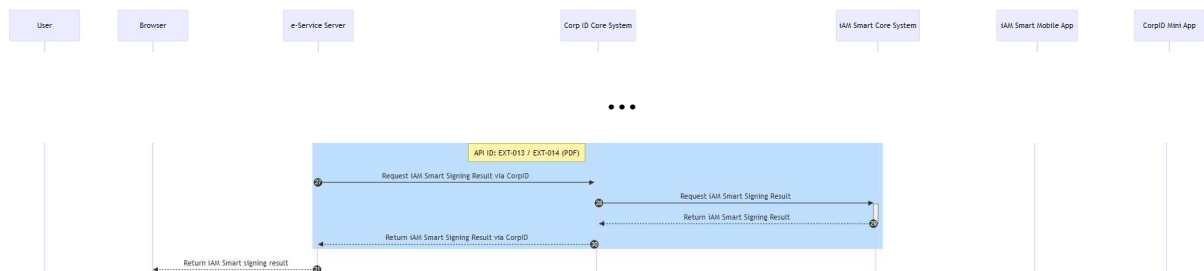
Please refer to Section 3.6.1.5.

### 3.8.2.7 Implementing (6) POST: Obtain Anonymous Digital Signing Result



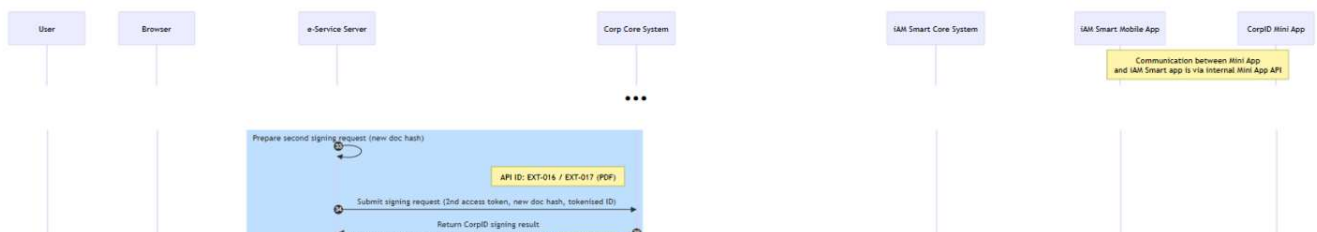
Please refer to Section 3.8.1.6.

### 3.8.2.8 Implementing (6) POST: Obtain Anonymous PDF Digital Signing Result



Please refer to Section 3.8.1.7.

### 3.8.2.9 Implementing (7): EXT-016: Request Corporation Digital Signing (For Integrated Signing)



#### Pre-conditions

- Please refer to Section 3.7.1.5, except:
  - No “signToken”

- e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

#### Post-conditions

- Please refer to Section 3.7.1.5, except:
  - e-Service should discard the cAccessToken as they can only be used once.

#### Error conditions

- Please refer to Section 3.7.1.5.

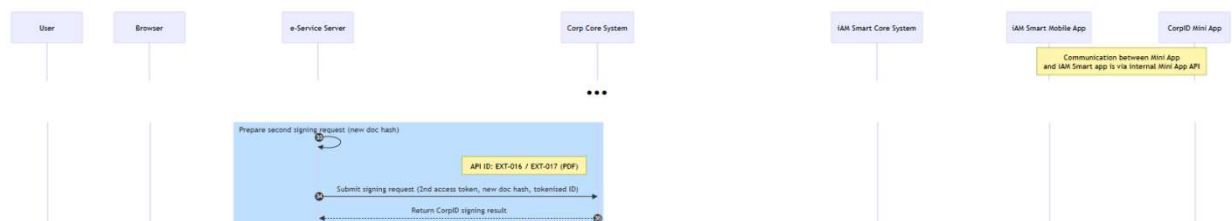
#### Request and Response Parameters

- Please refer to CorpID API Specification.

#### Notes

- Please refer to Section 3.7.1.5 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

### 3.8.2.10 Implementing (7): EXT-017: Request Corporation PDF Digital Signing (For Integrated Signing)



#### Pre-conditions

- Please refer to Section 3.7.1.6, except:
  - No “signToken”
  - e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

#### Post-conditions

- Please refer to Section 3.7.1.6, except:
  - e-Service should discard the cAccessToken as they can only be used once.

#### Error conditions

- Please refer to Section 3.7.1.6.

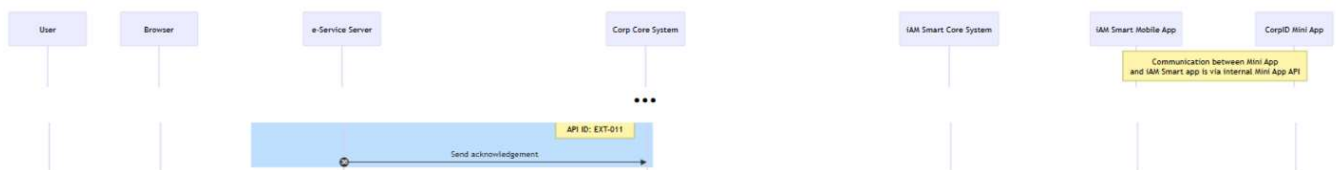
## Request and Response Parameters

- Please refer to CorpID API Specification.

## Notes

- Please refer to Section 3.7.1.6 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

### 3.8.2.11 Implementing (8): EXT-011: Acknowledges Digital Signing result



## Pre-conditions

- Please refer to Section 3.7.1.7.

## Post-conditions

- Please refer to Section 3.7.1.7.

## Error conditions

- Please refer to Section 3.7.1.7.

## Request and Response Parameters

- Please refer to Section 3.7.1.7.

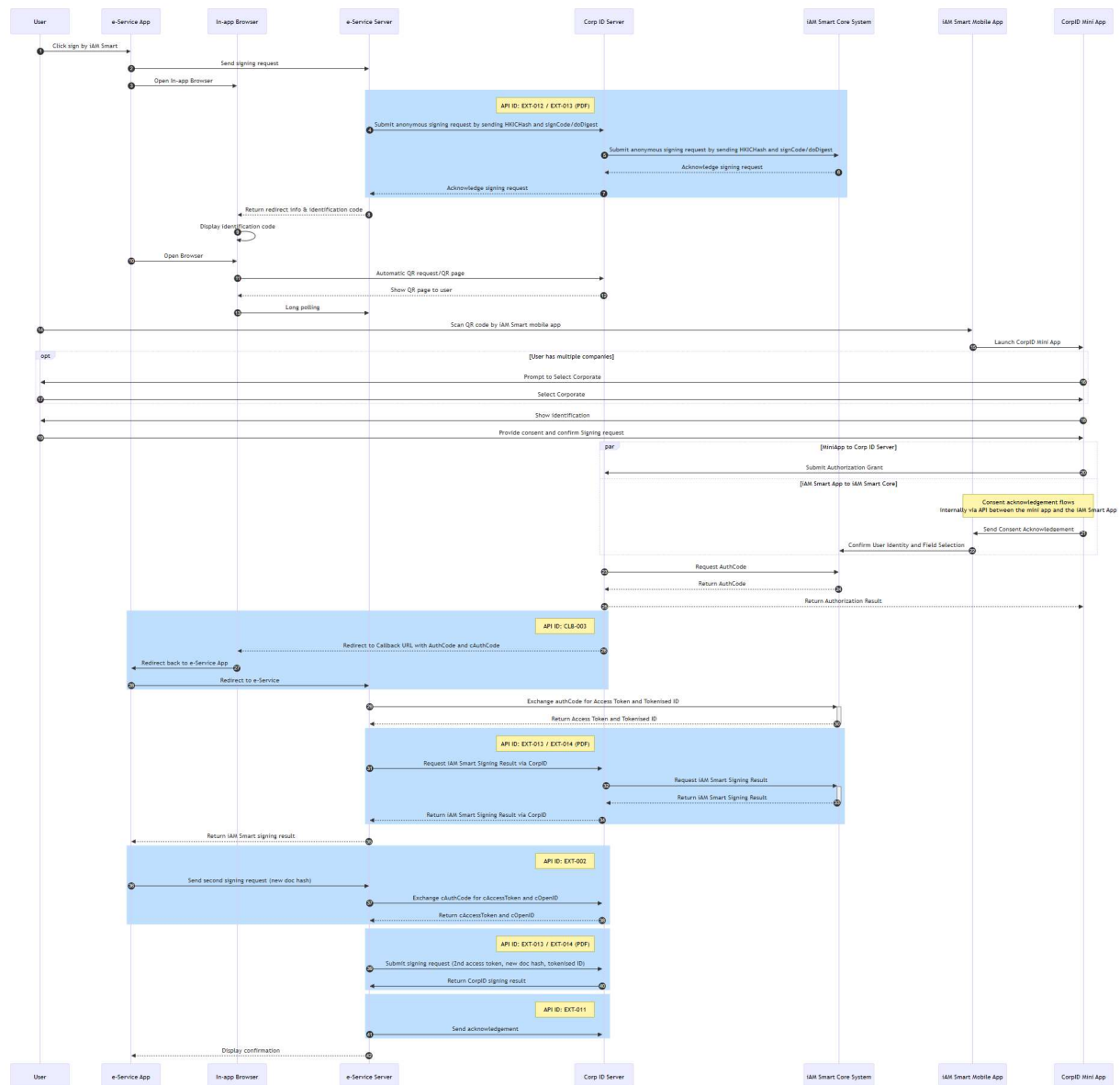
## Notes

- Please refer to Section 3.7.1.7.

### 3.8.3 Scenario 3: Anonymous Digital Signing (e-Service App in Different Device)

The sequence diagram below shows how an anonymous user authorises and signs the Document Hash when e-Service App and the “iAM Smart” Mobile App are running in different devices.

CorpID Anonymous Digital Signing (e-Service App in Different Device)



- Step 1-2. The user clicks “Sign by CorpID” in the e-Service App to start digital signing. The App initiates an anonymous digital signing request to the e-Service Server.
- Step 3-7. The e-Service Server initiates an anonymous digital signing request to invoke the CorpID API with the “businessID” of this request, Document Hash / PDF digest, HKIChash, signature algorithm (only for “Request Anonymous Digital Signing”), etc. API encryption is required;

*CorpID API (EXT-012: Request Anonymous Digital Signing)*

*CorpID API (EXT-013 (PDF): Request Anonymous PDF Digital Signing)*

- Step 8. The CorpID Server verifies and confirm receipt of the anonymous digital signing request to the e-Service server by returning response with parameter “ticketID”.
- Step 9-10. e-Service server uses the Document Hash / PDF Hash, HKICHash and hash of clientID to calculate a 4-digit identification code and instructs e-Service website to display the instructions with a 4-digit identification code to inform CorpID user to process the digital signing authorisation request in CorpID Mini-program.  
e-Service server prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as in-app browser’s user agent value), “scope”, “ticketID”, etc. and constructs the GET request to invoke the CorpID API using in-app browser redirection;  
*CorpID API (EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request)*
- Step 11-13. CorpID System returns the broker page (if e-Service has set “brokerPage” to “true”) and after the broker page fails to find the “iAM Smart” Mobile App, it will request QR page to be displayed in in-app browser. If e-Service does not request for broker page (i.e. no broker page will be returned), the in-app browser will directly display QR Code page.
- Step 13. QR Code page of CorpID and “iAM Smart” System polls “iAM Smart” System for the QR Code status;
- Step 14-15. CorpID user logs in “iAM Smart” Mobile App to scan the QR code, and launch the CorpID Mini-program;
- Step 16-17. If CorpID user has multiple corporations, he / she will be prompted to select a corporate account.
- Step 18-25. CorpID Mini-program requests and displays 4-digit identification code from CorpID System. CorpID user reviews the digital signing authorisation request (e.g. continue or reject the request) and verifies the 4-digit identification code with e-Service App; CorpID System and “iAM Smart” System verifies the validity of QR code and other necessary information, and return the authorisation result to CorpID Mini-program;
- Step 26-28. QR Code page of CorpID System receives the polling result returned by CorpID and “iAM Smart” System (i.e. response for the polling in step 12) which includes cAuthCode “cCode” of CorpID System, authCode “code” of “iAM Smart System” and businessID or any error code (e.g. CorpID user rejects the request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection. the flow redirects to the callback URL with the AuthCode.  
*CorpID API (CBL-003, Callback with AuthCode for Anonymous)*
- Step 29-30. When e-Service server gets the cAuthCode, authCode, and businessID, it verifies businessID as generated in Step 3. e-Service then invoke “iAM Smart” API to obtain

the “iAM Smart” accessToken and the Tokenised ID (i.e., openID). API data encryption is required;

*“iAM Smart” API (POST: Request accessToken & Tokenised ID)*

Step 31-35. e-Service Server invokes the “iAM Smart” API with the accessToken and openID to obtain the result of anonymous digital signing. API data encryption is required;  
*CorpID API (POST: Get Anonymous Digital Signing Result (Personal Signing))*  
*CorpID API (POST: Get Anonymous PDF Digital Signing Result (Personal Signing))*

Step 36-38. e-Service then invoke CorpID API to obtain the CorpID cAccessToken and the Tokenised ID (cOpenID). API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*

Step 39-40. e-Service Server use the 1<sup>st</sup> signing result and combine with raw document to create a “iAM Smart”-only signed document. A new document hash is generated and call the CorpID API with required parameters to obtain the result of CorpID signing. Api data encryption is required.

Under API ID: EXT-002, a second signing request is sent with a new document hash.

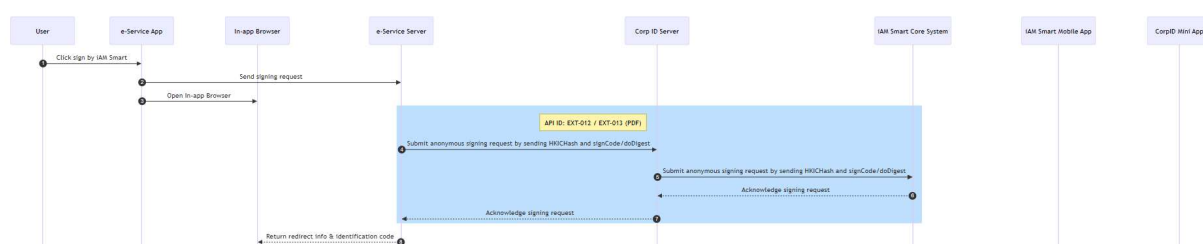
*CorpID API (EXT-016: Request Corporation Digital Signing (For Integrated Signing))*

*CorpID API (EXT-017: Request Corporation PDF Digital Signing (For Integrated Signing))*

Step 41. e-Service Server completes the document digital signing process, verify the result and acknowledges CorpID System the digital signing result using CorpID API with the same “businessID” of the digital signing request. API data encryption is required;  
*CorpID API (EXT-011: Acknowledges Digital Signing result)*

Step 42. The App displays the digital signing result to the user.

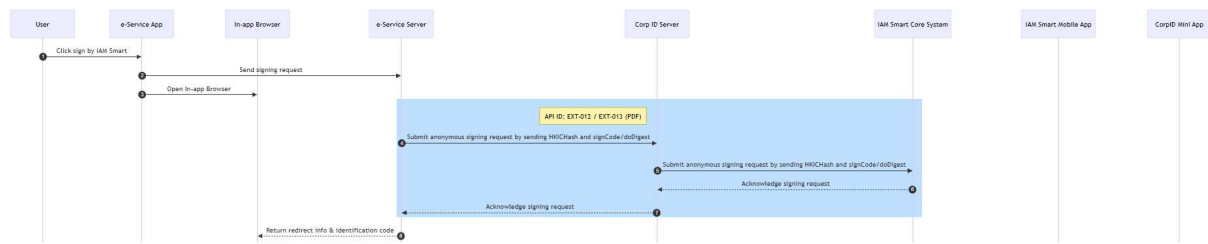
### 3.8.3.1 Implementing (1) POST: Request Anonymous Digital Signing



Please refer to Section 3.8.1.1 except:

- e-Service should determine the “iAM Smart” Mobile App is not in the same device of e-Service App using program code.

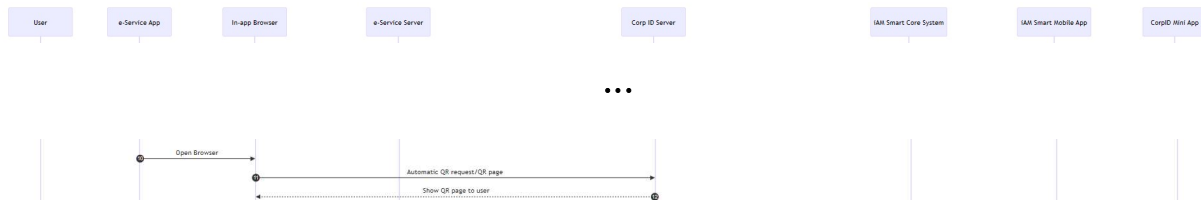
### 3.8.3.2 Implementing (1) POST: Request Anonymous PDF Digital Signing



Please refer to Section 3.8.1.2 except:

- e-Service should determine the “iAM Smart” Mobile App is not in the same device of e-Service App using program code.

### 3.8.3.3 Implementing (2) GET: Request QR Page



#### Pre-conditions

- Please refer to Section 3.4.3.1 except:
  - e-Service has the “ticketID” provided by “iAM Smart” System for the anonymous request in step 4.

#### Post-conditions

- Please refer to Section 3.4.3.1.

#### Error conditions

- Please refer to Section 3.4.3.1.

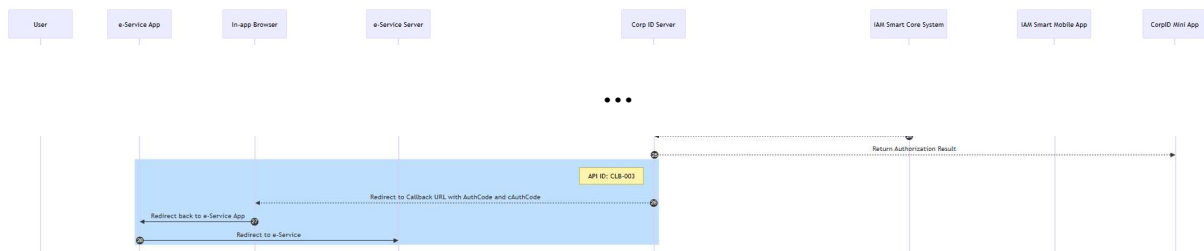
#### Request Parameters

- Please refer to “iAM Smart” API Specification.

#### Notes

- Please refer to Section 3.4.3.1 except the following parameter:
  - Request parameter “ticketID”: Value provided by “iAM Smart” System for the anonymous request.

### 3.8.3.4 Implementing (3) GET: Callback with authCode to e-Service Server



Please refer to Section 3.8.1.4.

### 3.8.3.5 Implementing (4) POST: Request accessToken & Tokenised ID

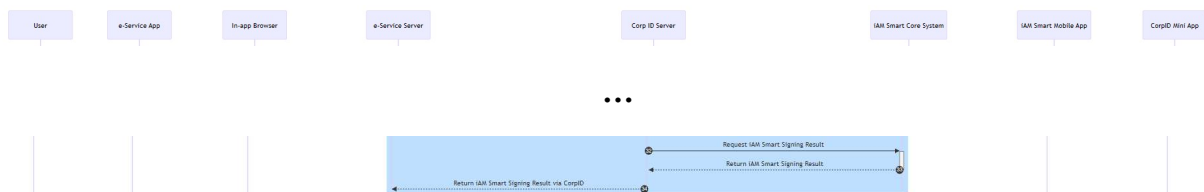


Please refer to Section 3.6.1.4.

### 3.8.3.6 Implementing (5) POST: EXT-002: Get Access Token

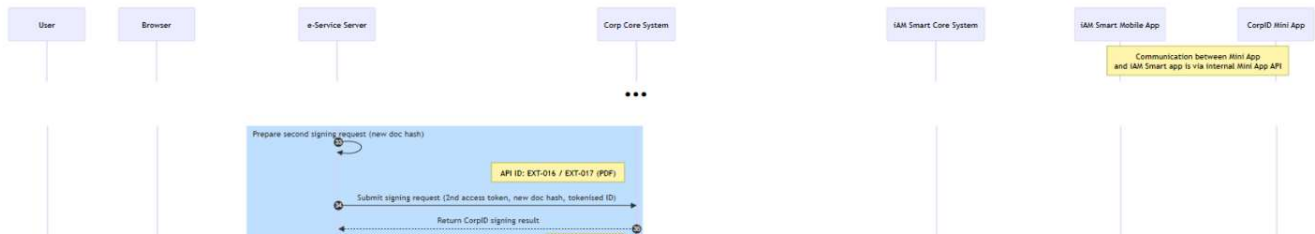
Please refer to Section 3.6.1.5.

### 3.8.3.7 Implementing (6) POST: Obtain Anonymous Digital Signing Result



Please refer to Section 3.8.1.6.

### 3.8.3.8 Implementing (7): EXT-016: Request Corporation Digital Signing (For Integrated Signing)



#### Pre-conditions

- Please refer to Section 3.7.1.5, except:
  - No “signToken”
  - e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

#### Post-conditions

- Please refer to Section 3.7.1.5, except:
  - e-Service should discard the cAccessToken as they can only be used once.

#### Error conditions

- Please refer to Section 3.7.1.5.

#### Request and Response Parameters

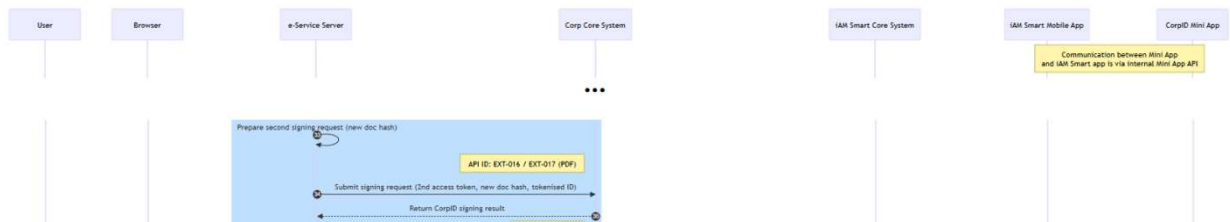
- Please refer to CorpID API Specification.

#### Notes

- Please refer to Section 3.7.1.5 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

### 3.8.3.9 Implementing (7): EXT-017: Request Corporation PDF Digital Signing (For Integrated

## Signing



### Pre-conditions

- Please refer to Section 3.7.1.6, except:
  - No “signToken”
  - e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

### Post-conditions

- Please refer to Section 3.7.1.6, except:
  - e-Service should discard the cAccessToken as they can only be used once.

### Error conditions

- Please refer to Section 3.7.1.6.

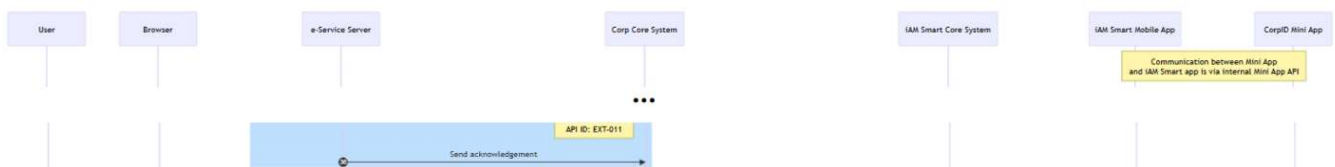
### Request and Response Parameters

- Please refer to CorpID API Specification.

### Notes

- Please refer to Section 3.7.1.6 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

### 3.8.3.10 Implementing (8): EXT-011: Acknowledges Digital Signing result



### Pre-conditions

- Please refer to Section 3.7.1.7.

### Post-conditions

- Please refer to Section 3.7.1.7.

#### **Error conditions**

- Please refer to Section 3.7.1.7.

#### **Request and Response Parameters**

- Please refer to Section 3.7.1.7.

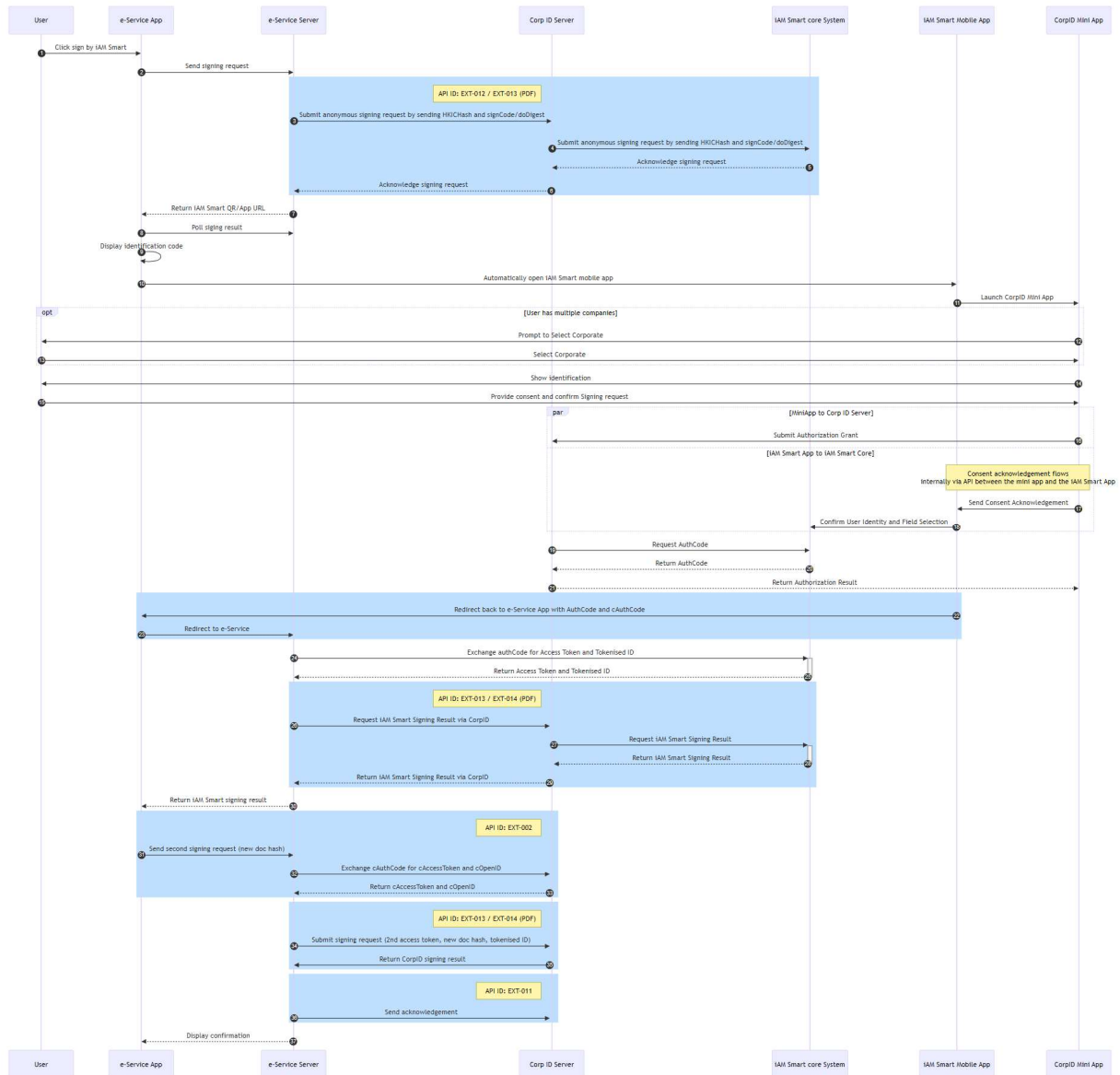
#### **Notes**

- Please refer to Section 3.7.1.7.

### 3.8.4 Scenario 4: Anonymous Digital Signing (e-Service App in Same Device)

The sequence diagram below shows how an anonymous user authorises and signs the Document Hash when e-Service App and the “iAM Smart” Mobile App are running in the same device.

CorpID Anonymous Digital Signing (e-Service App in Same Device)

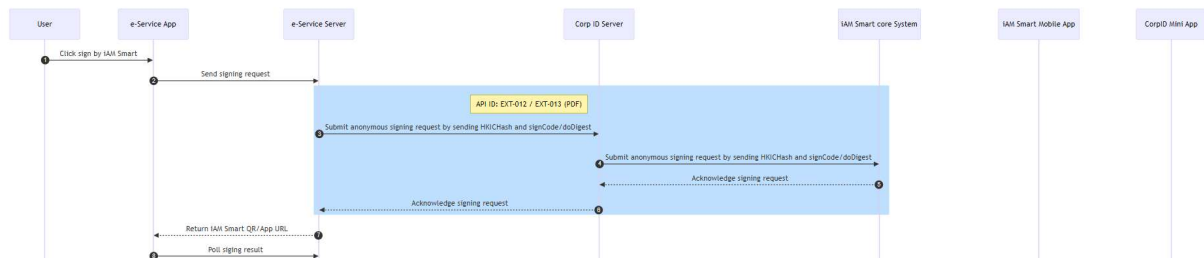


- Step 1-2. The user clicks “Sign by CorpID” in the e-Service App to start digital signing. The app initiates an anonymous digital signing request to the e-Service Server.
- Step 3-4. The e-Service Server initiates an anonymous digital signing request to invoke the CorpID API with the “businessID” of this request, Document Hash / PDF digest, HKICHash, signature algorithm (only for “Request Anonymous Digital Signing”), etc. API encryption is required;  
*CorpID API (EXT-012: Request Anonymous Digital Signing)*  
*CorpID API (EXT-013 (PDF): Request Anonymous PDF Digital Signing)*

- Step 5-6. The CorpID Server verifies and confirm receipt of the anonymous digital signing request to the e-Service server by returning response with parameter “ticketID”.
- Step 7-8. e-Service server uses the Document Hash / PDF Hash, HKICHash and hash of clientID to calculate a 4-digit identification code and instructs e-Service website to display the instructions with a 4-digit identification code to inform CorpID user to process the digital signing authorisation request in CorpID Mini-program.  
e-Service server prepares the request parameters such as “clientID”, “redirectURI”, “source” (set as browser’s user agent value), “scope”, “ticketID”, etc. and constructs the GET request to invoke the CorpID API using in-app browser redirection;  
*CorpID API (EXT-001: Authentication by Requesting QR Page / Broker Page for Anonymous Request)*  
  
CorpID System returns the broker page (if e-Service has set “brokerPage” to “true”).
- Step 8. Broker page of CorpID and “iAM Smart” System polls “iAM Smart” System for the status;
- Step 11. Broker Page trigger “iAM Smart” Mobile App, and launch the CorpID Mini-program;
- Step 12-13. If CorpID user has multiple corporations, he / she will be prompted to select a corporate account.
- Step 14-15. CorpID Mini-program requests and displays 4-digit identification code from CorpID System. CorpID user reviews the digital signing authorisation request (e.g. continue or reject the request) and verifies the 4-digit identification code with e-Service Website;
- Step 16-21. CorpID System and “iAM Smart” System verifies the validity and other necessary information, and return the authorisation result to CorpID Mini-program;
- Step 22-23. CorpID System receives the polling result returned by CorpID and “iAM Smart” System (i.e. response for the polling in step 12) which includes cAuthCode “cCode” of CorpID System, authCode “code” of “iAM Smart System” and businessID or any error code (e.g. CorpID user rejects the request), assembles and invokes the e-Service callback API given in “redirectURI” using browser redirection. the flow redirects to the callback URL with the AuthCode.  
*CorpID API (CBL-003, Callback with AuthCode for Anonymous)*
- Step 24-25. When e-Service server gets the cAuthCode, authCode, and businessID, it verifies businessID as generated in Step 3. e-Service then invoke “iAM Smart” API to obtain the “iAM Smart” accessToken and the Tokenised ID (i.e., openID). API data encryption is required;  
*“iAM Smart” API (POST: Request accessToken & Tokenised ID)*
- Step 26-30. e-Service Server invokes the “iAM Smart” API with the accessToken and openID to obtain the result of anonymous digital signing. API data encryption is required;  
*CorpID API (POST: Get Anonymous Digital Signing Result (Personal Signing))*  
*CorpID API (POST: Get Anonymous PDF Digital Signing Result (Personal Signing))*

- Step 31-33. e-Service then invoke CorpID API to obtain the CorpID cAccessToken and the Tokenised ID (cOpenID). API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*
- Step 34-35. e-Service Server use the 1<sup>st</sup> signing result and combine with raw document to create a “iAM Smart”-only signed document. A new document hash is generated and call the CorpID API with required parameters to obtain the result of CorpID signing. Api data encryption is required.  
 Under API ID: EXT-002, a second signing request is sent with a new document hash.  
*CorpID API (EXT-016: Request Corporation Digital Signing (For Integrated Signing)*  
*CorpID API (EXT-017: Request Corporation PDF Digital Signing (For Integrated Signing)*
- Step 36. e-Service Server completes the document digital signing process, verify the result and acknowledges CorpID System the digital signing result using CorpID API with the same “businessID” of the digital signing request. API data encryption is required;  
*CorpID API (EXT-011: Acknowledges Digital Signing result)*
- Step 37. The browser displays the digital signing result to the user.

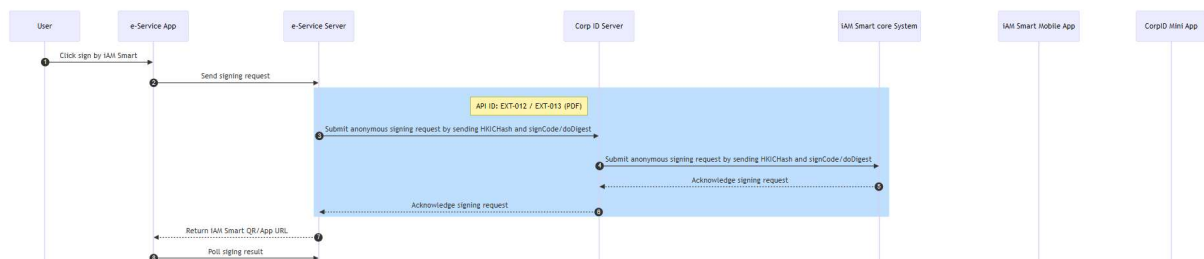
### 3.8.4.1 Implementing (1) POST: Request Anonymous Digital Signing



Please refer to Section 3.8.1.1 except:

- e-Service should determine the “iAM Smart” Mobile App is on the same device of e-Service App using program code.

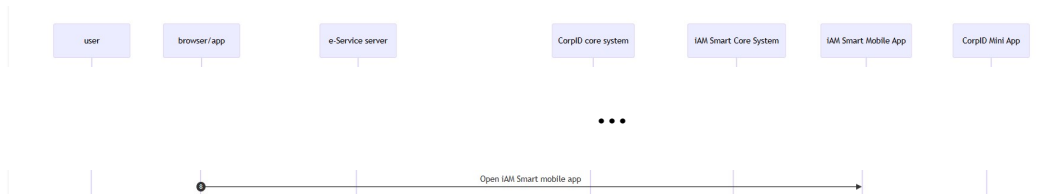
### 3.8.4.2 Implementing (1) POST: Request Anonymous PDF Digital Signing



Please refer to Section 3.8.1.2 except:

- e-Service should determine the “iAM Smart” Mobile App is on the same device of e-Service App using program code.

### 3.8.4.3 Implementing (2) URL Scheme: Open iAM Smart Mobile App for CorpID Digital Signing



#### Pre-conditions

- e-Service Website and “iAM Smart” Mobile App are in the same device.

#### Post-conditions

- “iAM Smart” Mobile App will be launched.
- e-Service Website should keep synchronising with e-Service server for the callback response of the digital signing request from “iAM Smart” System.

#### Error conditions

- Nil

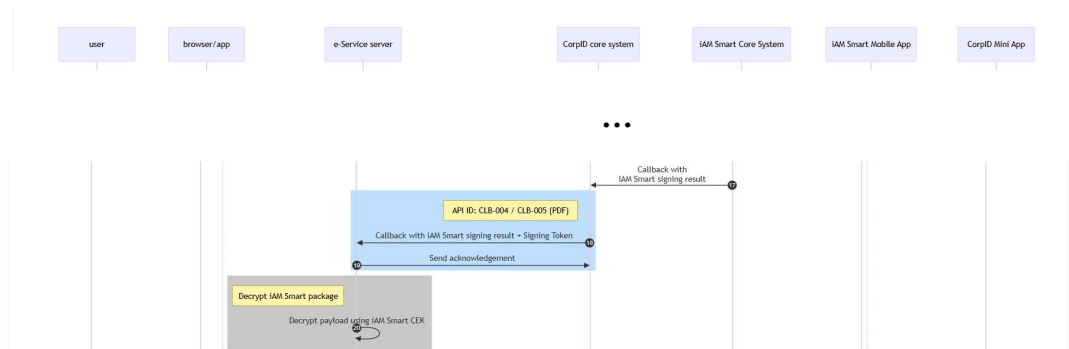
#### Request Parameters

- Please refer to “iAM Smart” for CorpID API Specification.

#### Notes

- Set <Context> as “corpId?params=corpapi\_auth%20corpapi\_sign” in URL Scheme.

### 3.8.4.4 Implementing (3) Callback with authCode to e-Service App



## Pre-conditions

Please refer to Section 3.4.2.2.

## Post-conditions

- Please refer to Section 3.4.2.2 except:
  - e-Service Server should match the callback result with corresponding e-Service App using the “businessID”.

## Error conditions

- Please refer to Section 3.8.1.4 except:
  - This e-Service callback API is a URL Scheme, or Universal Link/App Link.

## Callback Parameters

- Please refer to “iAM Smart” API Specification.

## Notes

- Please refer to Section 3.4.2.2 except the following parameter:
  - Callback parameter “businessID”: Value returned by “iAM Smart” System should be the same one as e-Service submitted in the anonymous request.

### 3.8.4.5 Implementing (4) POST: Request accessToken & Tokenised ID



Please refer to Section 3.6.1.4.

### 3.8.4.6 Implementing (5) POST: EXT-002: Get Access Token

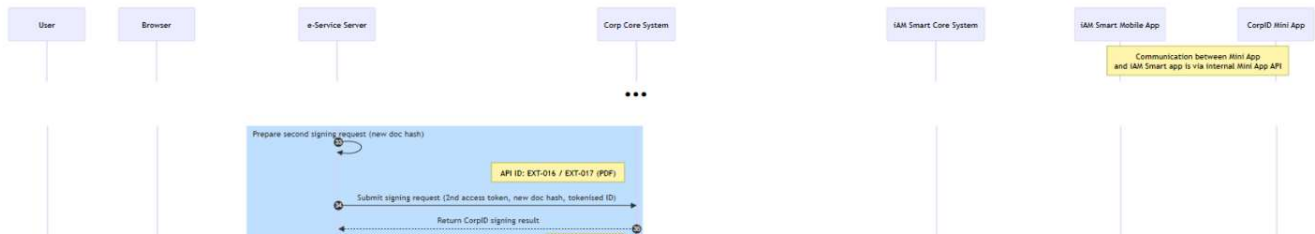
Please refer to Section 3.6.1.5.

### 3.8.4.7 Implementing (6) POST: Obtain Anonymous Digital Signing Result



Please refer to Section 3.8.1.6.

### 3.8.4.8 Implementing (7): EXT-016: Request Corporation Digital Signing (For Integrated Signing)



#### Pre-conditions

- Please refer to Section 3.7.1.5, except:
  - No “signToken”
  - e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

#### Post-conditions

- Please refer to Section 3.7.1.5, except:
  - e-Service should discard the cAccessToken as they can only be used once.

#### Error conditions

- Please refer to Section 3.7.1.5.

#### Request and Response Parameters

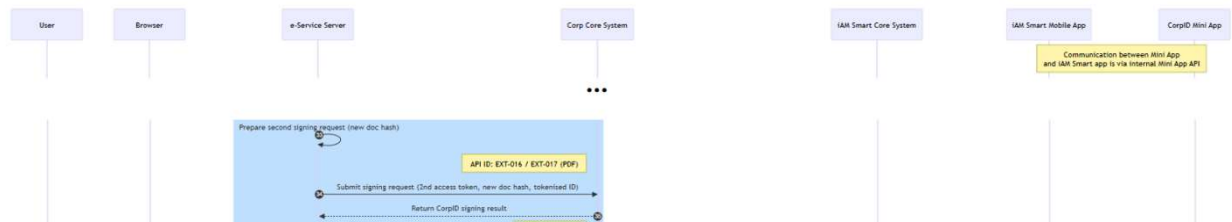
- Please refer to CorpID API Specification.

#### Notes

- Please refer to Section 3.7.1.5 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

### 3.8.4.9 Implementing (7): EXT-017: Request Corporation PDF Digital Signing (For Integrated

## Signing



### Pre-conditions

- Please refer to Section 3.7.1.6, except:
  - No “signToken”
  - e-Service control whether to call personal signing with “iAM Smart” Digital Signing API, instead of passing “signType” in request.

### Post-conditions

- Please refer to Section 3.7.1.6, except:
  - e-Service should discard the cAccessToken as they can only be used once.

### Error conditions

- Please refer to Section 3.7.1.6.

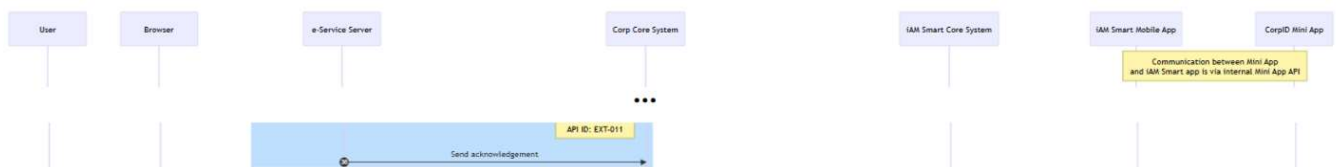
### Request and Response Parameters

- Please refer to CorpID API Specification.

### Notes

- Please refer to Section 3.7.1.6 except:
  - No “signToken”
  - No “signType”, e-Service control the signing flow
  - cAccessToken can only use once.

#### 3.8.4.10 Implementing (8): EXT-011: Acknowledges Digital Signing result



### Pre-conditions

- Please refer to Section 3.7.1.7.

### Post-conditions

- Please refer to Section 3.7.1.7.

#### **Error conditions**

- Please refer to Section 3.7.1.7.

#### **Request and Response Parameters**

- Please refer to Section 3.7.1.7.

#### **Notes**

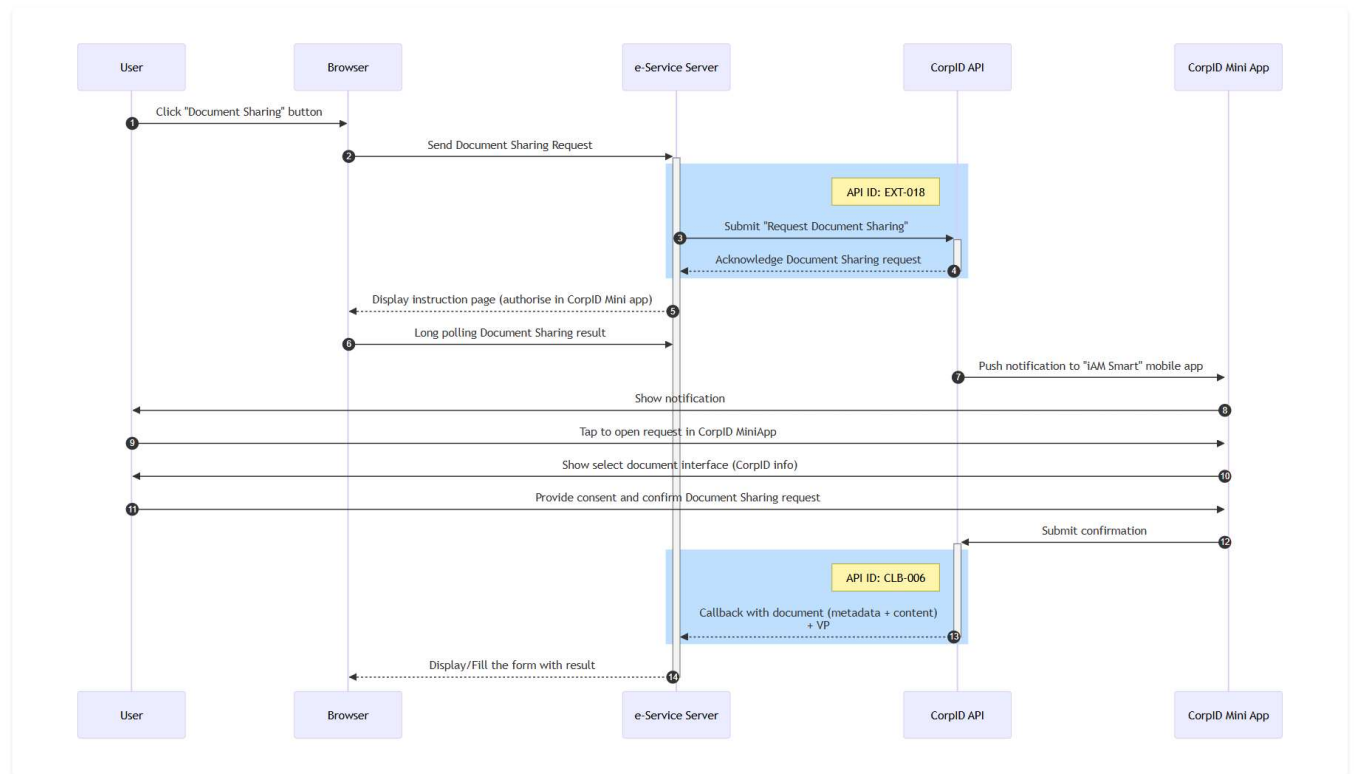
- Please refer to Section 3.7.1.7.

### 3.9 DOCUMENT SHARING WITH SERVICE LOGIN

#### 3.9.1 Scenario 1: Document Sharing (e-Service Website in Different Device)

The sequence diagram below shows how an authenticated user authorises and share document(s) when e-Service website and the “iAM Smart” Mobile App are running in different devices.

##### CorpID Document Wallet (e-Service Website in Different Device)



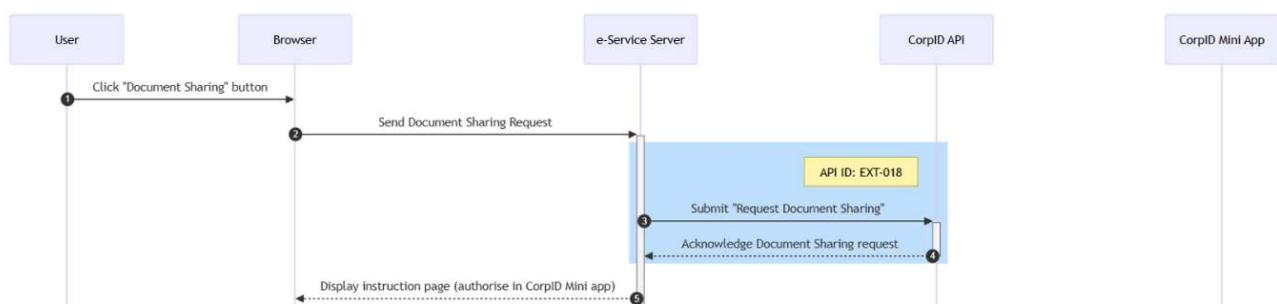
- Step 1. The user clicks the “Document Sharing” button in the browser.
- Step 2-3. The browser sends a Document Sharing request to the e-Service Server. The e-Service Server submits “Request Document Sharing” request to the CorpID Server with the “businessID”, “cAccessToken”, “cOpenID”, “docType”, “docExpTs”, “redirectURI” and “state”, etc.  
*CorpID API (EXT-018: Request Document Sharing).*
- Step 4. The CorpID API acknowledges the Document Sharing request and returns true for “authByQR” to indicate it is different device.
- Step 5-6. The e-Service Server displays an instruction page telling the user to authorise in the CorpID Mini-program. The browser performs long polling to check the Document Sharing result.

- Step 7. The CorpID API pushes a notification to the “iAM Smart” mobile app.
- Step 8-9. The user sees the notification and taps it to open the request in CorpID Mini-program.
- Step 10. The Mini-program shows the select document interface with CorpID information.
- Step 11. The user select document(s) for sharing, provides consent and confirms the Document Sharing request.
- Step 12-13. The Mini-program submits the confirmation back through the flow. Under API ID: CLB-006, the system performs a callback with document data including metadata + content + VP and required parameters. The e-Service Server receives the result.  
*CorpID API (CLB-006: Callback to Receive Document Sharing).*
- Step 14. e-Service Server displays/fills the form with the result in the Browser.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No. | API Name                                      | API Reference (Section)               |
|-----|-----------------------------------------------|---------------------------------------|
| 1   | Request Document Sharing                      | CorpID API Specification Section 4.22 |
| 2   | Open CorpID Mini-program for Document Sharing | CorpID API Specification Section 4.30 |
| 3   | Callback to Receive Document Sharing          | CorpID API Specification Section 4.23 |

### 3.9.1.1 Implementing (1): EXT-018: Request Document Sharing



#### Pre-conditions

- e-Service must possess a valid cAccessToken of the CorpID user with authorisation scope “Document Wallet”.
- e-Service Website/App and “iAM Smart” Mobile App are in different devices in this scenario.

- e-Service can set the “docType” for e-Service to specific which document type it want.
  - For more than one document type required, use “,” to separate.
  - If e-Service does not pass this parameter, CorpID would allow the CorpID user to select any of type of documents.
  - Detail document list will be provided in separate document.
- e-Service can set the “docExpTs” to specific the oldest document expiry range it want.
- The CorpID user who share the document with CorpID must have a “user\_submit” in “userPrivilege” during authentication or switching corporation’s response.
- The CorpID user’s company RP / AR / ADMIN should grant the document to share in correct folder that this CorpID user can access, or grant access right to user within valid time period and/or number of use.
- e-Service generates a “businessID” for this Digital Signing request and uses this identifier to match the callback result returned from CorpID System.
- The URL in “redirectURI” should be registered in CorpID Self-Service Portal’s whitelist.
- If the “state” parameter is presented in the request message, the same state value will be returned to e-Service during the callback. It is used to prevent the CSRF attack. The value of state is defined by e-Service, and it should be a secure random string of any length less than or equal to 36 (UUID string length). Only ASCII letters, numbers, underscore and hyphens are accepted.
- API data encryption is required.

#### **Post-conditions**

- e-Service server should check the response parameter “authByQR” and “ticketID” and determine the next action. For details, please refer to CorpID API Specification. “ticketID” will not be provided by CorpID System in this scenario.
- e-Service Website/App should keep synchronising with e-Service Server for the document sharing result returned from CorpID System.
- API data decryption is required.

#### **Error conditions**

- For common errors, please refer to CorpID API Specification.

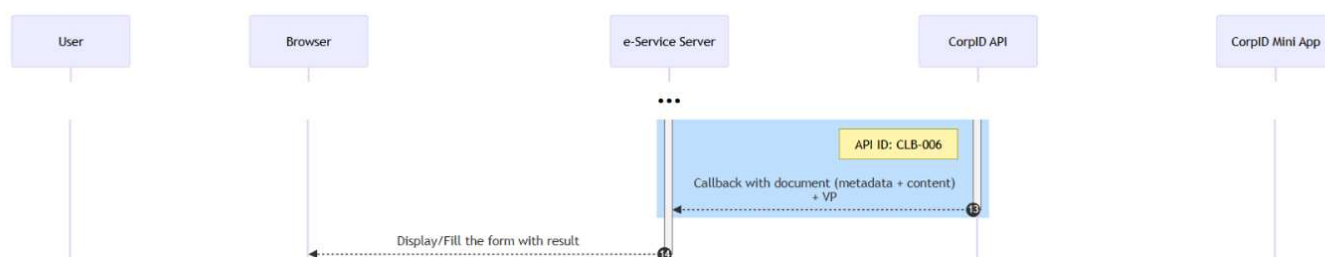
#### **Request and Response Parameters**

- Please refer to CorpID API Specification.

#### **Notes**

- For common parameters in request and response, please refer to CorpID API Specification.
- CorpID Server will not return the response parameter “ticketID” in this scenario.

### 3.9.1.2 Implementing (2): CLB-006: Callback to Receive Document Sharing



#### Pre-conditions

- CorpID user can accept, select document(s) and complete the document sharing request.
- CorpID user can also reject the document sharing request.

#### Post-conditions

- API data decryption is required.
- e-Service Server should match the callback result with corresponding e-Service Website/App using the “businessID”.
- “docList” will be returned in an array. Depends on the document type, the document content can be a base64 encoded file or a Verifiable Certificate. For detail and other parameters, please refer to CorpID API specification.

#### Error conditions

- For common errors, please refer to CorpID API Specification. Other errors of this API include:

| Error Code    | Error Description                | Suggested Action                                                                             |
|---------------|----------------------------------|----------------------------------------------------------------------------------------------|
| code - M80000 | user cancelled sharing request   | Inform user the CorpID document sharing request is cancelled                                 |
| code - M80001 | user rejected signing request    | Inform user the CorpID document sharing request is rejected                                  |
| code - M80002 | failed to request sharing        | Inform user the CorpID document sharing request is failed and retry later                    |
| code - M80003 | Document sharing request timeout | Inform user the CorpID document sharing request is timeout and provide way for user to retry |
| code - M88001 | failed to share                  | Inform user the CorpID document sharing request is failed and retry later                    |

#### Callback Parameters

- Please refer to CorpID API Specification.

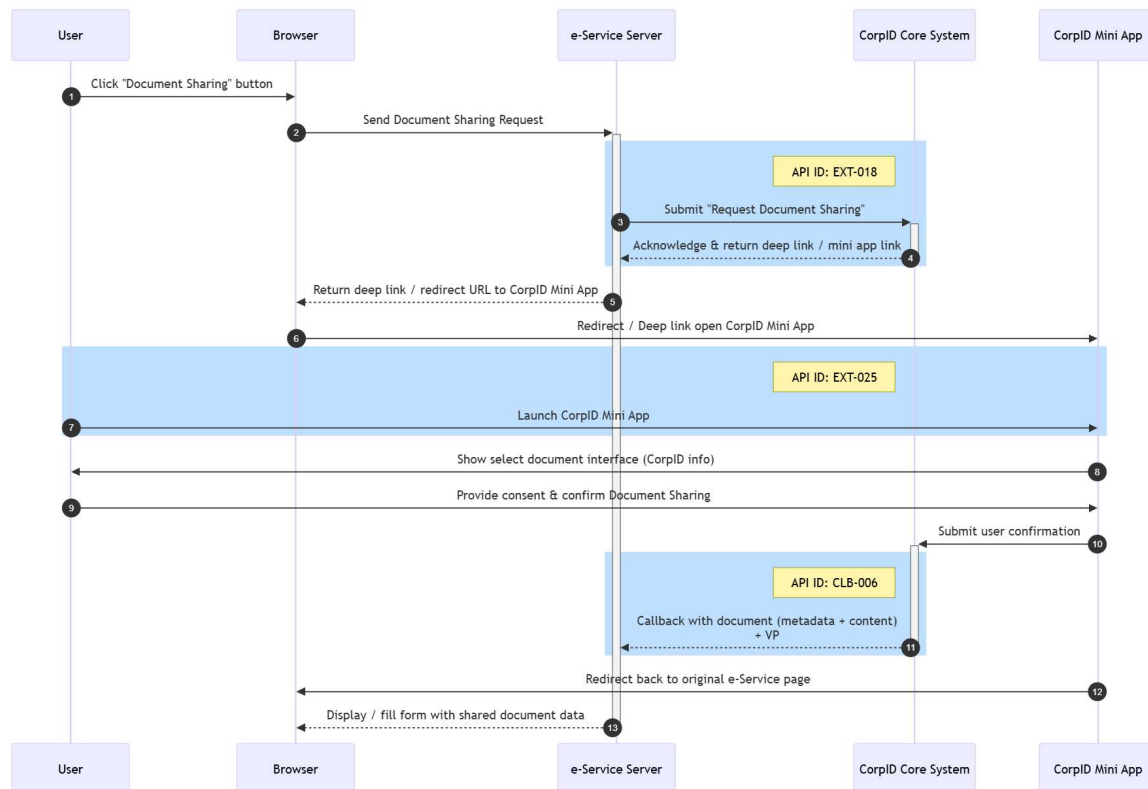
## Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- Callback parameter “docList”: Consists of list of documents. For each document, the “docContent” contains the “metadata”, “file” and “vc”. For detail please refer to CorpID API Specification.

### 3.9.2 Scenario 2: Document Sharing (e-Service Website in Same Device)

The sequence diagram below shows how an authenticated user authorises and share document(s) when e-Service website and the “iAM Smart” Mobile App are running in same device.

#### CorpID Document Wallet (e-Service Website in Same Device)



Step 1. The user clicks the “Document Sharing” button in the e-Service App.

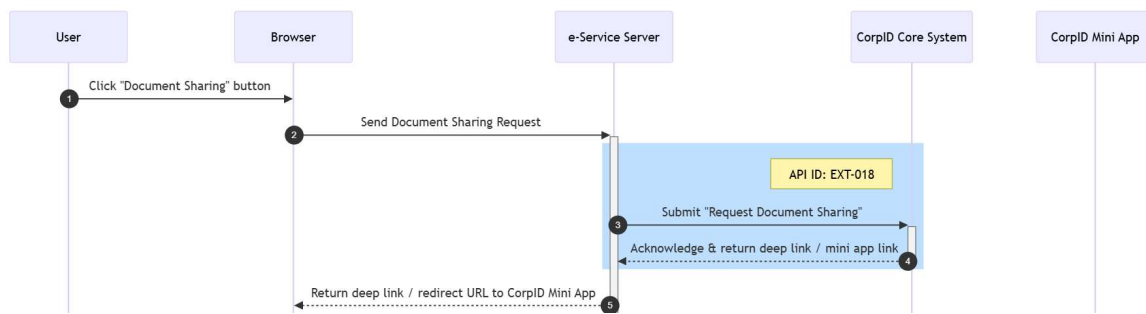
Step 2-3. The App sends a Document Sharing request to the e-Service Server. The e-Service Server submits “Request Document Sharing” request to the CorpID Server with the “businessID”, “cAccessToken”, “cOpenID”, “docType”, “docExpTs”, “redirectURI” and “state”, and “clientRedirectURI” for redirect to browser callback, etc.  
*CorpID API (EXT-024: Request Document Sharing (for App e-Service)).*

- Step 4-5. The CorpID API acknowledges the Document Sharing request and returns false for “authByQR” to indicate it is same device. e-Service Server return a redirect / deeplink to e-Service App
- Step 6-7. The CorpID API redirect / call deeplink to launch the “iAM Smart” mobile app and CorpID Mini-program.  
*CorpID API (EXT-025: Open CorpID Mini-program for Document Sharing).*
- Step 8. The Mini-program shows the select document interface with CorpID information.
- Step 9. The user select document(s) for sharing, provides consent and confirms the Document Sharing request.
- Step 10-11. The Mini-program submits the confirmation back through the flow. Under API ID: CLB-006, the system performs a callback with document data including metadata + content + VP and required parameters. The e-Service Server receives the result. The CorpID Mini-program launch and return to the e-Service App  
*CorpID API (CLB-006: Callback to Receive Document Sharing).*
- Step 12. CorpID Mini-program redirects back to original e-Service page in Browser.
- Step 13. Browser displays / fills form with shared document data.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No. | API Name                                      | API Reference (Section)               |
|-----|-----------------------------------------------|---------------------------------------|
| 1   | Request Document Sharing                      | CorpID API Specification Section 4.22 |
| 2   | Open CorpID Mini-program for Document Sharing | CorpID API Specification Section 4.28 |
| 3   | Callback to Receive Document Sharing          | CorpID API Specification Section 4.23 |

### 3.9.2.1 Implementing (1): EXT-018: Request Document Sharing



- Please refer to Section 3.9.1.1, except the following:

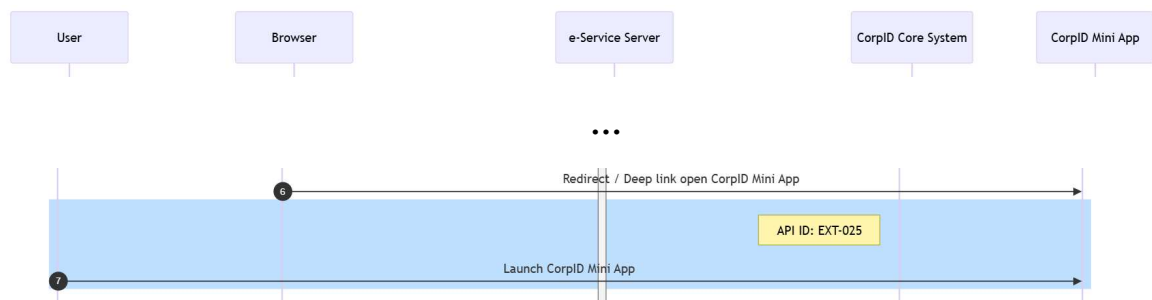
#### Post-conditions

- “ticketID” will be provided by CorpID System in this scenario.

#### Notes

- CorpID Server will return the response parameter “ticketID” in this scenario.

### 3.9.2.2 Implementing (2): EXT-025: Open CorpID Mini-program for Document Sharing



#### Pre-conditions

- “ticketID” is a unique identifier provided by CorpID System. e-service retrieves ticketID while requesting the document share. It is an ASCII string with a length of less than or equal to 36 chars.

#### Post-conditions

- e-Service Server construct the URL Scheme with required parameters and launch “iAM Smart” Mobile App

#### Error conditions

- For common errors, please refer to CorpID API Specification.

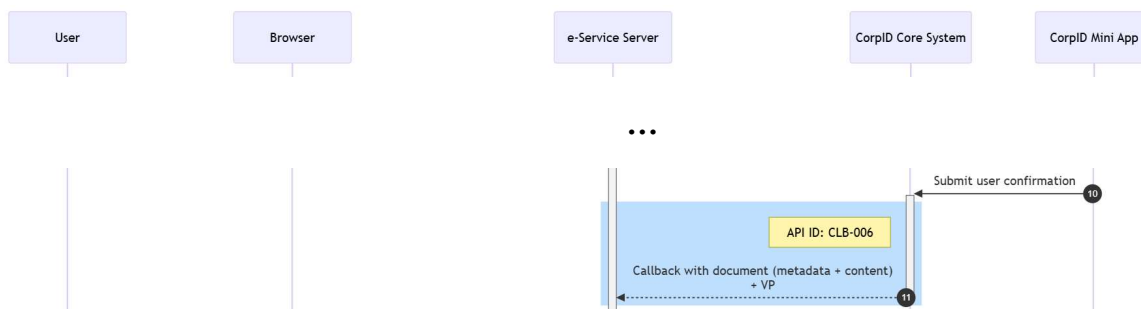
#### Callback Parameters

- Please refer to CorpID API Specification.

## Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- The URL scheme makes use of deep linking to redirect users to specific document share in CorpID Mini-program. The URL schemes are supported on iOS and Android versions of CorpID Mini-Program.

### 3.9.2.3 Implementing (3): CLB-006: Callback to Receive Document Sharing

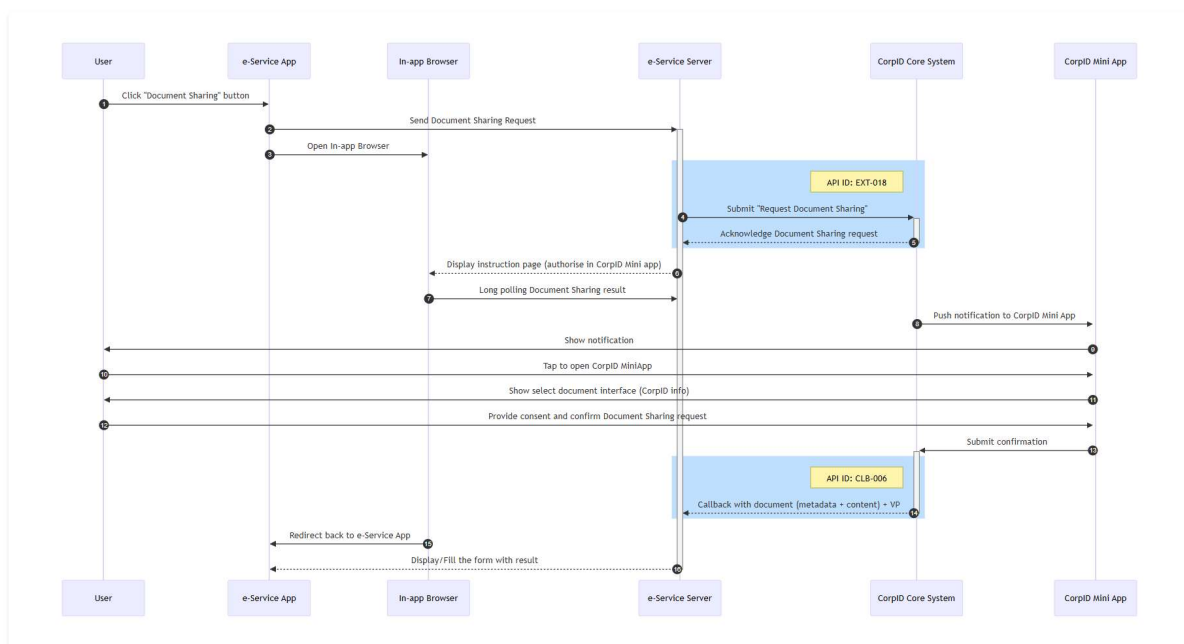


- Please refer to Section 3.9.1.2

### 3.9.3 Scenario 3: Document Sharing (e-Service App in Different Device)

The sequence diagram below shows how an authenticated user authorises and share document(s) when e-Service app and the “iAM Smart” Mobile App are running in different devices.

#### CorpID Document Wallet (e-Service App in Different Device)

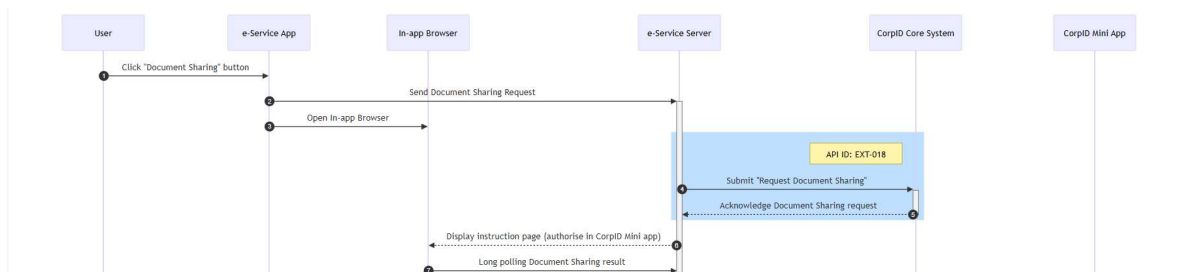


- Step 1. The user clicks the “Document Sharing” button in the e-Service App.
- Step 2-4. The App sends a Document Sharing request to the e-Service Server. The e-Service Server submits “Request Document Sharing” request to the CorpID Server with the “businessID”, “cAccessToken”, “cOpenID”, “docType”, “docExpTs”, “redirectURI” and “state”, etc.  
*CorpID API (EXT-024: Request Document Sharing (for App Sharing)).*
- Step 5. The CorpID API acknowledges the Document Sharing request and returns true for “authByQR” to indicate it is different device.
- Step 6-7. The e-Service App displays an instruction page telling the user to authorise in the CorpID Mini-program. The e-Service App performs long polling to check the Document Sharing result.
- Step 8. The CorpID API pushes a notification to the “iAM Smart” mobile app.
- Step 9-10. The user sees the notification and taps it to open the request in CorpID Mini-program.
- Step 11. The Mini-program shows the select document interface with CorpID information.
- Step 12. The user select document(s) for sharing, provides consent and confirms the Document Sharing request.
- Step 13-14. The Mini-program submits the confirmation back through the flow. Under API ID: CLB-006, the system performs a callback with document data including metadata + content + VP and required parameters. The e-Service Server receives the result.  
*CorpID API (CLB-006: Callback to Receive Document Sharing).*
- Step 15. In-app Browser redirects back to e-Service App.
- Step 16. e-Service App displays/fills the form with result.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No. | API Name                                      | API Reference (Section)               |
|-----|-----------------------------------------------|---------------------------------------|
| 1   | Request Document Sharing (for App e-Service)  | CorpID API Specification Section 4.27 |
| 2   | Open CorpID Mini-program for Document Sharing | CorpID API Specification Section 4.30 |
| 3   | Callback to Receive Document Sharing          | CorpID API Specification Section 4.23 |

### 3.9.3.1 Implementing (1): EXT-024: Request Document Sharing (for App Sharing)



#### Pre-conditions

- e-Service must possess a valid cAccessToken of the CorpID user with authorisation scope "Document Wallet".
- e-Service Website/App and "iAM Smart" Mobile App are in different devices in this scenario.
- e-Service can set the "docType" for e-Service to specific which document type it want.
  - For more than one document type required, use "," to separate.
  - If e-Service does not pass this parameter, CorpID would allow the CorpID user to select any of type of documents.
  - Detail document list will be provided in separate document.
- e-Service can set the "docExpTs" to specific the oldest document expiry range it want.
- The CorpID user who share the document with CorpID must have a "user\_submit" in "userPrivilege" during authentication or switching corporation's response.
- The CorpID user's company RP / AR / ADMIN should grant the document to share in correct folder that this CorpID user can access, or grant access right to user within valid time period and/or number of use.
- e-Service generates a "businessID" for this Digital Signing request and uses this identifier to match the callback result returned from CorpID System.
- The URL in "redirectURI" should be registered in CorpID Self-Service Portal's whitelist.
- The App-to-App callback parameters of "clientRedirectURI", "packageName", "activityClass", "activityParams" should be registered in CorpID Self-Service Portal's whitelist.
- If the "state" parameter is presented in the request message, the same state value will be returned to e-Service during the callback. It is used to prevent the CSRF attack. The value of state is defined by e-Service, and it should be a secure random string of any length less than or equal to 36 (UUID string length). Only ASCII letters, numbers, underscore and hyphens are accepted.
- API data encryption is required.

#### Post-conditions

- e-Service server should check the response parameter "authByQR" and "ticketID" and determine the next action. For details, please refer to CorpID API Specification. "ticketID" will not be provided by CorpID System in this scenario.

- e-Service Website/App should keep synchronising with e-Service Server for the document sharing result returned from CorpID System.
- API data decryption is required.

### Error conditions

- For common errors, please refer to CorpID API Specification.

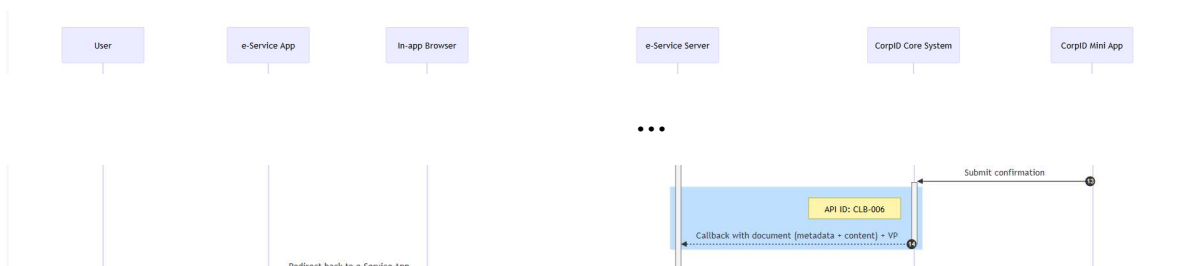
### Request and Response Parameters

- Please refer to CorpID API Specification.

### Notes

- For common parameters in request and response, please refer to CorpID API Specification.
- CorpID Server will not return the response parameter “ticketID” in this scenario.

### 3.9.3.2 Implementing (2): CLB-006: Callback to Receive Document Sharing

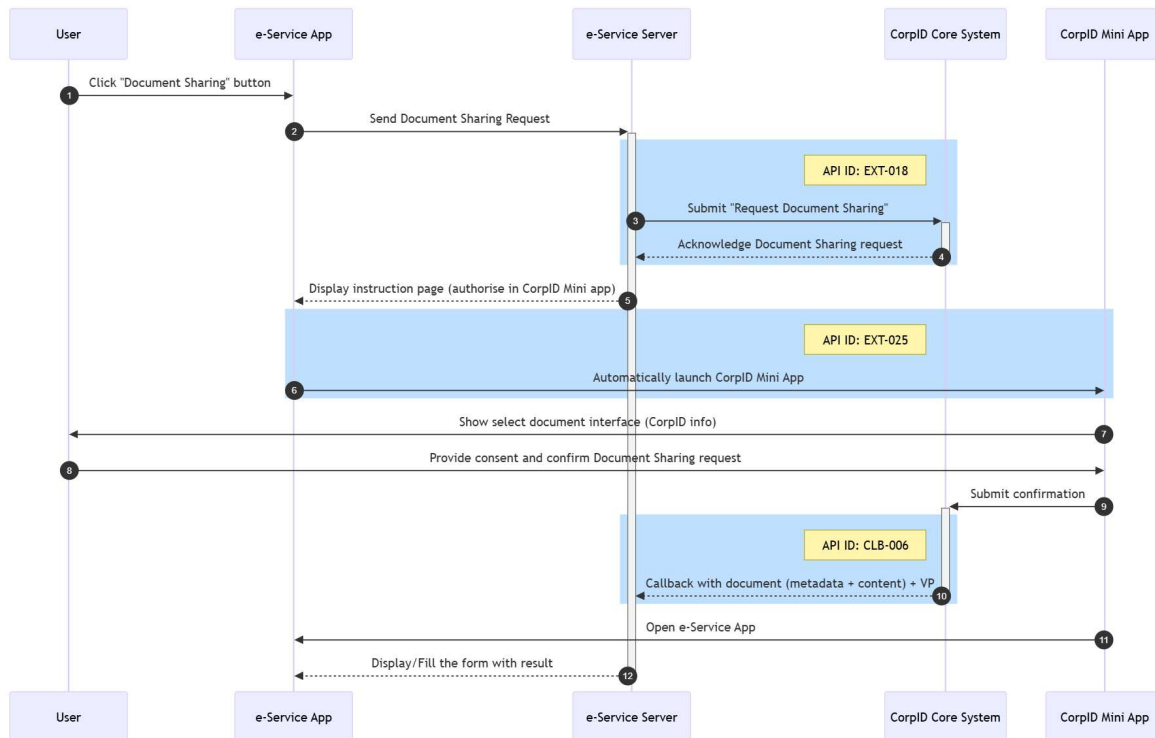


- Please refer to Section 3.9.1.2

### 3.9.4 Scenario 4: Document Sharing (e-Service App in Same Device)

The sequence diagram below shows how an authenticated user authorises and share document(s) when e-Service App and the “iAM Smart” Mobile App are running in same device.

## CorpID Document Wallet (e-Service App in Same Device)



- Step 1. The user clicks the “Document Sharing” button in the e-Service App.
- Step 2-3. The App sends a Document Sharing request to the e-Service Server. The e-Service Server submits “Request Document Sharing” request to the CorpID Server with the “businessID”, “cAccessToken”, “cOpenID”, “docType”, “docExpTs”, “redirectURI” and “state”, and “clientRedirectURI”, “packageName”, “activityClass”, “activityParams” for redirect to browser callback, etc.  
*CorpID API (EXT-024: Request Document Sharing (for App e-service)).*
- Step 4-5. The CorpID API acknowledges the Document Sharing request and returns false for “authByQR” to indicate it is same device. e-Service Server return a redirect / deeplink to e-Service App for the displays instruction page (authorise in CorpID Mini-program).
- Step 6. The CorpID API redirect / call deeplink to launch the “iAM Smart” mobile app and CorpID Mini-program.  
*CorpID API (EXT-025: Open CorpID Mini-program for Document Sharing).*
- Step 7. The Mini-program shows the select document interface with CorpID information.
- Step 8. The user select document(s) for sharing, provides consent and confirms the Document Sharing request.
- Step 9-10. The Mini-program submits the confirmation back through the flow. Under API ID: CLB-006, the system performs a callback with document data including metadata + content + VP and required parameters. The e-Service Server receives the result. The CorpID Mini-program launch and return to the e-Service App

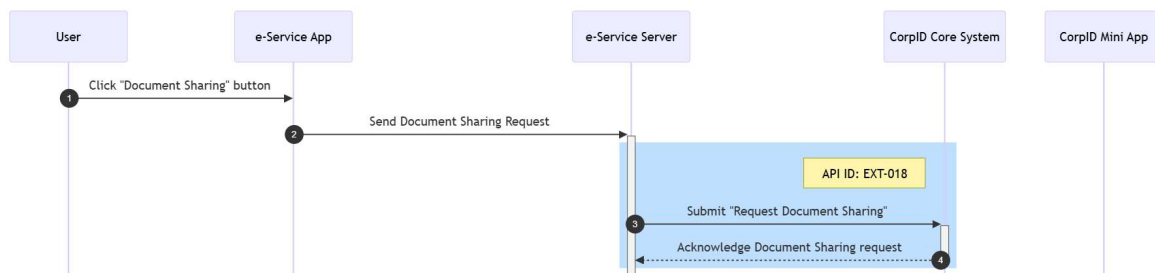
*CorpID API (CLB-006: Callback to Receive Document Sharing).*

- Step 11. CorpID Mini-program opens e-Service App.
- Step 12. e-Service App displays/fills the form with result.

API interactions between e-Service servers and CorpID System are listed below. Technical details of respective APIs can be found at the following sections.

| No. | API Name                                      | API Reference (Section)               |
|-----|-----------------------------------------------|---------------------------------------|
| 1   | Request Document Sharing (for App e-Service)  | CorpID API Specification Section 4.27 |
| 2   | Open CorpID Mini-program for Document Sharing | CorpID API Specification Section 4.28 |
| 3   | Callback to Receive Document Sharing          | CorpID API Specification Section 4.23 |

### 3.9.4.1 Implementing (1): EXT-024: Request Document Sharing (for App e-service)



- Please refer to Section 3.9.3.1, except the following:

#### Pre-conditions

- The URL in “clientRedirectURI”, “packageName”, “activityClass”, “activityParams” should be registered in CorpID Self-Service Portal’s whitelist.

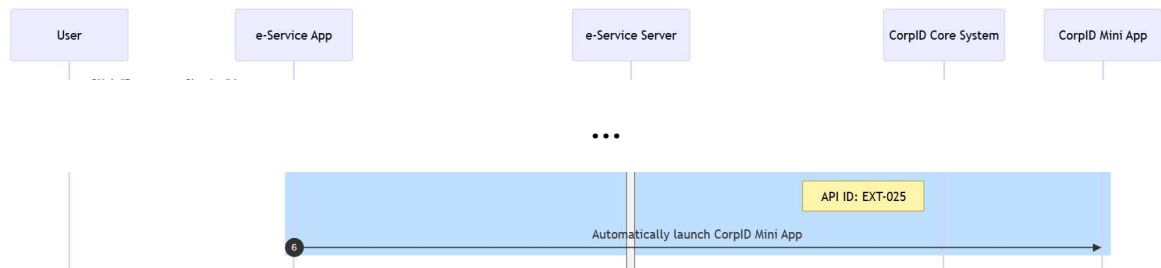
#### Post-conditions

- “ticketID” will be provided by CorpID System in this scenario.

#### Notes

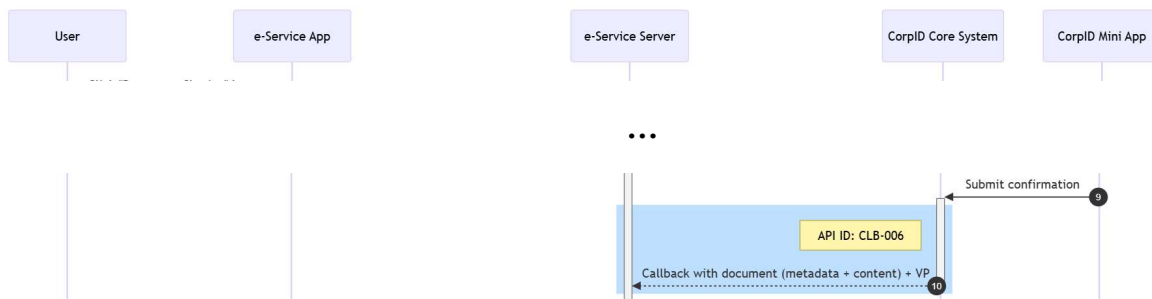
- CorpID Server will return the response parameter “ticketID” in this scenario.

### 3.9.4.2 Implementing (2): EXT-025: Open CorpID Mini-program for Document Sharing



Please refer to Section 3.9.2.2

### 3.9.4.3 Implementing (3): CLB-006: Callback to Receive Document Sharing



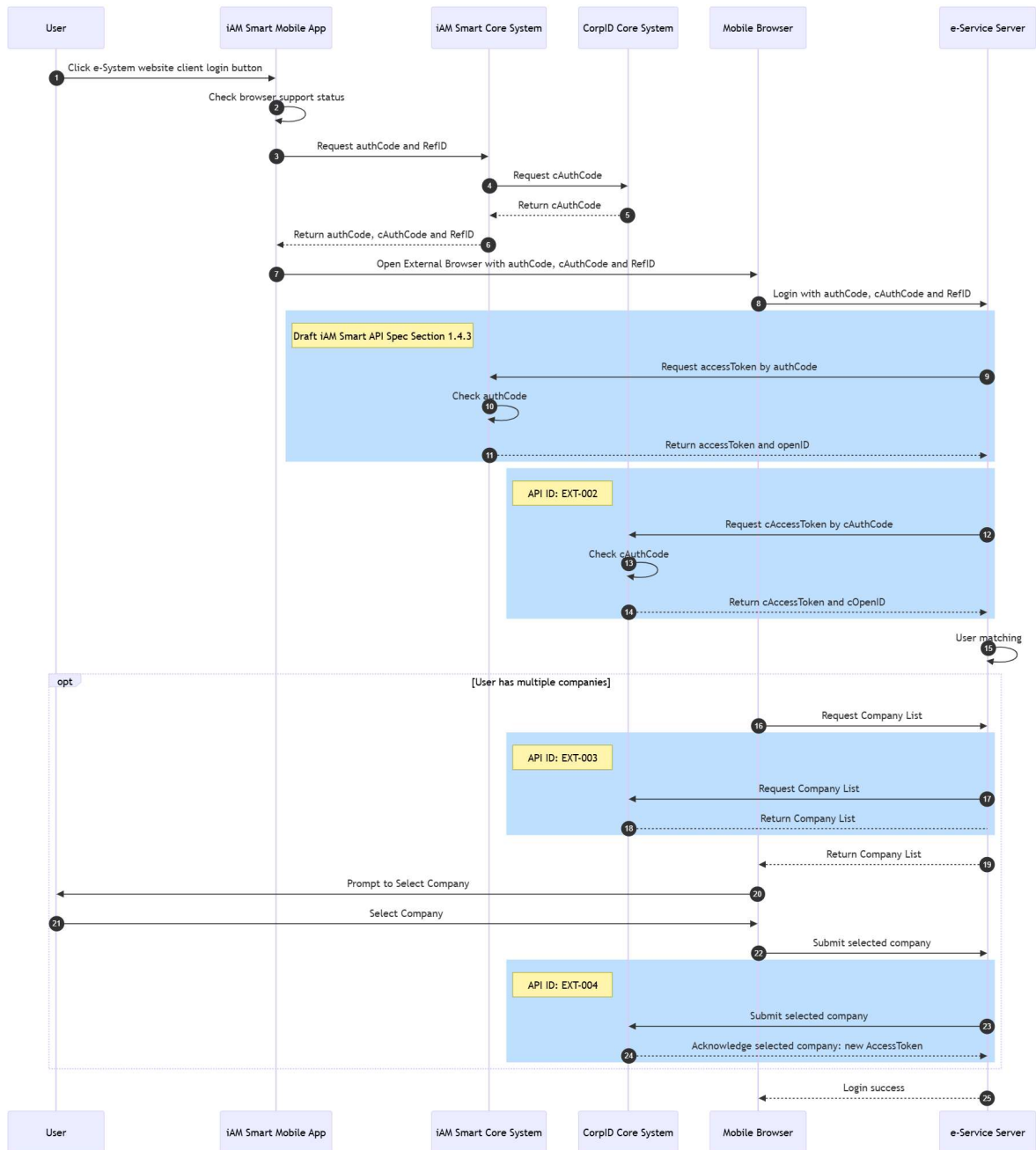
Please refer to Section 3.9.1.2

## 3.10 WORKFLOWS FOR DIRECT LOGIN & DIRECT ACCESS

### 3.10.1 Scenario 1: Direct Login with e-Service Website (Direct Login v2)

The sequence diagram below shows how users initiate Direct Login in the CorpID Mini-program and then automatically login e-Service website opened in a supported mobile browser.

## CorpID Authentication (e-Service Website in Same Device)



- Step 1. User opens “iAM Smart” Mobile App / CorpID Mini -App, browses e-Service catalogue or views the detailed information of e-Service, and clicks the button to invoke the e-Service;
- Step 2. CorpID Mini-program checks if the supported mobile browser and its browser version is installed. If there is no supported browser found, the default browser in the mobile device will be opened and redirected to the fallback URL that has been registered in the CorpID Platform.
- Step 3. iAM Smart Mobile App requests authCode and RefID from the iAM Smart Core System.

- Step 4. iAM Smart Core System requests cAuthCode from the CorpID Core System.
- Step 5. CorpID Core System returns cAuthCode to the iAM Smart Core System.
- Step 6. iAM Smart Core System returns authCode, cAuthCode, and RefID to the iAM Smart Mobile App.
- Step 7-8. “iAM Smart” Mobile -6. CorpID Mini-program opens the external browser, and requests the direct login callback endpoint with the cAuthCode and RefID according to the current UI display language. e-Service shall provide three direct login endpoints with three different languages (EN/TC/SC) and register in the CorpID Platform in advance. If there is any exception happened, CorpID Mini-program will open the fallback URL without the cAuthCode and RefID instead of the direct login URL. In addition, the three fallback URLs of e-Service for three languages (EN/TC/SC) should have been registered so that the CorpID Mini-program can decide which URL is for fallback with respect to the current UI display language. e-Service should check if any login session exists and perform session management properly according to the business needs;  
*Draft “iAM Smart” API for CorpID (GET: Callback with cAuthCode to Online Service Server (Direct Login v2 to Web))*
- Step 9-11. When the e-Service Server gets the cAuthCode, it should invoke the CorpID API to obtain the cAccessToken and the tokenised ID (cOpenID) of the CorpID user for the selected e-Service. API data encryption is required;  
*“iAM Smart” API (POST: Request accessToken & tokenised ID)*
- Step 12-14. When the e-Service Server gets the cAuthCode, it should invoke the CorpID API to obtain the cAccessToken and the tokenised ID (i.e., cOpenID) of the CorpID user for the selected e-Service. API data encryption is required;  
*CorpID API (EXT-002: Get Access Token)*
- Step 15. The e-Service Server performs user matching;
- Step 16-20. If corpcount return from CorpID “Get Access Token” API is larger than 1, the user has more than 1 corporate associated. The e-Service should call “Get Corporate List” API with cAccessToken and cOpenID. This API gets the list of corporation profiles associated with the logged-in CorpID user. The e-Service uses the result to show selectable corporation options and let CorpID user to choose which corporate profile to use in this session;  
*CorpID API (EXT-003: Get Corporation List)*
- Step 21-24. The user selects a corporate profile in the browser. The browser submits the selected company to the e-Service Server. The e-Service Server submits the selected corporate “ubi” to the corpID Core System and retrieve the updated cAccessToken and cOpenID, new user permission, new role, and other parameters of the corporation selected.  
*CorpID API (EXT-004, Request Token Exchange for Switching Corporation with UBI)*
- Step 25. The e-Service Website shows the login result (e.g., successful when Tokenised ID (cOpenID) matched with a user account of e-Service).

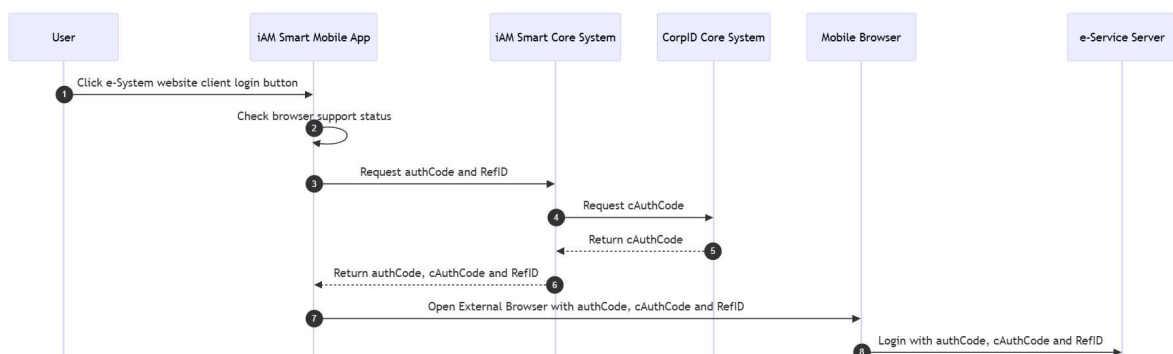
**Notes:**

e-Services shall provide the following URLs to DPO for setup:

| No. | URL Type                                       | URL Description                                                                                                                                                                                                                    | Described in       |
|-----|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| 1   | Direct Login (English)                         | Receive authorisation code of web direct login with English language                                                                                                                                                               | Step 7-8           |
| 2   | Direct Login (Traditional Chinese)             | Receive authorisation code of web direct login with Traditional Chinese language                                                                                                                                                   | Step 7-8           |
| 3   | Direct Login (Simplified Chinese)              | Receive authorisation code of web direct login with Simplified Chinese language                                                                                                                                                    | Step 7-8           |
| 4   | Fallback / Direct Access (English)             | Provide fallback for web direct login with English language when there is no supported browser, i.e., e-Service login page; or provide direct access to the e-Service in English without login requirement                         | Step 2<br>Step 7-8 |
| 5   | Fallback / Direct Access (Traditional Chinese) | Provide fallback for web direct login with Traditional Chinese language when there is no supported browser, i.e., e-Service login page; or provide direct access to the e-Service in Traditional Chinese without login requirement | Step 2<br>Step 7-8 |
| 6   | Fallback / Direct Access (Simplified Chinese)  | Provide fallback for web direct login with Simplified Chinese language when there is no supported browser, i.e., e-Service login page; or provide direct access to the e-Service in Simplified Chinese without login requirement   | Step 2<br>Step 7-8 |

### 3.10.1.1 Implementing (1) GET: An e-Service endpoint to receive authCode & RefID from external browser

#### CorpID Authentication (e-Service Website in Same Device)



This endpoint is to be implemented by e-Service. The implementation uses “Callback with cAuthCode to e-Service Server”.

#### **Pre-conditions**

- CorpID user has initiated e-Service website via direct login in CorpID Mini-program. CorpID Mini-program has verified that there is a supported browser installed in the same mobile phone.

#### **Post-conditions**

- cAuthCode will expire in 1 minute and e-Service can only use the cAuthCode once for requesting the cAccessToken from CorpID System.

authCode will expire in 1 minute and e-Service can only use the authCode once for requesting the accessToken from “iAM Smart” System.

#### **Error conditions**

- This API is a HTTP GET request. If parameter “error\_code” is returned, it means the direct login request is failed.

#### **Callback Parameters**

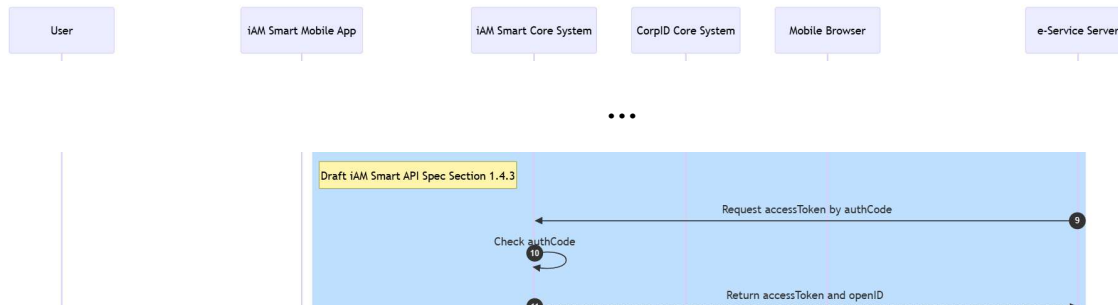
- Please refer to Draft “iAM Smart” API for CorpID.

#### **Notes**

- Callback parameter “cCode”: It is the cAuthCode for requesting the accessToken from CorpID System. Its validity lasts for 1 minute and can only be used once.
- Callback parameter “code”: It is the authCode for requesting the accessToken from “iAM Smart” System. Its validity lasts for 1 minute and can only be used once.
- If errors occur, “error\_code” instead of “code” / “cCode” will be returned.
- RefID is a unique value for troubleshooting when necessary.
- e-Service should check if any login session exists and perform session management properly according to business needs.

### 3.10.1.2 Implementing (2) POST: Request accessToken & tokenised ID

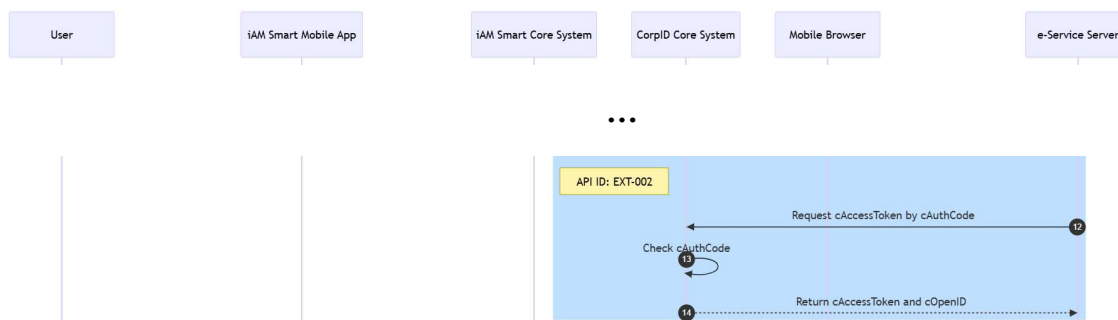
#### CorpID Authentication (e-Service Website in Same Device)



Please refer to Section 3.4.1.3.

### 3.10.1.3 Implementing (3): EXT-002: Get Access Token

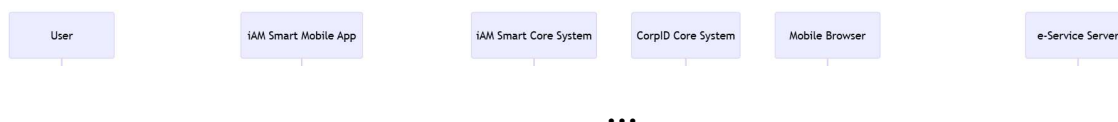
#### CorpID Authentication (e-Service Website in Same Device)

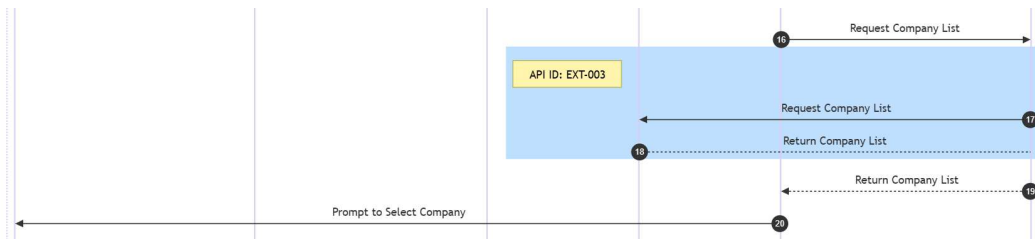


Please refer to Section 3.4.1.4.

### 3.10.1.4 Implementing (4): EXT-003: Get Corporation List

#### CorpID Authentication (e-Service Website in Same Device)

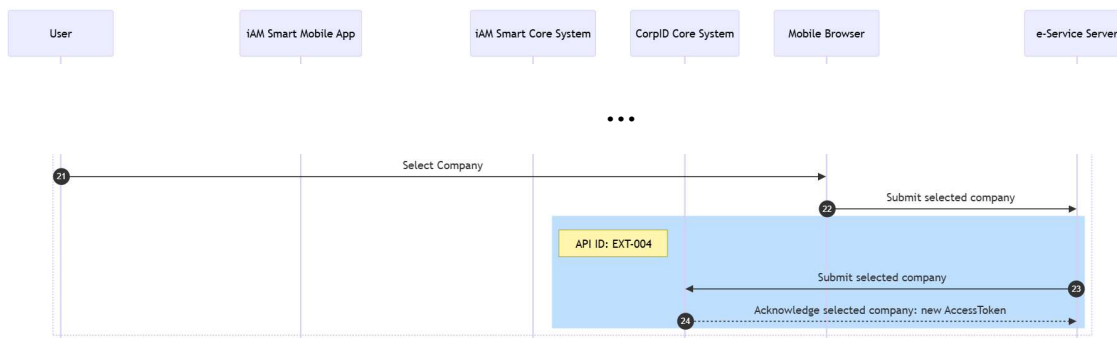




Please refer to Section 3.4.1.5.

### 3.10.1.5 Implementing (5): EXT-004, Request Token Exchange for Switching Corporation with UBI

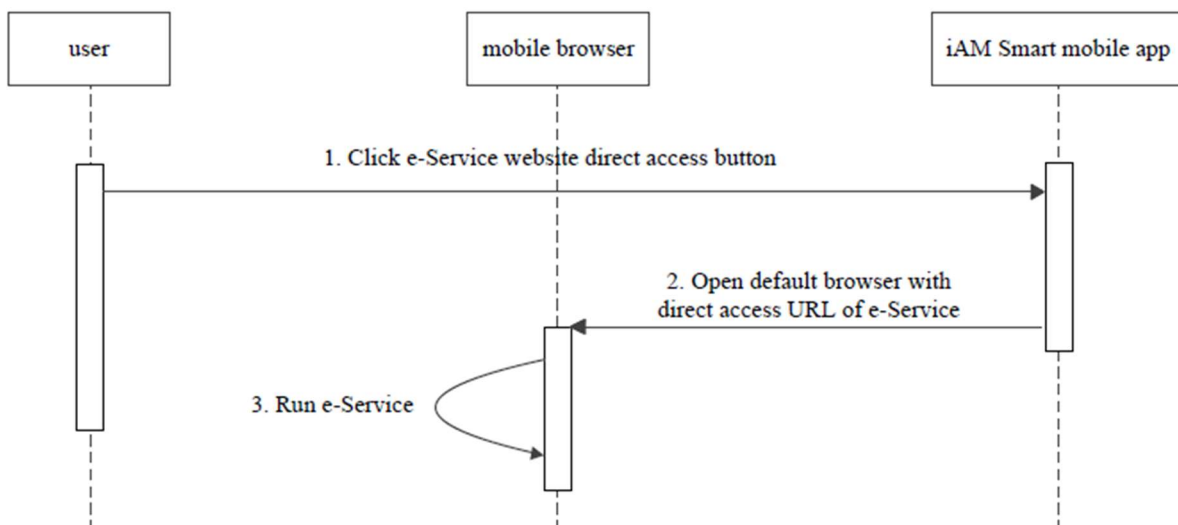
#### CorpID Authentication (e-Service Website in Same Device)



Please refer to Section 3.4.1.6.

### 3.10.2 Scenario 2: Direct Access with e-Service Website

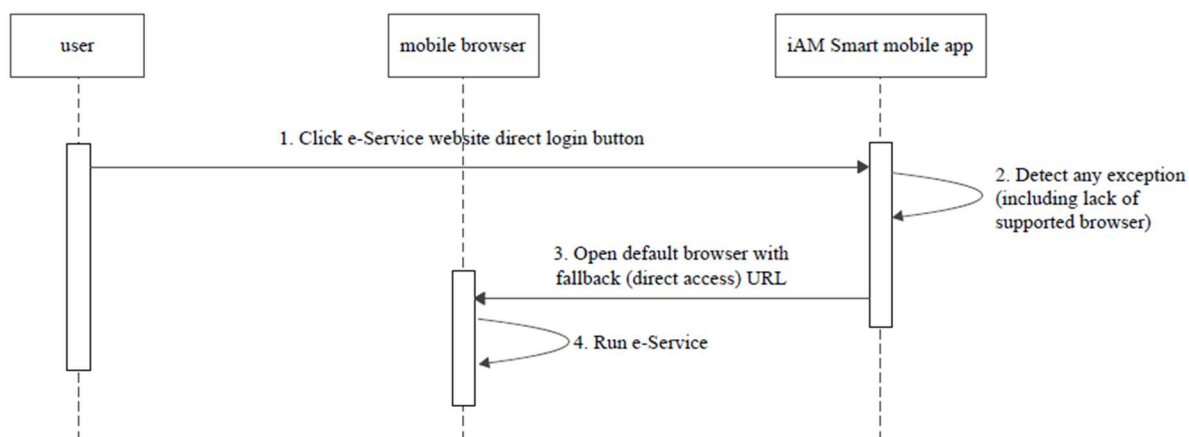
The sequence diagram below shows how users initiate Direct Access in the CorpID Mini-program.



- Step 1. User opens “iAM Smart” Mobile App / CorpID Mini-program, browses e-Service catalogue or views the detailed information of e-Service and clicks the button to invoke the e-Service.
- Step 2. “iAM Smart” Mobile App opens the default browser with direct access URL of e-Service. The three direct access URLs (or fallback URLs, for the English, Traditional Chinese, and Simplified Chinese languages) for the e-Service must be registered in advance in the CorpID and “iAM Smart” Platform. “iAM Smart” Mobile App chooses the URL based on its current UI display language.
- Step 3. User starts using e-Service website in the default browser.

### 3.10.3 Scenario 3: Direct Login with e-Service Website (Fallback Mechanism)

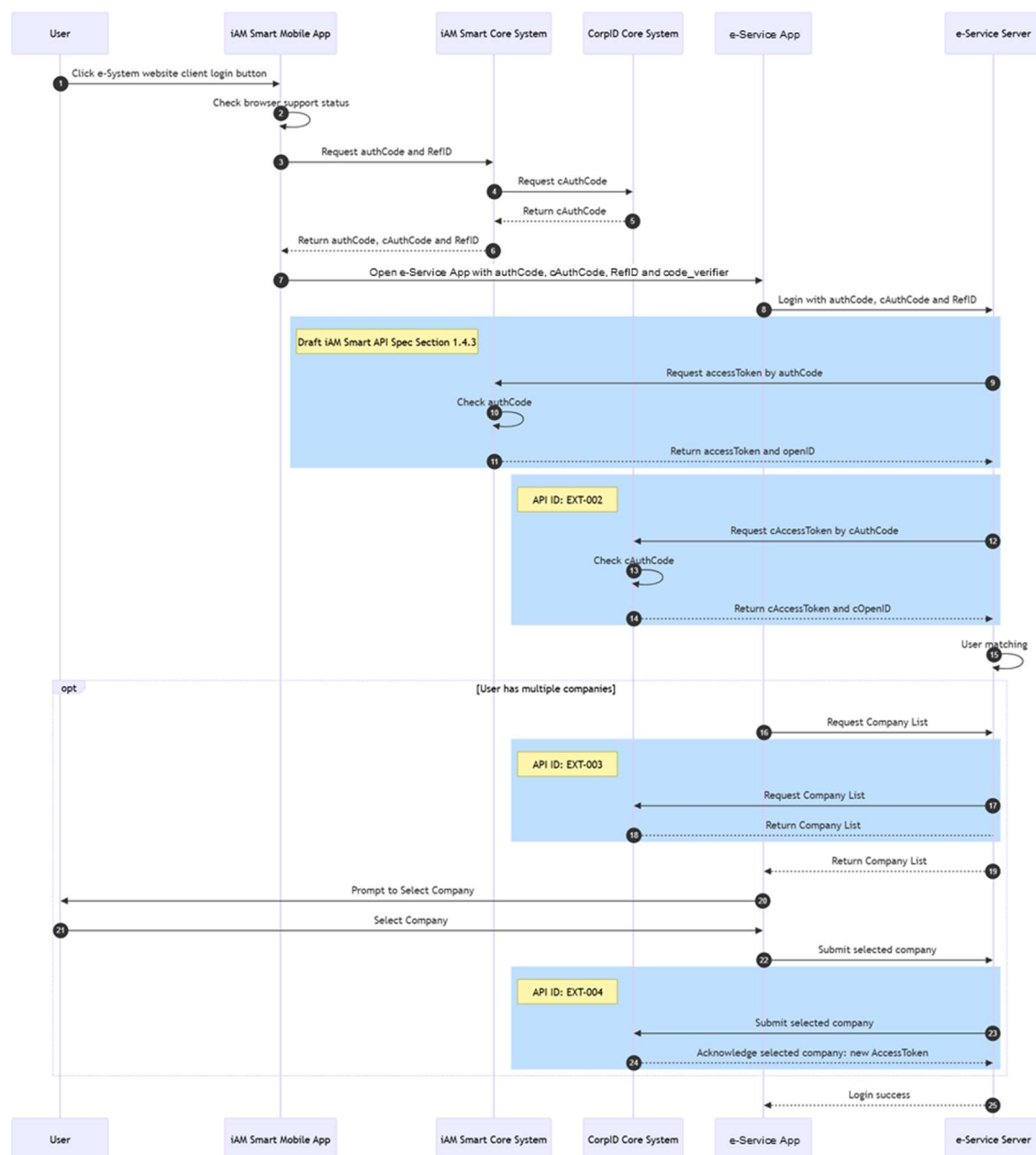
The sequence diagram below shows how users initiate Direct Login in that CorpID Mini-program but fallback to direct access because there is no supported browser or exception happens.



- Step 1. User opens “iAM Smart” Mobile App / CorpID Mini-program, browses e-Service catalogue or views the detailed information of e-Service and clicks the button to invoke the e-Service.
- Step 2. “iAM Smart” Mobile App checks if any exception is detected. For example, “iAM Smart” Mobile App checks and does not find browser that support the direct login process.
- Step 3. “iAM Smart” Mobile App opens the fallback URL with the default browser. The three fallback URLs (or direct access URLs, for the English, Traditional Chinese, and Simplified Chinese languages) for the e-Service must be registered in advance in the “iAM Smart” Platform. “iAM Smart” Mobile App chooses the URL based on its current UI display language.
- Step 4. User starts using e-Service website with the default browser.

### 3.10.4 Scenario 4: Direct Login with e-Service App (Direct Login v2)

The sequence diagram below shows how users initiate Direct Login in the CorpID Mini-program and then automatically log in to the e-Service app installed in the same mobile device.



#### Pre-conditions

The e-Service App is registered to CorpID and iAM Smart team with the following details,

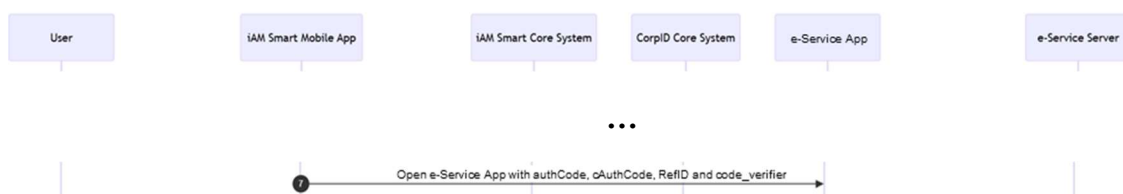
- Basic information, including the service name, scope
- Package name (Submission should be made 3 months in advance and should be registered in iAM Smart App)

- iOS Universal Link and Android app fingerprint of the signing certificate correspond to package name in ESP

- Step 1. User opens “iAM Smart” Mobile App / CorpID Mini-program, browses e-Service catalogue or views the detailed information of e-Service and clicks the “(Direct Login) App” button;
- Step 2. iAM Smart Mobile App checks browser support status.
- Step 3. iAM Smart Mobile App requests authCode and RefID from the iAM Smart Core System.
- Step 4. iAM Smart Core System requests cAuthCode from the CorpID Core System.
- Step 5. CorpID Core System returns cAuthCode to the iAM Smart Core System.
- Step 6. iAM Smart Core System returns authCode, cAuthCode, and RefID to the iAM Smart Mobile App.
- Step 7. “iAM Smart” Mobile App invokes e-Service App with required parameters using Universal Link (iOS) or package name (Android) according to the operating system;  
*Draft “iAM Smart” API for CorpID (URL scheme: Callback with cAuthCode and code\_verifier to Online Service App (Direct Login v2 to App))*  
 For iOS platform, the system browser will open the Universal Link and forward the request to e-Service server. Upon receiving the browser request, e-Service is recommended to instruct user to download and install the e-Service App.  
 For Android platform, “iAM Smart” will validate the e-Service App. If the e-Service App is installed and both the fingerprint of signing certificate and package name are valid, it will be launched by package name and the login request data, including the authCode and code\_verifier will be forwarded. Otherwise, an alert dialog will be shown, the user can open the fallback URL via system browser by choosing the “More” option. Upon receiving the browser request, e-Service is recommended to instruct the user to download and install the e-Service App.  
 Please note that the e-Service app must check if another login session is already exists and perform session management properly according to the business needs. e-Service is strongly recommended to terminate the existing login session before it proceeds with the rest of the Direct Login workflow. The e-Service is recommended to ask the user if he/she confirms to logout the existing session and use “iAM Smart” Mobile App to login.
- Step 8. e-Service App sends cAuthCode, authCode, code\_verifier, etc. to e-Service Server;
- Step 9-11. e-Service Server provides the authCode and code\_verifier to invokes the “iAM Smart” API to obtain the accessToken and the tokenised ID (i.e., openID) of the “iAM Smart” user with the selected e-Service. API data encryption is required.  
*“iAM Smart” API (POST: Request accessToken & Tokenised ID)*

- Step 12-14. e-Service Server provides the cAuthCode and code\_verifier to invokes the CorpID API to obtain the cAccessToken and the tokenised ID (i.e., cOpenID) of the CorpID user with the selected e-Service. API data encryption is required.  
*CorpID API (EXT-002: Get Access Token)*
- Step 15. e-Service then performs user matching using the openID with its user repository and determines the result of this login request;
- Step 16-20. If corpcount return from CorpID “Get Access Token” API is larger than 1, the user has more than 1 corporate associated. The e-Service should call “Get Corporate List” API with cAccessToken and cOpenID. This API gets the list of corporation profiles associated with the logged-in CorpID user. The e-Service uses the result to show selectable corporation options and let CorpID user to choose which corporate profile to use in this session;  
*CorpID API (EXT-003: Get Corporation List)*
- Step 21-24. The user selects a corporate profile in the browser. The browser submits the selected company to the e-Service Server. The e-Service Server submits the selected corporate “ubi” to the corpID Core System and retrieve the updated cAccessToken and cOpenID, new user permission, new role, and other parameters of the corporation selected.  
*CorpID API (EXT-004, Request Token Exchange for Switching Corporation with UBI)*
- Step 25. e-Service App shows the login result (e.g., successful when Tokenised ID (cOpenID) is matched with a user account of e-Service).

#### 3.10.4.1 Implementing (1) Callback with cAuthCode and code\_verifier to E-Service App (Direct Login v2 to App)



##### Pre-conditions

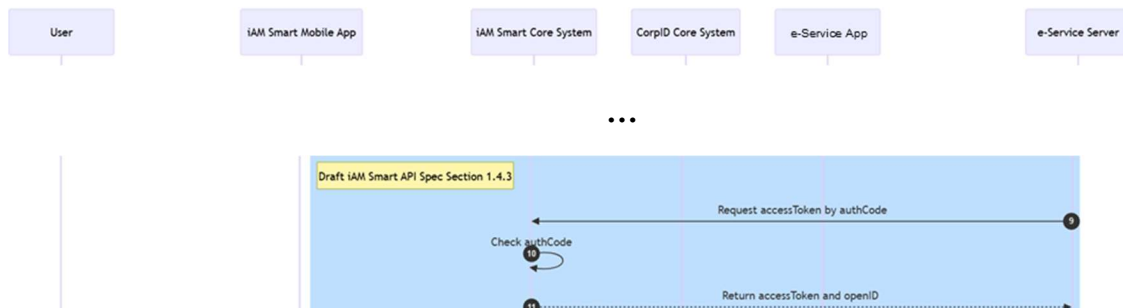
- CorpID user has initiated e-Service app via direct login in “iAM Smart” Mobile App.
- e-Service has set up Universal Link and/or Android package name with fingerprint of signing certificate for the e-Service App.

##### Post-conditions

- cAuthCode will be expired in 1 minute and e-Service can only use the cAuthCode once for requesting the cAccessToken from CorpID System.

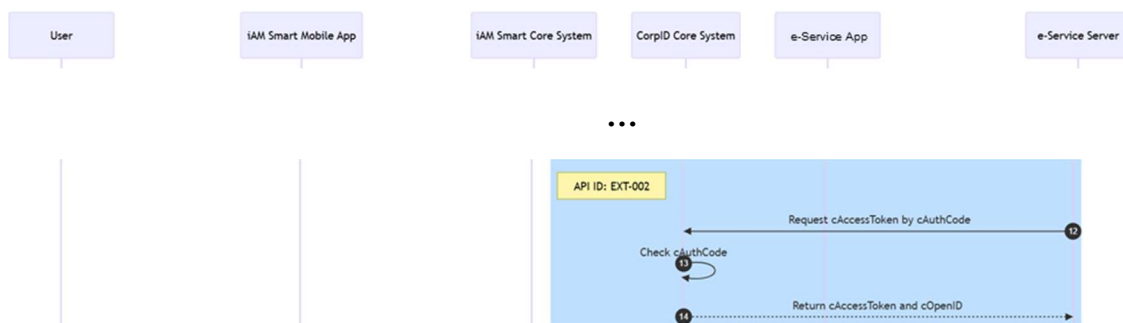
- authCode will be expired in 1 minute and e-Service can only use the authCode once for requesting the accessToken from “iAM Smart” System.

#### 3.10.4.2 Implementing (2) POST: Request accessToken & tokenised ID



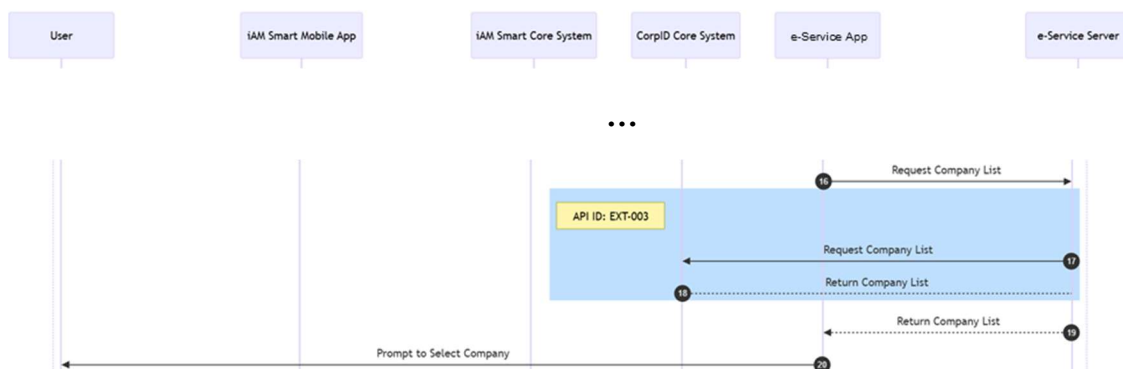
Please refer to Section 3.4.1.3.

#### 3.10.4.3 Implementing (3): EXT-002: Get Access Token



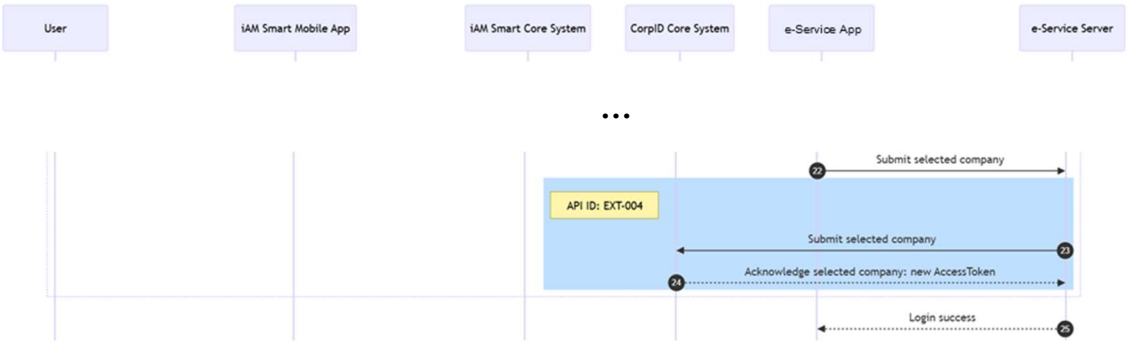
Please refer to Section 3.4.1.4.

#### 3.10.4.4 Implementing (4): EXT-003: Get Corporation List



Please refer to Section 3.4.1.5.

**3.10.4.5 Implementing (5): EXT-004, Request Token Exchange for Switching Corporation with UBI**



- Please refer to Section 3.4.1.6.

## 4. APPENDIX

### A. REFERENCE APPLICATIONS

#### A.1 Overview

The Reference Application (“RA”) is a set of sample e-Service programs to demonstrate how to make use of CorpID APIs to provide functions such as authentication, form pre-filling and digital signing. The RA consists of a java backend program (rm-backend-main) and a vue frontend (rm-frontend-main).

##### rm-backend-main

It is an integration layer for the CorpID and iAM Smart platforms. It orchestrates OAuth 2.0 flows, encrypts/signs API requests via HMAC-SHA256 + AES-256-GCM, and provides REST endpoints for authentication, corporate profile listing, form pre-filling, document wallet sharing, and digital signing.

##### rm-frontend-main

It is a Vue 3 + Vite 8 SPA that demonstrates the CorpID e-Service user journey. It implements a 3-step wizard (applicant info → business info/doc sharing → confirm & sign), plus anonymous flows for form filling, digital signing, and PDF signing. It uses Vue Router, native fetch(), sessionStorage for step data, and a polling utility for async CorpID operations.

The RA is intended only for a reference of application development and it is not a mandatory way of implementation nor a standard package to be included in the e-Service Application. Not all return codes of CorpID APIs are processed by the RA. e-Services should design and implement their own business logic and handle different CorpID API return codes according to their business requirements. e-Services developers must ensure that their applications should be implemented with sufficient security measures on handling the data, according to their departmental security policy as well as their application security requirements.

#### A.2 Software Directory Structure

The “rm-backend-main” contains a Springboot 3.5.13 application written in Java JDK 17. These components can be found in the following directories of the RA package:

```
rm-backend-main/
├── pom.xml                # Root Maven POM (Spring Boot parent)
├── Dockerfile             # Multi-stage Docker build
├── .gitlab-ci.yml         # GitLab CI pipeline
├── mvnw / mvnw.cmd       # Maven wrapper
├── README.md
├── src/                  # Main CorpID RM Backend application
│   └── main/
│       ├── java/hk/gov/corpid/eserv/rm/
│       │   ├── CorpIdRmBackendApplication.java # @SpringBootApplication entry point
│       │   ├── async/                        # Async flow store (file-based polling store)
│       │   ├── client/                      # Client interfaces (CorpID, iAM Smart APIs)
│       │   ├── clienthttp/                 # RestTemplate HTTP implementations + interceptors
│       │   └── clientmock/                 # Mock clients for local dev
```

```

├── config/           # CORS, client config beans
├── controller/       # 11 REST controllers
├── dto/              # Request/response DTOs
├── exception/        # Custom exceptions + global handler
├── security/         # JWT, AES/RSA encryption, CEK management
├── service/          # Service interfaces + implementations
├── util/             # PDF digest, signing digest, HKID hash, etc.
├── resources/        # application.yml, profiles, keystores
├── test/            # Unit tests
├── application/      # Separate demo "eservice.demo" Spring Boot app
│   ├── pom.xml
│   ├── src/main/java/eservice/demo/
│   │   ├── WebApplication.java
│   │   ├── config/
│   │   └── controller/test/
│   │       # Security, Jackson, Swagger, global handlers
│   │       # Test health-check controller
├── common/           # Shared common module (eservice.demo)
│   ├── pom.xml
│   ├── src/main/java/eservice/demo/
│   │   ├── constants/CodeMessage.java # Error code enum
│   │   ├── exception/                 # BusinessException, AccessDeniedException
│   │   ├── mapper/MapperUtils.java    # JSON mapper utility
│   │   ├── model/ApiResponse.java     # Generic API response record
│   │   └── test/TestService.java      # Demo test service
├── model/            # Simple model module (eservice.demo)
│   ├── pom.xml
│   └── src/main/java/eservice/demo/model/
│       ├── TestReqModel.java
│       └── TestResModel.java

```

The “rm-frontend-main” Vue 3 SPA has a static assets + SDK scripts in public/, Vue source in src/, environment configs at the root, and Docker/Nginx deployment configs in docker/. The 8 page components map directly to 8 Vue Router routes, and the 3-step wizard flow (applicant info → business info → confirm & sign) uses sessionStorage for cross-step state.

```

rm-frontend-main/
├── index.html          # Vite entry HTML
├── package.json        # Dependencies (Vue 3, Vue Router 4, Vite 8)
├── vite.config.js      # Vite config (HTTPS on :8443, @ alias)
├── Dockerfile          # Multi-stage Docker build
├── README.md
├── .env.local          # Environment: local / DCI dev
├── .env.sit            # Environment: SIT
├── .env.isit          # Environment: iSIT (staging)
├── .gitlab-ci.yml      # GitLab CI pipeline
├── .vscode/extensions.json # Recommends Vue Volar extension
├── certs/
│   ├── frontend-cert.pem # Local dev SSL cert
│   └── frontend-key.pem  # Local dev SSL key
├── docker/
│   ├── nginx.conf       # Nginx config (Nexify deployment)
│   ├── nginx-dci-dev.conf # Nginx config (DCI Dev)
│   ├── nginx-dci-isit.conf # Nginx config (DCI iSIT)
│   ├── nginx-dci-uat.conf # Nginx config (DCI UAT)
│   ├── dci-ARMDockerfile # ARM64 DCI Dockerfile
│   └── dci-Dockerfile    # x86 DCI Dockerfile
├── public/
│   ├── favicon.svg
│   └── icons.svg

```

```

├── corpid/eservice/index.html # Standalone "Select Corp" page (vanilla JS)
├── libs/js/
│   ├── init.js                # Dynamic adaptor loader
│   ├── index.js               # SDK demo page scripts
│   ├── adaptor-mini-program.js # Mini-program bridge
│   ├── adaptor-web-page.js    # Web page localStorage fallback
│   ├── adaptor-mock.js        # Mock data adaptor
│   ├── tcmpp-jssdk-1.0.0.js   # Tencent Cloud Mini Program SDK
│   ├── tcsas-jssdk-1.0.1.js  # Tencent Cloud Service App SDK
│   └── uni.webview.1.5.8.js   # UniApp WebView bridge SDK
├── src/                        # ★ Main application source
│   ├── main.js                # App bootstrap (createApp + router + mount)
│   ├── App.vue                # Root component (<router-view />)
│   └── style.css               # Global stylesheet (~2188 lines)
│   ├── router/index.js        # 8 Vue Router routes
│   ├── components/
│   │   └── LoginButton.vue    # Reusable login button
│   ├── pages/
│   │   ├── LoginPage.vue      # Landing / login page
│   │   ├── SelectCorpPage.vue # Corp selection after login
│   │   ├── ApplicantInfoPage.vue # Step 1: Applicant info form
│   │   ├── BusinessInfoPage.vue # Step 2: Business info + doc sharing
│   │   ├── ConfirmSignPage.vue # Step 3: Confirm + sign
│   │   ├── FormFillingPage.vue # Anonymous form filling demo
│   │   ├── DigitalSigningPage.vue # Anonymous digital signing demo
│   │   └── PdfDigitalSigningPage.vue # Anonymous PDF signing demo
│   ├── utils/
│   │   ├── util.js            # URL param parser
│   │   ├── pollingUtil.js     # Async result polling helper
│   │   └── uniBridge.js       # UniApp JS bridge readiness
│   └── assets/
│       ├── corpid.png / corpid_v2.png / corpid_signed.png # CorpID icons
│       ├── iamsmart.png / verified.png / upload.jpg
│       └── mock/corpid-shared-doc.pdf # Mock doc for demo

```

### A.3 Implementation Reference

This section provides a directory of functions to be demonstrated by the RA and the corresponding location of the program code in the RA package.

There are 16 endpoints across 10 functional controllers:

1. AuthController.java — src/main/java/.../controller/AuthController.java

| Endpoint                      | Description                            |
|-------------------------------|----------------------------------------|
| POST /api/auth/getCorpIdToken | Exchange CorpID auth code for token    |
| GET /api/auth/iamsmart/login  | Initiate iAM Smart QR login (redirect) |

2. CallbackController.java — src/main/java/.../controller/CallbackController.java

| Endpoint               | Description                                                                      |
|------------------------|----------------------------------------------------------------------------------|
| GET /api/auth/callback | Universal OAuth callback (handles login, Form Pre-Filling, signing, PDF signing) |

### 3. CorpController.java — src/main/java/.../controller/CorpController.java

| Endpoint                                  | Description                                 |
|-------------------------------------------|---------------------------------------------|
| GET /api/corp/list                        | List corporations for authenticated user    |
| POST /api/corp/requestDocSharing          | Initiate document wallet sharing            |
| POST /api/corp/requestDocSharing/callback | Document sharing callback (decrypts result) |

### 4. FormFillingController.java — src/main/java/.../controller/FormFillingController.java

| Endpoint                                 | Description                                            |
|------------------------------------------|--------------------------------------------------------|
| POST /api/formFilling/initiateRequest    | Initiate authenticated Form Pre-Filling                |
| POST /api/formFilling/callback           | Form Pre-Filling callback (decrypts CorpID + iAM data) |
| POST /api/formFilling/anonymous/initiate | Initiate anonymous (no-auth) Form Pre-Filling          |

### 5. SigningController.java — src/main/java/.../controller/SigningController.java

| Endpoint                             | Description                                           |
|--------------------------------------|-------------------------------------------------------|
| POST /api/signing/initiateRequest    | Initiate authenticated digital signing                |
| POST /api/signing/callback           | Signing callback (decrypts result)                    |
| POST /api/signing/requestCorpSigning | Request corporate-level signing                       |
| POST /api/signing/anonymous/initiate | Initiate anonymous digital signing (multipart upload) |

### 6. PdfSigningController.java — src/main/java/.../controller/PdfSigningController.java

| Endpoint                                | Description                                       |
|-----------------------------------------|---------------------------------------------------|
| POST /api/pdfsigning/anonymous/initiate | Initiate anonymous PDF signing (multipart upload) |

### 7. AsyncFlowController.java — src/main/java/.../controller/AsyncFlowController.java

| Endpoint              | Description                               |
|-----------------------|-------------------------------------------|
| GET /api/async/result | Poll async operation result by businessID |

### 8. MockAuthController.java — src/main/java/.../controller/MockAuthController.java (local/sit only)

| Endpoint                  | Description                        |
|---------------------------|------------------------------------|
| POST /api/mock/auth/login | Mock login without real CorpID/iAM |

9. SignatureController.java — src/main/java/.../controller/SignatureController.java

| Endpoint                    | Description                            |
|-----------------------------|----------------------------------------|
| GET /api/v1/signature/debug | Debug HMAC-SHA256 signature generation |

10. TestController.java — src/main/java/.../controller/TestController.java

| Endpoint            | Description                |
|---------------------|----------------------------|
| GET /test/getKey    | Test CorpID CEK loading    |
| GET /test/getIamKey | Test iAM Smart CEK loading |

## B. STREAMLINE WORKFLOW SCENARIOS

The Streamline Workflow is one of the important enhancement to realise “single portal for online government services (一網通辦)”, aiming to provide the simplified and seamless CorpID

workflow with a view to enhancing user experience without extra switching back and forth between CorpID Mini-program and e-Services.

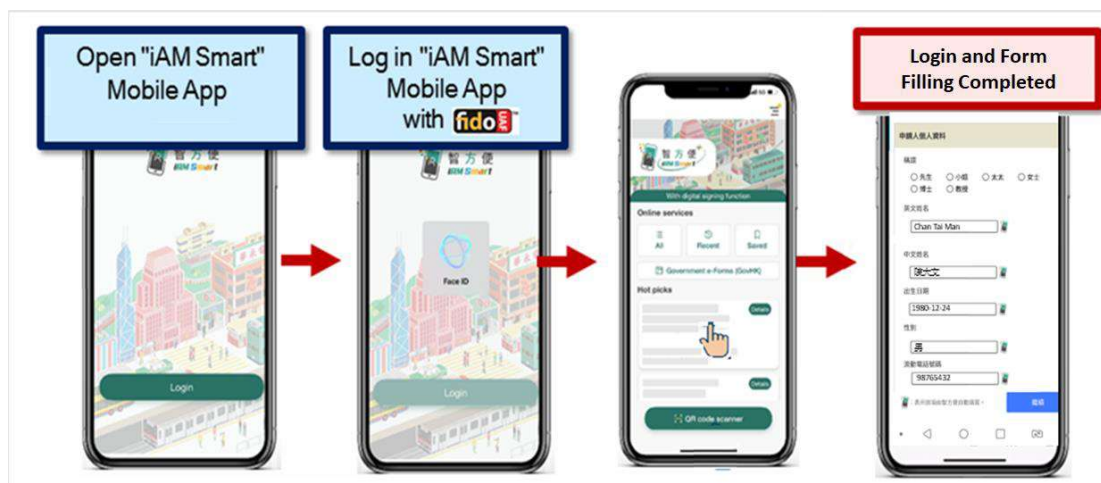
e-Services shall follow both “CorpID API Specification” and “CorpID Developer Guide” for CorpID Adoption. In addition, DPO has appointed a Consultant to gauge user needs, expectations and aspiration on the future single portal for online government services and will provide new “Reference System Flow and User Interface Specifications” for reference in late 2023.

The below examples illustrate how the Streamline Workflow can be fully utilised among different business scenarios.

Remark: e-Services can request or collect personal data of users from CorpID in accordance with the requirements under the Personal Data (Privacy) Ordinance (“PD(P)O”) (Cap. 486). After the personal data is transferred to e-Services (data users), e-Services undertake to ensure the personal data are held, processed and used in accordance with the PD(P)O, including but not limited to **giving a personal information collection statement (“PICS”) stating clearly how they will collect, use and process the personal data of the users from the CorpID system.**

### B.1 Scenario 1

User directly login to e-Service Website via CorpID service catalogue and then retrieve “eMEFields” for form pre-filling. User can modify form data if required and then submit the form.



e-Services should refer to below table for above implementation.

|                                                             | Developer Guide | API Specification |
|-------------------------------------------------------------|-----------------|-------------------|
| <b>Direct Login v2</b>                                      | Section 3.10.1  | Section 3.3.2.2   |
| <b>Form Pre-Filling with Service Login (i.e., Profiles)</b> | Section 3.5     | Section 5         |

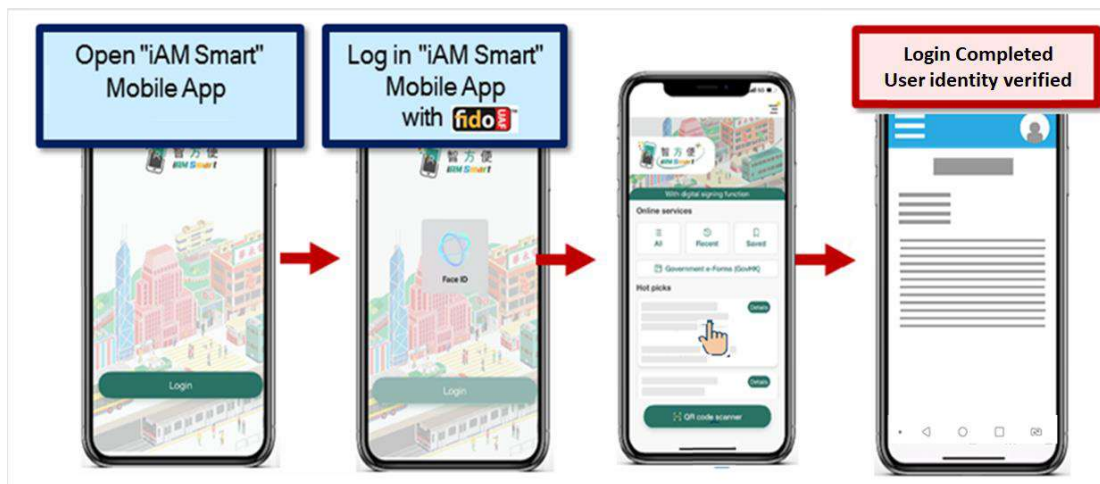
**Remark:**

*Existing e-Service hosted in Service Catalogue should also refer below table to take necessary upgrade action.*

| For e-Service adopting below API             | Impact | Action                                                  |
|----------------------------------------------|--------|---------------------------------------------------------|
| Direct Login v1                              | Yes    | Migrate to Direct Login v2                              |
| Anonymous Form pre-filling                   | Yes    | Implement Authentication process<br>Migrate to Profiles |
| Form pre-filling with Service Login v1 or v2 | Yes    | Migrate to Profiles                                     |

## B.2 Scenario 2

User directly login to e-Service Website via CorpID service catalogue and then e-Service requires “profileFields” for user identity verification, account link up or account on boarding. User is not allowed to update “profileFields” in the e-Service Website.



e-Services should refer to below table for above implementation.

|                                                             | Developer Guide | API Specification |
|-------------------------------------------------------------|-----------------|-------------------|
| <b>Direct Login v2</b>                                      | Section 3.10.1  | Section 3.3.2.2   |
| <b>Form Pre-Filling with Service Login (i.e., Profiles)</b> | Section 3.5     | Section 5         |

Remark:

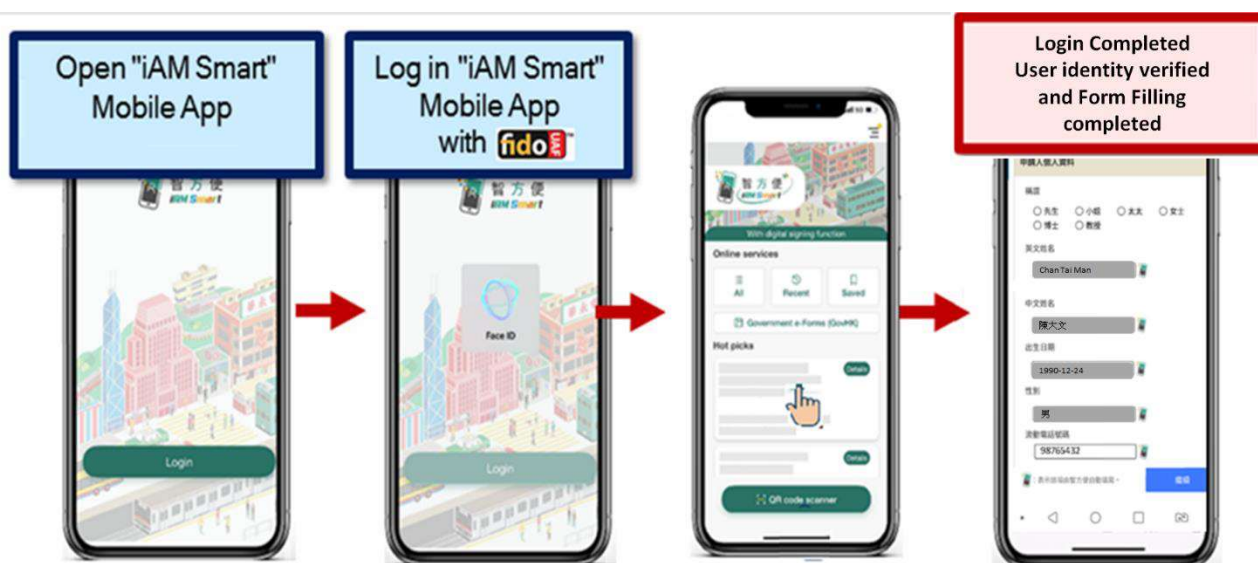
Existing e-Service hosted in Service Catalogue should also refer below table to take necessary upgrade action.

| For e-Service adopting below API | Impact | Action                     |
|----------------------------------|--------|----------------------------|
| Direct Login v1                  | Yes    | Migrate to Direct Login v2 |
| getProfile                       | Yes    | Migrate to Profiles        |

### B.3 Scenario 3

User directly login to e-Service Website via CorpID service catalogue and then e-Service requires both “profileFields” and “eMEFields” for user identity verification and form pre-filling respectively.  
(Scenario 1 + 2)

The account information in “profileFields” and “eMEFields” are the same for same user (e.g., e-Service would get the same HKIC number of specific CorpID user, no matter the information is from “profileFields” or “eMEFields”). However, both authentication and form pre-filling statistic report in CorpID system would be updated if e-Service requests both “profileFields” and “eMEFields” in one Profiles API call. In addition, unlike “profileFields” which is for the purpose of identity verification, e-Service can allow CorpID user to modify online form data filled with “eMEFields”.



e-Services should refer to below table for above implementation.

|                                                             | Developer Guide | API Specification |
|-------------------------------------------------------------|-----------------|-------------------|
| <b>Direct Login v2</b>                                      | Section 3.10.1  | Section 3.3.2.2   |
| <b>Form Pre-Filling with Service Login (i.e., Profiles)</b> | Section 3.5     | Section 5         |

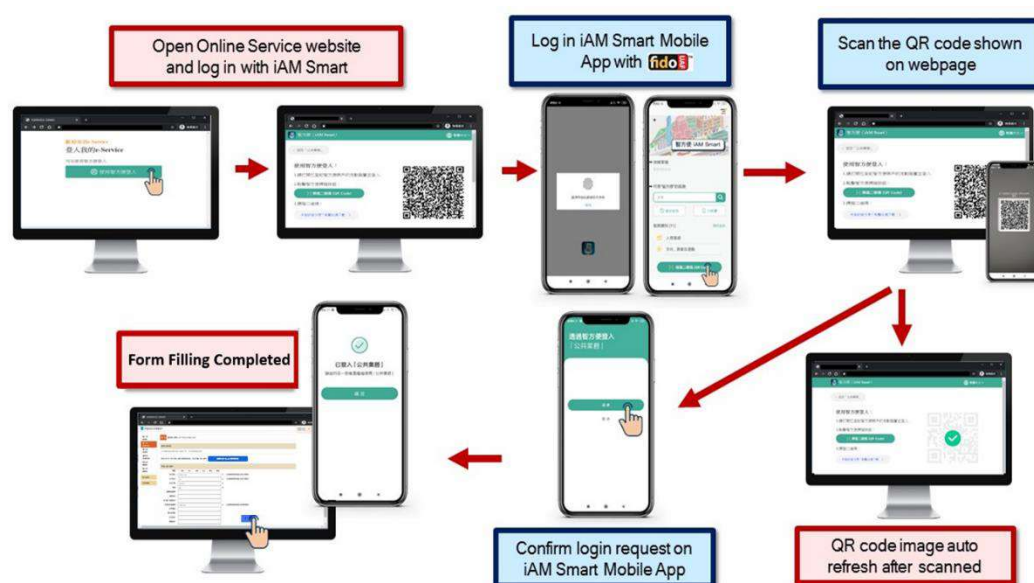
Remark:

Existing e-Service hosted in Service Catalogue should also refer below table to take necessary upgrade action.

| For e-Service adopting below API             | Impact | Action                     |
|----------------------------------------------|--------|----------------------------|
| Direct Login v1                              | Yes    | Migrate to Direct Login v2 |
| getProfile                                   | Yes    | Migrate to Profiles        |
| Form pre-filling with Service Login v1 or v2 | Yes    | Migrate to Profiles        |

#### B.4 Scenario 4

User opens e-Service Website via web browser (different device) and then login e-Service with CorpID. Upon successful of Authentication process, e-Service retrieves “eMEFields” for form pre-filling.



e-Services should refer to below table for above implementation.

|                                                      | Developer Guide | API Specification |
|------------------------------------------------------|-----------------|-------------------|
| Authentication                                       | Section 3.4.1   | Section 3.3.2.1   |
| Form Pre-Filling with Service Login (i.e., Profiles) | Section 3.5     | Section 5         |

Remark

Existing e-Service should also refer below table to take necessary upgrade action.

| For e-Service adopting below API                                                                   | Impact | Action                                   |
|----------------------------------------------------------------------------------------------------|--------|------------------------------------------|
| getProfile                                                                                         | Yes    | Migrate to Profiles                      |
| Form pre-filling with Service Login v1 or v2                                                       | Yes    | Migrate to Profiles                      |
| Anonymous Form pre-filling v1<br>(support multiple “iAM Smart” users to fill the same online form) | Yes    | Migrate to Anonymous Form pre-filling v2 |

### B.5 Scenario 5

User opens e-Service Website via web browser (different device) and then login e-Service with CorpID. Upon successful of Authentication process, e-Service retrieves “eMEFields” for form pre-filling and perform digital signing.



|                                                             | Developer Guide | API Specification |
|-------------------------------------------------------------|-----------------|-------------------|
| <b>Authentication</b>                                       | Section 3.4.1   | Section 3.3.2.1   |
| <b>Form Pre-Filling with Service Login (i.e., Profiles)</b> | Section 3.5     | Section 5         |
| <b>Digital Signing with Service Login</b>                   | Section 3.7     | Section 8         |

e-Services should refer to below table for above implementation.

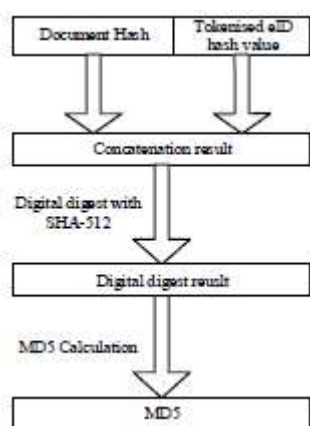
Remark:

Existing e-Service should also refer below table to take necessary upgrade action.

| For e-Service adopting below API                                                                   | Impact | Action                                        |
|----------------------------------------------------------------------------------------------------|--------|-----------------------------------------------|
| getProfile                                                                                         | Yes    | Migrate to Profiles                           |
| Form pre-filling with Service Login v1 or v2                                                       | Yes    | Migrate to Profiles                           |
| Anonymous Form pre-filling v1<br>(support multiple “iAM Smart” users to fill the same online form) | Yes    | Migrate to Anonymous Form pre-filling v2      |
| Anonymous Digital Signing (single user digital signing)                                            | Yes    | Migrate to Digital Signing with Service Login |
| Anonymous Digital Signing<br>(support multiple “iAM Smart” users to sign the same application)     | No     | N/A                                           |

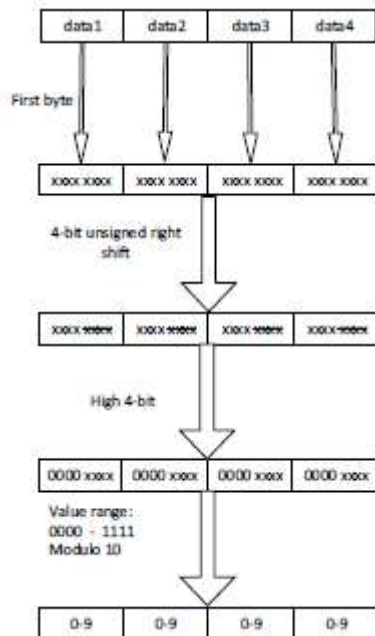
#### C. 4-DIGIT IDENTIFICATION CODE GENERATION ALGORITHM

- 1) Concatenate the Document Hash and the Tokenised ID hash value (calculated with SHA-512), and perform the digital digest for the concatenation result with SHA-512.
- 2) Perform the MD5 calculation to obtain a unique MD5 hash value.



- 3) Divide the 128-bit MD5 hash value into four groups of 4 bytes binary value and then extract the first byte from each group as a data sample.

- 4) Perform 4-bit unsigned right shift to each byte of the result obtained above for the hexadecimal data.
- 5) Perform the modulo operation on each byte (modulo 10) to obtain a number of 0-9.
- 6) Assemble the 4 numbers obtained above as a 4-digit identification code.



### Pseudo code for generating 4-digit identification code

```
private static final char hexDigits[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
public static String getSignCode (String hashCode, String openID) {
    // Assume Document Hash is BASE64 encoded and perform decode
    byte[] hashCodeBytes = Base64Util.base64Decode(hashCode);
    // Calculate SHA-512 hashcode of Tokenised ID
    byte[] openIDBytes = DigestUtil.digestToByte(openID, Alg.SHA512);
    char code[] = new char [4];
    byte[] source = new byte[hashCodeBytes.length + openIDBytes.length];
    System.arraycopy(hashCodeBytes, 0, source, 0, hashCodeBytes.length);
    System.arraycopy(openIDBytes, 0, source, hashCodeBytes.length, openIDBytes.length);
    // Calculate SHA-512 hashcode from concatenated Document Hash and
    // SHA-512 hashed Tokenised ID
    byte[] inData = DigestUtil.digestToByte(source, Alg.SHA512);
    // Calculate MD5 hashcode
    byte[] digestMD5Data = DigestUtil.digestToByte(inData, Alg.MD5);
    // Convert MD5 value 4-digit identification code
    for (int i = 0; i < 4; i++) {
        code[i] = hexDigits[(digestMD5Data[i * 4] >>> 4 & 0xf) % 10];
    }
    return new String(code);
}
```

**End of Document**