Technical Reference
for the
Integration of In-app Browser
for Web-based Online Service in “iAM Smart”

Version: *1.3*

JUNE 2025

Document Revision History

Ver. No.	Date	Section Affected	Summary of Changes
1.0	18 Nov 2024	All	First release
1.1	07 Jan 2025	All	Amendments on Blob Download Support, Cookies Handling, Apple Pay Consideration and Special Attention on wildcard URL usage in whitelist.
1.2	12 Mar 2025	2.1, 2.4, 3.1, Appendix	Amendments on Support Information for Onboarding, Handling of Browsing Data, Payment Flows, Reference Web Applications and new permission related JSAPI.
1.3	30 Jun 2025	1, 3.1.3, Appendix	Amendments on Overview. Amendments on Appendix for updated permission dialogue JSAPI and new section for common issues.

Table of Contents

1. OVERVIEW

1.1 Introduction of In-app Browser

The use of In-app browser allows mobile application developers to display web content, without having users to leave the mobile application. They are commonly adopted in popular social networking, news, search, and HTML5 web applications. In-app browsers are often powered by mobile OS components (like, WebView and WKWebView). "iAM Smart" has implemented In-app browser solution which serves as a substitute of external mobile browsers (e.g. Safari, Chrome) with the benefits of providing seamless experience to users while accessing Online Service Websites in "iAM Smart".

Previously, Online Service Websites used to be run independently under external browsers and integrate with the "iAM Smart" system through RESTful API calls on the server side to conduct authentication, digital signing and "e-ME" form-filling functions. With the introduction of the In-app browser, Online Service Websites will be able to be run in the In-app browser and to leverage the same RESTful API calls on the server side for completing the "iAM Smart" business workflows without leaving the "iAM Smart" mobile application. This streamlines and improves user experience by reducing the back-and-forth app-switching between the "iAM Smart" mobile application and the Online Service Websites.

Generally, no major changes are required for Online Service to adopt In-app browser for calling features/services of "iAM Smart". The "iAM Smart" business workflows are seamlessly utilised via the In-app browser to provide users with consistent experience.

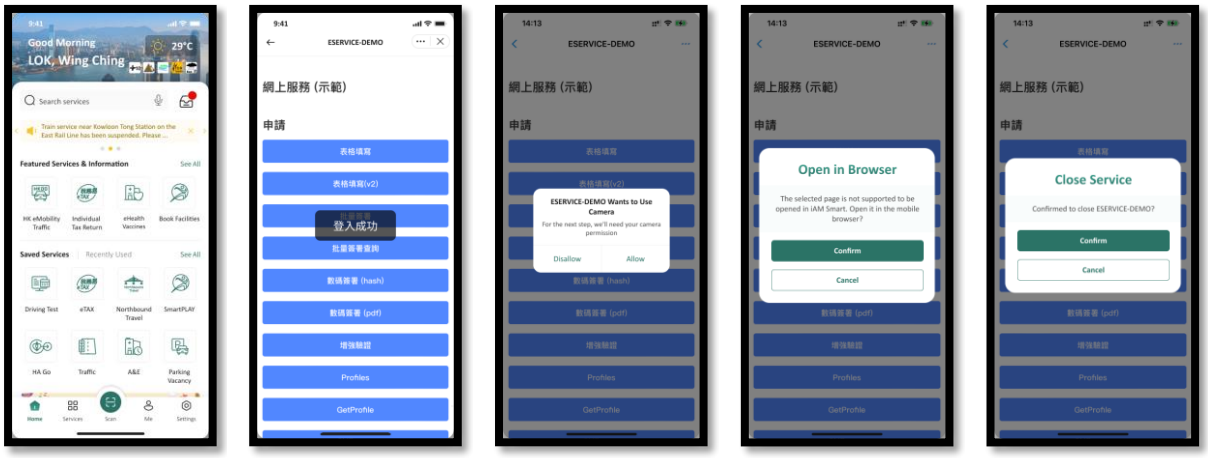
For some occasions, Online Service may need to make minor configuration changes to its Website based on individual design. This technical reference document aims to provide Online Service with descriptions, settings, potential changes as well as the onboarding procedures in details.

1.2 Operation Flows between Online Service Website, In-App browser and "iAM Smart" Mobile Application

1.2.1 Typical user interaction

The sequence of diagrams below illustrates typical user interaction when Online Service adopts the In-app browser.

Note: The diagrams in step 2 to 5 are captured from e-Service Demo web app which currently supports Traditional Chinese only.



Step 1 Step 2 Step 3 Step 4 Step 5

Step 1. "iAM Smart" mobile application retrieves catalogue information. User navigates and selects a service catalogue item with In-app browser adoption.

Step 2. "iAM Smart" mobile application initialises In-app browser and performs Direct Login or Direct Access within In-app browser.

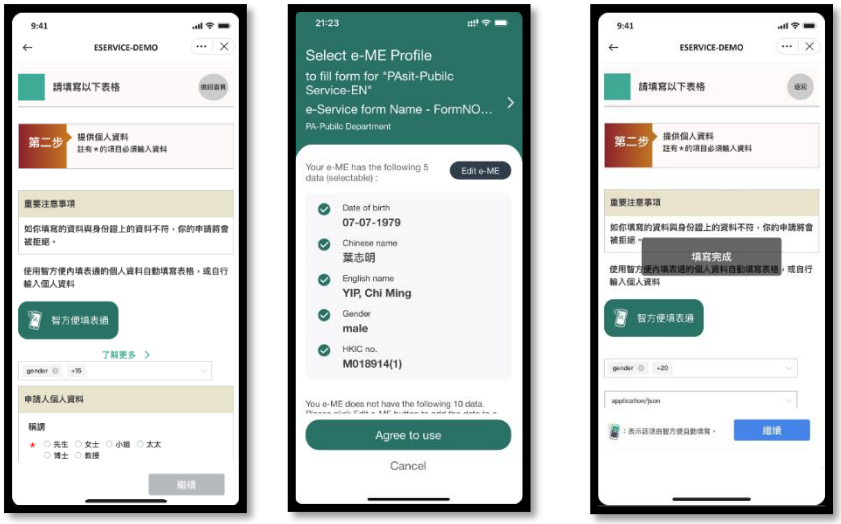
Step 3. When Online Service has applied special device permission and requests the permission during runtime of its Website. "iAM Smart" mobile application will ask user to confirm granting of such permission to the Online Service Website at the first time of its usage. User may revoke permissions anytime afterwards through Settings page accessible through the "three horizontal dots" menu button located on the top right-hand corner.

Step 4. When user navigates to a URL which is not on the application URL whitelist submitted by Online Service, "iAM Smart" mobile application will ask user to stay or to confirm opening the URL in external mobile browser.

Step 5. When user completes the workflow and closes In-app browser, "iAM Smart" mobile application will prompt user to confirm exit.

1.2.2 Typical user interaction with “iAM Smart” workflows

The sequence of diagrams below illustrates typical user interaction when Online Service with “iAM Smart” workflows adopts the In-app browser.
Note: The diagrams in step 1 and 3 are captured from e-Service Demo web app which currently supports Traditional Chinese only.



Step 1 Step 2 Step 3

Step 1. User initiates “iAM Smart” form-filling.
Step 2. “iAM Smart” mobile application displays confirmation page. User confirms the form-filling request.
Step 3. User continues to access the Online Service Website in the In-app browser.

Online Service must perform testing to ensure its Website is fully functional in both external mobile browser and In-app browser during the testing stage of the integration. “iAM Smart” mobile application may launch Online Service Website in external mobile browser according to following conditions including but not limited to –

i) Device model and operating system compatibility;
ii) Webpage not included in the whitelist submitted by Online Service during onboarding; or
iii) Quick responses from “iAM Smart” on reported vulnerabilities affecting the In-app browser. In-app browser will be disabled for all Online Services until remedial action has been taken.

2. ONBOARDING PROCEDURES

Online Service is required to submit onboarding request to DPO for vetting before its Online Service Website can be integrated with the In-app browser.

2.1 Onboarding Journey

Testing Environment

1. Online Service should follow the steps described in Section 1.3 of the "iAM Smart" API Specification to set up its Website.

2. Online Service should prepare the following information for approval to use the In-app browser. The details can be found in Section 2.4 of this document.
   - Permission settings
   - URL whitelist
   - Support Information

3. Upon application approval, DPO will provide Online Service with a **Testing App** and relevant testing account and information. Online Service should verify and test its Website in the In-app browser to ensure every function as well as the UI/UX are working as expected. DPO will provide support to Online Service if any issues are encountered.

Production Environment

1. Online Service should follow the steps described in Section 1.3 of the "iAM Smart" API Specification to set up its Website.
2. Online Service should prepare the following information for approval to use the In-app browser. Details can be found in Section 2.4 of this document.
   - Permission settings;
   - URL whitelist
   - Support Information

3. Upon application approval, the Support Team will notify the Online Service to verify whether its Website is fully functional.

2.2 Environments

2.2.1 Testing Environment

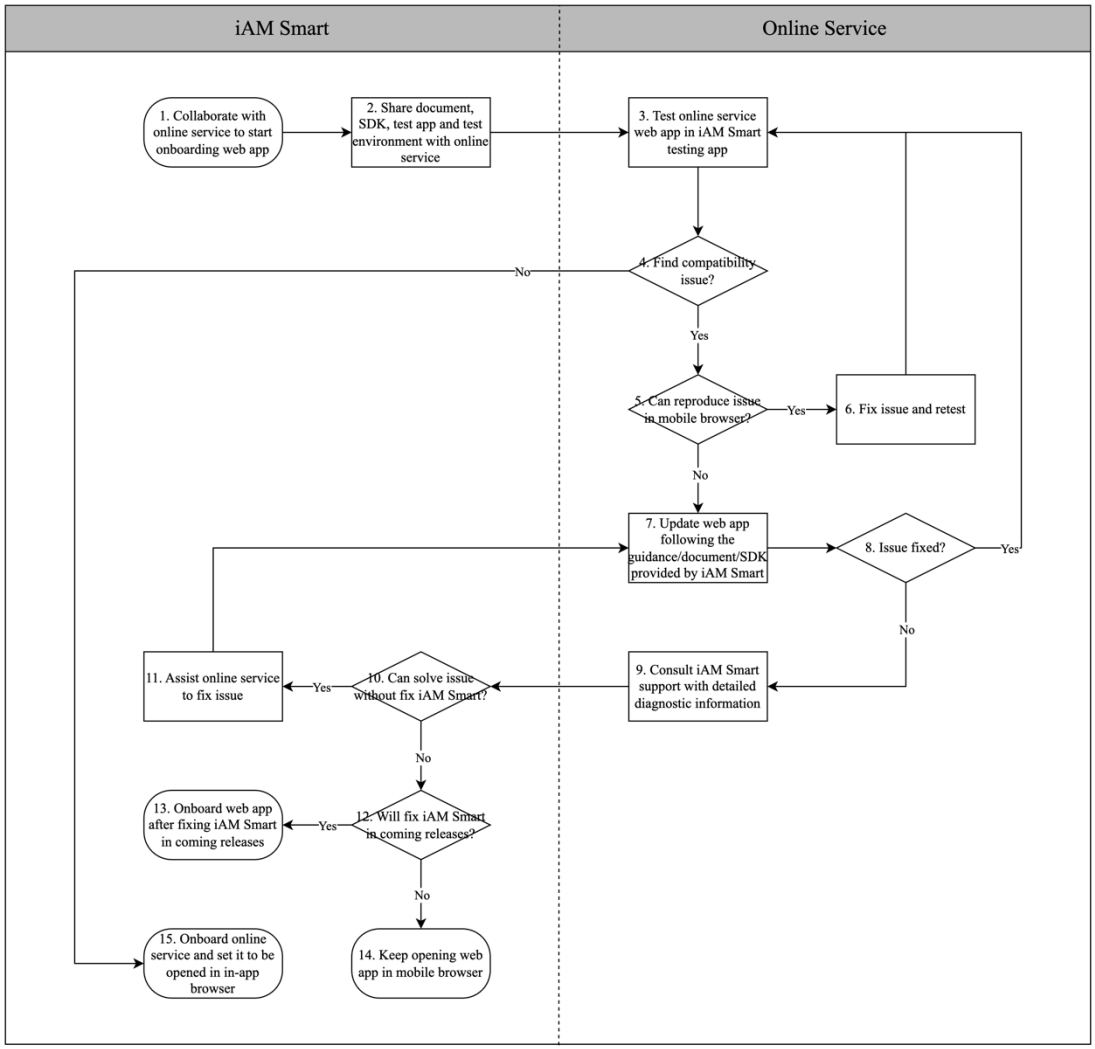
Domain Name	apigw-isit.staging-eid.gov.hk
URL Scheme of "iAM Smart" Mobile Application (iOS and Android)	hk.gov.iamsmart.testapp://
"iAM Smart" Mobile Application Installation	By invitation with AppStore and PlayStore with the tester list.
Self-Service Portal URL	https://portal-isit.staging-eid.gov.hk/ESP/index.html
KEK Certificate (Encipherment Certificate)	Trial certificate from Hong Kong Recognised Certification Authority ("RCA") is allowed
HTTPS Callback Endpoint	HTTPS with production certificate

2.2.2 Production Environment

Domain Name	apigw.iamsmart.gov.hk
URL Scheme of "iAM Smart" Mobile Application (iOS and Android)	hk.gov.iamsmart://
"iAM Smart" Mobile Application Installation	Officially download from App Store, Google Play, AppGallery or apk file from "iAM Smart" thematic page
Self-Service Portal URL	https://portal.iamsmart.gov.hk/ESP/index.html
KEK Certificate (Encipherment Certificate)	Production certificate from Hong Kong RCA
HTTPS Callback Endpoint	HTTPS with production certificate

2.3 Testing Process

Online Service Website should be fully tested before it is integrated into the "iAM Smart" System and opened in the In-app browser. The diagram below shows how Online Service tests and verifies its Website with the assistance of the "iAM Smart" team.



2.4 Mandatory Settings from Online Service during Onboarding

Online Service should provide the following information during the onboarding stage -

2.4.1 Permission Settings

The In-app browser allows Online Service Website to utilise device camera, microphone and location upon request. Online Service should register the need of corresponding permissions during onboarding. Any attempts from web content requesting unauthorised permissions will be discarded without explicit alerts on screen. User can also deny or revoke permissions to "iAM Smart" mobile application or to individual Online Service. Hence Online Service should prepare proper exception handling when access to protected resources is denied.

2.4.2 URL Whitelist

Online Service should provide the URL whitelist during onboarding. Only HTTPS enabled URLs are supported. When user navigates the Online Service Website in the In-app browser, external mobile browser will be opened instead of In-app browser if the target URL is not on the URL whitelist.

The asterisk (*) is the only valid wildcard character that can be used when defining a URL. The asterisk can be used in the URL host, the URL port, the URL path, and the URL parameters. The asterisk wildcard character represents any combination of characters and is applicable to where it is placed. For example:

- `iamsmart*` is equivalent to any string of characters that begins with `iamsmart`
- `*iamsmart` is equivalent to any string of characters that ends with `iamsmart`
- `*iamsmart*` is equivalent to any string of characters that has `iamsmart` in it somewhere
- `iam*smart` is equivalent to any string of characters that begins with `iam` and ends with `smart`

By default, an asterisk is considered to be a wildcard. If asterisk is used as a literal in a string and not as a wildcard, Online Service should precede it with a backslash (\). For example, the string `iam\*smart*` is matched with `iam*smart123`, but is not matched with `iam123smart123`.

The Direct Login URLs, Fallback URLs and "iAM Smart" URLs are automatically added to the whitelist and therefore Online Service is not required to include these URLs to the whitelist explicitly.

The following is a sample URL whitelist.

```
https://www.td.gov.hk/*

https://queue.td.gov.hk/*

https://www.gov.hk/*
```

2.4.3 Support Information

Online Service should provide support information including support email, support hotline and other information. Users can access such kind of information in the In-app browser menu and make enquires through the provisioned channels.

3. SPECIAL ATTENTION TO ONLINE SERVICE

3.1 Special Attention / Potential Changes Required for Online Service Applications

Subject to the implementation of the Online Service Website, Online Service may require to perform changes manually to ensure its Website is fully functional in both the In-app browser and mobile browser.

3.1.1 Support New User Agent Value

The In-app browser user agent sample values are shown in the Appendix A of this document. It always contains the "`iAM Smart In-App Browser`" sub-string at the end of the user agent value. Online Service is responsible for checking the compatibility and accepting the In-app browser's user agent value if there is any browser compatibility check based on user agent value.

3.1.2 Avoid Relying on Push Notification to Invoke "iAM Smart" Mobile Application

When initiating the "iAM Smart" workflow, Online Service Website running in the In-app browser should not solely rely on "Push Notification" flow to invoke "iAM Smart" mobile application. Instead, it should invoke the "iAM Smart" mobile native page explicitly to request user to confirm the authorisation request. Otherwise, user will have no way to invoke the confirmation page when the push notification function is disabled. For example, in the following implementation, user can still click "Open iAM Smart" to invoke the "iAM Smart" mobile application even if the push notification has been disabled. Note: For illustrative purpose, the diagram below shows the notification together.

3.1.3 Ensure Popup Works as Expected

Online Service Website with popups should be fully tested to ensure its popup function will work as expected when adopting the In-app browser. For example, notice popups using `window.open()` immediately after initial loading of website may be blocked silently. Please refer to Appendix E for common issues related to `window.open()`.

3.1.4 Ensure third-party integration URLs are included in URL whitelist

Online Service Website with third party components integration should be fully tested to ensure its function will work as expected when adopting the In-app browser. For example, virtual waiting room and CAPTCHA service may redirect user or load URL in embedded inline frame <iframe>. These URLs should be included in URL whitelist.

3.1.5 Special attention on using wildcard in URL whitelist

Online Service should not use wildcard URLs on public search engines. Search engines may lead user to unsafe and unsuitable web contents for all age groups of "iAM Smart" mobile application users. For example, https://youtube.com/* or https://google.com/* are not allowed.

3.1.6 Ensure Inline Frame <iframe> Works as Expected

Online Service Website with iframe usage should be fully tested to ensure its intended function will work as expected when adopting the In-app browser. For example, printing content within iframe should be avoided in In-app browser as it is not supported.

3.1.7 Ensure Blob Content Download Works as Expected

Online Service Website with blob download usage (URLs created by `URL.createObjectURL()`) combined with Content Security Policy **"CSP"** should be fully tested to ensure it works as expected. Online Service should add `blob:` to `connect-src` in existing CSP to be compatible in In-app browser.
If file name cannot be detected, ("`Download_YYMMDDHHmmss`") will be used as default file name. Similarly, if file extension cannot be detected, it will be stored based on the MIME type and may not be the same as the original one. If the downloaded file will be uploaded in future (e.g. saved form data file), Online Service should review HTML upload controls to include appropriate parameters in "accept" attribute so the file can be selected and uploaded.

Detected file name	Detected file extension	Downloaded file
Yes	Yes	`<filename>.<ext>`
No	Yes	`Download_YYMMDDHHmmss.<ext>`

No	No	`Download_YYMMDDHHmmss`

3.1.8 Update the Source Parameter of the /getQR Request

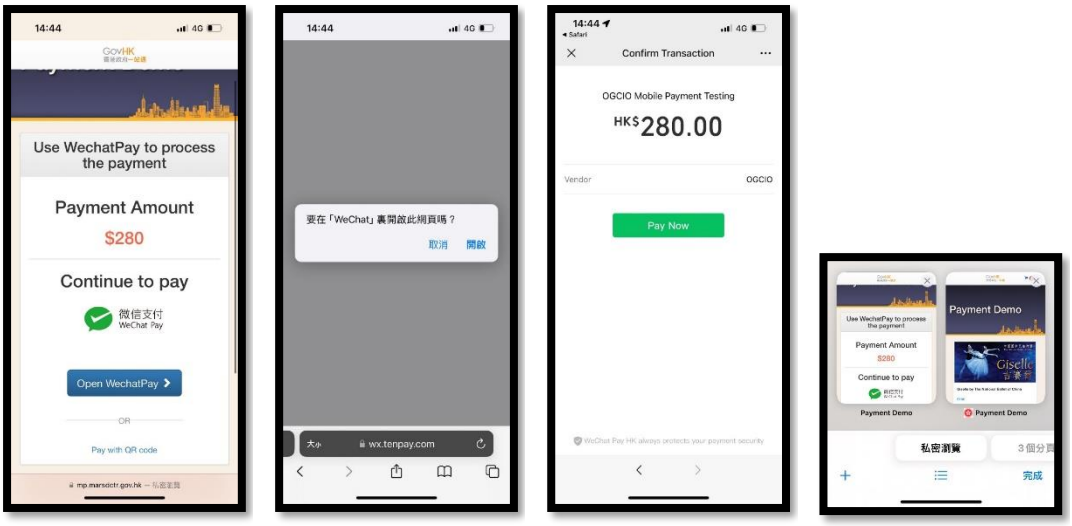
The Appendix B of the "iAM Smart" API Specification will be extended to support two more source parameter values, `iOS_IMS_InAppBrowser` and `Android_IMS_InAppBrowser`. The QR/App broker page API ("/getQR") will automatically set the source parameter when being detected to be running in the In-app browser. However, if Online Service calls other APIs that require the source parameter, the source value should correctly be set.

3.1.9 Unsupported Mobile OS, Webview or Device Model

The invocation of In-app browser function will be disabled in the "iAM Smart" mobile application under unsupported mobile OS, webview or device models. In such case, Online Service Website will be accessed via the original approach, i.e. external mobile browser. Online Service must perform testing to ensure its Website is fully functional in both external mobile browser and In-app browser during the testing stage of the integration.

3.1.10 Web-to-app application flow

For “Web-to-App”, Online Service may open an intermediate webpage to invoke external mobile application during its business workflow. One typical scenario of “Web-to-App” is the invocation of a payment mobile application from Online Service Website.



Step 1 Step 2 Step 3 Step 4

Step 1. Webpage provides invocation method for a mobile application.

Step 2. System opens mobile application after user confirmation.

Step 3. User interacts with mobile application for its workflows.

Step 4. Mobile application launches acknowledgement URL in external mobile browser when workflow completed.

It is observed that in Step 4, the mobile application may directly open a new webpage for acknowledgement of completing the payment workflow in default external mobile browser instead of returning to In-app browser. User may need extra time to locate the “iAM Smart” mobile application to continue his/her original operation. Hence it is recommended that if “Web-to-App” application flows are involved, Online Service must test and confirm that the flow will satisfy user experience requirements, or should follow original approach to run its Website in external mobile browser instead of adopting the In-app browser to avoid degradation of user experience.

3.1.11 No switching between “iAM Smart” mobile application and Online Service Website

When using external mobile browser, user can switch between Online Service Website running in Mobile Browser and “iAM Smart” mobile application.

Note: The diagrams in screen 1 and 3 to 5 are captured from e-Service Demo web app which currently supports Traditional Chinese only.

Screen 1
Online Service Website in Mobile Browser

Screen 2
App Switching

Screen 3
“iAM Smart” mobile application

Upon adoption of In-app browser, Online Service Website will be running in the “iAM Smart” mobile application, therefore user can only proceed the following screen flows in sequence.

Screen 4
Online Service Website in “iAM Smart” mobile application

Screen 5
“iAM Smart” mobile application

3.1.12 Handling of Browsing History, Cookies and Caches

Browsing history, cookies and caches (collaboratively called “browsing data”) will not be retained for long period after a browsing session is closed.

User is allowed to perform manual deletion of browsing data at any time. In addition, such browsing data stored on user device will be automatically removed upon the launch of “iAM Smart” mobile application if the mobile application has not been accessed for more than 24 hours by default.

For example, Online Service Website that integrates with virtual waiting rooms (e.g. queue-it service) relies on cookies to reserve user position and queue completion status. If browsing data are removed, they will need to queue again.

3.1.13 Device Permissions Not Supported by “iAM Smart”

Requesting permissions from Online Service Website not supported by “iAM Smart” mobile application may lead to unexpected results. Online Service should carefully prepare exception handling, test and ensure its website is fully functional when adopting the “iAM Smart” In-app browser.

3.1.14 Change of UX

There are a number of changes with respect to user experience after the adoption of the In-app browser, e.g. the “iAM Smart” Direct Login (v1) is simplified to reduce user interactions. Details are described in Section D of Appendix of this document.

3.1.15 Closing of In-app Browser

Online Service may have implemented close button at the end of their workflow. For the adoption of the In-app browser, Online Service should fully test if its own original implementation still works, otherwise it should leverage the provided JavaScript APIs to close the In-app browser and all its opened webpages. Details are described in Section B.4 of Appendix of this document.

3.1.16 Payment flows are Unsupported in In-app Browser

There exist different constraints for prevailing e-payment services, e.g. Apply Pay is only supported in Safari browser and e-payment service provider may not be able to provide exhaustive URLs to whitelist for their payment flow. As such, e-payment flows will not be supported in In-app browser at this stage. However, users should be able to continue their business journey (i.e. e-payment flow) with default mobile browser.

3.1.17 Use of Reference Web Applications and Sample Source Codes

Reference Web Applications and sample source codes are provided for reference to demonstrate the interaction of "iAM Smart" workflows in the In-app browser. Textual items in workflow-related UIs should be revised to wordings natural to users in both In-app browser and mobile/desktop browsers according to the "Reference System Flow and User Interface Specifications for iAM Smart Adoption". For examples, rephrasing similar phrases that ask user to open "iAM Smart" mobile application and use "Continue with iAM Smart" instead of "Open iAM Smart".

APPENDICES

A. Sample In-App Browser User Agent Values

Platform	Sample In-App Browser User Agent Value
Android	Mozilla/5.0 (Linux; Android 14; 23013RK/5C Build/UKQ1.230804.001; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/118.0.0.0 Mobile Safari/537.36 NebulaSDK/1.8.100112 Nebula AlipayDefined(nt:WIFI,ws:411|0|2.625) useStatusBar/true isConcaveScreen/false mPaaSClient iAM Smart In-App Browser
iOS	Mozilla/5.0 (iPhone; CPU iPhone OS 17_5_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile/21F90 ChannelId(22) NebulaSDK/1.8.100112 Nebula Mobile SafariWK PSDType(1) mPaaSClient/(null) iAM Smart In-App Browser

B. JavaScript APIs

The In-app browser implements several add-on functions for Online Service Website to get the settings of the current "iAM Smart" mobile application and to close all the pages and In-app browser directly. These add-on functions are only available when Online Service Website is loaded in In-app browser. Therefore, for all examples below, Online Service should check if its Website is being accessed in the In-app browser and the `eIDJSBridge` object exists before calling these functions.

```
if (navigator.userAgent.includes('iAM Smart In-App Browser') &&
window.eIDJSBridge)
{
  // call the Javascript APIs
}
```

B.1 Get Language Setting

The In-app browser implements the `getAppLanguage()` function for Online Service Website to get the language setting of the current "iAM Smart" mobile application. The following is the sample program that shows how this function can be called via JavaScript.

```
<script>
  function onClick() {
    var appLang = window.eIDJSBridge.getAppLanguage();
    /*
    * use the appLang value
    */
  }
</script>
```

The function returns "`tc`", "`en`" or "`sc`". Online Service may choose to use similar language when getting the value from the "iAM Smart" mobile application.

B.2 Get Font Setting

The In-app browser implements the `getAppFontSize()` function for Online Service Website to get the font size of the current "iAM Smart" mobile application. The following is the sample program that shows how this function can be called via JavaScript.

```
<script>
  function onClick() {
    var appFontSize = window.eIDJSBridge.getAppFontSize();
    /*
    * use the appFontSize value
    */
  }
</script>
```

The function returns "`small`", "`medium`" or "`large`". The normal (medium) font size of the "iAM Smart" mobile application is 14 sp/pt, the small font size is 0.83 times of the normal font size, and the large font size is 1.27 times of the normal font size. The font size for the Lite version of the "iAM Smart" mobile application is always 21 sp/pt. Online Service may use the `getAppFontSize()` value in combination with the application mode described below to determine the scale of their applications.

B.3 Get Application Mode

The In-app browser implements the `getAppMode()` function for Online Service Website to get the mode of the current "iAM Smart" mobile application. The following is the sample program that shows how this function can be called via JavaScript.

```
<script>
  function onClick() {
    var appMode = window.eIDJSBridge.getAppMode();
    /*
    * use the appMode value
    */
  }
</script>
```

The function returns "`normal`" or "`lite`". The UI of the `lite` mode enlarges the fonts and buttons, reduces background colors and animations, and reformats the page layouts to ensure it's friendly for senior citizens. Online Service may choose to use similar mode when getting the value from the "iAM Smart" mobile application.

B.4 Close In-App Browser

The In-app browser implements the `closeInAppBrowser()` function for Online Service Website to close the In-app browser. For example, Online Service may have already implemented a close button using native JavaScript at the end of a workflow. If the implementation does not work in the In-app browser, Online Service can call this function to close the In-app browser and all its opened webpages. The following is the sample program that shows how this function can be called via JavaScript.

```
<script>
  function onClick() {
    window.eIDJSBridge.closeInAppBrowser();
  }
</script>
```

B.5 Request Permission Dialogue

The In-app browser implements the `requestSysPermission (permissionName)` function for Online Service Website to request permission setting instruction dialogue. This function provides a dialogue for redirecting user to app permission settings offered by the mobile operating system. Online Service Website should modify the exception handler when using device permission web APIs to prompt setting dialogue when permission is denied. This function will only display the dialogue if the related permission is registered by Online Service or the related permission is explicitly revoked by User for "iAM Smart" mobile application.

Parameter	Type	Presence	Description
`permissionName`	String	Required	This specifies which permission the dialogue should guide the user to setup. The possible values are: `camera`, `microphone`, and `location`

The following is the sample program that shows how this function can be called via JavaScript.

```
<script>
    function onClick() {
        const stream = navigator.mediaDevices.getUserMedia({ video:
true }).catch((err) => {
            if (navigator.userAgent.includes('iAM Smart In-App Browser') &&
window.eIDJSBridge) {
                // guide user to enable permission if applicable
                window.eIDJSBridge.requestSysPermission("camera");
                // window.eIDJSBridge.requestSysPermission("microphone");
                // window.eIDJSBridge.requestSysPermission("location");
            } else {
                // display generic dialogue for mobile browser
            }
        });
    }
</script>
```

C. Debugging Support

Online Service can debug its Website in the In-app browser function under iOS version of "iAM Smart" testing mobile application by following the steps below.

1. On iPhone, go to the Settings app and then Safari > Advanced and toggle the Web Inspector on.
2. Use cable to connect the iPhone to Mac computer.
3. On Mac computer, run Safari, click to open the Develop menu and highlight the connected iPhone.
4. Run the Online Service Website in the "iAM Smart" test application in iPhone. The highlighted device menu in the Mac Safari browser will show the corresponding webpages.

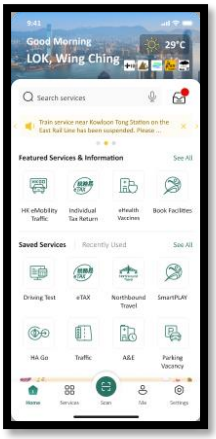
Online Service can also debug its Website in the In-app browser under Android version of "iAM Smart" testing mobile application by following the steps described in https://developer.chrome.com/docs/devtools/remote-debugging.

Please note that the debugging feature is disabled in the production release.

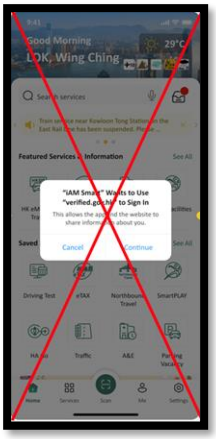
D. User Experience Changes in In-App Browser

User interactions are simplified for Direct Login v1 and the Authentication workflows while most of the remaining "iAM Smart" workflows, e.g. form filling and digital signing, remain unchanged. The simplified workflows are available after on-boarding and do not require modification from Online Service.

1. When running the Online Service Website in the In-app browser, the Direct Login workflow does not require user to confirm cookie sharing anymore. The diagram below shows the simplified Direct Login v1 workflow. Note: The diagram in screen 4 are captured from e-Service Demo web app which currently supports Traditional Chinese only.



Screen 1



Screen 2

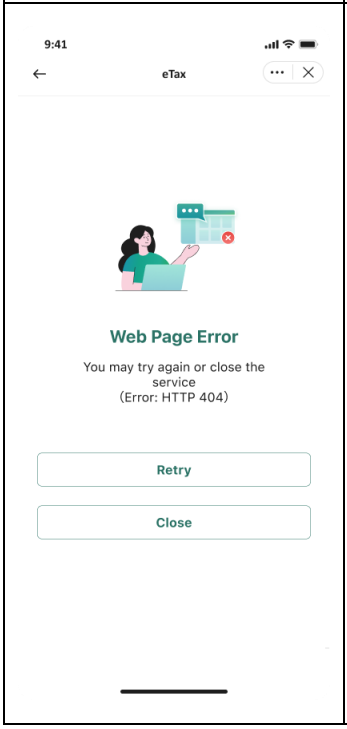
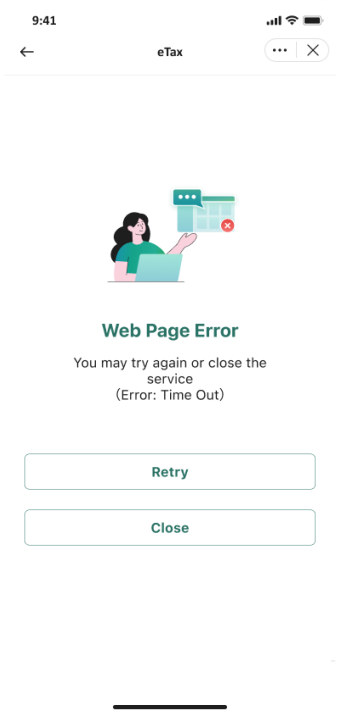
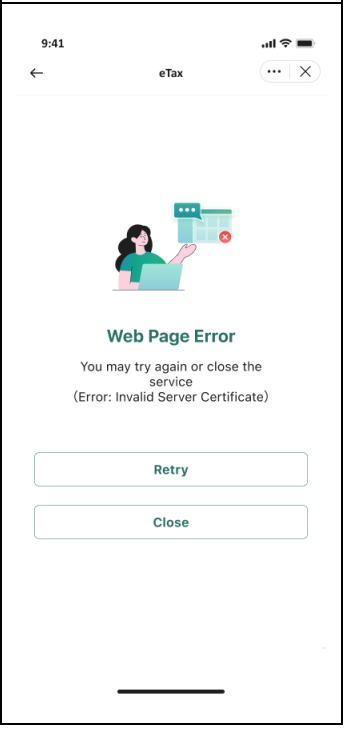
Screen 3

Screen 4

2. The In-app browser will display error pages in situations where the Online Service Website fails to load. Common issues, such as HTTP errors, connection timeouts and certificate errors, will be handled by the In-app browser.

   If error is encountered at the entry point of the Online Service Website (i.e. Direct Login URLs and Fallback URLs) in the In-app browser, "iAM Smart" mobile application will allow user to start the Website in external mobile browser (Retry) or exit the In-app browser (Close). If user chooses "Retry", the In-app browser will be closed after the launching of external mobile browser.

   If error is encountered during the navigation of the Online Service Website in the In-app browser, "iAM Smart" mobile application will allow user to reload the same webpage (Retry) or exit the In-app browser (Close).

HTTP Errors	Connection timeout	Invalid Server Certificate
		

E. Common issues

This section outlines common technical issues encountered when implementing the In-app browser, in order to help developers identify and resolve problems efficiently.

E.1 window.open()

Since the In-app browser does not provide a tab bar like desktop or mobile browsers, users cannot close or switch between opened tabs. The most recently opened view appears at the top of the view stack. Therefore, Online Service should be aware of the use of window.open(). It is recommended to call window.close() once the relevant request or action is completed in the new window. Otherwise, a blank page may be shown, negatively affecting the user experience.

MDN Window: open() method ref: https://developer.mozilla.org/en-US/docs/Web/API/Window/open

MDN Window: close() method ref: https://developer.mozilla.org/en-US/docs/Web/API/Window/close

Example 1

Online Service reported an issue when opening a window to trigger the "iAM Smart" Digital Signing workflow. After the digital signing process was completed, the window could not be closed (see the original code in the image below).

The issue can be resolved by storing the reference of the opened window in a variable, so that the opened window can be closed or handled appropriately once the digital signing process is completed. Please refer to the sample code below:

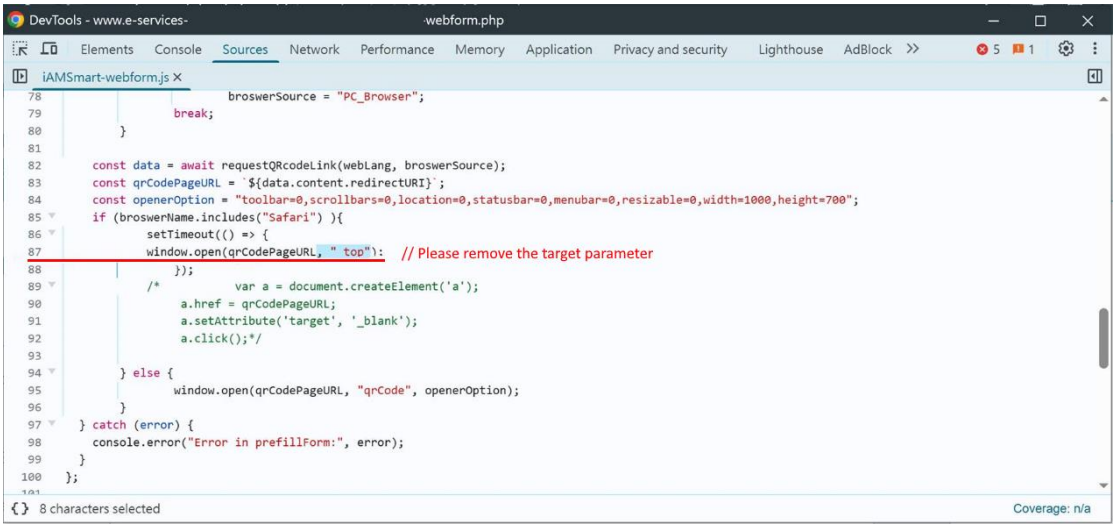
Untitled-1.js

```
loopGetSigningInfo = function (count){
    runCount = count;
    loopTask = setInterval(function(){
        if (runCount < 0) {
            clearInterval(loopTask);
            location.href = 'personalInfo.do?updateErr=IAMSMART_GET_INFO_FAIL';
        } else {
            if (signingApiCalling) {
                console.log("signingApiCalling = true, skip calling - " + runCount);
            } else {
                console.log("call get signing - " + runCount);
                runCount--;
                signingApiCalling = true;
                $.ajax({
                    url: "../api/getIAMSmartSigningResult.do",
                    method: "POST",
                    success: function(result){
                        signingApiCalling = false;
                        if (result.status == "OK") {
                            clearInterval(loopTask);
                            location.href = "updatePersonalInfoComplete.do?refNo="+result.refNo+"&date="+result.date+"&time="+result.time;
                            console.log("Cilla");
                            // Close the opened window if the signing is successful   2.
                            window.Cilla.close();
                        } else if (result.errorCode == 'IAMSMART_NOT_RETURNED'){

                        } else {
                            clearInterval(loopTask);
                            location.href = 'personalInfo.do?updateErr='+result.errorCode;
                        }
                    },
                    erro: function () {
                        signingApiCalling = false;
                        location.href = 'personalInfo.do?updateErr=IAMSMART_GET_INFO_FAIL';
                    }
                });
            }
        }
    }, 5000);
};
openIAMSmartAppForSigning = function(){
    $('#iAMSmartLoadingIcon').removeClass('hide');
    $('#iAMSmartOpenButton').haddClass('dim');

    // Create a variable for the opened window
    window.Cilla = window.open(deeplink,'_blank');   1.
};
```

The above logic can be implemented to other “iAM Smart” workflows as well, such as Form Filling request.

Example 2

Online Service reported that a blank page could not be dismissed on macOS and iOS devices upon completion of the "iAM Smart" Form Filling authentication. After investigation, it was noticed that the issue was caused by the use of the `target` (see image below).



The problem was resolved after removing this parameter, i.e. the highlighted code was updated from `window.open(qrCodePageURL,"_top")` to `window.open(qrCodePageURL)`.

E.2 Downloading PDF

Some users reported that the PDF files could not be downloaded or saved as .bin files in the In-app browser. In most cases, the issue was resolved by setting the content type to "`application/pdf`" when the user agent was detected as the "`iAM Smart In-app browser`". Please refer to the sample code below:

```
<script>
    function onClick() {
        if(navigator.userAgent.includes("iAM Smart In-App Browser")) {
            // set the content type to application/pdf for In-App Browser
              var contentType = "application/pdf";
              ...
         } else {
            // keep the original setting for desktop or mobile browser
              var contentType = "application/octet-stream";
            // or other appropriate value
              ...
         }
        // continue to proceed download
    }
</script>
```