Towards a Mobile-First Cross-Border eID Framework

ROLAND CZERNY, Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology and Secure Information Technology Center Austria (A-SIT), Austria
CHRISTIAN KOLLMANN, A-SIT Plus GmbH, Austria
BLAŽ PODGORELEC, Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology and Secure Information Technology Center Austria (A-SIT), Austria
BERND PRÜNSTER, A-SIT Plus GmbH, Austria
THOMAS ZEFFERER, A-SIT Plus GmbH, Austria

The eIDAS technical framework has been successfully enabling cross-border e-government processes for many years. When initially conceived, today's user habits and the prevalence and ubiquity of smartphones was nothing but a glimmer on the horizon. As a consequence, the concepts, technologies chosen, and technical standards used to carry out cross-border authentication were designed and chosen with browser-based user flows in mind. In this context, the network of eIDAS nodes and the interfaces defined to integrate them with all kinds of different national eID systems has stood the test of time. At the same time, however, transitioning these workflows to a mobile setting presents various significant challenges: Instead of using a single application (a web browser) to orchestrate the interaction of eID systems, eIDAS nodes and e-government service frontends (mostly using SAML), users are accustomed to using distinct native apps for every service and for interacting with eID systems. This work discusses different concepts essential for transitioning from such browser-based user flows to native app-to-app communication and combines them into a coherent concept. It presents a framework, which maintains browser compatibility, while at the same time providing all the benefits of native mobile apps, taking currently deployed eIDAS-based cross-border authentication to the next level by making it mobile-first, all without requiring invasive changes to existing infrastructure. As will be shown, a slew of technical constraints to overcome makes this a lofty goal, especially considering the heterogeneity of national eID systems which must obviously integrate well with the proposed concept.

CCS Concepts: • **Applied computing → E-government**; • **Human-centered computing → Mobile phones**; • **Security and privacy → Domain-specific security and privacy architectures**; • **Information systems → Information systems applications**.

Additional Key Words and Phrases: eGovernment, mGovernment, eIDAS, mobile cross-border authentication

ACM Reference Format:
Roland Czerny, Christian Kollmann, Blaž Podgorelec, Bernd Prünster, and Thomas Zefferer. 2023. Towards a Mobile-First Cross-Border eID Framework. In . ACM, New York, NY, USA, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Cross-border authentication has been massively pushed in the European Union for years, with various large-scale research and innovation projects leading up to and going beyond the eIDAS[1] regulation. The current technical implementation of this regulation is a tried-and-true staple of European e-government. While its technological foundation

[1]https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv%3AOJ.L_.2014.257.01.0073.01.ENG

has proven itself to be reliable and robust, its underlying design revolves around usage patterns and paradigms, which are heavily centered on the traditional personal computing metaphor of a user operating a traditional PC or laptop device. In reality, however, user habits have drastically changed since, and users nowadays typically do most of their computing on mobile devices such as smartphones and tablets. Consequently, users have come to expect the applications they rely on to conform to radically different UX flows. Most prominently, native mobile apps are available for many services as the primary way to consume them, compared to websites, which are oftentimes hard to comprehend on small smartphone screens and cumbersome to operate using touch input. In addition, such an app-based, clear separation of concern ensures that sensitive user data cannot flow between applications without user consent. This, again, presents a stark contrast to ubiquitous web tracking practices and the general difficulty of securing web pages. The main reason for these privacy and security benefits is the wholly different security model of mobile operating systems compared to traditional desktop OSes. This aspect by itself can already be considered a strong motivator for bringing e-government processes to mobile platforms.

This work presents a novel approach to transform established eIDAS-based cross-border authentication to mobile-first paradigms. We propose a dedicated eIDAS app to accomplish this goal, and provide interfaces for existing service providers to integrate. Our concept remains backwards-compatible and makes very few assumptions with respect to authentication processes on mobile platforms – all of which effectively reflect current best practices. The main motivation behind this strategy is the simple fact that trying to push something, which is incompatible with systems currently used in practice as part of multilateral deployments and SLAs, is illusory and will realistically never come to fruition. Hence, our envisioned system must adhere to the following constraints:

- Browser-based cross-border authentication must still work
- Both relevant mobile platforms (iOS and Android) must be supported
- No changes to the member states' eIDAS nodes must be required. (This whole document follows the terminology used by the *eIDAS Node Integration Package*[2] provided by the European Commission)
- The least possible integration effort on national eID systems and SPs is aspired

This paper is structured as follows: We first provide technical background information and argue why and how the currently deployed browser-based systems should be revised in Section 2. This also includes related work on previous research on m-government. Section 3 then introduces our concept in detail, providing the rationale behind our vision. We subsequently describe a demo implementation of our concepts and how it was used to evaluate our concept in Section 4 before finally concluding this work and providing an outlook in Section 5.

2 BACKGROUND

The general utility of smartphones in e-government contexts has been discussed with respect to a variety of aspects, soon after their rise in popularity. In those early days of smartphones gaining traction, the focus was rather different than what we aim for in this work. More general issues, like the inability of using then mandatory smartcards to create qualified electronic signatures on smartphones, needed solving as mentioned by Rath et al. [11], as well as Zefferer, Golser and Lenz [16], for example. At the same time, portability challenges were identified early on as well, specifically noting the fact that many e-government, and e-commerce workflows assume a traditional desktop computing setup [15].

[2] https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eIDAS-Node+Integration+Package

Luckily, the situation has since changed, and neither technological and regulatory barriers outright blocking a transition to mobile-first cross-border e-government still exist. At the same time, however, cross-border authentication workflows – integrating service providers with eID systems from another country through eIDAS – still very much do not fit with native smartphone usage paradigms. As a consequence, user experience is poor and the full potential of modern smartphone platforms remains unused.

To tackle this issue, the European Commission has recently published a proposal[3] for a successor of the current eIDAS Regulation. This proposal changes the way how cross-border authentication shall be achieved in the EU in the future. The core concept of the European Commission's proposal is the so-called IW. The EUDIW is supposed to be a technical component under full control of the user and enables secure storage and presentation of asserted identity information (identifiers, names, date of birth, etc.). The fundamental idea is that this information is no longer provided by central national identity management systems but in a decentralized way by the user and their wallet. This improves privacy, as national identity management systems are no longer involved in authentication processes and hence unable to track users' behavior.

The EUDIW does not only improve privacy by putting the users in full control of their data, it will also enable mobile usage scenarios involving mobile end-user devices. This way, the EUDIW will be a key enabler for future European mobile cross-border e-government services. Currently, concepts are tested and piloted on a European level by several projects (Large Scale Pilots). However, at this stage, various technical aspects are still open and no fully-fledged implementation of the EUDIW is available yet. Our solution, on the other hand, is readily usable with existing infrastructure and within the bounds of already approved policies. Moreover, our proposal could serve as a compatibility layer, bridging the gap between both approaches and smoothing the transition from one paradigm to the other.

Before discussing our proposed solution in a comprehensive manner, the background of the relevant technological foundation needs to be laid out, and a more detailed problem description is required. The remainder of Section 2 serves this purpose.

2.1 eIDAS Cross-Border Authentication

Figure 1 shows an abstract view of an eIDAS cross-border authentication workflow. The depicted sequence diagram refers to the traditional browser-based approach.
eIDAS allows users to authenticate to an SP in another country using their national eID system. Part 1 of Figure 1 shows how an authentication request is issued in the SP country: First, the SP creates an authentication request. The SP relays the authentication request to the eIDAS node of the SP country. The SP eIDAS node generates an eIDAS SAML2 AuthnRequest, which is included in a redirect to the MS eIDAS node of the user. In part 2, the authentication request is handled in an MS specific way by the MS eID implementation. Regardless of the technical details of the MS specific eID system, the user's MS eIDAS node creates an authentication response in the form of a SAML assertion and posts it to the eIDAS node of the SP country. This is depicted in part 3 of Figure 1. In the last step, the SP receives the authentication response to handle the authentication information.

As shown in Figure 1, the communication is HTTPS-based and uses either redirects or auto-POST forms. All communication is handled by the browser of the user. The driving force behind this eIDAS workflow are the eIDAS nodes of each member state. The nodes are responsible for connecting member states with each other through a defined

[3] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52021PC0281&from=EN

interface. However, some details like country selection are not standardized and thus are implemented in a MS-specific way.

Although this workflow works well with a browser, we will see in the following sections that additional steps are required in order to transform it to a mobile flow.

2.2 SAML and its Drawbacks in Mobile Scenarios

SAML [14] is an XML-based open standard to exchange authentication and authorization information between parties. These parties consist of a service provider (SP) and an IdP. In the eIDAS use case, two eIDAS nodes take the roles of SP and IdP.

When SAML was designed in the early 2000s, mobile use-cases were not taken into consideration. SAML depends on certain keys that are pre-shared between the SP and the IdP to establish integrity and authenticity. This is trivial in the standard use-case, where SP and IdP are hosted on dedicated servers. The key material is securely stored on these servers. In the mobile use-case, however, this is an issue. If, for example, the SP is implemented as a native app, key management becomes a very cumbersome task. The IdP can not trust the keys in the same way as it does with a SP that is hosted on a server. Also, if the IdP changes its keys, the SP app needs to be updated on every device.

As discussed by Choudry [5], there are other reasons why SAML is not a good choice for a mobile scenario. SAML heavily relies on back-channels or HTTP POST requests for communication. For example, the sign-on response is redirected to the SP using HTTP POST or HTTP Artifact binding [9]. HTTP Redirect is not a feasible solution, since the response can get too large.

Another problem is that an SP is required to share its configuration data with the IdP. Again, this is trivial for a server-side web application. However, a mobile application can not host its metadata on a chosen URI path for the IdP to fetch.

To summarize, SAML simply was not designed to be used in SPAs or mobile applications. Although workarounds solving some of the issues exist, many vendors move to OIDC [7] and OAuth [8]. This is also the reason not to use SAML in a mobile scenario, since it is very likely that compatibility issues will arise in the future.

2.3 OpenID Connect

OIDC is an authentication layer built on top of the OAuth 2.0 [8] authorization protocol. In contrast to SAML, OIDC was designed to allow a variety of clients, including web-based, mobile, and JavaScript clients, to obtain information about authenticated sessions and end-users [7]. The OIDC specification is extensible and comes with optional features like encryption, signing of identity data, and discovery of OpenID providers. As described in Section 2.2, SAML flows are usually happening between SP and IdP. The rough equivalent of an SP it is often referred to colloquially as *client*. However, the concept of a client is more versatile for OIDC, which is why it is called a RP throughout specifications. Due to this work specifically targeting SAML-based setup, involving traditional service providers the term SP is used for the sake of simplicity aside from this subsection.

A major drawback of SAML is that it usually requires a back-channel due to payload sizes (see below). OIDC has been specifically designed to overcome this issue. All communication is API-friendly and can be implemented on a variety of different clients in a straightforward manner.

OIDC allows for a dynamic client registration to establish trust between the RP and the OIDC Provider [12]. In combination with OIDC Discovery [13], OIDC overcomes the biggest issue of SAML – how can a mobile client obtain and provide information required for a trusted interaction between a RP and an OIDC Provider.

Another advantage of OIDC is that it uses JSON [6] as the underlying data exchange format. One of the issues of SAML is that the complex XML format leads to large response messages. Often these messages get too large to be sent over a front-channel like a query parameter. With JSON, OIDC uses a format that is lightweight and easily legible even for a human reader. The simplicity of JSON leads to shorter messages, which can easily be transmitted via a front-channel.

2.4 App-to-App Communication

Since the proposed eIDAS app is only responsible for parts of the eIDAS communication, app-to-app communication is a strong requirement. App-to-app communication allows apps installed on one device to exchange data with each other. Additionally, apps need to be able to start other apps. If those other apps are not installed, a fallback solution must handle this case gracefully.

Another requirement for the eIDAS app is that it should be available on Android and iOS. Therefore, we require a concept that works on both platforms.

Luckily there is already a solution available on Android [1] and iOS [3], that fulfills all of the requirements mentioned above. In a generic way, this works as follows: An app can register URLs at compile time. This informs the operating system that the app is capable of handling these URLs, which are commonly referred to as *claimed* URLs. In order to verify that only valid apps are opened when such a URL is requested, a file confirming this validity (referred to as *asset links file* throughout this work) needs to be hosted on this URL's host. This is very similar on both Android [2] and iOS [4].

When another app, like an SP app or a mobile web browser, sends a GET request to a claimed URL, the app responsible is opened. If the app is not installed on the device, the request is simply handled by the browser. Therefore, a fallback solution that does not require any changes in the backend is given.

Since a URL may contain query parameters, this solution additionally enables transferring data between apps. As such, it is a good fit to enable app-to-app communication while remaining compatible with browser-based workflows.

3 TOWARDS A UNIFIED EIDAS APP

The main challenge this work seeks to address is splitting responsibilities between distinct apps to handle cross-border authentication in a mobile-first way, i.e. it has to *look and feel* native to the user. We propose a dedicated eIDAS app to achieve this goal and minimal changes to member-state specific components already deployed for cross-border authentication workflows. To provide more added value to the user, we introduce well-defined means for citizens to specify their home country through the eIDAS app, as this provides a coherent user experience across countries and service providers and aligns user actions with the involved component's responsibilities.

On the one hand, all technical constraints elaborated on in the previous section map out a well-defined space of possible solutions for the proposed eIDAS app to occupy. On the other hand, organisational, operative, and legislative policies further restrict the bounds of this space of technical possibilities. Hence, this section first lays out the technical constraints of a mobile-first, multi-app cross-border authentication framework followed by relevant non-technical considerations – some of which indeed assume some leeway with respect to various policies and current practices.

Overall, transitioning eIDAS-based cross-border authentication towards mobile-first paradigms as outlined here effectively only requires technical changes to the interactions on the SP side, corresponding to Part 1 of Figure 1. An overview of these altered workflows is depicted in Figure 2.

3.1 Technical Requirements and Constraints

As the proposed solution strives for backwards compatibility, while heavily relying on native app-to-app communication, protocols based on the OpenID Connect suite are effectively the only way to go whenever it comes to crossing app-boundaries. This should not come as a surprise, since OIDC has been explicitly defined with browser-compatible mobile use cases in mind. Consequently, any service seeking to utilise mobile cross-border authentication must natively use OIDC based on `HTTP GET` for authentication requests. Naturally, an eIDAS node handing off an authentication request to a national eID system must also employ OIDC using `HTTP GET`.

Based on these constraints, one might intuitively assume that using SAML to communicate between eIDAS nodes presents a problem. However, two properties apply to all communications between eIDAS nodes:

- Only Messages from an overall well-defined set of possible exchanges are ever passed between eIDAS nodes. Communication is based on redirects and auto-post forms, whose behaviour is equally well-defined.
- No app-to-app communication is necessary, only emulation of browser behaviour.

As a consequence of these properties, a single *eIDAS app* can handle all communication between eIDAS nodes transparently while providing a native mobile user interface and not requiring any alterations to existing eIDAS node deployments. Without any further measures such an eIDAS app will, however, be required to claim the URLs of all deployed eIDAS node connectors[4] across Europe in order to respond to all cross-border authentication requests. Declaring such claimed URLs must happen at app compile time – hence, additional steps need to be taken to make the operation of such an eIDAS app feasible in practice, as will be discussed in the following section.

In order to provide the unified country selection mechanic mentioned in Section 3, changes to member states' specific eIDAS connector implementations are required. In short, a standardised way is needed such that a request to select a citizen country can be relayed to the user when the eIDAS app takes over from a service app. As service provider frontends may still permit users to pre-select their citizen country, the whole country selection must remain optional. In addition, making standardised country selection an optional process allows for smooth upgrade paths. Still, we are proposing a rather significant augmentation of established procedures, which is why country selection is discussed separately in more detail later on (see Section 3.4).

3.2 Operative Challenges

Having laid out the technical framework to implement mobile-first cross-border authentication, their consequences need to be considered. Most prominently, having an eIDAS app claim the URLs of all member states' connectors (and proxy services, for handling responses of eID apps) is utterly illusory:

- Every change in any national deployment (which could otherwise be handled by each country internally) would need to be communicated to a central entity to prepare a new release of the eIDAS app.
- Every connector and proxy service would require an asset links file to be deployed and maintained.
- Users would need to update the eIDAS app upon every deployment change, even though its functionality would not change.

Simply coordinating the first procedure amounts to an unmanageable task in reality, which is why we propose the introduction of a dedicated relay service, operated by a trusted entity. Details on how this strategy circumvents all of the above issues are provided in the following section.

[4]As per the documentation of the eIDAS Node Integration Package, the connector is a dedicated component to connect SPs to an eIDAS node, whose communication with an SP is to be customised to fit whatever protocol is natively supported by the SP.

3.3 Relay Service

The general idea for tackling operative challenges, is simply not having to deal with them. This can be achieved by taking the issue of claiming URLs of all involved parties out of the equation. Instead of directly issuing authentication requests to member-state-specific connectors and responding to member-state-specific proxy services, we introduce a single central relay service. In native app-to-app use cases, this relay service does not actually perform any operation. Its sole purpose, in this case, is to provide a single URL which must be claimed by the app. Services would not issue authentication requests to their country's native connector, but instead to the relay service, with the original request URL encoded into a URL parameter. By hosting an asset links file and the eIDAS app claiming this relay service's URL, the mobile operating system will invoke the eIDAS app whenever an authentication request is issued. The eIDAS app can easily decode the original request URL and subsequently interface with a country's eIDAS node in a fully transparent manner. This approach requires no functional changes to services, connectors, proxies and eIDAS nodes, but simply an altered configuration at the service provider to use the relay service as IdP.

Compatibility with browser-based authentication can be achieved easily as well: The only functionality the relay service needs to provide is a simple URL rewrite to perform the same operation as the eIDAS app. This can easily be achieved by few lines of pure client-side JavaScript[5], such that serving a static website is sufficient. As such, system requirements for running such a relay service are modest, while providing enough bandwidth is far more crucial. Such bandwidth requirements can be easily fulfilled, as demonstrated by various cloud service providers time and again.

3.4 Country Selection

The (optional) country selection mechanic mentioned in Section 3 amounts to a single exchange between either the SP app (if supported natively) or the MS-specific connector and the eIDAS app, as depicted in Figure 2. This should not come as a surprise, since the complexity of presenting the user with a list of countries to choose from is inherently limited. Still, a set of inherently required properties of both the message containing the list of supported countries can be derived:

- A protocol identifier for forward compatibility
- The actual list of supported countries, where each entry consists of:
  - Humanly readable country name
  - Country identifier (using the two-letter representation mandated by the eIDAS Light protocol is the obvious choice)
  - Potentially a country icon (flag)
- The target URL to transmit the selected country to
- A free-form data field (see below)

Similar requirements apply to the message conveying the user-chosen country:

- The country identifier
- A free-form data field (see below)
- An optional protocol identifier for forward compatibility

[5]or actual URL rewriting as part of a webserver configuration

In order to maintain high compatibility rates with existing deployment, stateful communication must be possible, which can easily be achieved through a free-form data field to store cookie and/or other session information, such that it can be piggybacked on the country selection process.

As will be laid out in the following section, we have implemented prototypes of all components and successfully deployed a demo service provider and connected to an unmodified national eID system, incorporating unmodified eIDAS nodes. More details on this implementation and the deployment are provided in the following section.

4 EVALUATION THROUGH IMPLEMENTATION

Successfully operating components managed by different entities across borders in concordance heavily relies on strictly following well-specified interfaces (as argued in Sections 2 and 3). As such, this section discusses concrete technical details on how and why different procedures were implemented in a certain way and thus provides a rationale as well as a technical reference – both of which are key for the concepts proposed in this paper to be implemented outside the well-defined safe space of a demo deployment.

4.1 Deployment Overview

On a high level, little changes are required compared to traditional deployments of components involved in cross-border authentication, although the deployment depicted in Figure 3 emphasises the use of native mobile service and eID apps. The real novelty is the dedicated eIDAS app and the clear separation of concerns this brings for the end user. As sketched by the dotted areas in Figure 3, each app has well-defined responsibilities and app-to-app interfaces are defined along clear borders, which, in turn, align with traditional back-end components. All involved entities communicate with each other by handing off requests and responses through redirects. Hence, more detailed representations of all these interactions are hard to comprehend, and even impossible to create, since no assumptions about the internals of SP systems and national eID systems can be safely made. Instead, the remainder section enumerates deployment assumptions and requirements for our proposed systems, while Section 4.2 provides more in-depth technical details on the interfaces and protocols used.

- eIDAS nodes of SP country and citizen country are peered; i.e. it is already possible for a citizen to identify themselves to a service provider operated in the SP country using their home country's eID system through eIDAS.
- Aside from peering, no further coordination between SP and citizen country is required.
- SP and connector support and use OIDC for handling authentication requests and responses.
- The national eID system also relies on OIDC to interface with the eID-facing portion of the proxy service.
- The protocols and interfaces between connector and eIDAS-node, eIDAS node and proxy, as well as among eIDAS nodes are unmodified and conform to the specification provided as part of the *eIDAS Node Integration Package*[6].
- Endpoints of the connector are not hardcoded in an SP's codebase, but can be reconfigured as desired.
- Endpoints of the proxy are not hardcoded in a national eID system's codebase, but can be reconfigured as desired.
- URL rewrite rules can be deployed on national eID systems and SPs to support the proposed relay service[7].

[6]https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eIDAS-Node+Integration+Package

[7]Technically, a simple reconfiguration of the endpoint should suffice, as URL-encoded authentication requests and responses remain untouched, but only authority and path need to be changed

- Country selection may happen either at the SP or the connector. The interface remains the same either way and the operation remains optional in general.

Based on this high-level overview, more intricate technical details, such as the concrete OIDC flow and the protocol specification of the country selection can be discussed. While some of the requirements and the following details may seem presumptuous at first glance, these are the result of analysing current best practices, recommendations and technical constraints. In addition, some of the protocol specifications are simply taken from production SPs and production eID systems.

4.2 Interface and Protocol Specifications

In general, a bijective mapping between various OIDC flavours and the eIDAS 2 profile must, by definition, exist. After all, various production systems already harness the combination of both. Austria's next-generation eID platform, called *ID Austria*, for example, supports OIDC. Estonia also relies on OpenID Connect for their eID platform, as reflected by the technical documentation of the *TARA* authentication service[8]. Technical details on our proposal for using OIDC are provided in Section 4.2.1.

As for the country selection, our work obviously needs to be compatible with existing ways of SPs communicating with eIDAS node connectors. Since each country has some peculiarities in this setup, our solution needed to remain generic enough to allow for integration in such a diverse landscape. Germany, in particular, follows a very special way of using eIDAS nodes. This has already presented challenges in the past, which is why we approached the German company *ECSEC GmbH*[9] for feedback on our concept to make sure that our ideas would work in practice. This exchange ultimately led to the technical specification of a generic IdP selection API, which integrates well with cross-border authentication to provide a unified country selection experience to users. The full protocol specification is provided as part of Section 4.2.3.

4.2.1 OIDC. OpenID Connect specifies different so-called *flows* for authenticating users. Since this work discusses cross-border authentication within the realm of e-government, security is crucial. Hence, we follow current best practices and recommendations from standardisation bodies when it comes to the security properties of different flows. Consequently, we recommend service providers and eID systems, as well as eIDAS node connectors and proxy services to implement the *authorization code flow* in accordance with the current specification [10]. Figure 4 provides a graphical overview of this procedure, using terminology as defined by the OIDC specification.

While the wording used in Figure 4 based on *IBM Security Verify Access* documentation[11] may seem foreign in the context of this work, it resonates with the official OIDC specification and various available documentation resources, libraries and tutorials. We hence refer so such material for an in-depth technical insight. On a conceptual level, the auth code flow essentially states that user information is not returned directly in an authentication response. Instead, an SP backend must actively fetch this information directly from the eIDAS node connector based on an authorization code obtained through the client (i.e. the eIDAS app). The connector then responds with all requested information. This same flow is mandated between an eIDAS node connector and the national eID system in the citizen country.

So far, no details on the mapping between the eIDAS SAML2 profile (or more, precisely, the `Light` protocol used between the eIDAS node and connector/proxy) were discussed. Although this may seem like essential information, it is

[8]https://e-gov.github.io/TARA-Doku/TechnicalSpecification
[9]https://www.ecsec.de/en/home/
[10]https://openid.net/specs/openid-connect-core-1_0.html#CodeFlowAuth
[11]https://www.ibm.com/docs/en/sva/9.0.5?topic=SSPREK_9.0.5/com.ibm.isam.doc/config/concept/con_oauth20_workflow.htm

actually not: Since the eIDAS app orchestrates communication between eIDAS nodes, connector, and proxy service, it is not required to interpret and operate on any of these data structures. Instead, these tasks are still handled by the involved backends. As such, the eIDAS app is, by definition, compatible with member state systems utilising OIDC, as it is capable of handling redirects. In effect, this mirrors the behaviour of the eIDAS nodes translating between `Light` protocol messages and SAML messages. A key requirement for this to work is having the eIDAS app claim all relevant connector, proxy service, and eIDAS node endpoint URLs through the relay service.

4.2.2 Relay Service. We propose a simple yet effective implementation of the relay service. As argued before, the relay service is not actually involved, whenever the eIDAS app is installed on a client device. For compatibility with browser-based authentication flows, however, a basic redirection logic is required. This logic itself is defined as follows:
redirect_url = `https://`*authority*`/#`*original_target_url*
original_target_url is simply the original version of the URL, which would be used without the relay service.

Such a simple redirection is trivial to implement both as part of a smartphone app, as well as in a pure JavaScript function, which is served as part of an otherwise static placeholder page served by the relay service, if accessed through a regular web browser.

4.2.3 Country Selection. The country selection's API spec, as implemented on the eIDAS node connector, has been made available by *ECSEC GmbH* [10] and works as follows: First, a GET request is sent to the endpoint serving the country options. We piggyback the session information on the country selection by appending it as a query parameter. If the request header contains the `Accept: application/json` field, the endpoint responds with a JSON response. An example for such a response is depicted in Listing 1. If the sender does not specify that it can accept JSON in the request header, the endpoint will respond with a rendered HTML view containing a list with available countries.

For the country flags, we provide base64 encoded PNG images instead of URLs. This has the advantage that the flags are provided in the response and no further requests have to be sent.

Listing 1. Response with available countries

```
{
"get_options":
 {
 "profile":"GetOptions",
 "display_options":
 [
   {
     "display_type":"option",
     "display_data":
     {
       "en":
       {
         "country":["AT"],
         "loa":null,
         "name":"AT",
         "description":"Austria",
         "logos":
            [
              {
                "type":"pixel",
                "url":"iVBORw0KGgoAA ...",
                "mimetype":"image/png",
                "width":30,
                "height":40
              }
            ]
          }
        },
        "option_id":"AT"
      },
      {
        "display_type":"option",
        "display_data":
        {
          "en":
          {
            "country":["DE"],
            "loa":null,
            "name":"DE",
            "description":"Germany",
            "logos":
            [
              {
                "type":"pixel",
                "url":"iVBORw0KGgoAA ...",
                "mimetype":"image/png",
                "width":30,
                "height":40
              }
            ]
          }
        },
        "option_id":"DE"
      },
    ],
    "options":
    [
      {
        "id":"AT",
        "activation_type":"Browser",
        "type":"EID",
        "protocol":"eIDAS",
      "issuers":[]
    },
    {
      "id":"DE",
      "activation_type":"Browser",
      "type":"EID",
      "protocol":"eIDAS",
      "issuers":[]
    },
  ]
  }
}
```

The eIDAS app displays the options to the user and sends the response to the `select` endpoint. An example of such a response is depicted in Listing 2. Note that the request contains the session information. The eIDAS node connector uses this information to initiate the authentication flow with the selected countries eIDAS node.

Listing 2. POST request with selected country

```
{
  "session": "c2NvcGU9b3BlbmlkJmNsYWlt ...",
  "selected_option": "AT"
}
```

5 CONCLUSIONS AND OUTLOOK

Cross-border e-government will only become more important and smartphone-centric user habits are also here to stay. This work showed a way forward, respecting user habits, as well as established infrastructure. As such, we have presented a framework, which augments eIDAS-based cross-border authentication in a mobile-first manner by introducing a dedicated eIDAS app. In addition, this work also introduces a standardised way to implement a country selection mechanic. On the one hand, this enhancement means that service providers are not required to implement such procedures individually. On the other hand, this has additional potential to improve user experience: For future work, the eIDAS app can be extended to remember a citizen's home country and thus completely remove the need for manually selecting one's home country after initially providing this information.
We have also tackled the fundamental technical issue of coherent app-to-app communication, such that it feels native without requiring the coordination of multiple governing bodies. We achieve this through the introduction of a dedicated relay service, which is leveraged for this sole purpose.

All the functionality presented here is aimed at providing a smooth migration path of traditionally browser and desktop-oriented workflows towards a mobile-first user experience, while maintaining compatibility. We have demonstrated the feasibility of our approach by implementing, deploying, and successfully operating all involved components. The potential of a dedicated eIDAS app, however, goes beyond established cross-border authentication. In effect, the eIDAS app can be extended to also act as a local service app towards mobile wallets. In doing so it becomes possible to bridge the gap between upcoming wallet-based authentication concepts and established ones. This enables the user to authenticate using a mobile wallet on their smartphone, while the eIDAS app translates this authentication information into a format, which is compatible with existing service providers. Initial experiments on this matter show promising results and highlight the potential of a dedicated eIDAS app far beyond providing a mobile-first user experience. Still, further investigation and thorough evaluation on this matter is required. This, however, is out of this work's scope.

ACKNOWLEDGMENTS

This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 959072, project mGov4EU. The paper relies and is based on work conducted in this project.

REFERENCES

[1] Android Open Source Project. 2023. Create Deep Links to App Content. Retrieved January 19, 2023 from https://developer.android.com/training/app-links/deep-linking

[2] Android Open Source Project. 2023. Verify Android App Links. Retrieved January 19, 2023 from https://developer.android.com/training/app-links/verify-android-applinks

[3] Apple, Inc. 2023. Allowing apps and websites to link to your content. Retrieved January 19, 2023 from https://developer.apple.com/documentation/xcode/allowing-apps-and-websites-to-link-to-your-content

[4] Apple, Inc. 2023. Supporting associated domains. Retrieved January 19, 2023 from https://developer.apple.com/documentation/Xcode/supporting-associated-domains

[5] Sundas Choudry. 2021. Why You Wouldn't Use SAML in a SPA and Mobile App. Retrieved January 19, 2023 from https://www.identityserver.com/articles/why-you-wouldn-t-use-saml-in-a-spa-and-mobile-app

[6] T. Bray (Eds.). 2017. The JavaScript Object Notation (JSON) Data Interchange Format. Retrieved January 19, 2023 from https://www.rfc-editor.org/rfc/rfc8259

[7] OpenID Foundation. 2014. Welcome to OpenID Connect. Retrieved January 19, 2023 from https://openid.net/connect/

[8] IETF OAuth Working Group. 2012. OAuth 2.0. Retrieved January 19, 2023 from https://oauth.net/2/

[9] IBM. 2021. SAML 2.0 bindings. Retrieved January 19, 2023 from https://www.ibm.com/docs/en/sva/9.0.4?topic=federations-saml-20-bindings

[10] Mike Prechtl. 2022. options-api. Retrieved January 19, 2023 from https://git.ecsec.de/mike.prechtl/options-api

[11] Christof Rath, Simon Roth, Manuel Schallar, and Thomas Zefferer. 2014. A Secure and Flexible Server-Based Mobile eID and e-Signature Solution. In *The Eighth International Conference on Digital Society*. ., 7–12. The Eighth International Conference on Digital Society ; Conference date: 23-03-2014 Through 27-03-2014.

[12] N. Sakimura, J. Bradley, and M. Jones. 2014. OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 1. Retrieved January 19, 2023 from https://openid.net/specs/openid-connect-registration-1_0.html

[13] N. Sakimura, J. Bradley, M. Jones, and E. Jay. 2014. OpenID Connect Discovery 1.0 incorporating errata set 1. Retrieved January 19, 2023 from https://openid.net/specs/openid-connect-discovery-1_0.html

[14] OASIS Security Services TC. 2008. Security Assertion Markup Language (SAML) V2.0 Technical Overview. Retrieved January 19, 2023 from http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

[15] Thomas Zefferer. 2014. A Server-Based Signature Solution for Mobile Devices. In *The 12th International Conference on Advances in Mobile Computing and Multimedia*. ., 175–184. 12th International Conference on Advances in Mobile Computing & Multimedia (MoMM2014) ; Conference date: 08-12-2014 Through 10-12-2014.

[16] Thomas Zefferer, Fabian Golser, and Thomas Lenz. 2013. Towards Mobile Government: Verification of Electronic Signatures on Smartphones. In *Technology-Enabled Innovation for Democracy, Government and Governance*. ., 140–151. International Conference on Electronic Government and the Information Systems Perspective ; Conference date: 26-08-2013 Through 30-08-2013.

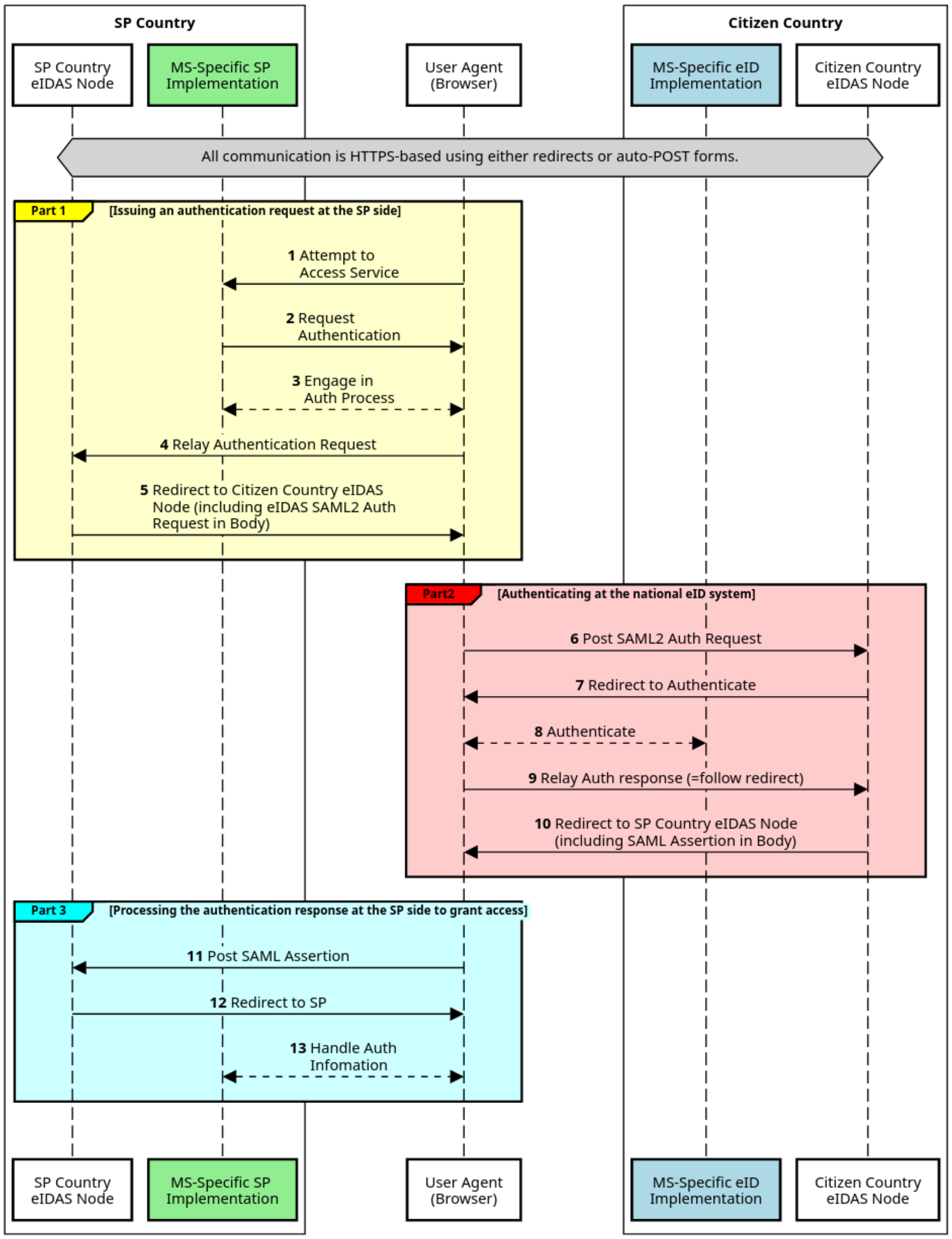
SP Country
Citizen Country
SP Country eIDAS Node
MS-Specific SP Implementation
User Agent (Browser)
MS-Specific eID Implementation
Citizen Country eIDAS Node
All communication is HTTPS-based using either redirects or auto-POST forms.
Part 1 [Issuing an authentication request at the SP side]
1 Attempt to Access Service
2 Request Authentication
3 Engage in Auth Process
4 Relay Authentication Request
5 Redirect to Citizen Country eIDAS Node (including eIDAS SAML2 Auth Request in Body)
Part2 [Authenticating at the national eID system]
6 Post SAML2 Auth Request
7 Redirect to Authenticate
8 Authenticate
9 Relay Auth response (=follow redirect)
10 Redirect to SP Country eIDAS Node (including SAML Assertion in Body)
Part 3 [Processing the authentication response at the SP side to grant access]
11 Post SAML Assertion
12 Redirect to SP
13 Handle Auth Infomation

Fig. 1. Abstract Cross-Border (eIDAS) Authentication

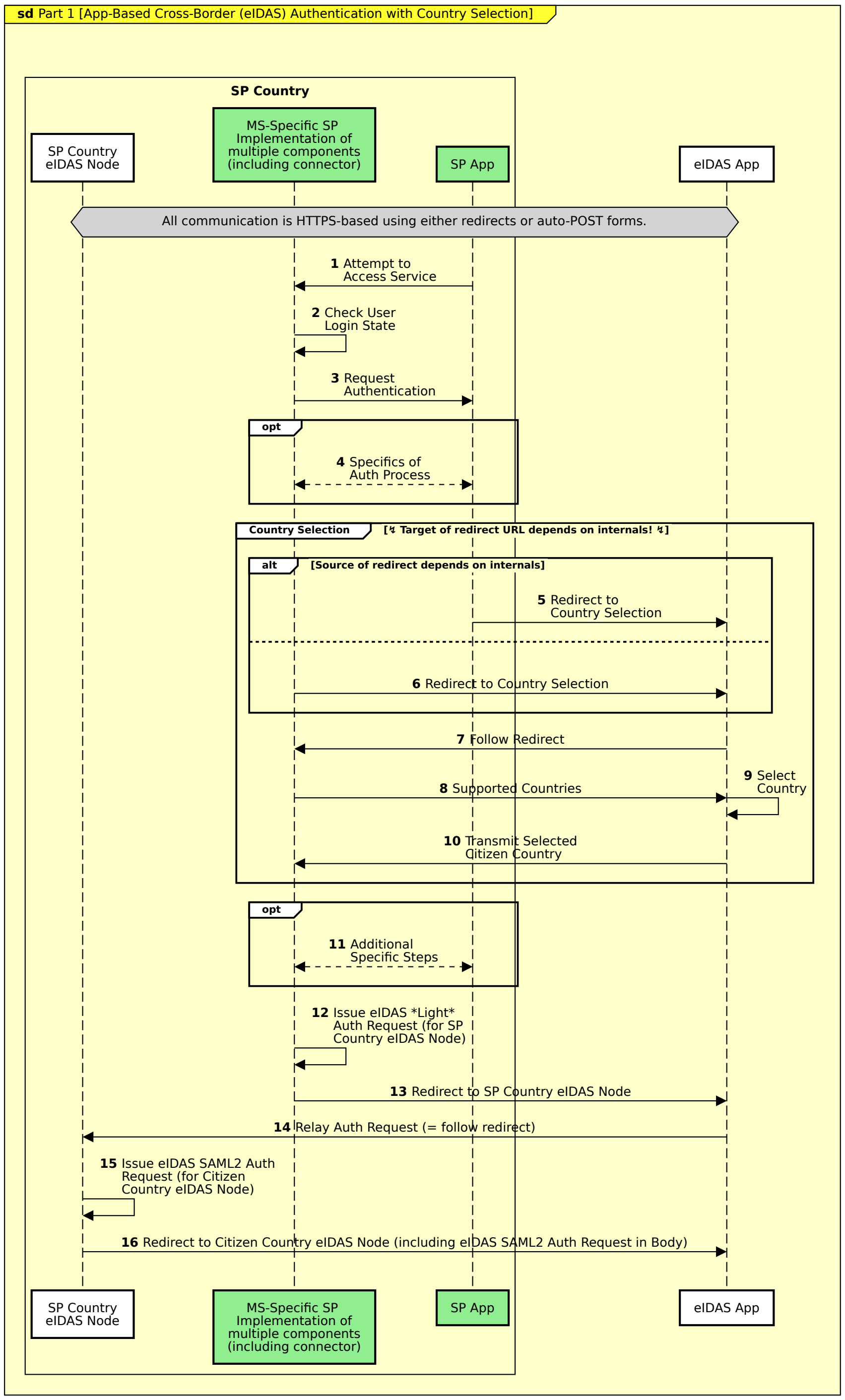
sd Part 1 [App-Based Cross-Border (eIDAS) Authentication with Country Selection]
SP Country
SP Country eIDAS Node
MS-Specific SP Implementation of multiple components (including connector)
SP App
eIDAS App
All communication is HTTPS-based using either redirects or auto-POST forms.
1 Attempt to Access Service
2 Check User Login State
3 Request Authentication
opt
4 Specifics of Auth Process
Country Selection
[↯ Target of redirect URL depends on internals! ↯]
alt
[Source of redirect depends on internals]
5 Redirect to Country Selection
6 Redirect to Country Selection
7 Follow Redirect
8 Supported Countries
9 Select Country
10 Transmit Selected Citizen Country
opt
11 Additional Specific Steps
12 Issue eIDAS *Light* Auth Request (for SP Country eIDAS Node)
13 Redirect to SP Country eIDAS Node
14 Relay Auth Request (= follow redirect)
15 Issue eIDAS SAML2 Auth Request (for Citizen Country eIDAS Node)
16 Redirect to Citizen Country eIDAS Node (including eIDAS SAML2 Auth Request in Body)
SP Country eIDAS Node
MS-Specific SP Implementation of multiple components (including connector)
SP App
eIDAS App

Fig. 2. App-based issuance of a cross-border authentication request, optionally supporting country selection

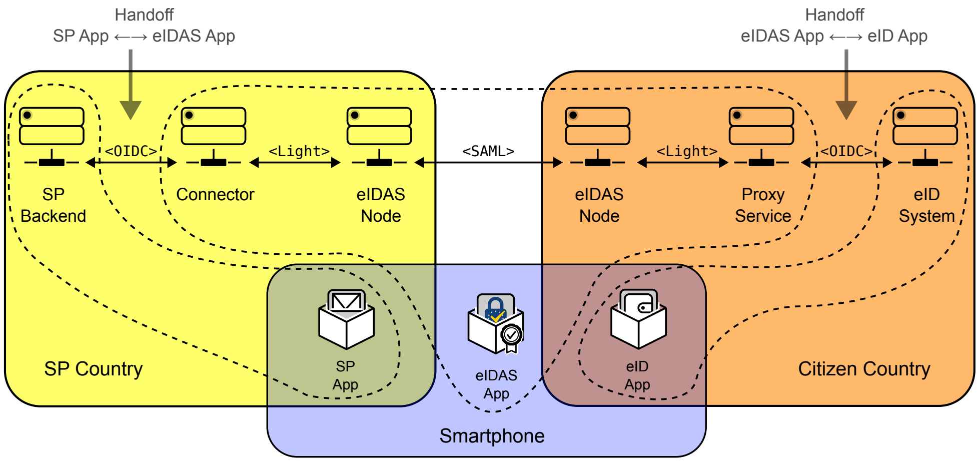

Fig. 3. Deployment and operation of involved components. Dotted areas indicate which app is responsible for interfacing with certain components. Two areas bordering each other indicate an app-to-app handoff.

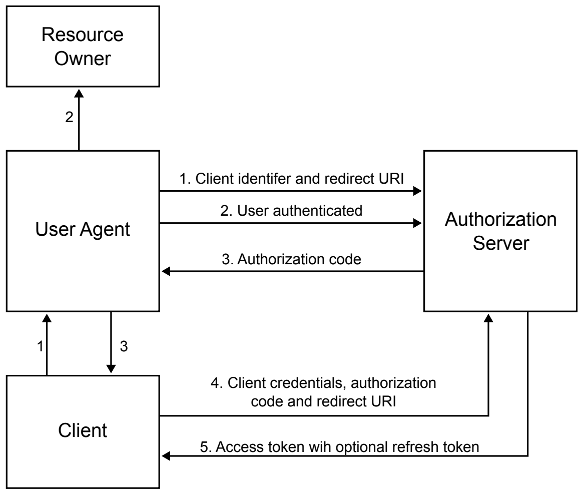
Resource Owner
2
User Agent
1. Client identifer and redirect URI
2. User authenticated
3. Authorization code
Authorization Server
1
3
4. Client credentials, authorization code and redirect URI
5. Access token wih optional refresh token
Client

Fig. 4. Steps of the OIDC auth code flow based on *IBM Security Verify Access* documentation. The original source provides all technical details on each step depicted here.