Federated User Experience

Federated User Experience Guidelines

Probably the most difficult aspect of federation is producing a good user experience. With many applications, the use of SSO or externalized authentication from even one source can introduce user interface complexity. Federation adds additional considerations that can easily lead to a bad or confusing user experience.

Complicating this issue, applications leverage federation in very different ways, and the lack of experience with the technology has led us down a number of dead ends (or at least suboptimal paths) that have left deployers with a lot of choices and confusing advice over the last decade.

Note that the bulk of this material represents a "snapshot in time" based on the currently prevalent user interface metaphors and capabilities in wide use. Adjustments for mobile devices, lacking screen size or pointing devices, as well as for touch screen devices are inevitable. Some of the material reflects the limitations of such interfaces already.

Elements of the Federated User Experience

Services share a mostly common set of user experience elements with respect to federation, even though there are specific considerations (or entire elements missing) in some cases.

- Initiating the Login Process
- Discovery of the IdP
- Login at the IdP
- User Consent
- Outcome

We elaborate on these steps below, but the overarching principle to be understood is *consistency*, in two senses.

First, consistency across services: studies have shown that the lack of consistency across SPs is the single largest source of confusion for users. People can accomodate fairly ugly workflows if they're familiar ones, but even simple workflows that are "new" create barriers. This is something that can be difficult to convince application owners, who frequently think they have a special need to be different or to stand out. Experience suggests that such an approach incurs costs in user training and user avoidance, and federation is often a scapegoat.

Second, consistency of "story" throughout the process. Evidence suggests that users react best when the transition between steps of the process maintains a degree of consistency with respect to language and presentation. Users want the context of the overall action (logging into the SP from the IdP) to be clear at each step. If the UI changes in a jarring way, and/or lacks a visual and/or textual reference back to the overall goal/activity, they get confused and lost more easily. A number of new software features and federation initiatives (e.g., User Interface Elements) relate to this need, so we know improvement will be gradual.

Initiating Login

Recommended Practice

- A "Login" link is placed in the upper right corner.
- The main application screen is uncluttered by choices of different login mechanisms.

Sites have demonstrated a substantial degree of "creativity" in presenting users with the option of federated login. This is defensible because of the wide array of "starting points". Some applications are starting from scratch, but most have preexisting mechanisms, often numerous such mechanisms, that have to be supported at the same time. The trap, however, is that treating federation as a secondary (or worse) option is a self-fulfilling prophecy. Federation will always compare poorly to an integrated login experience by number of clicks, steps, etc., but users aren't counting clicks, they just want it to be obvious what to do.

The solution advocated here is a consistent experience of one discrete "Login" link in the upper right hand corner of a site. Login options that involve user interaction (e.g., not IP-based methods) should be accessible through that one starting point, with the user indicating what method to use through a "discovery" process (the next step). It is important, both for security and consistency reasons, that "local" login options, however prevalent, be presented on an equal footing to federated options.

Discovery

The bane of federation historically has been the need for user selection of the IdP to use, something that is largely unknown to single-domain authentication, with a few minor exceptions (e.g., selecting a login domain on a Windows desktop). In the past, progress was stunted by an overemphasis on **avoiding** discovery, using cookies to minimize the number of times users would encounter it, and a lack of good tools for implementing scalable approaches to the problem.

URL-Based Discovery and Deep Linking

Recommended Practice

- Application resources shared among users from multiple home organizations can access those resources with stable, authentication-neutral URLs.

First, note that there are scenarios in which discovery can be avoided. In particular there are applications that, while federated, are designed as a silo in which access to the system self-determines the IdP to use. This is common in outsourced/hosted scenarios. The IdP can be implicit based on the URL (hostname/domain or path). In other cases, access to the application requires users to start at a link that initiates the login process from the IdP and "pushes" the client into the site.

While superficially elegant, this is a model to avoid (or at least not to use exclusively) if there is ever a possibility that the same resources will be shared by users from different organizations. The inability to directly address resources in an authentication-neutral way with a single URL is a limitation that will frustrate users deeply. So-called deep linking isn't just a good idea, it's the basis of the web. Mechanisms that bypass discovery are acceptable, but not to the exclusion of this capability.

General Guidelines

Recommended Practice

- Discovery either overlays the application (an embedded or pop-up design), or includes contextual information identifying the service accessed by the user.
- Different login options/mechanisms, including federated IdPs, are presented uniformly to the user.
- Preferred or remembered choices are highlighted, but not automatically chosen (i.e., no automatic "Use this choice next time" behavior).
- Dynamic search via text box is the primary interface for general selection.
- Help and "go back" links are available.

Earlier it was suggested that all login mechanisms be offered behind a single entry point. The discovery interface is that entry point; the user should be presented with an interface to select from among all the possible login mechanisms, including federated IdPs. This interface can be presented as an "overlay" or pop-up that supplants, without replacing, the underlying application. Alternatively, a traditional "full page" interface can replace the application, allowing for more room and fewer accessibility challenges.

The discovery component may be part of the application or SP, or a separate service. The former enables a more consistent UI experience (color, style, etc.), while the latter enables work to be offloaded and shared by multiple services with identical or at least compatible discovery needs. In either case, the UI should remain as consistent with the originating application as possible. If the UI replaces that of the service, the UI should reference the service with some combination of descriptive name, icon, URL, etc. This provides the user with context for the selection of a login mechanism or IdP. When implemented separately from the service itself, SP user interface elements in federation metadata can supply the UI with this information.

As an interface, discovery should be minimal, focused on the task at hand. Past selections, as well as common or predominant choices, should be offered prominently at the top for easy selection. For example, an application used mostly by users from one organization can offer that choice by default to all users to make selection easy for the widest audience. It also makes sense to directly offer options that don't lend themselves to easy identification by the user (e.g., local accounts). However, do not assume that a previous choice is the only choice; give the user the chance to select it again, or make a different one; this is essential when mistakes are made or shared machines are used, or when users have multiple accounts.

Other choices (i.e., the "rest") should be selectable primarily through a dynamic search text box, akin to most search engines. This scales well and is simple to use provided the user can enter a variety of terms, such as geography, organization names, mascots/brands, etc. If implemented consistently, one successful "find" is sufficient to teach a user how to find their choice somewhere else. When multiple results are found and displayed, tool tips containing more extensive descriptive text might be used to disambiguate like-named choices (but beware of mobile devices that don't support mouseover events). Icons may also be used in some cases. IdP user interface elements in federation metadata can help supply the UI with this information.

While it is attractive to present "a complete list of options", bear in mind that this limits the scalability of the service. This may be appropriate for services that limit the number of IdPs they support, but is not suitable for "promiscuously" federated services that want to support the widest number of choices possible. Design for success, not failure.

It may be appropriate to leverage information about the client, such as IP address or geolocation data, as hints to aid discovery (e.g., to determine a default set of choices to favor). Hints work well, just not as a replacement for explicit user search/selection.

Finally, the UI should include a link to explanatory information about the purpose of the process and how to search, as well as a way for the user to signal "no choice" and return to the application. There should be no "dead ends."

The Boarding Problem

Recommended Practice

- The choice of IdP is not artificially limited, but left open to selection of any trusted option.

In the context of discovery, a brief word on the implications of privacy on federation. Traditionally, most IdPs take an active stance with respect to user privacy, to the point that releasing information to SPs outside of explicit agreements is rare. An exception to this approach is one of "consent" (discussed later in more detail), allowing users to opt-in to release of data. This is uncommon today, but is likely to be more common in the future.

The upshot is that today, the general case is that any given IdP is unlikely to "just work" if selected by a user accessing a service unknown to the operators of the IdP. This leads to one of two conclusions:

- Discovery should only make available IdPs that are known or reasonably expected to work.
- Discovery should make any trusted/acceptable IdP available, leaving the application to handle the case of privacy controls preventing success.

Historically, the first option was considered more attractive, because it unburdens applications from some additional error handling. In practice, the need to "board" IdPs to get them added to the Discovery list by proving successful use created much more work for deployers than error handling would. It also confused users by creating "dead ends" that failed to offer them expected choices, depending on the service being accessed. This lack of consistency, recall, has been highlighted as a problem.

Therefore, the second choice, one of asking "forgiveness" rather than permission seems the better option. The cost is that applications (or one's federation software) **MUST** handle the case of insufficient data to obtain access in a manner that is acceptable to users. This is discussed in more detail in the **Outcome** section below.

In parallel, it is critical for IdPs to address the problem of defining an effective and accessible Attribute Release Process.

Mistakes were Made

Historically, discovery used to be handled by a stand-alone application, often called a "WAYF", containing lists of IdPs limited to single federations like InCommon. The big advantage to this approach is simplicity for SPs, and the sharing of a cookie to remember selections on behalf of multiple SPs. The latter is a hugely overblown feature that is undercut by structural failings. Limiting the choice of IdP to one federation rarely addresses the requirements of most services; often the limitation was a consequence of this approach to discovery rather than any inherent decision to do so by the service.

A more subtle failing is that the choice of IdP became available only after reaching the WAYF screen via an option with no good label: federated login? InCommon? Shibboleth? None are understandable by the poor end user.

Login at the IdP

Recommended Practice

- Login pages identify the service requesting authentication.
- Applications use full-frame windows to present the IdP's interface, or at least full "chrome" in the sense of title bars, menus, location bars, etc.

The user interface of the IdP is not for the most part within the reach of standardization, but one can apply lessons learned to achieve a better, more consistent approach that will feel more natural to users. The most essential point is that the "story" of the login be maintained by identifying the service involved using a name, descriptive text, an icon, etc. SP user interface elements in federation metadata can supply the UI with this information.

Another consideration is that reducing phishing is best achieved by ensuring the IdP page is visible within a complete browser window so that SSL-protected location information can be verified by the user (no laughter please, it's still the right thing to do).

User Consent

Recommended Practice

- IdPs that seek broad usage provide a mechanism for users to opt-in to the release of personally identifiable information to SPs without prearranged contracts/agreements.
- Consent pages identify the service requesting the information and its privacy policy.
- Consent pages are kept as short and simple as possible. Users are not asked to consent to the release of complex data they're unlikely to understand.

For an IdP to be broadly useful, the decision whether to allow authentication to proceed and particular attributes be released must be automated in some fashion, rather than in the hands of university staff. One popular mechanism is to add an interface to collect the user's consent to the transaction before allowing it. There is likely to be substantial variability in consent UI in the near term, but there are some obvious general principles that are likely to emerge.

In keeping with earlier recommendations, a consent UI must identify the service receiving the information. SP user interface elements in federation metadata can supply the UI with this information, including links to privacy policies that are obviously relevant to the act of consent.

Common sense also dictates that users need to clearly understand what they're consenting to. Keeping the text to a minimum, and avoiding overly technical descriptions of attributes that won't make sense to most users is important.

For some IdPs, it may be sufficient for some users to merely offer a generic "terms of use" agreement asking for permission to release a typical set of information to SPs that guarantee appropriate protection and use of the information. This is much less technically complex to deploy.

Error Handling

Obviously the outcome of the whole process is either success or failure. Success is easy: simply give the user what they asked for. Failure is more complex, since there can be many causes and degrees of failure to handle.

The case of outright technical failure is discussed in the Error Handling topic. The Federated Error Handling topic discusses failure due to access control issues and the hand-off between IdPs and SPs.