SeamlessAccess.org
User and Data Flows

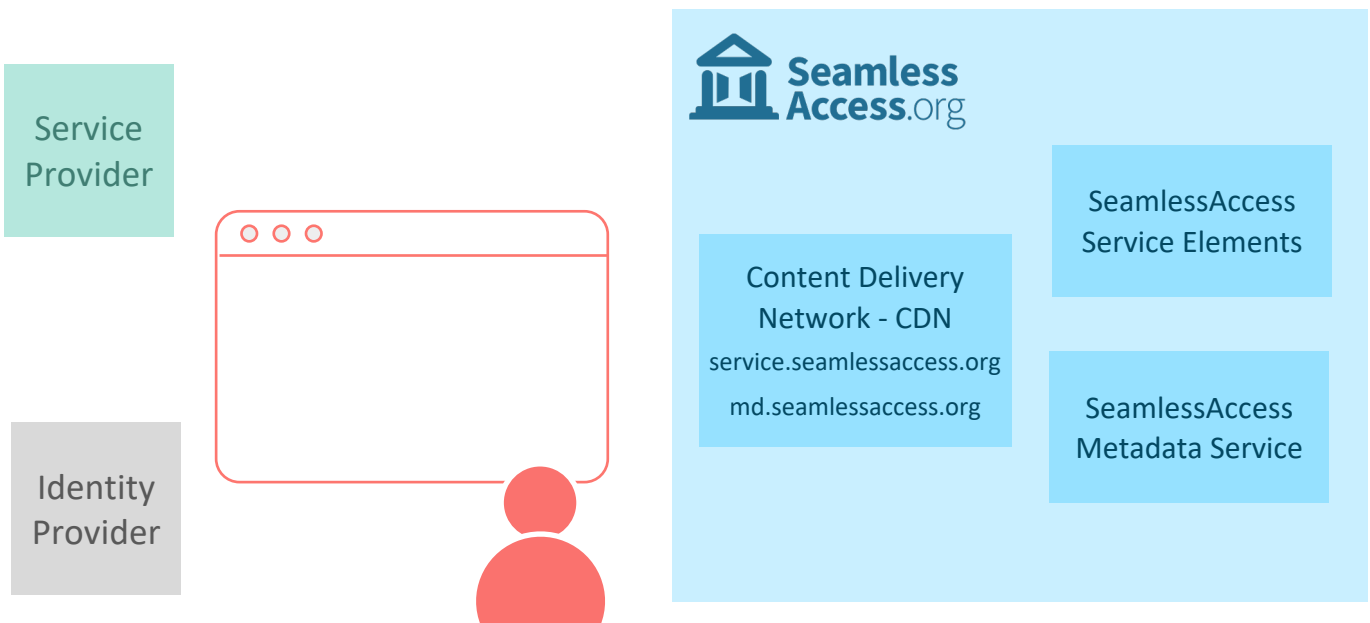
This document explains what happens ”under the hood” for three variants of SeamlessAccess implementation: **Limited**, **Standard** and **Advanced**.

For each implementation, one possible user journey is described. For brevity, these diagrams show overall functionality but are not intended as technical whitepapers.

Color coding is used to show different actors and the flows they initiate (green for Service Provider, blue for Seamless Access, pink for calls initiated by the browser or user action).

The following layout is used to show user journey:

- **User and browser** in the middle, showing various calls and content loaded
- **SeamlessAccess site** on the right, showing the high-level architecture: CDN frontend (serving service.seamlessaccess.org and md.seamlessaccess.org) and backend service elements
- Cylinders are used to document the resulting user data flows



Seamless Access.org

Limited Integration

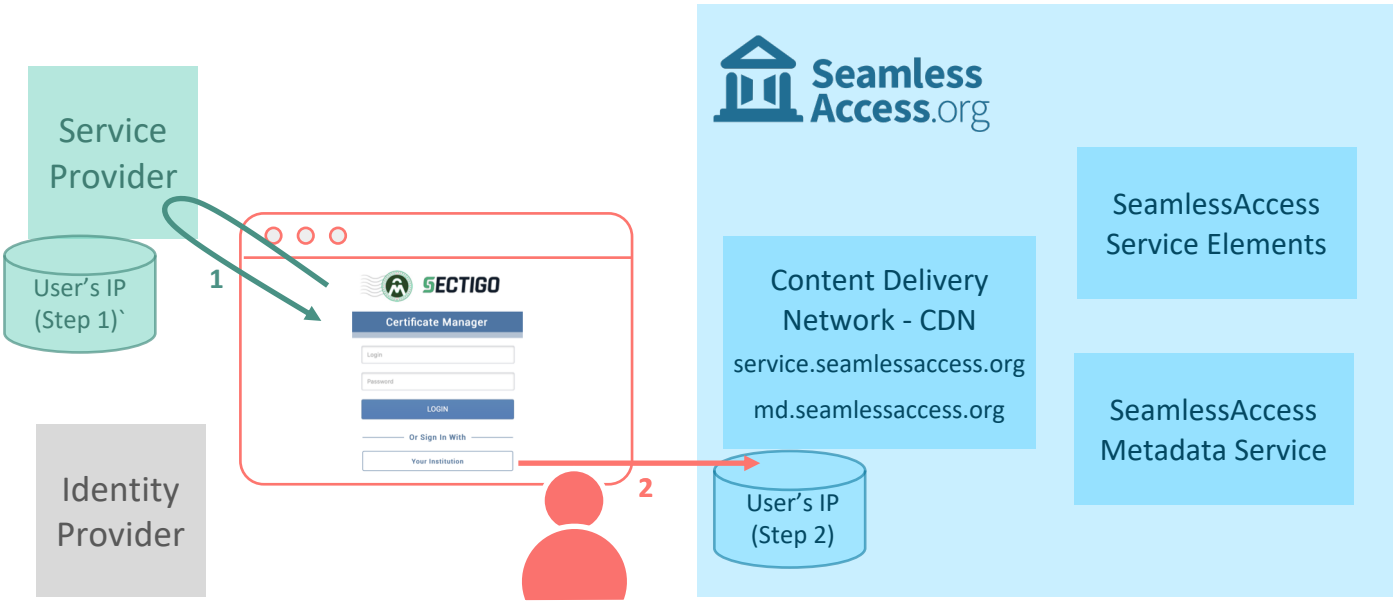
No previously remembered institution

Example from GÉANT TCS service implementation

SeamlessAccess.org

Limited Integration, no previously remembered institution

Phase 1 - SP site login button



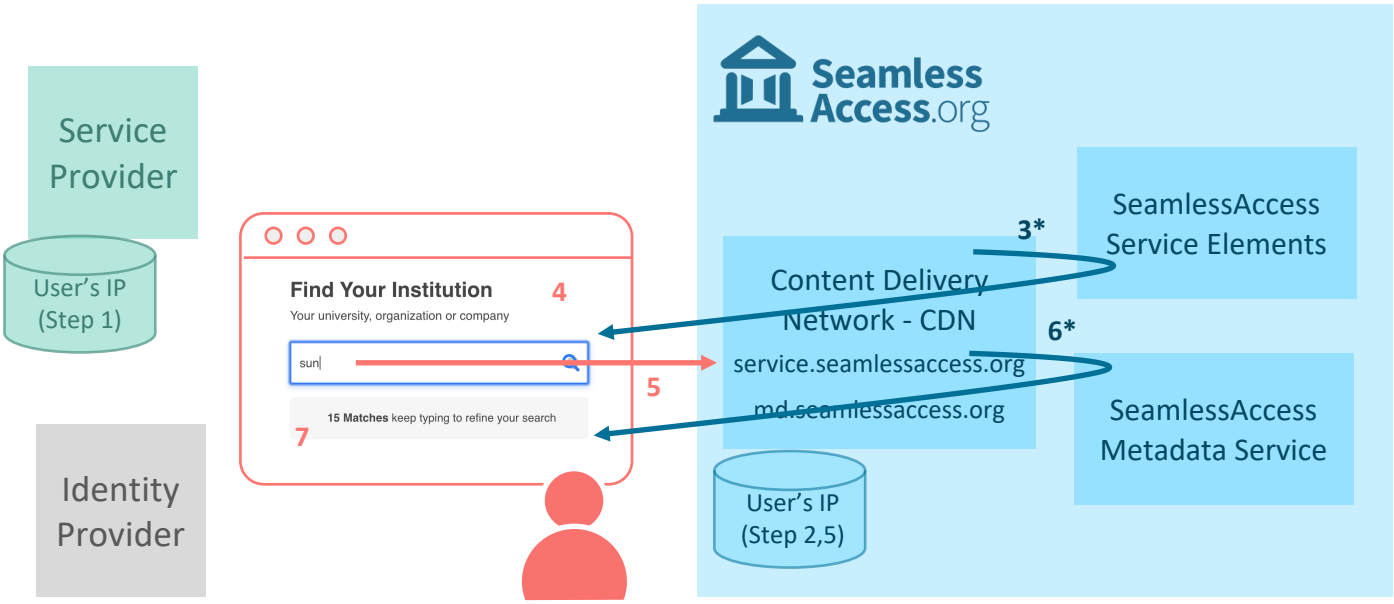
1. User accesses the Service Provider's (SP) site, which will likely log user's IP address.

2. On user's request to login, the SP site redirects user to **service.seamless.access.org/ds** which resolves to CDN used by SeamlessAccess. The CDN provider records user's IP address according to its policy.

Seamless Access.org

Limited Integration, no previously remembered institution

Phase 2 - Discovery and Persistence service, finding institution



3. The CDN is setup to fetch **origin-service.seamlessaccess.org,** hosted by the SeamlessAccess Service Elements. It serves the JavaScript used for the Discovery and Persistence services.

4. These JavaScripts are used to render the Discovery Service user interface. The Persistence Service JavaScript **reads the browser local storage** to show (up to the last three) institution choices that user has previously made. In this scenario, there are no institution choices remembered.

5. User starts to search for an institution by typing its name. As characters are entered, the call to **md.seamlessaccess.org/entites/?q=string** is triggered. This redirects to the CDN, which is also serving the **md.seamlessacess.org**.

6. The CDN uses the load balancer to query one of the SeamlessAccess Metadata Service nodes in the backend, which will answer with the list of matching institutions. Since this list will initially be too large, it will return a number of matches only.

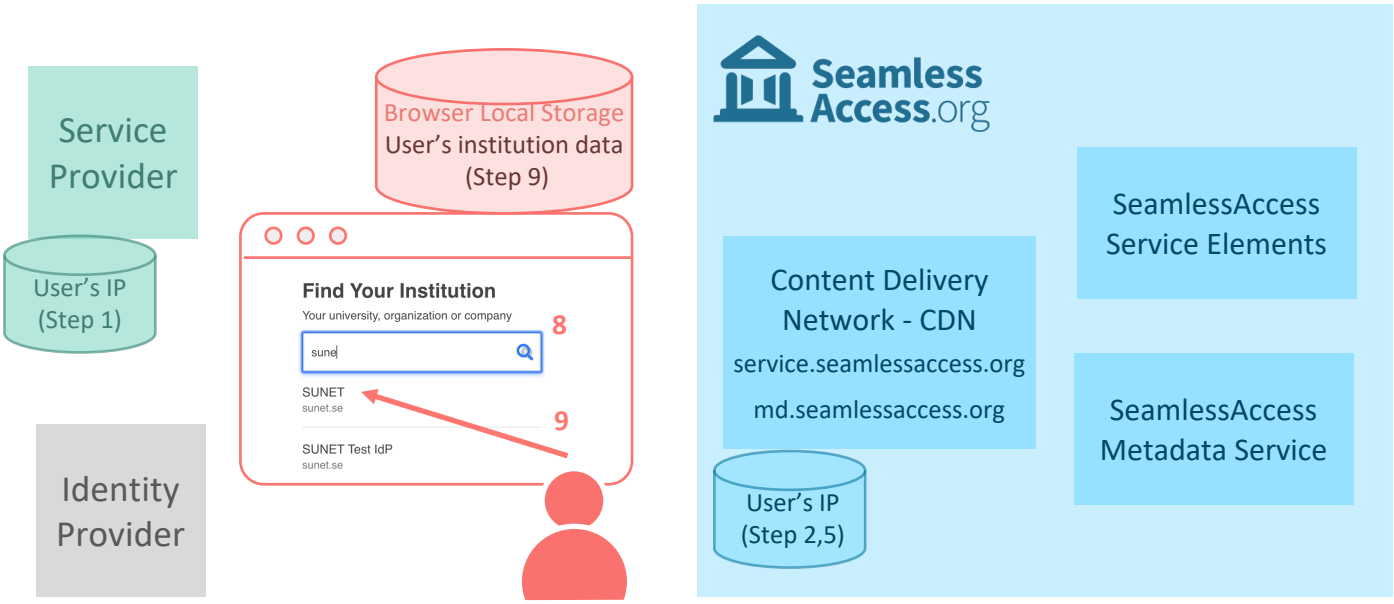
7. The Discovery service renders the message: "***N Matches*** *keep typing to refine your search*", so user repeats Steps 5 & 6 until the list of matching institutions small enough.

*In most cases, the CDN will have already cached this information and will not need to query the SemlessAccess backend

Seamless Access.org

Limited Integration, no previously remembered institution

Phase 3 - Discovery and Persistence service, choosing institution



8. Once the number of matching institutions is small enough, the metadata query in Steps 5 & 6 will return the list of matching institutions. The Discovery service in Step 7 will display this list.

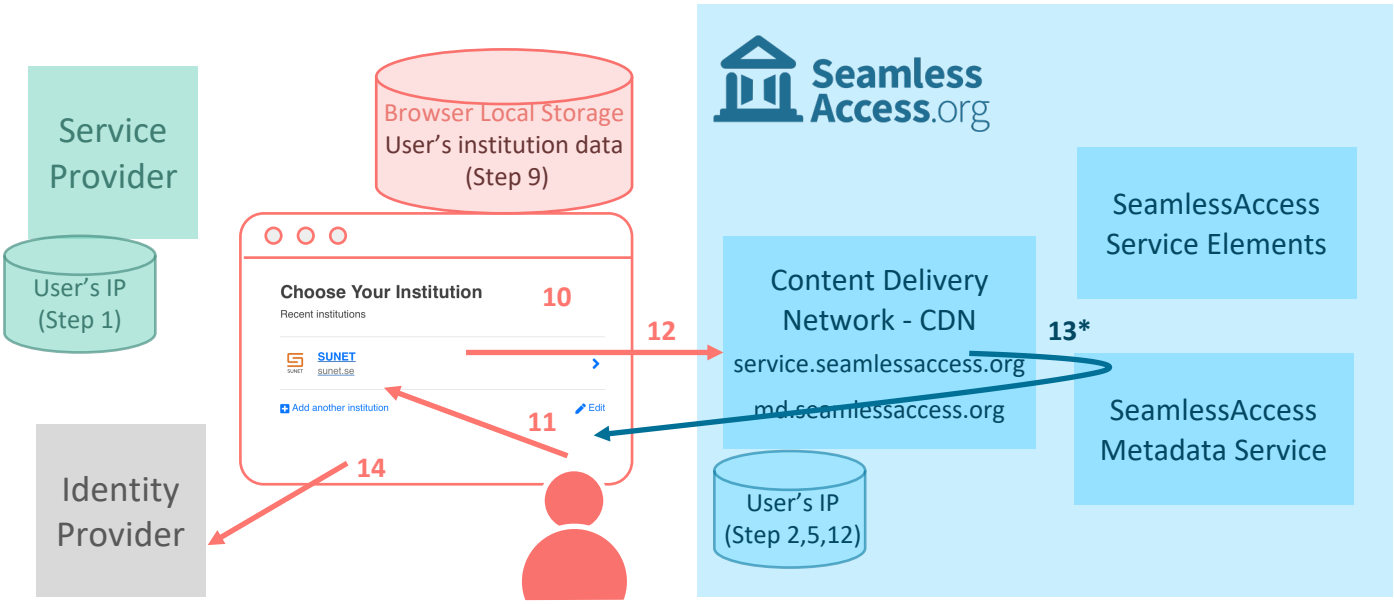
9. User clicks to select an institution, triggering the Persistence service to **write the chosen institution to the browser local storage**. Following information is written (example from SUNET):

```
entity: {title: "SUNET", descr: "Login for SUNET employees", auth: "saml",…}
      title: "SUNET"
      descr: "Login for SUNET employees"
      auth: "saml"
      entityID: "https://idp.sunet.se/idp"
      type: "idp"
      hidden: "false"
      scope: "sunet.se"
      domain: "sunet.se"
      name_tag: "SUNET"
      entity_icon_url: {url: "https://static.sunet.se/images/sunet256.png",
      width: "256", height: "205"}
      entity_id: "https://idp.sunet.se/idp"
```

Seamless Access.org

Limited Integration, no previously remembered institution

Phase 4 - Discovery and Persistence service, selecting institution



10. Steps 3 & 4 are now repeated. This time, the Persistence Service **reads the previously remembered institutions in the browser local storage**. The Discovery interface is rendered to display these institutions.

11. User clicks on their preferred institution.

12. The **md.seamlessaccess.org/entites/entities/{sha1}...** call is triggered to query the metadata of the chosen institution's IdP. This redirects to the CDN, serving the **md.seamlessacess.org.**

13. The CDN uses load balancer to query the SeamlessAccess Metadata Service nodes in the backend, which will answer with the institution's IdP metadata.

14. User's browser redirects to chosen institution IdP.

*In most cases, the CDN will have already cached this information and will not need to query the SeamlessAccess backend

Seamless Access.org

Standard Integration

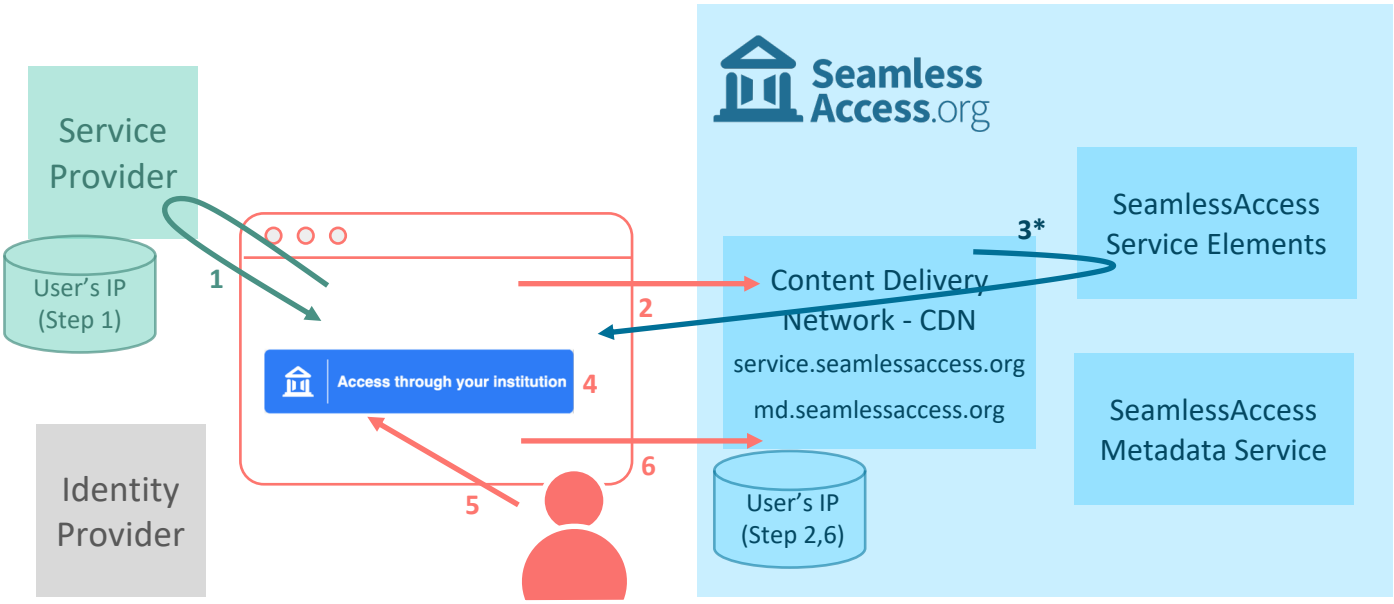
No previously remembered institution

Generic example

SeamlessAccess.org

Standard Integration, no previously remembered institution

Phase 1 – SP site, Button and Persistence service



1. User accesses the Service Provider's (SP) site, which will likely log user's IP address. In order to render the SeamlessAccess Button, the SP uses the component library provided by SeamlessAccess at **service.seamlessaccess.org/thiss.js**.

2. To fetch this library, the SP calls **service.seamlessaccess.org**, which resolves to the CDN. The CDN records user's IP address according to its policy.

3. The CDN is setup to fetch **origin-service.seamlessaccess.org,** hosted by the SeamlessAccess Service Elements. It serves the JavaScript for rendering the Seamless Access Button and Persistence service.

4. The JavaScript renders the SeamlessAccess Button. Persistence service JavaScript is used to **read from the browser local storage** to show (up to the last three) institution choices that user has previously made. As there are no previous choices in this scenario, the SeamlessAccess Button renders the text: "*Access through your institution*".

5. User clicks on the SeamlessAccess Button.

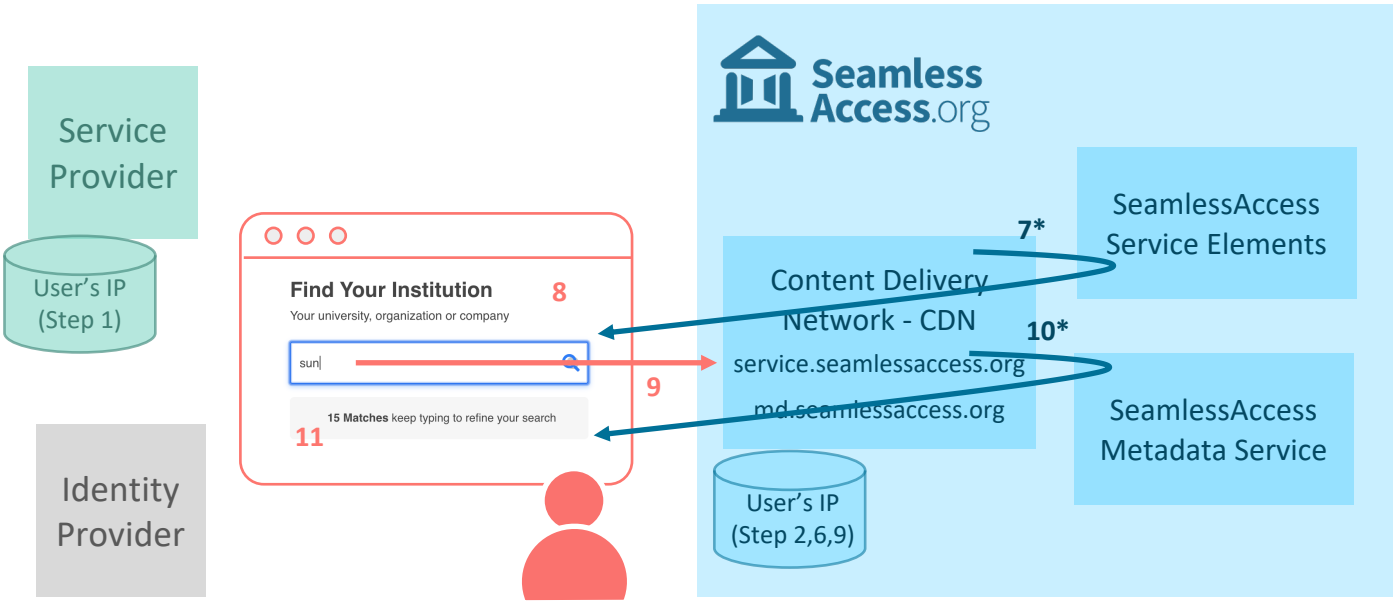
6. This makes another browser redirect to **service.seamless.access.org/ds.**

*In most cases, the CDN will have already cached this information and will not need to query the SeamlessAccess backend

Seamless Access.org

Standard Integration, no previously remembered institution

Phase 2 - Discovery and Persistence service, finding IdP



7. In the same way, the CDN fetches **origin-service.seamlessaccess.org**. Another set of JavaScripts gets served now, which are used for the Discovery and Persistence services.

8. The JavaScripts are used to render the page with the Discovery service, and the Persistence service **reads the browser local storage** to show (up to the last 3) institution choices that user has previously made. In this scenario, there are no institution choices remembered.

9. User starts to search for an institution by typing its name. As characters are entered, the call to **md.seamlessaccess.org/entites/?q=string** is triggered. This redirects to the CDN, which is also serving the **md.seamlessacess.org**.

10. The CDN uses load balancer to query one of the SeamlessAccess Metadata Service nodes in the backend, which will answer with the list of matching institutions. Since this list will initially be too large, it will return a number of matches only.

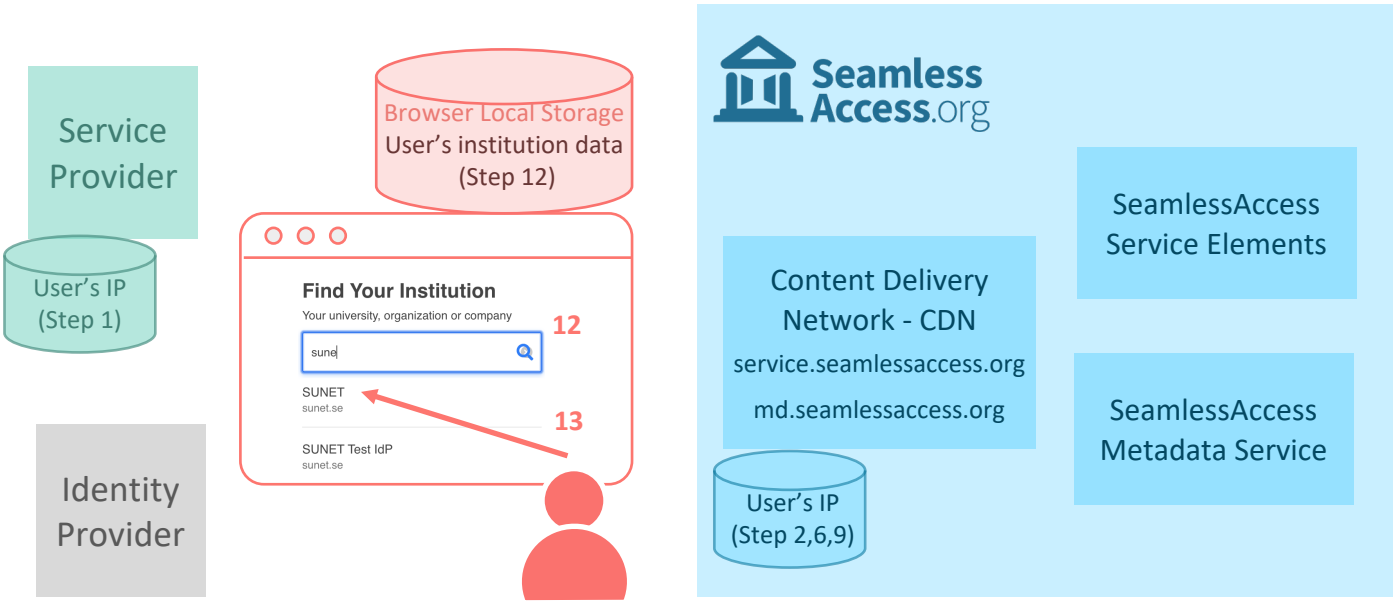
11. The Discovery service renders the message: "***N Matches*** *keep typing to refine your search*", so user repeats Steps 9 & 10 until the list of matching institutions is small enough.

* In most cases, the CDN will have already cached this information and will not need to query the SeamlessAccess backend

Seamless Access.org

Standard Integration, no previously remembered institution

Phase 3 - Discovery and Persistence service, choosing IdP



12. Once the number of matching institutional choices is less than the defined maximum, the metadata query in Steps 9 & 10 will return the list of matching institutions, and the Discovery service in Step 11 will display this list.

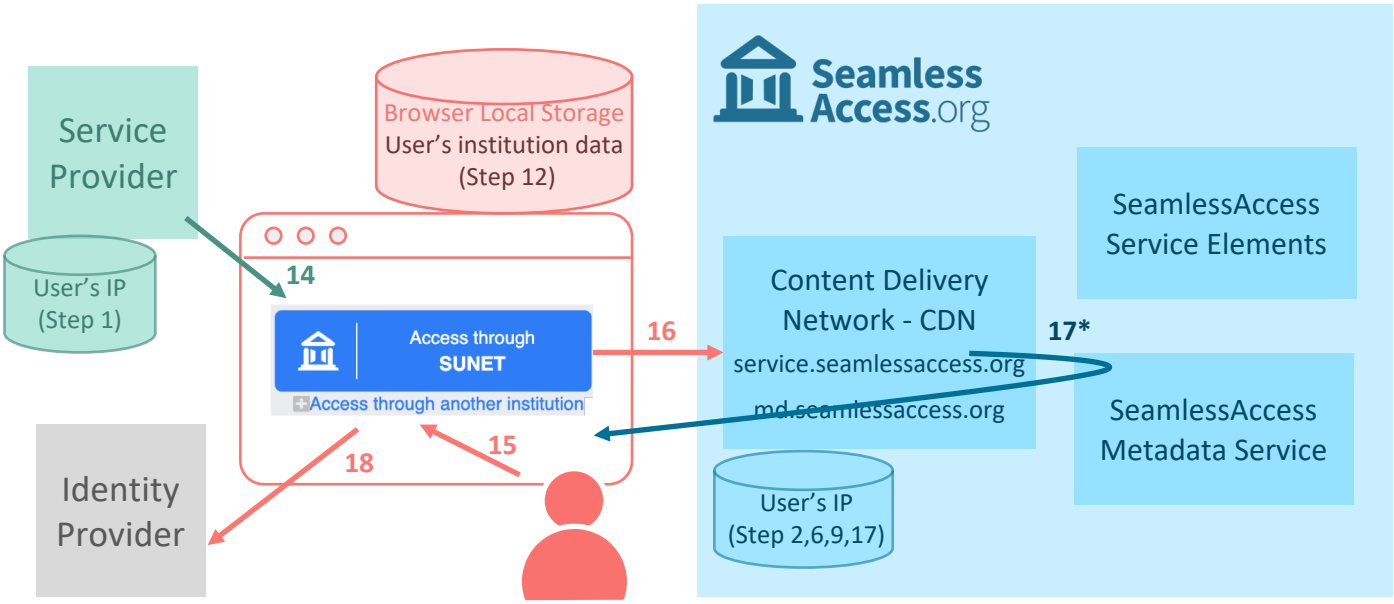
13. User clicks to select an institution, triggering the Persistence service to **write the chosen institution to the browser local storage**. The following information is written (example SUNET IdP):

```
entity: {title: "SUNET", descr: "Login for SUNET employees", auth: "saml",…}
      title: "SUNET"
      descr: "Login for SUNET employees"
      auth: "saml"
      entityID: "https://idp.sunet.se/idp"
      type: "idp"
      hidden: "false"
      scope: "sunet.se"
      domain: "sunet.se"
      name_tag: "SUNET"
      entity_icon_url: {url: "https://static.sunet.se/images/sunet256.png",
      width: "256", height: "205"}
      entity_id: "https://idp.sunet.se/idp"
```

Seamless Access.org

Standard Integration, no previously remembered institution

Phase 4 - Button and Persistence service, selecting IdP



14. User is redirected back to the SP site, where Steps 3 & 4 are now repeated. This time, the Persistence service **reads the last remembered institution from browser local storage** and that institution is rendered in the SeamlessAccess Button.

15. User clicks on the SeamlessAccess Button, which now displays their last remembered institution e.g. *"Access through SUNET"*.

16. The **md.seamlessaccess.org/entites/entities/{sha1}...** call is triggered to query for the metadata of the institution's IdP. This redirects to the CDN, which is serving the **md.seamlessacess.org**.

17. The CDN uses load balancer to query one of the SeamlessAccess Metadata Service nodes in the backend, which will answer with the institution's IdP metadata.

18. User's browser redirects chosen institution IdP.

* In most cases, the CDN will have already cached this information and will not need to query the SeamlessAccess backend

Seamless Access.org

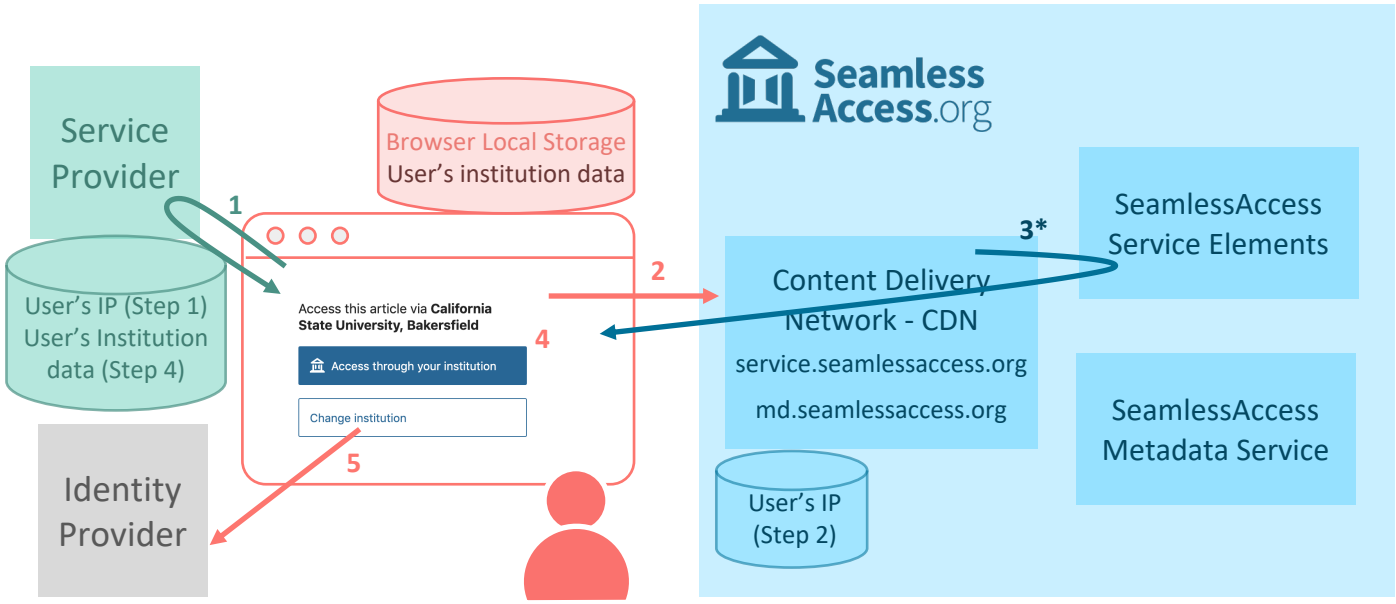
Advanced Integration

Previously remembered institution

Example from Springer Nature implementation

SeamlessAccess.org

Advanced Integration, previously remembered institution



1. User accesses the Service Provider's (SP) site, which will likely log user's IP address. User navigates to a page where the SP presents login options. Springer Nature have their own discovery service and so, in this example, the SP's own implementation of JavaScript is used to integrate with SeamlessAccess (**wayf.springernature.com/seamless-access.js**).

2. JavaScript uses the SeamlessAccess Persistence service API to read and write to the browser local storage. JavaScript loads the SeamlessAccess Persistence service **service.seamlessaccess.org/ps,** which resolves to the CDN. Browser will only allow access to the common-domain browser local store if the script is loaded from that same domain. The CDN records user's IP address according to its policy.

3. The CDN is configured to fetch **origin-service.seamlessaccess.org.** It downloads JavaScripts used by the Persistence service.

4. wayf.springernature.com/seamless-access.js uses the SeamlessAccess Persistence service APIs to **read from the browser local storage**. It will read the institution that was last remembered and, if it can resolve the entityID, it will load the institution name and entityID. With this information the SP site renders its implementation of SeamlessAccess button.

5. User clicks to "*Access through your institution*", which will redirect to institution's IdP.

* In most cases, the CDN will have already cached this information and will not need to query the SeamlessAccess backend

Seamless Access.org