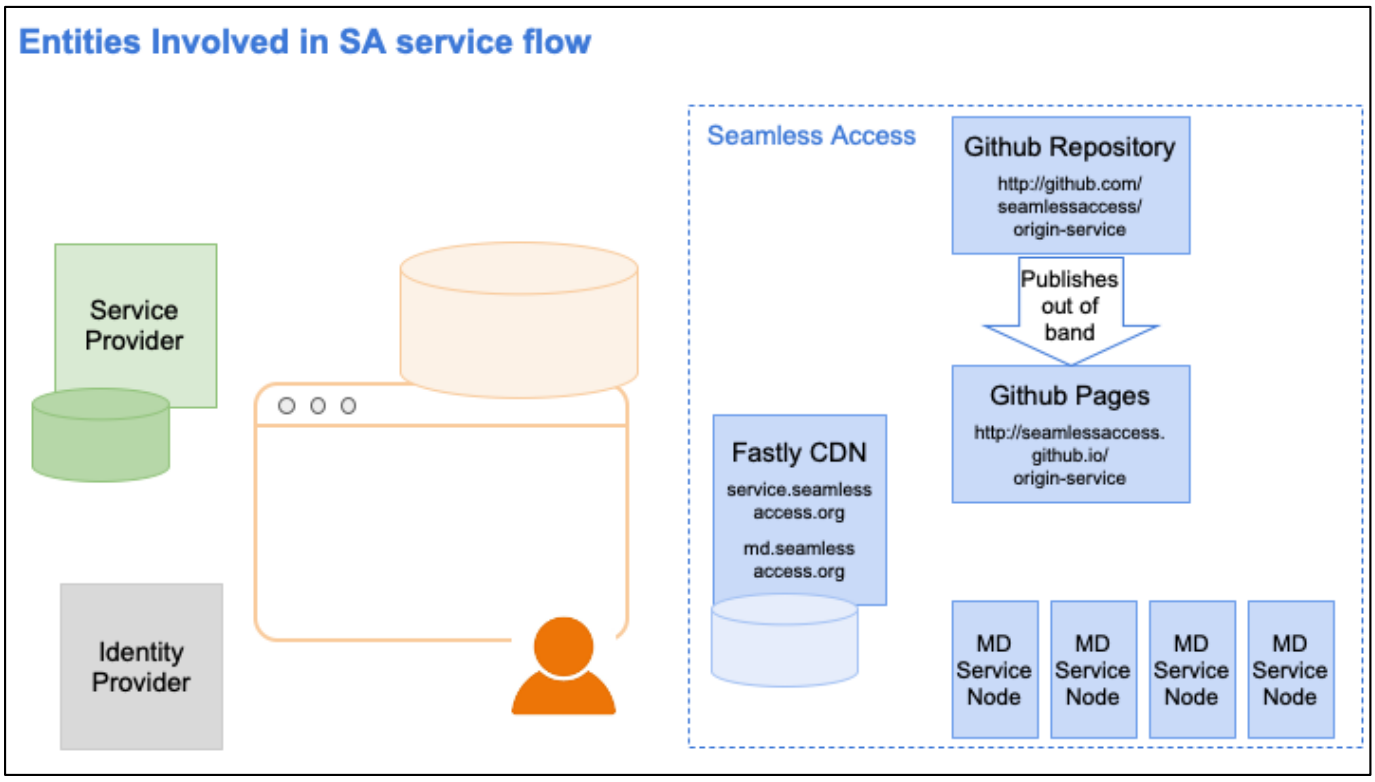
Seamless Access
User and Data Flow

How to read the slides!

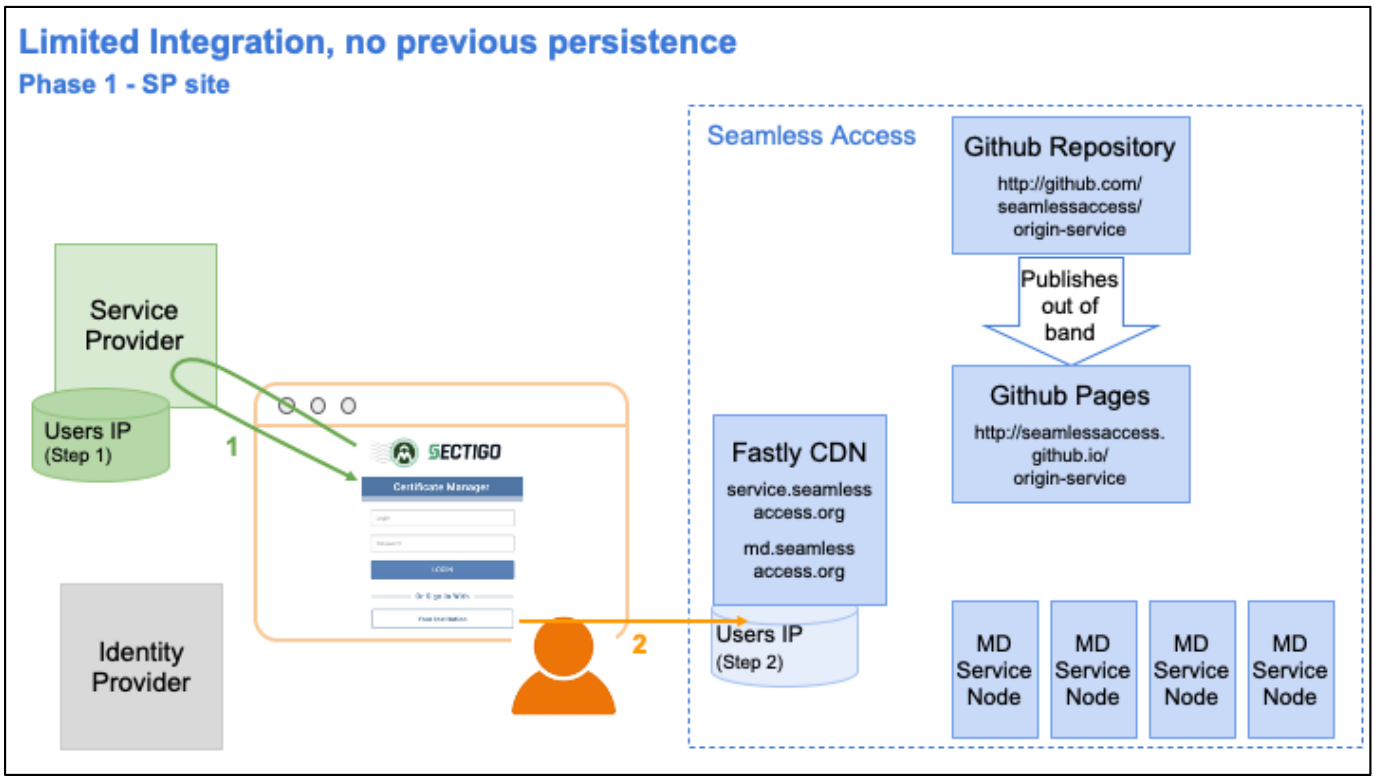
- Slides show diagrams with the user and resulting data flows. Those are broken down in multiple slides for better overview.
- Color coding is used to show different actors and the flows they initiate (green for Service Provider, blue for Seamless Access, orange from calls initiated from the browser or user action etc.)
- Slides notes describe steps depicted in diagrams. Refer to the notes to understand what is happening in each step.
- The cylinder by each party holds the data that would be processed at given step of the flow.



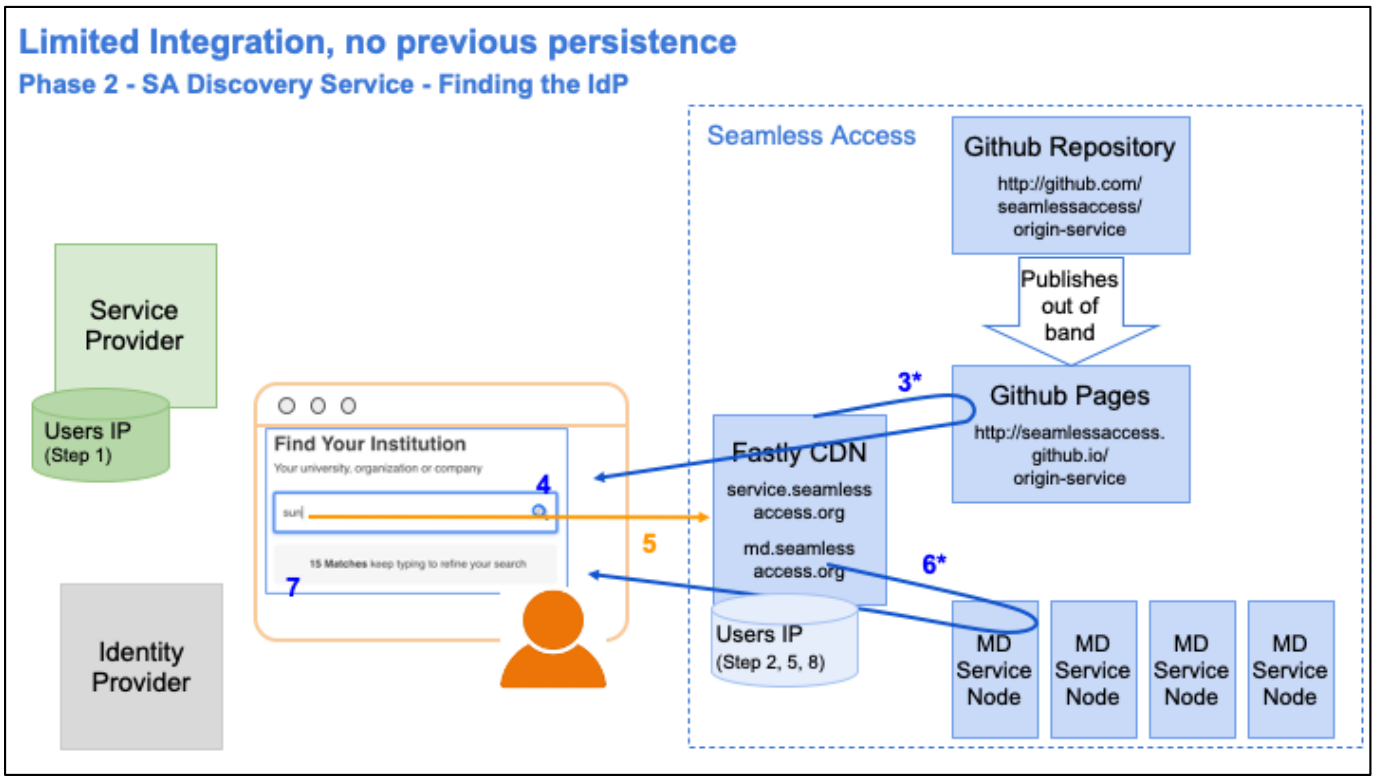
- Fastly CDN serves:
  - https://service.seamlessaccess.org - which is the SA button used by the Standard Integration
  - https://service.seamlessaccess.org/ds - which is the SA discovery, used by the Limited and Standard Integrations
  - https://service.seamlessaccess.org/ps - which are the scripts used to deliver the SA persistence service, used by all three integrations
  - https://md.seamlessaccess.org - which is the metadata service used by Limited and Standard Integrations

Limited Integration

No Previous Persistence
Example of Sectigo Implementation



1. User goes to Service Providers site, which will most probably log the users IP address.
2. On users request to login, Service Provider site redirects user to the **service.seamless.access.org/ds** which resolves to the CDN, provided by Fastly. CDN records the users IP address according to its policy



3.CDN is configured to fetch **http://origin-service.seamlessaccess.org/**, which in turn is an alias for **seamlessaccess.github.io/origin-service**. It serves the Javascripts used for the discovery and persistence service. *CDN will in most cases have the cache already so it will not need to go back to origin service to fetch the resources. Following resources are downloaded (this is just an example, the actual files that download can differ):

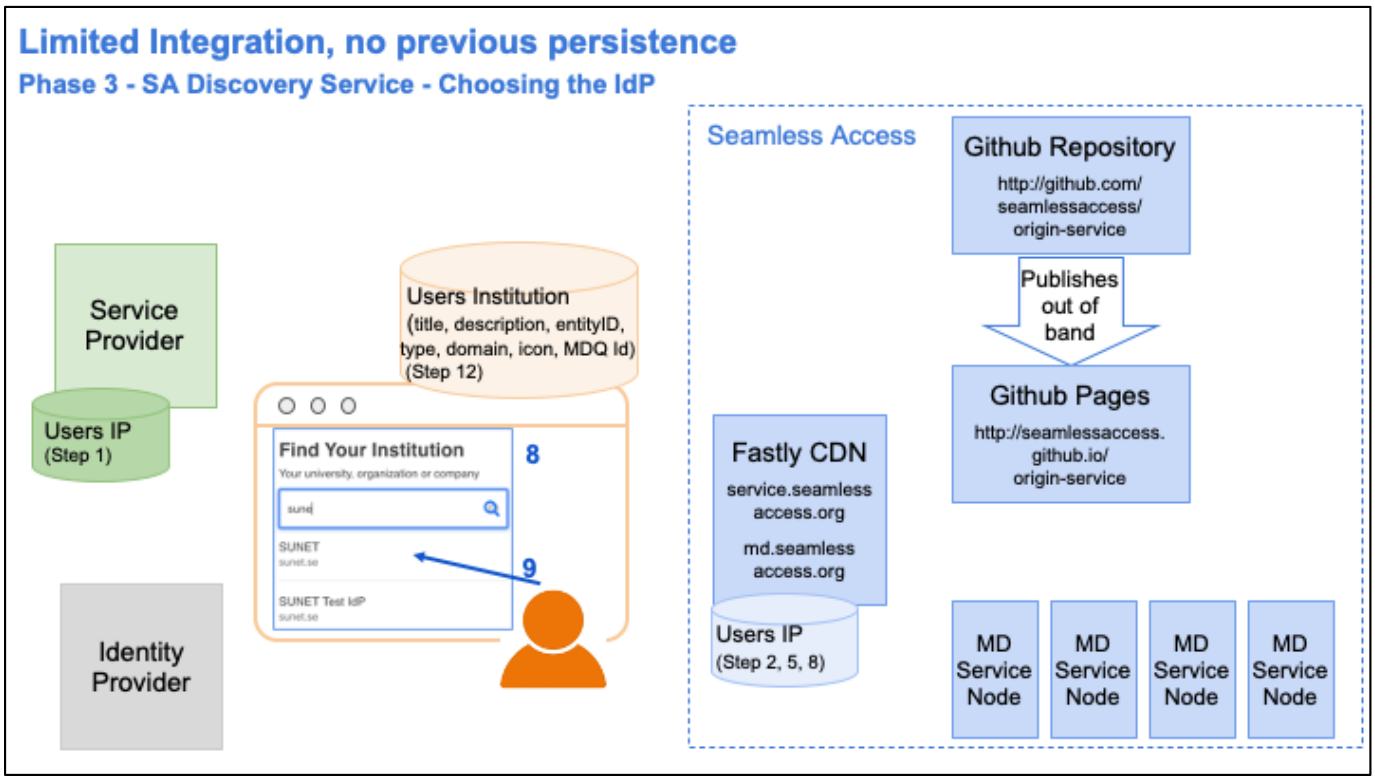
- `https://service.seamlessaccess.org/vendors~cta~ds~index.js`
- `https://service.seamlessaccess.org/ds.css`
- `https://service.seamlessaccess.org/vendors~ds.js`
- `https://service.seamlessaccess.org/ds.js`
- `https://service.seamlessaccess.org/ps/`
- `https://service.seamlessaccess.org/vendors~ps.js`
- `https://service.seamlessaccess.org/ps.js`
- `Images...`

4.These render the page with the discovery service UI, and the persistence service reads the browser local storage to show (up to last 3 institution) choices that user made. In this case, there are no institution choices remembered.

5. User starts to search for the institution by typing its name. On user entering character, the **md.seamlessaccess.org/entites/?q=string** is triggered. (There is algorithm, so that not every keystroke triggers a search - for example when user types string, only when more the certain amount of time (15ms) is logged between subsequent keystrokes, the string triggers the md query).This redirects to Fastly CDN which is also serving the **md.seamlessacess.org**.

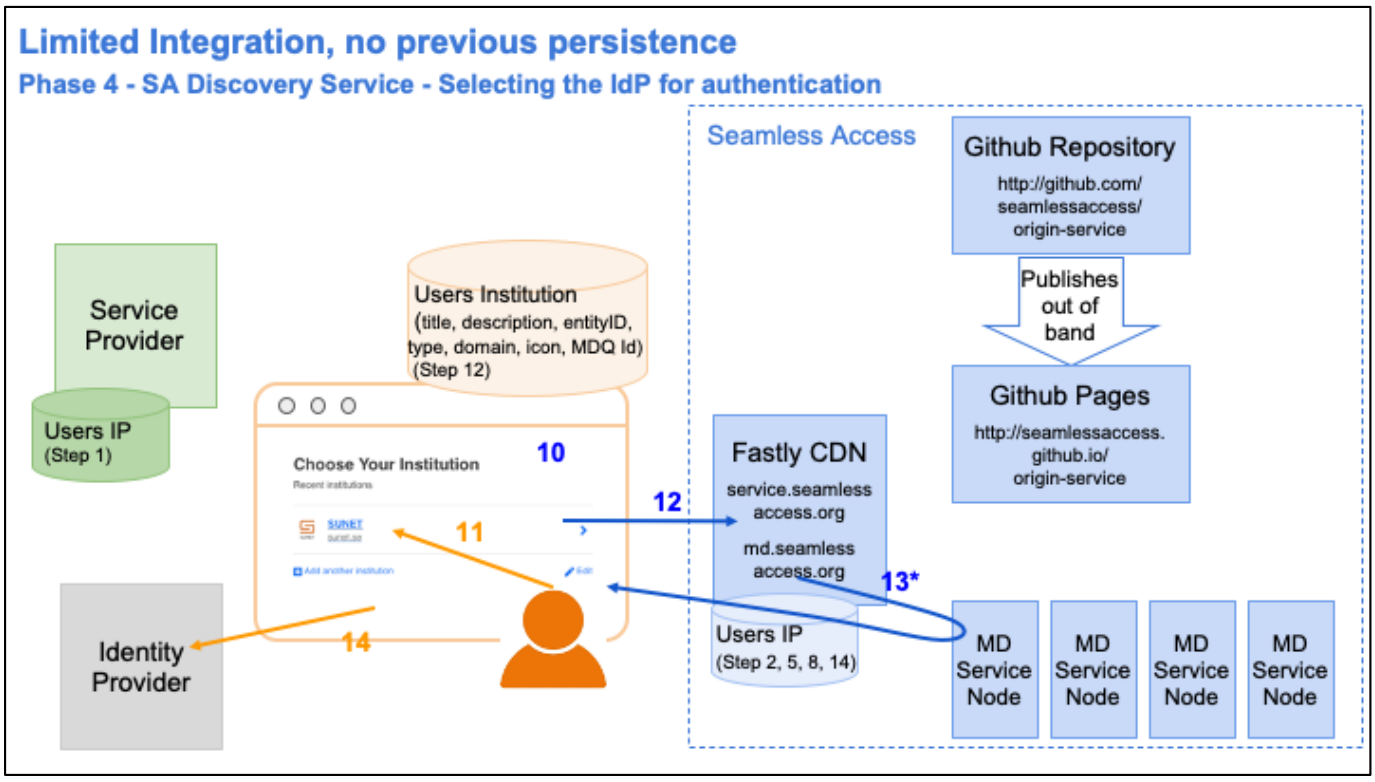
6. CDN uses loadbalancer to query one of the MD Service Nodes in the background, which will

answer with the list of the matching institutions. * In most cases, CDN will already have cache so it will not need to call the MD nodes. Since the list is to large, it will return number of matches only.
7. The discovery DS will render the text: “N Matches, keep typing to refine your search”, so the user repeats step 8 and 9 until the list of matching institutions get below a defined number.



8. Once when the entered string is good enough so that the number of choices is less than the defined number, MDQ in Step 9 will return the list of matching institutions, and the Discovery Service in step 10 will render the list of the matching institutions.
9. User Clicks on its institution. This triggers the Persistence Service to save the choice in the browser store. Following information is being saved (in example from Sunet):

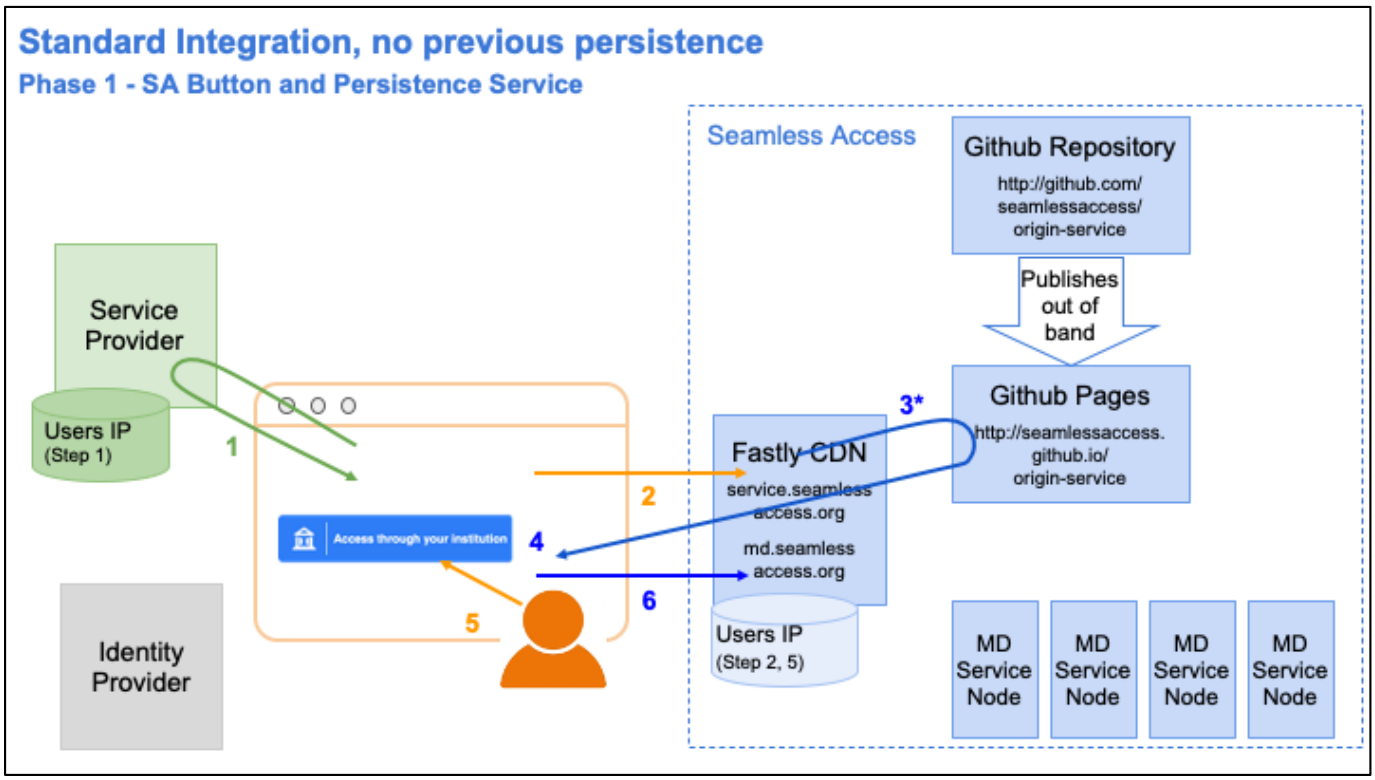

```
entity: {title: "SUNET", descr: "Login for SUNET employees", auth:
   "saml",…}
      title: "SUNET"
      descr: "Login for SUNET employees"
      auth: "saml"
      entityID: "https://idp.sunet.se/idp"
      type: "idp"
      hidden: "false"
      scope: "sunet.se"
      domain: "sunet.se"
      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"
      id: "{sha1}2f260e8b792a91db581bb3833731ade6773c56e9"
```



10. Steps 3 and 4 are now repeated, and the discovery UI with the IdP from the Persistence Service is rendered.
11.User clicks on the preferred institution.
12. The **md.seamlessaccess.org/entites/entities/{sha1}...** is triggered to query for the metadata of the IdP. This redirects to Fastly CDN which is also serving the **md.seamlessacess.org**.
13. CDN uses loadbalancer to query one of the MD Service Nodes in the background, which will answer with the IdP metadata.* In most cases, CDN will already have cache so it will not need to call the MD nodes
14. The browser redirects to the Identity Provider.

Standard Integration

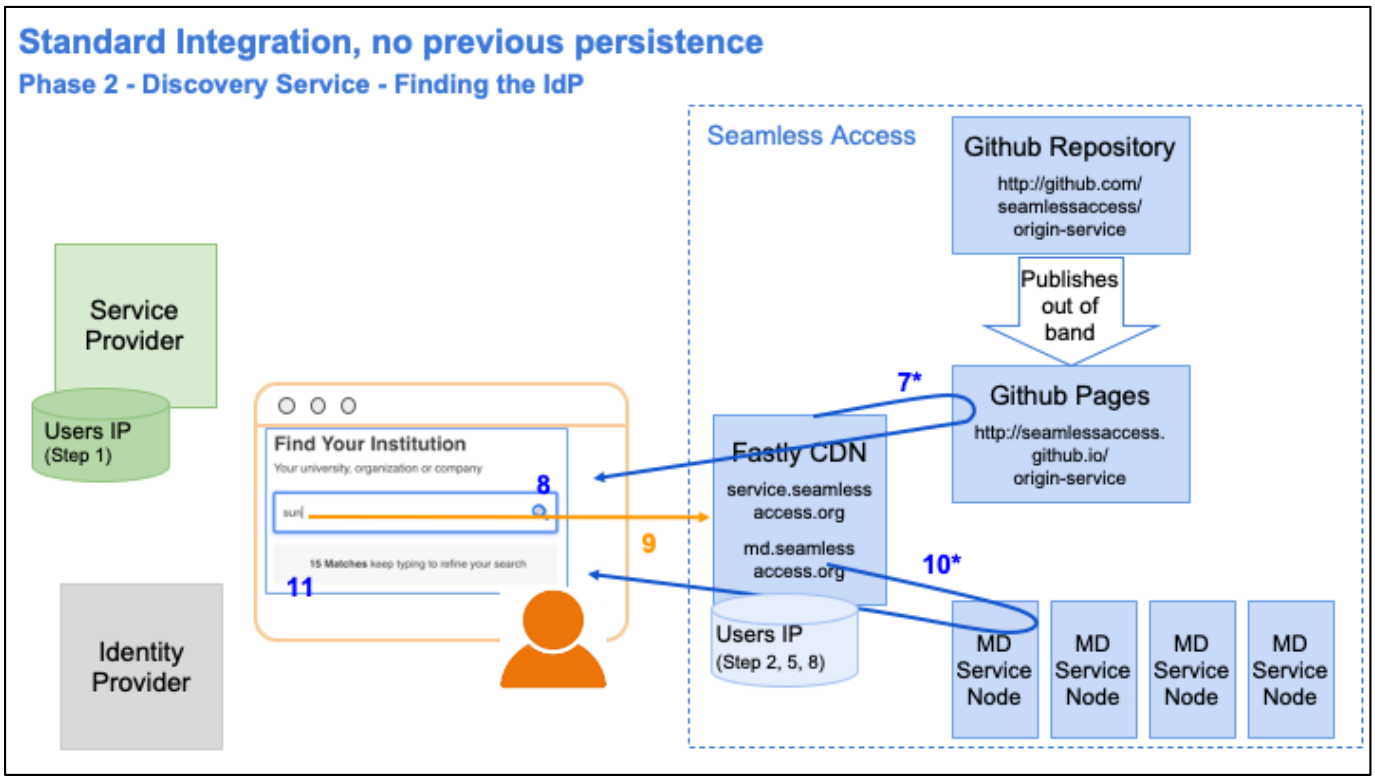
No Previous Persistence



1. User goes to Service Providers site, which will most probably log the users IP address. In order to render the Button, Service Provider, uses the Seamless Access component library **https://service.seamlessaccess.org/thiss.js**.
2.To fetch the library, Service Provider calls the **service.seamlessaccess.org**, which resolves to the CDN, provided by Fastly. CDN records the users IP address according to its policy
3. CDN is configured to fetch **http://origin-service.seamlessaccess.org/**, which in turn is an alias for **seamlessaccess.github.io/origin-service**. It serves the Javascripts used for rendering the button and using the persistence service. *CDN will in most cases have the cache already so it will not need to go back to origin service to fetch the resources. Following resources are downloaded (this is just an example, the actual files that download can differ):

- https://service.seamlessaccess.org/thiss.js (this is the original call from the step 2)
- https://service.seamlessaccess.org/vendors~cta~ds~index.js
- https://service.seamlessaccess.org/vendors~cta~index.js
- https://service.seamlessaccess.org/cta/
- https://service.seamlessaccess.org/cta.css
- https://service.seamlessaccess.org/cta.js
- https://service.seamlessaccess.org/ps/
- https://service.seamlessaccess.org/vendors~ps.js
- https://service.seamlessaccess.org/ps.js
- Images that are SA logo

4. Javascripts render the button, check whether there there are previous user choices stored in the browser local storage. As there are no previous choices, the button shows the “Access through your institution ”.
5. User clicks on the button
6. This makes another browser redirect to the **service.seamless.access.org/ds**.



7.CDN in the same manner fetches **http://origin-service.seamlessaccess.org/**, i.e **seamlessaccess.github.io/origin-service**. Another set of Javascripts gets served now, as they are used for the discovery service and for persistence service. *CDN will in most cases have the cache already so it will not need to go back to origin service to fetch the resources. Following resources are downloaded (this is just an example, the actual files that download can differ):

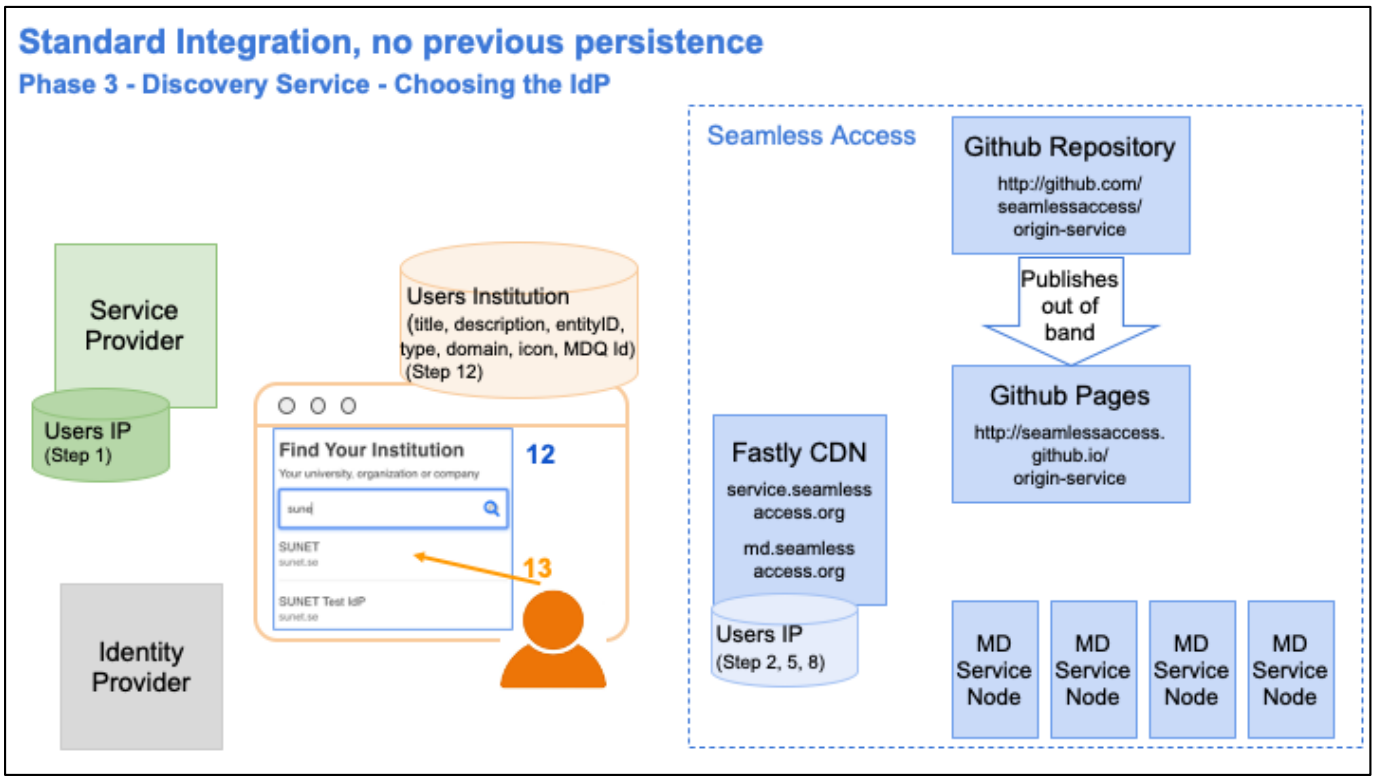
- `https://service.seamlessaccess.org/vendors~cta~ds~index.js`
- `https://service.seamlessaccess.org/ds.css`
- `https://service.seamlessaccess.org/vendors~ds.js`
- `https://service.seamlessaccess.org/ds.js`
- `https://service.seamlessaccess.org/ps/`
- `https://service.seamlessaccess.org/vendors~ps.js`
- `https://service.seamlessaccess.org/ps.js`

8.These render the page with the discovery service UI hosted by the seamless access service, and the persistence service reads the browser local storage to show (up to last 3 institution) choices that user made. In this case, there are no institution choices remembered.

9. User starts to search for the institution by typing its name. On user entering character, the **md.seamlessaccess.org/entites/?q=string** is triggered. (There is algorithm, so that not every keystroke triggers a search - for example when user types string, only when more the certain amount of time (15ms) is logged between subsequent keystrokes, the string triggers the md query).This redirects to Fastly CDN which is also serving the **md.seamlessacess.org**.

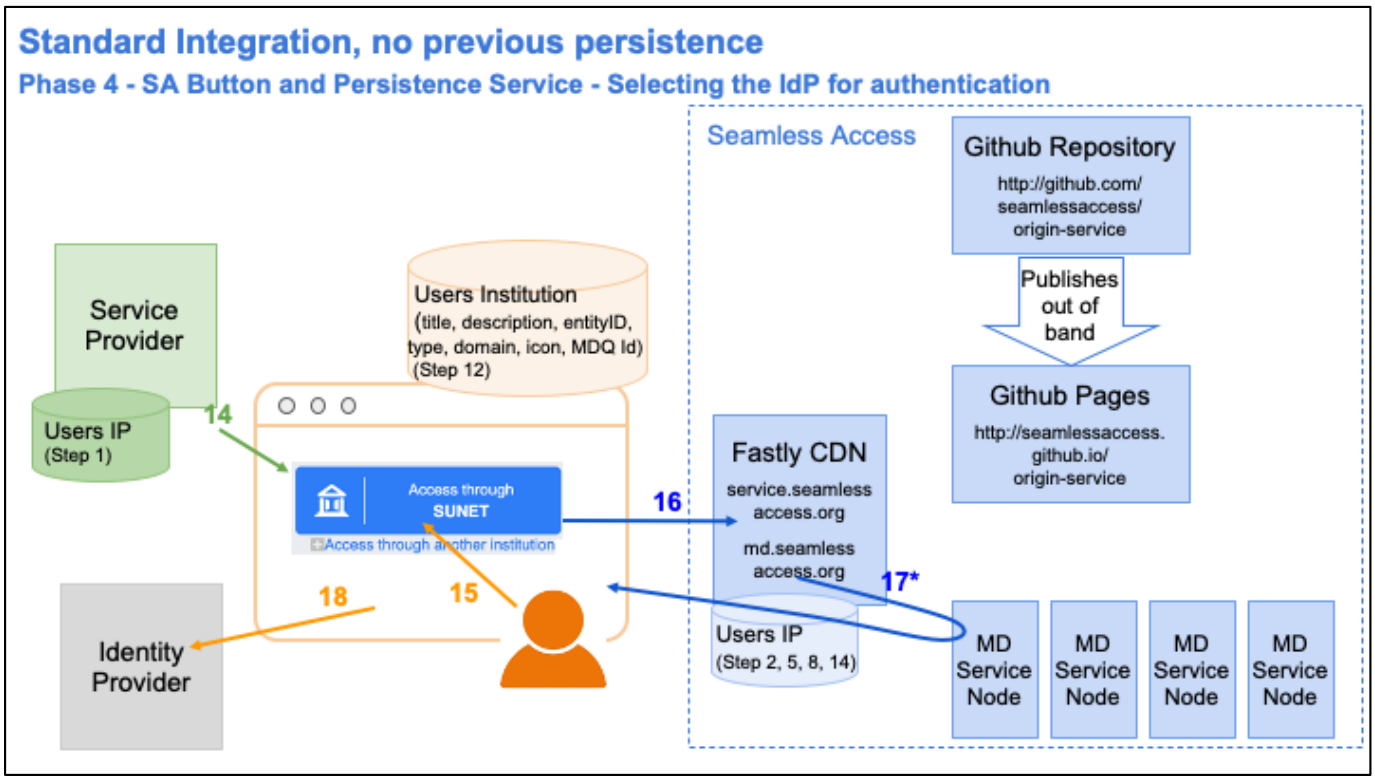
10. CDN uses loadbalancer to query one of the MD Service Nodes in the background, which

will answer with the list of the matching institutions. * In most cases, CDN will already have cache so it will not need to call the MD nodes. Since the list is to large, it will return number of matches only.
11. The discovery DS will render the text: “N Matches, keep typing to refine your search”, so the user repeats step 8 and 9 until the list of matching institutions get below xxxx?



12. Once when the entered string is good enough so that the number of choices is less then xxx the MDQ in Step 9 will return the list of matching institutions, and the Discovery Service in step 10 will render the list of the matching institutions.
13. User Clicks on its institution. This triggers the Persistence Service to save the choice in the browser store. Following information is being saved (in example from Sunet)

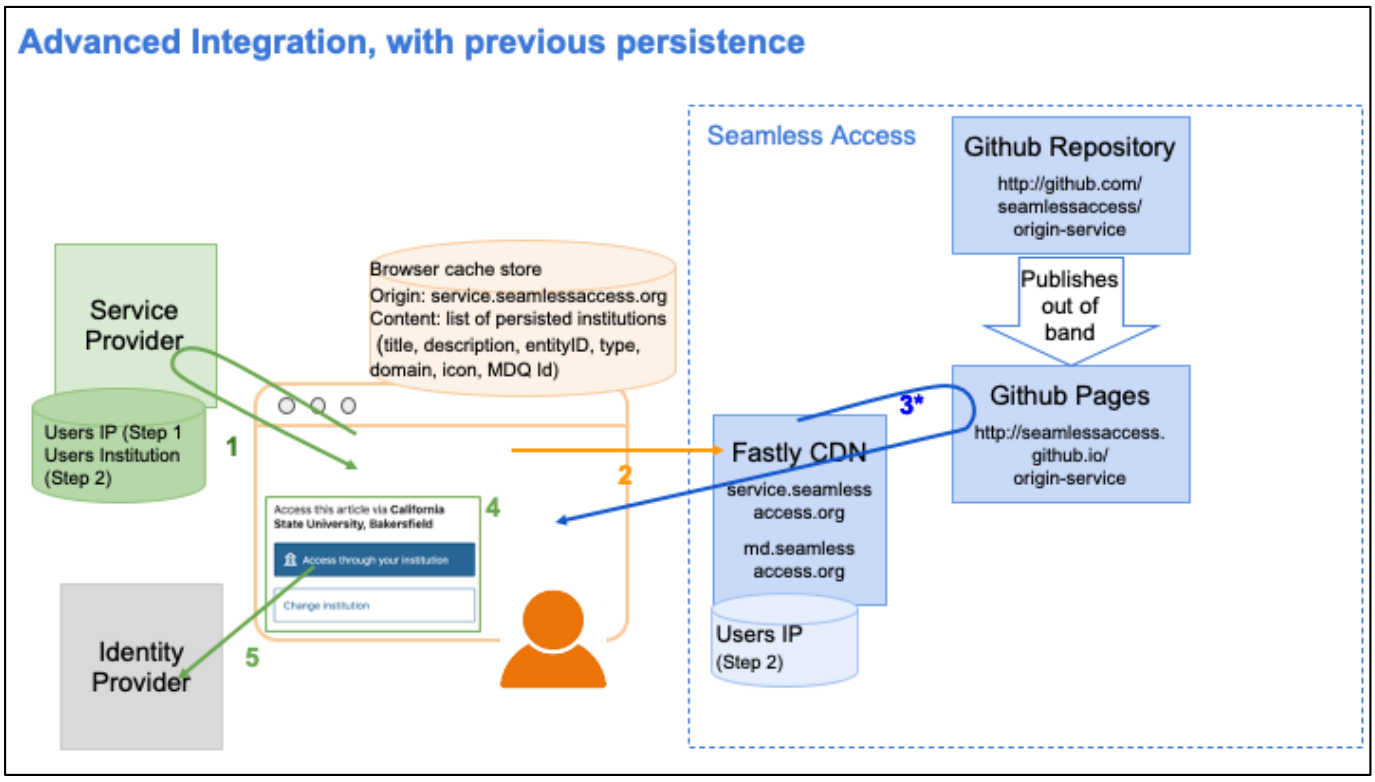
```
entity: {title: "SUNET", descr: "Login for SUNET employees", auth:
   "saml",…}
      title: "SUNET"
      descr: "Login for SUNET employees"
      auth: "saml"
      entityID: "https://idp.sunet.se/idp"
      type: "idp"
      hidden: "false"
      scope: "sunet.se"
      domain: "sunet.se"
      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"
      id: "{sha1}2f260e8b792a91db581bb3833731ade6773c56e9"
```



14. User gets redirected back to the SP site, where steps 3 and 4 are now repeated, and the button with the IdP from the Persistence Service is rendered.
15. User clicks on the button which now has its chosen IdP, saying “Access trough SUNET ”.
16. The **md.seamlessaccess.org/entites/entities/{sha1}...** is triggered to query for the metadata of the IdP. This redirects to Fastly CDN which is also serving the **md.seamlessacess.org**.
17. CDN uses loadbalancer to query one of the MD Service Nodes in the background, which will answer with the IdP metadata.* In most cases, CDN will already have cache so it will not need to call the MD nodes
18. The discovery redirects to the Identity Provider.

Advanced Integration

With Previous Persistence
Example of Springer Nature implementation



1. User goes to Service Providers (SP) site, which will most probably log users IP address. User navigates to a page with resource, where SP wants to present option for user to authenticate. SP has its own discovery service, in this case https://wayf.springernature.com/seamless-access.js. This JavaScript is springer nature own implementation, that uses the seamless access persistence service API calls to read and write to the browser cache store for service.seamlessaccess.org origin.

2, https://wayf.springernature.com/seamless-access.js loads the seamless access persistent service **service.seamlessaccess.org/ps.** which resolves to the CDN, provided by Fastly. The browser will only allow access to the common-domain browser local store if the script is loaded from that same domain. *CDN will in most cases have the cache already so it will not need to go back to origin service to fetch the resources. CDN records the users IP address according to its policy.

3. CDN is configured to fetch **http://origin-service.seamlessaccess.org/**, which in turn is an alias for **seamlessaccess.github.io/origin-service/**. It downloads the JavaScripts used by the persistent service:

- https://service.seamlessaccess.org/ps.js
- https://service.seamlessaccess.org/vendors~ps.js

4. https://wayf.springernature.com/seamless-access.js will use Persistance Service APIs to read from the browser cache store It will read institution that was lastly used and if it can

resolve the entityID, it will load the institution name and entity ID. SP renders the discovery service UI using this information.

5. User clicks to “Access to your institution” which will redirect to IdP.