GitHub Enterprise

This information was last reviewed in November 2018, by mdomingues@uiowa.edu.

Change Log:

11/30/2018: Initial version.

This is not a replacement for the actual documentation and you cannot cut and paste your way to a working system. The examples are **not** usable without taking into consideration your local needs and requirements.

GitHub Enterprise Server is the on-premises version of GitHub, the well-known source control hosting system. Provided as a virtual-appliance, it supports a variety of authentication methods, the most relevant of which (to this documentation) is SAML.

Note that GitHub Enterprise Server is distinct from GitHub Enterprise Cloud, the GitHub-hosted enterprise offering. GitHub Enterprise Server used to be known simply as "GitHub Enterprise". You can review GitHub's product offerings here: https://help.github.com/en/articles/githubs-products#github-enterprise

GitHub's official configuration documentation can be found here: https://help.github.com/en/enterprise/admin/user-management/using-saml

While the official documentation covers most of the basics, there are a number of important caveats the bear mentioning below. If these caveats are ignored, your appliance can end up in a state that officially requires GitHub Enterprise Support to intervene, lest you risk voiding your support contract.

Identity Provider Metadata

The GitHub Enterprise Server appliance cannot consume federated or standalone metadata, but does let you configure settings in a self-service manner. In the Admin Console, it provides you with a number of text-fields and drop-downs, allowing you to provide the following pieces of information:

- **Single-sign on URL:** Your IDP's SAML2 Redirect endpoint, e.g. https://idp.myuniversity.edu/idp/profile/SAML2/Redirect/SSO
- **Issuer:** Your IDP's entityID. This supports both URL-style and URN-style entityIDs.
- **Verification Certificate**: Your IDP's SAML signing certificate. Provided via file-upload.

> *At the University of Iowa, we found that filesystem permissions on the uploaded certificate (which gets stored at /data/user/common /idp.crt) weren't being set correctly. They default to (Unix-style) 660, with the "admin" user and group set as the owner and group respectively. The SP component for the GitHub appliance runs under a different (indeterminate) user, so we changed the permissions to 664 by accessing the administrative shell: https://help.github.com/enterprise/2.15/admin/guides/installation/accessing-the-administrative-shell-ssh/. This resolved the issue, but the 660 permissions were re-applied whenever we saved the settings. This may be an issue with our specific appliance — we believe our GitHub Enterprise admins have an open case about it — but we figured we would note it here in case others encounter the same problem.*

Service Provider Metadata

GitHub Enterprise Server provides access to its metadata at `http(s)://[hostname]/saml/metadata`. The file is unsigned, so deployers will want to download a copy and load it statically. The certificate contained within is long-lived (~10 years), and does not seem to change if SAML authentication is disabled and re-enabled, so keeping a static copy should not prove any more bothersome than usual.

Note that the SP's certificate is only present in metadata when SAML authentication is enabled. You will likely want to stage all of your IdP configuration first (using the metadata without a certificate), then enable SAML authentication, wait 1-2 minutes *after* the appliance says it has restarted for the SAML settings to actually take effect, then retrieve the updated metadata and load it into your IdP.

Profile Requirements

The SAML SSO profile behavior is fairly standard and relies on signed responses and no encryption. The former is the Shibboleth default and the latter would require setting the **Idp.encryption.optional** property or disabling encryption for the service.

Early in our testing, we found that whatever GitHub Enterprise Server uses as its SP implementation (we suspect https://github.com/onelogin/ruby-saml but the appliance code is, unsurprisingly, obfuscated) did not properly account for clock skew, and was not tunable. As such, we preemptively remove the NotB efore condition from assertions we send to them.

Example Shibboleth Configuration

Refer to the RelyingPartyConfiguration topic and be cognizant that creating overrides for every service is generally an inefficient use of the software. Consider identifying common requirements across services and create overrides tied to multiple services that share those requirements, or that reference profile configuration beans containing common settings.

Example relying-party.xml override

```
    <!-- Container for any overrides you want to add. -->

    <util:list id="shibboleth.RelyingPartyOverrides">

            <!-- other overrides... -->

        <!-- GitHub Enterprise Server doesn't handle allowable clock skew correctly, so
         disable the inclusion of a NotBefore condition in assertions sent to their
         SP. -->
    <bean parent="RelyingPartyByName" c:relyingPartyIds="https://github.myuniversity.edu">
        <property name="profileConfigurations">
            <list>
                <bean parent="SAML2.SSO"
                      p:encryptAssertions="false"
                      p:includeConditionsNotBefore="false" />
                <ref bean="SAML2.Logout" />
            </list>
        </property>
    </bean>

 </util:list>
```

Account Provisioning

Accounts in GitHub Enterprise Server are provisioned just-in-time during sign-in, but can also be created out-of-band if you so choose. Account provisioning relies on both attributes released to the service, as well as the NameID value. It is imperative to get these right the first time, since certain data about SAML federated users (including the NameID) **cannot** be edited without contacting GitHub Enterprise Server, or potentially violating your support contract. See below for details.

GitHub Enterprise Username Normalization

GitHub Enterprise Server normalizes usernames according to a scheme described here: https://help.github.com/enterprise/admin/guides/user-management/using-saml/#username-considerations-with-saml. If multiple usernames normalize to the same value in GitHub Enterprise Server, only the first account is created.

NameID Requirements

Proper NameID Configuration is Imperative

Be sure you release the correct NameID format to GitHub Enterprise Server before enabling SAML authentication. **The first NameID value GitHub Enterprise Server receives** for a given username is persisted in a (non-administrator accessible) database, and **cannot be officially cleared without contacting GitHub Enterprise Support**. If this is done incorrectly (by releasing a NameID format that can vary independently from username, like the Transient ID, or a Persistent ID not generated based on whatever attribute you use for the username) you can and will end up locking your users out.

GitHub Enterprise Server claims to require a persistent (`urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`) NameID format, but then in the next sentence claims (of the NameID value) "it may only contain alphanumeric characters or dashes, and cannot begin with a dash".

What they *really* mean is **for a given username, the NameID value released to GitHub Enterprise Server must never vary independently from the username**. GitHub Enterprise Server does not care what you actually call the NameID format.

From the GitHub Enterprise Server documentation:

> *If multiple accounts are normalized into the same GitHub Enterprise Server username, only the first user account is created. Subsequent users with the same username won't be able to sign in.*

In our testing, we (the University of Iowa) found that GitHub Enterprise Server detects this condition by comparing both the NameID and username attributes released in a SAML assertion. If the username attribute value normalizes to a username GitHub Enterprise Server is already aware of, but the NameID of the assertion does not match the first NameID GitHub Enterprise Server received (and persisted) for said username, the login attempt is prevented.

At this point, short of contacting GitHub Enterprise Support, you are (officially) stuck. No capabilities are exposed in the administrative GUI (the only officially supported method of interacting with the appliance) to clear out the stale / bad NameID value.

Thus, in choosing a NameID format to use with GitHub Enterprise Server, deployers can take one of the following approaches:

1. Release a custom NameID format based on the same attribute that you release to GitHub Enterprise Server as the username.
2. Release the Persistent NameID format *if and only if* it cannot vary independently from your username attribute. Given that many deployments generate their Persistent NameID based on a more stable attribute than the username (typically, an identifier internal to their IAM platform) this is unlikely to be the case.

Given the caveats on option two, you will most likely just want to release a custom NameID format based on the attribute you release to GitHub Enterprise Server as the username. Note that in either approach, if a user's username changes, they will be given a new account (and lose access to their old data), but given the caveats on how GitHub Enterprise Server handles the NameID, there isn't a better option.

After deciding on a NameID format, make sure you add that to the GitHub Enterprise Server metadata in a `<md:NameIDFormat>` element to trigger its use.

Example Shibboleth Configuration

Refer to the NameIDGenerationConfiguration topic for a full treatment of NameID features.

Example saml-nameid.xml changes

```
        <!-- SAML 2 NameID Generation -->
        <util:list id="shibboleth.SAML2NameIDGenerators">

                <ref bean="shibboleth.SAML2TransientGenerator" />

                <!--
                <ref bean="shibboleth.SAML2PersistentGenerator" />
                -->

        <!--
        Custom eduPersonPrincipalName NameID format, named after attribute name
        as per https://wiki.shibboleth.net/confluence/x/cgBSAQ. This assumes
        you've released the source attribute (ePPN) to any SPs expecting to get it.
        -->
        <bean parent="shibboleth.SAML2AttributeSourcedGenerator"
            p:format="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
            p:attributeSourceIds="#{ {'eduPersonPrincipalName'} }" />

        </util:list>
```

Example attribute-filter.xml changes

```
    <!-- ePPN only -->
    <AttributeFilterPolicy id="eduPersonPrincipalName">
        <PolicyRequirementRule xsi:type="Requester" value="https://github.myuniversity.edu" />

        <AttributeRule attributeID="eduPersonPrincipalName" permitAny="true" />
    </AttributeFilterPolicy>
```

Attribute Requirements

GitHub Enterprise Server supports a number optional attributes as documented here: https://help.github.com/enterprise/admin/guides/user-management/using-saml/#saml-attributes

The attribute names used in mapping default to non-standard values, but administrators can switch them to the standard SAML and Shibboleth OID nomenclature in the Admin Console. Of note (and interest), there are a couple of discrepancies between the documentation on their website, and the options available in the Admin Console:

1. If you do not explicitly specify an attribute mapping for username, GitHub Enterprise Server will use the value of the NameID by default. If you've released a custom human-readable NameID format sourced from your username attribute, you're set. If not, carefully choose an attribute that makes sense for your institution, is unique and non-reassignable across all of your users, that cannot vary independently of your NameID value, and is human readable. **See the "NameID Requirements" section above to understand why this is very, very important.**
2. While it doesn't appear in the Admin Console, GitHub Enterprise Server can consume an attribute (strictly) named "administrator", that contains a Boolean value indicating whether or not the authenticated user should be promoted to an administrator (or demoted to a regular user) during log-in. See the aforementioned official documentation for more details.