# Administrator's Guide for Helix Authentication Service



## Overview

The Helix Authentication Service is designed to enable certain Perforce products to integrate with your organization's Identity Provider (IdP), such as Okta (identity management), Ping Identity, OneLogin, or Cisco Duo Security.

This feature supports:

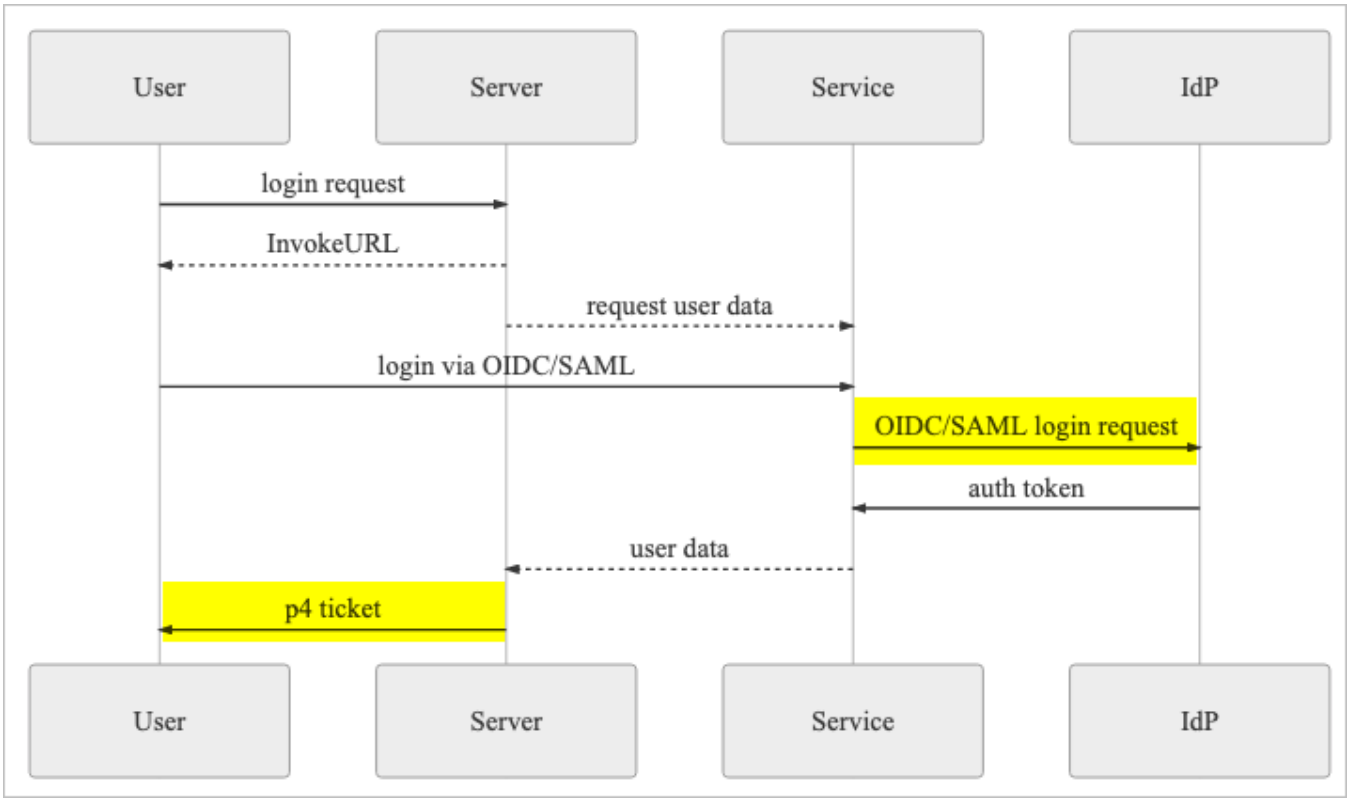
- Security Assertion Markup Language (SAML) and OpenID Connect standards
- Security-Enhanced Linux (SELinux) enabled in **enforcing** mode

After reading this document, if you want to use this integration with the Perforce Helix Core Server command-line client (P4) and the Perforce Helix Core GUI client, P4V, see Next Steps.

Although the same IdP can be used for both authentication with a Perforce Product AND authentication with web apps, these two types of authentication are completely separate from each other:

| Perforce Helix authentication | Web App authentication |
|---|---|
| Authentication with certain Perforce products, such as:<br>• P4V, the Helix Core Visual Client<br>• P4, the Helix Core command-line client<br>• P4VS, the Helix Plugin for Visual Studio<br>• P4EXP, the Helix Plugin for File Explorer<br>• Helix ALM clients | Authentication with web apps, such as Salesforce, Workday, Gmail, JIRA, Box, Splunk, ADP, and so on |

### Sequence for Helix Core




Note that this diagram indicates that the IdP authentication precedes and is separate from the Helix Core "ticket".
Therefore, when the user logs out of Helix Core, the user is not necessarily logged out from the IdP's perspective.

For Helix ALM clients, instead of a p4 ticket, the user gets a login response.

## SECURITY NOTICE ABOUT LOGGING OUT

Logging out of a Helix Core or Helix ALM client does not invoke a logout with the IdP.
Depending on the IdP, subsequently starting a Helix Core or Helix ALM client might result with the user being logged in again without the user being prompted to provide credentials.

## Download Contents

A single tarball/zip on GitHub.

Contents are:

- Helix Authorization Service source code, written in JavaScript
- Administrator's Guide for Helix Authentication Service, this PDF

## Prerequisites

### Helix Authentication Service

- Administrative expertise with the software of your Identity Provider.
- Node.js, version 12 or later.
  (The installation script installs Node.js, version 12)
- (recommended) A process manager for the Node.js runtime, such as pm2, forever, or StrongLoop.
  (The installation script installs pm2)
- A web browser. Any client using the authentication service requires a web browser.
- Any client (even the p4 command-line client) is still required to authenticate through your IdP's website. We recommend that at least one user with super level access use Perforce authentication instead of Helix Authentication Service. See the Authorizing Access section of the Helix Core Server Administrators Guide.

### Helix ALM

- Helix ALM 2019.4 or later

### Helix Core

- Helix Core Server version 2019.1 or later
- Knowledge of Perforce administration for authentication with tickets - see https://www.perforce.com/manuals/p4sag/Content/P4SAG/DB5-21975.html

### Surround SCM

- Surround SCM 2019.2 or later

## Support

The configuration of the Helix Authentication Service to work with both the Identity Provider (IdP) and the Perforce server product requires an experienced security administrator.
This effort might require assistance from Perforce Support.

# Installing Helix Authentication Service

### Installation Script

1. Run the bash script named `install.sh`, which is provided to set up a Linux-based system for running the authentication service.
   This script installs Node and pm2 process manager (you can change this recommended default), and then builds the service dependencies.
2. Modify the service configuration by editing the `ecosystem.config.js` file
   Configuration consists of defining the identity provider (IdP) details for either OIDC or SAML, and setting the `SVC_BASE_URI` of the authentication service.
3. (Recommended) For better security, replace the example self-signed SSL certificates with ones signed by a trusted certificate authority.
4. Restart the service by using `pm2 startOrReload ecosystem.config.js`

#### CentOS and Ubuntu

CentOS and Ubuntu lack Node.js packages, but there are packages available from nodesource.com that are easy to install.

| CentOS | Ubuntu |
|---|---|
| `$ sudo yum -q -y install git gcc-c++ make`<br><br>`$ curl -sL https://rpm.nodesource.com/setup_12.x \| sudo -E bash -`<br><br>`$ sudo yum -q -y install nodejs` | `$ sudo apt-get install -q -y build-essential`<br><br>`$ sudo apt-get install -q -y curl $ curl -sL https://deb.nodesource.com/setup_12.x \| sudo -E bash -`<br><br>`$ sudo apt-get install -q -y nodejs` |

### Build

With the authentication service code downloaded, open a terminal window and change to the directory containing the service code. Then download and build the dependencies using the `npm` command:

```
$ npm install
```

### Configure

The authentication service is configured using environment variables. Because there are numerous settings, it is easiest to create a file called `.env` that contains all of the settings. If you change the `.env` file while the service is running, the service must be restarted for the changes to take effect. See the **Deploy** section below because the choice of process manager affects how environment variables are defined.

#### OpenID Connect settings

| Name | Description |
|---|---|
| `OIDC_CLIENT_ID` | The client identifier as provided by the OIDC identity provider |
| `OIDC_CLIENT_SECRET` | The client secret as provided by the OIDC identity provider |
| `OIDC_ISSUER_URI` | The OIDC provider issuer URL |

OpenID Connect also has a discovery feature in which the identity provider advertises various properties. The URI path is `/.well-known/openid-configuration`, which is described in the OIDC specification. For guidance with several popular identity provider, see Configuring your IdP for Helix Authentication Service Identity Providers.

#### SAML settings

| Name | Description | Default |
|---|---|---|
| SAML_IDP_SSO_URL | URL of IdP Single Sign-On service. | *none* |
| SAML_IDP_SLO_URL | URL of IdP Single Log-Out service. | *none* |
| SAML_SP_ISSUER | The service provider identity provider that will be using the Helix Authentication Service as a SAML IdP. | urn:example:sp |
| IDP_CONFIG_FILE | Path of the configuration file that defines SAML service providers that will be connecting to the authentication service. | Note: When the authentication service is acting as a SAML identity provider, it reads some of its settings from a configuration file in the auth service installation. By default, this file is named saml_idp.conf.js and is identified by the IDP_CONFIG_FILE environment variable. It is evaluated using the Node.js require() |
| SAML_SP_AUDIENCE | Service provider audience value for AudienceRestriction assertions. | *none* |
| SAML_AUTHN_CONTEXT | The authn context defines the method by which the user will authenticate with the IdP. Normally the default value works on most systems, but it may be necessary to change this value. For example, Azure may want this set to urn:oasis:names:tc:SAML:2.0:ac:classes:Password in certain cases. | urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport |
| SAML_NAMEID_FIELD | Name of the property in the user profile to be used if nameID is missing, which is likely to be the case when using another authentication protocol (such as OIDC) with the identity provider (such as Okta). | Note: Changing the configuration file requires restarting the service because Node caches the file contents in memory. |
| SAML_NAMEID_FORMAT | The desired NameID format expected from the SAML identity provider. Defaults to urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, and can be set to any of the formats defined in the SAML specifications. | Note: If not specified, the service will try email and sub, and if those fail, the service will generate a unique identifier. The value is used as a unique key for the user data. To see the raw user profile returned by the identity provider, enable the debug logging (see the DEBUG entry below) and watch for **"legacy setting nameID"** in the log output. |

SAML identity providers advertise some of this information through their metadata URL. The URL is different for each provider, unlike OIDC. See Configuring your IdP for Helix Authentication Service Identity Providers.

### Other Settings

| Name | Description | Default |
|---|---|---|
| DEBUG | Set to auth:* to enable debug logging in the service (writes to standard error). | *none* |
| FORCE_AUTHN | If set to any non-empty value, will cause the service to require the user to authenticate, even if the user is already authenticated. For SAML, this means setting the forceAuthn field set to true, while for OIDC it will set the max_age parameter to 0. This is not supported by all identity providers, especially for OIDC. | *none* |
| SESSION_SECRET | Password used for encrypting the in-memory session data. | keyboard cat |
| SVC_BASE_URI | The authentication service base URL visible to end users. Needs to match the application settings defined in IdP configuration. | *none* |
| SP_CERT_FILE | The service provider public certificate file, needed with SAML. | *none* |
| SP_KEY_FILE | The service provider private key file, typically needed with SAML. | *none* |
| SP_KEY_ALGO | The algorithm used to sign the requests. | sha256 |
| CA_CERT_FILE | Path of certificate authority file for service to use when verifying client certificates. | *none* |
| DEFAULT_PROTOCOL | The default authentication protocol to use. Can be oidc or saml. | saml |

| | | |
|---|---|---|
| `LOGIN_TIMEOUT` | How long in seconds to wait for user to successfully authenticate. | `60` |

### Certificates

Although it is possible to use a self-signed certificate, we recommend that you use proper certificates and a trusted certificate authority (CA).

The Helix Authentication Service reads its certificate and key files using the paths defined in `SP_CERT_FILE` and `SP_KEY_FILE`, respectively. The path for the CA certificate is read from the `CA_CERT_FILE` environment variable. Providing a CA file is only necessary if the CA is not one of the root authorities whose certificates are already installed on the system. Clients accessing the `/requests/status/:id` route will require a valid client certificate signed by the certificate authority.

If the certificate files are changed, the service will need to be restarted because the service only reads the files at startup.

## Deploy

### Overview

Helix Authentication Service does not rely on a database because all data is stored temporarily in memory. The configuration is defined by environment variables. The service can serve multiple Helix Server installations because the client application that interacts with Helix Authentication Service initiates the requests and pulls data as needed. The Helix Core Server Extension asks the service for a request identifier, and the user logs in through the Helix Authentication Service with that request identifier.

### npm

The simplest way to run the Helix Authentication Service is using `npm start` from a terminal window. However, that is not robust because if the service fails, it must be restarted. Therefore, we recommend that you use a Node.js process manager to start and manage the service. We recommend the pm2 process manager, but forever or StrongLoop are among the valid alternatives. These Node.js process managers typically hook into the system process manager (e.g. `systemd`) and thus will only go down if the entire system goes down.

### pm2

The pm2 process manager has been used for testing this service. See the example configuration file, `ecosystem.config.js`, in the top-level of the service installation directory .

# Configuring your IdP for Helix Authentication Service Identity Providers

For every occurrence of `SVC_BASE_URI` in the instructions below, substitute the actual protocol, host, and port for the authentication service (e.g. `https://localhost:3000` for development environments). This address must match the URL that the identity provider is configured to recognize as the "SSO" or "callback" URL for the application.

## Configuration

### Environment Variables

In the instructions below, when referring to setting environment variables for Helix Authentication Service, there are several choices.

1. Set them in the environment, such as in the command shell.
2. Define them in a file called `.env` in the auth service source tree (in the same directory as the `package.json` file).
3. If you are using pm2, edit the `ecosystem.config.js` file.

### SAML entity identifiers

When configuring the service as a "service provider" within a SAML identity provider, provide an *entityID* that is unique within your set of registered applications. By default, the service uses the value `urn:example:sp`, which can be changed by setting the `SAML_SP_ISSUER` environment variable. Anywhere that `urn:example:sp` appears in the following instructions, replace it with the value you defined in the identity provider.

## Restarting the Service

Changing the environment settings requires restarting the service for the changes to take effect:

1. If using `npm start`, use **Ctrl-c** to stop the running process, and run **`npm start`** again.
2. If using pm2, use **`pm2 startOrReload ecosystem.config.js`**

## Auth0

### OpenID Connect

1. From the admin dashboard, click the *CREATE APPLICATION* button.
2. Enter a meaningful name for the application.
3. Select the *Regular Web Application* button, then click *Create*.
4. Open the *Settings* tab, copy the **Client ID** and **Client Secret** values to `OIDC_CLIENT_ID` and `OIDC_CLIENT_SECRET` settings in the service configuration.
5. For **Allowed Callback URLs** add `{SVC_BASE_URI}/oidc/callback`
6. For **Allowed Logout URLs** add `{SVC_BASE_URI}`
7. Scroll to the bottom of the *Settings* screen and click the **Advanced Settings** link.
8. Find the *Endpoints* tab and select it.
9. Open the **OpenID Configuration** value in a new browser tab to get the raw configuration values.
   Find `issuer` and copy the value to `OIDC_ISSUER_URI` in the service config.
   Close the configuration tab.
10. At the bottom of the page, click the *SAVE CHANGES* button.

### SAML 2.0

1. From the admin dashboard, click the *CREATE APPLICATION* button.
2. Enter a meaningful name for the application.
3. Select the *Regular Web Application* button, then click *Create*.
4. On the application *Settings* screen, add `{SVC_BASE_URI}/saml/sso` to the **Allowed Callback URLs** field.
5. For **Allowed Logout URLs** add `{SVC_BASE_URI}/saml/slo`
6. At the bottom of the page, click the *SAVE CHANGES* button.
7. Click the *Addons* tab near the top of the application page.
8. Click the *SAML2 WEB APP* button to enable SAML 2.0.
9. Enter `{SVC_BASE_URI}/saml/sso` for the **Application Callback URL**
10. Ensure the **Settings** block looks something like the following:

```
{
  "signatureAlgorithm": "rsa-sha256",
  "digestAlgorithm": "sha256",
  "nameIdentifierProbes": [
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
  ],
  "logout": {
    "callback": "{SVC_BASE_URI}/saml/slo"
  }
}
```

11. Click the *ENABLE* button at the bottom of the page.
12. On the *Usage* tab of the addon screen, copy the **Identity Provider Login URL** to the `SAML_IDP_SSO_URL` setting in the service configuration.
13. To get the SLO URL you will need to download the metadata and look for the `SingleLogoutService` element, copying the `Location` attribute value to `SAML_IDP_SLO_URL` in the config.

## Azure Active Directory

### OpenID Connect

1. Visit the Azure portal
2. Register a new application under Azure Active Directory
   - You can use a single app registration for both OIDC and SAML.
3. Enter the auth service URL as the **redirect URL**
4. Copy the *Application (client) ID* to the `OIDC_CLIENT_ID` environment variable
5. Open the OIDC metadata URL in the browser (click *Endpoints* button from app overview page)
6. Copy the `issuer` URL and enter as the `OIDC_ISSUER_URI` environment variable; if the issuer URI contains `{tenantid}` then replace it with the *Directory (tenant) ID* from the application overview page.
7. Under *Certificates & secrets*, click **New client secret**, copy the secret value to the `OIDC_CLIENT_SECRET` environment variable
8. Add a user account (*guest* works well) such that it has a defined **email** field; for whatever reason, "personal" accounts do not have the "email" field defined.
9. Make sure the Perforce user email address matches the user in Active Directory

### SAML 2.0

1. Visit the Azure portal
2. Register a new application under Azure Active Directory
   - You can use a single app registration for both OIDC and SAML.
3. Enter the auth service URL as the **redirect URL**
4. Copy the *Application (client) ID* to the `SAML_SP_ISSUER` environment variable
5. Open the API endpoints page: click the *Endpoints* button from app overview page
6. Copy the *SAML-P sign-on endpoint* value to the `SAML_IDP_SSO_URL` environment variable
7. Copy the *SAML-P sign-out endpoint* value to the `SAML_IDP_SLO_URL` environment variable
8. Set the `SAML_NAMEID_FORMAT` environment variable to the value `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`
9. Make sure the Perforce user email address matches the user in Active Directory

10. Configure the extension to use `nameID` as the `name-identifier` value.

# Okta

## OpenID Connect

1. On the Okta admin dashboard, click the **Create a New** application button (helps to use "classic ui").
2. Select *Web* as the **Platform** and *OpenID Connect* as the **Sign on method**.
3. Provide a meaningful name on the next screen.
4. For the **Login redirect URIs** enter `{SVC_BASE_URI}/oidc/callback`
5. For the **Logout redirect URIs** enter `{SVC_BASE_URI}`
6. On the next screen, find the **Client ID** and **Client secret** values and copy to the `OIDC_CLIENT_ID` and `OIDC_CLIENT_SECRET` service settings.
7. From the *Sign On* tab, copy the **Issuer** value to `OIDC_ISSUER_URI`.

If you are already logged into Okta, do ONE of the following:

- assign that user to the application you just created
- log out so you can log in again using the credentials for a user that is assigned to the application.

Otherwise you will immediately go to the "login failed" page, and the only indication of the cause is in the Okta system logs.

## SAML 2.0

1. On the Okta admin dashboard, click the **Create a New** application button (helps to use "classic ui").
2. Select *Web* as the **Platform** and *SAML 2.0* as the **Sign on method**.
3. Provide a meaningful name on the next screen.
4. Click *Save* to go to the next screen.
5. For the **Single sign on URL** enter `{SVC_BASE_URI}/saml/sso`
6. For the **Audience URI** enter `urn:example:sp`
7. Click the **Show Advanced Settings** link and check the **Enable Single Logout** checkbox.
8. For the **Single Logout URL** enter `{SVC_BASE_URI}/saml/slo`
9. For the **SP Issuer** enter `urn:example:sp`
10. For **Signature Certificate**, select and upload the `certs/server.crt` file.
11. Click the *Next* button to save the changes.
12. There may be an additional screen to click through.
13. From the *Sign On* tab, click the **View Setup Instructions** button and copy the values for IdP SSO and SLO URLs to the `SAML_IDP_SSO_URL` and `SAML_IDP_SLO_URL` settings in the environment.
14. Configure the extension to use `nameID` as the `name-identifier` value.
15. Configure the extension to use `user` as the `user-identifier` value.

If you are already logged into Okta, do ONE of the following:

- assign that user to the application you just created
- log out so you can log in again using the credentials for a user that is assigned to the application.

Otherwise you will immediately go to the "login failed" page, and the only indication of the cause is in the Okta system logs.

# OneLogin

## OpenID Connect

1. From the admin dashboard, create a new app: search for `OIDC` and select **OpenId Connect (OIDC)** from the list.
2. On the *Configuration* screen, enter `{SVC_BASE_URI}/oidc/login` for **Login Url**
3. On the same screen, enter `{SVC_BASE_URI}/oidc/callback` for **Redirect URI's**
4. Find the **Save** button and click it.
5. From the *SSO* tab, copy the **Client ID** value to the `OIDC_CLIENT_ID` environment variable.
6. From the *SSO* tab, copy the **Client Secret** value to `OIDC_CLIENT_SECRET` (you may need to "show" the secret first before the copy button will work).
7. From the *SSO* tab, find the **OpenID Provider Configuration Information** link and open in a new tab.
   Find the `issuer` and copy the URL value to the `OIDC_ISSUER_URI` environment variable.
8. Ensure the **Application Type** is set to *Web*
9. Ensure the **Token Endpoint** is set to *Basic*

## SAML 2.0

1. From the admin dashboard, create a new app: search for `SAML` and select **SAML Test Connector (Advanced)** from the list.
2. On the *Configuration* screen, enter `urn:example:sp` for **Audience**
3. On the same screen, enter `{SVC_BASE_URI}/saml/sso` for **Recipient**
4. And for *ACS (Consumer) URL Validator*, enter `.*` to match any value
5. For *ACS (Consumer) URL*, enter `{SVC_BASE_URI}/saml/sso`
6. For *Single Logout URL*, enter `{SVC_BASE_URI}/saml/slo`
7. For *Login URL*, enter `{SVC_BASE_URI}/saml/sso`

8. For *SAML initiator* select **Service Provider**
9. Find the **Save** button and click it.
10. From the *SSO* tab, copy the **SAML 2.0 Endpoint** value to the `SAML_IDP_SSO_URL` environment variable.
11. From the *SSO* tab, copy the **SLO Endpoint** value to `SAML_IDP_SLO_URL`.
12. Configure the extension to use `nameID` as the `name-identifier` value.

# Next Steps

If you want to configure Helix Authentication Service for Helix Core Server (P4) and the Helix Core visual client (P4V), see the Administrator's Guide for Helix Authentication Extension, which is available at https://github.com/perforce/helix-authentication-extension.

If you want to use the Helix Authentication Service to authenticate from Helix ALM or Surround SCM, see the Helix ALM License Server Admin Guide, which is available at https://help.perforce.com/alm/help.php?product=licenseserver&type=lsadmin.