Amtiri

TECHNICAL INFORMATION

Sign-in and Identity Providers

Generated on November 25, 2026

How operators sign in to Data Orchester, where their accounts come from, and who decides what they may do. Accounts can live in the installation itself or in a directory or single sign-on service your organization already runs; either way, **roles are always assigned in Data Orchester**.

Concepts

An **identity provider** is one source of operators. Each one is a record under `entities/identityProvider/`, and every provider is optional: it exists only while it is configured, and can be disabled or deleted like any other.

Kind	What it is
`LOCAL`	Accounts kept by the installation itself, with passwords hashed with BCrypt
`LDAP`	An LDAP directory or Active Directory, checked with the operator's directory password
`OIDC`	An OpenID Connect provider - Microsoft Entra ID, Google Workspace, Keycloak, Okta, AD FS - signed in to by redirecting the browser to it

A SAML identity provider is reached through a broker that speaks OpenID Connect to Data Orchester, and is configured as an `OIDC` provider - see *SAML through a broker*.

A **user** is either a local account or an external one. An external user is created the first time they sign in, with the account `{source}:{login}` - `corp-ad:JPEREZ`, for example - where `source` is the provider's id. That form can never be a local account's name: a local account may not contain `:`. So a directory user whose login name happens to be `ADMIN` becomes `corp-ad:ADMIN`, never the local `ADMIN`.

Behind the account, an external user is identified by the provider's **subject**: a stable id that is never reassigned, such as an Active Directory `objectGUID`. A user renamed in the directory keeps their Data Orchester record; a login name reused by somebody new gets a record of its own, numbered `~2`.

Roles are Data Orchester's. A `SECURITY` administrator assigns them in the Users pane, for local and external users alike. A directory group or an SSO claim may decide who may **enter**; it never decides what anyone may do.

Setting up the first authenticator

There is no default account and no default provider. A new installation shows **No authenticator configured** instead of a sign-in card until its first authenticator, and that authenticator's administrator, are configured - which happens when Data Orchester is installed:

Set-up	Effect
`ORCHESTER_ADMIN_ACCOUNT`	A local provider whose administrator is that account, and the local user. With `ORCHESTER_ADMIN_PASSWORD` the user gets that password; without it, the first sign-in asks the administrator to define one
`ORCHESTER_IDENTITY_PROVIDER`	The path of a provider record of any kind to install as it is, with its `administrators`. A `LOCAL` record's administrators are created as local users, as above
`java -jar data-orchester-X.Y.Z.jar identity-setup --local ACCOUNT [--must-change]` or `identity-setup --provider FILE`	The same, run from the instance home without starting Data Orchester: the jar's own command, which the install scripts run for you. With `--must-change` the administrator must replace the password at their first sign-in, so it needs `ORCHESTER_ADMIN_PASSWORD`. Installations from a Gradle distribution call it `bin/IdentitySetup`

The variables are read at start, and `identity-setup` does the same work without starting Data Orchester:

- Set-up acts **only while no identity provider exists**, and **never overwrites a user**. A second start, or a second run of `identity-setup`, changes nothing.
- When both are set, `ORCHESTER_IDENTITY_PROVIDER` is installed and `ORCHESTER_ADMIN_ACCOUNT` is ignored.
- The four roles - `CONFIGURATION`, `DASHBOARD`, `EXPORT`, `SECURITY` - are created where missing, so there is something to assign.
- A secret in a provider record is written as `env:NAME`, naming an environment variable; a literal secret is refused.
- The password is read from `ORCHESTER_ADMIN_PASSWORD` and never from the command line, where every process listing would show it.
- `identity-setup` exits `0` when it installed, `1` when it refused, such as a record that cannot be installed, `2` on a usage error and `3` when there was nothing to install because a provider already exists; the install scripts share these codes.

```
# Docker or systemd: in the service's environment
ORCHESTER_ADMIN_ACCOUNT=ADMIN
ORCHESTER_ADMIN_PASSWORD=<a long passphrase, from your secret store>

# By hand, from the instance home; restart the service afterwards.
# read -s keeps the passphrase off the screen and out of the shell's history.
read -rs ORCHESTER_ADMIN_PASSWORD && export ORCHESTER_ADMIN_PASSWORD
java -jar data-orchester-X.Y.Z.jar identity-setup --local ADMIN
unset ORCHESTER_ADMIN_PASSWORD
```

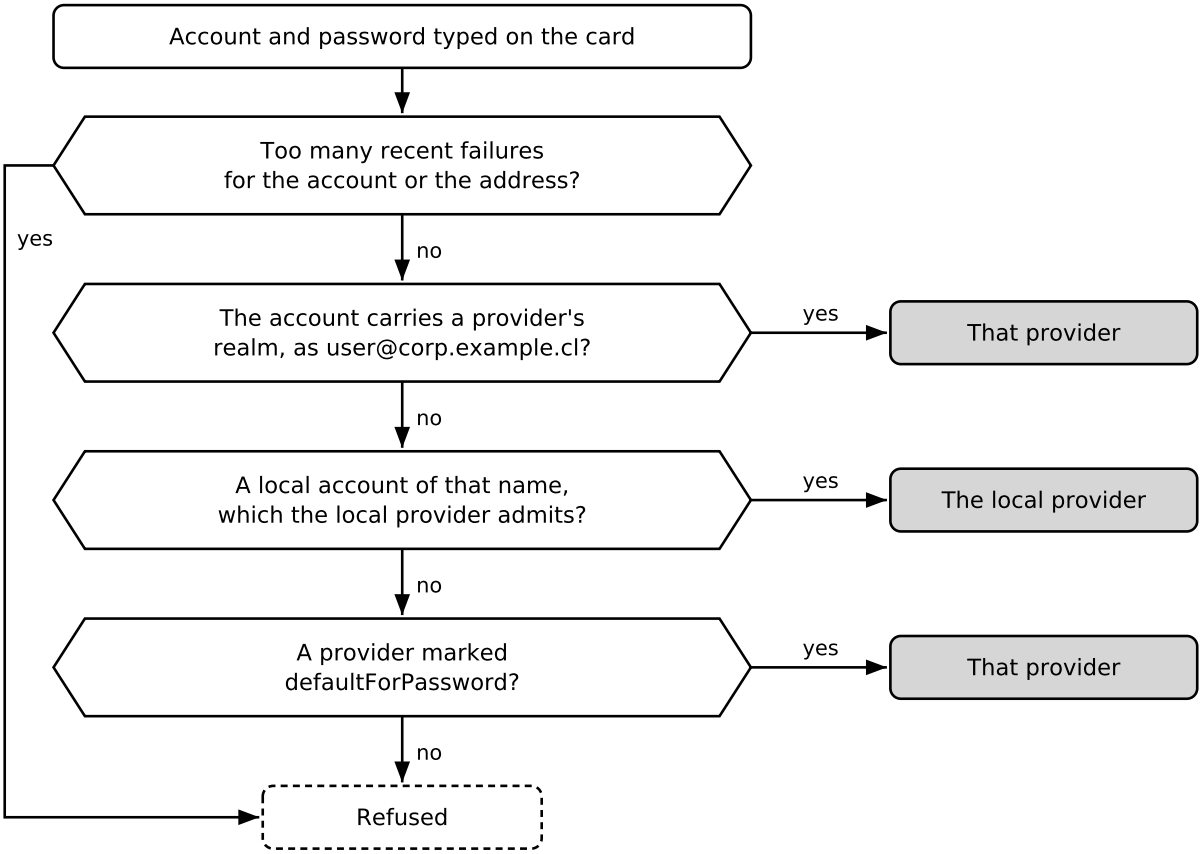
On Linux and macOS, run it as the service's account - `data-orchester` or `_dataorchester` - so that the files it writes stay the service's; on Windows, what an administrator writes in the instance home inherits its access list. On Linux, with the install scripts' paths:

```
read -rs ORCHESTER_ADMIN_PASSWORD && export ORCHESTER_ADMIN_PASSWORD
sudo --preserve-env=ORCHESTER_ADMIN_PASSWORD -u data-orchester sh -c 'cd /var/lib/data-orchester && exec /opt/d
unset ORCHESTER_ADMIN_PASSWORD
```

> **Note**
>
> An installation from before identity providers is **migrated**, not set up: on its first start after the upgrade, it gets a `local` provider whose administrators are the local users that hold `SECURITY`, and every account signs in exactly as before. No other file changes. This happens only when neither `ORCHESTER_IDENTITY_PROVIDER` nor `ORCHESTER_ADMIN_ACCOUNT` is set: with either one set, that set-up runs instead, and only the accounts it names are administrators.

How a password sign-in is routed

Every account and password typed on the card goes to **one provider at most**:



A refusal is final: a failed attempt is never passed on to a second source. Trying one password against every directory in turn would double lockouts in Active Directory. And the card says the same thing - **Wrong account or password** - whichever provider refused and whatever the reason, so it cannot be used to find out which accounts exist. The reason is in the security log - see *The security log* below.

Five failures within fifteen minutes lock the account, and separately the address they came from, for fifteen minutes. The window starts at the first failure; it does not slide.

Who may enter, and what they may do

What happens once a provider has verified the credential:

1. **Administrators** listed in the provider's `administrators` skip the entry gate (step 2) and the roles check (step 5) - see *Administrators* below. They are still provisioned, and still refused when disabled.
2. **The entry gate.** A provider may list groups whose members may sign in - `entryGroups` for a directory, an entry claim for OpenID Connect. Anyone in none of them is refused, and **no user is created** for them. An empty list admits everyone the provider authenticates.
3. **Provisioning.** The external user is found by provider and subject, or created, with **no roles**.
4. **Disabled** users are refused, whichever provider they sign in through.
5. **Awaiting roles.** An external user with no roles is refused with its own message: *Your account has no access yet. Ask an administrator to assign you roles, then sign in again.* It is shown only after the password was correct.

So the first time somebody from the directory signs in, they are told to ask for access - and appear in the Users pane under **Awaiting roles**. Once an administrator assigns roles, their next sign-in succeeds. Local accounts are not held to step 5: a local account without roles signs in as it always did.

Which roles gate what today:

Role	What it gates
`CONFIGURATION`	The Configuration section: the Program, starting and stopping the engine, the licence, configuration export and import
`SECURITY`	The Security section: users, roles and identity providers
`DASHBOARD`	Nothing in code. A dashboard with a `role` is listed only for users holding that role
`EXPORT`	Nothing in code

A dashboard whose `role` is empty is listed for every signed-in user - which is why a new external user is refused until somebody decides what they should see.

Administrators

Each provider's configuration names its **administrators** by login name. A listed user is an administrator whenever they sign in through that provider:

- the session holds `SECURITY` and `CONFIGURATION`, whatever their roles - nothing is written to their user;
- they pass the entry gate, and are never awaiting roles;
- taking a name off the list signs that administrator's sessions out at once.

Login names are written as the provider knows them, in capitals: the local account; the directory login without its realm (`JPEREZ`, not `JPEREZ@CORP.EXAMPLE.CL`); for OpenID Connect, the value of `loginNameClaim`.

Data Orchester refuses any change that would leave **no enabled provider with an administrator who can sign in** - emptying the last list, disabling or deleting the last provider that has one, or disabling or deleting the last local account listed as an administrator. A local administrator only counts while their account exists and is enabled; a provider that cannot run, because a secret it names is not set, does not count at all.

> **Important**
>
> Where the directory or the SSO service can be unreachable - a plant network, a site link that fails - keep a `local` provider with an administrator. With `allow` set to `ADMINISTRATORS_ONLY`, only its administrators can use it, and everybody else signs in through the directory. It is recommended, not enforced.

The local provider

Field	Meaning
`allow`	`ALL` (the default): every local account may sign in. `ADMINISTRATORS_ONLY`: only the provider's administrators may
`administrators`	The local accounts that are administrators

The password rules are set in the instance's `application.properties`, not in this record: `application.password.min-length`, 12 unless set, and `application.password.require-mixed-case`, `application.password.require-digit` and `application.password.require-special`, which are off unless set to `true`. The configuration reference describes them, with what NIST SP 800-63B-4 advises. From Data Orchester 1.25.0 the record's fields `passwordMinLength`, `passwordRequireMixedCase`, `passwordRequireDigit` and `passwordRequireSpecial` are ignored: a record that still holds them keeps them, but they no longer apply, so set any rule they held in the file.

A new local password must have at least the minimum number of characters, at most 128, and must not be the account name. The rules apply when a password is set or changed, never to one already stored. An account without a password is asked to define one at its first sign-in, and is granted nothing until it does.

LDAP and Active Directory

An `LDAP` provider checks a password the way a domain-joined machine does: it binds as a **service account**, finds the one entry the login names, and binds as that entry with the password typed. Data Orchester keeps nothing of the password, and roles are still assigned in the Users pane.

What the directory needs

- **A service account** that can read users and groups and nothing else: a plain domain user in no privileged group, denied interactive sign-in, and marked *Account is sensitive and cannot be delegated*. Its password lives in an environment variable, which the record names as `env:NAME`.
- **The certificate of the CA that issued the directory's own**, as PEM, under `instance/pki/ldap/` - by default `instance/pki/ldap/` followed by the provider's id and `.pem`. Every connection is verified against that file, and the host names in `servers` must be ones the directory's certificate carries.
- **An entry group** such as `CN=DO-Users,OU=Groups,DC=corp,DC=example,DC=cl`, whose members - directly or through nested groups - may sign in. Leave `entryGroups` empty only if everybody the directory knows may enter.

On an Active Directory certification authority, `certutil -ca.cert corp-ca.cer` writes its certificate; convert it to PEM on the Data Orchester host:

```
openssl x509 -inform der -in corp-ca.cer -out instance/pki/ldap/corp-ad.pem
```

Active Directory

```
{
  "id": "corp-ad",
  "kind": "LDAP",
  "name": "Corp domain",
  "enabled": true,
  "flavor": "ACTIVE_DIRECTORY",
  "transport": "LDAPS",
  "servers": ["ldaps://dc1.corp.example.cl:636", "ldaps://dc2.corp.example.cl:636"],
  "bindDn": "CN=svc-dataorchester,OU=Service Accounts,DC=corp,DC=example,DC=cl",
  "bindPassword": "env:LDAP_BIND_PASSWORD",
  "userBaseDn": "OU=Staff,DC=corp,DC=example,DC=cl",
  "groupBaseDn": "OU=Groups,DC=corp,DC=example,DC=cl",
  "entryGroups": ["CN=DO-Users,OU=Groups,DC=corp,DC=example,DC=cl"],
  "realms": { "suffixes": ["@corp.example.cl"], "prefixes": ["CORP\\"] },
  "defaultForPassword": true,
  "administrators": ["JPEREZ"]
}
```

`ACTIVE_DIRECTORY` fills in the rest: users are found by `sAMAccountName`, identified by `objectGUID`, and their groups read with `AD_NESTED` - one query that follows nesting. `jperez@corp.example.cl`, `CORP\jperez` and `jperez` are the same user. A disabled account is refused without its password being tried. An account whose password must change at next logon is told **Your directory password must be changed before you can sign in** - and only once the password proved right.

OpenLDAP

```
{
  "id": "plant-ldap",
  "kind": "LDAP",
  "name": "Plant directory",
  "enabled": true,
  "flavor": "OPENLDAP",
  "transport": "STARTTLS",
  "servers": ["ldap://ldap1.plant.example.cl:389"],
  "bindDn": "cn=dataorchester,ou=services,dc=plant,dc=example,dc=cl",
  "bindPassword": "env:LDAP_BIND_PASSWORD",
  "userBaseDn": "ou=people,dc=plant,dc=example,dc=cl",
  "groupBaseDn": "ou=groups,dc=plant,dc=example,dc=cl",
  "entryGroups": ["cn=do-users,ou=groups,dc=plant,dc=example,dc=cl"],
  "defaultForPassword": true,
  "administrators": []
}
```

`OPENLDAP` finds users by `uid` among `inetOrgPerson` entries, identifies them by `entryUUID`, and reads groups with `GROUP_SEARCH`: the `groupOfNames` entries whose `member` is the user, then the groups those belong to, in turn. With the `memberof` overlay, `MEMBER_OF` reads the user's own `memberOf` instead - direct groups only. For any other directory, `GENERIC` fills in nothing: set `userFilter` and `subjectAttribute` yourself.

Field	Default	Meaning
`flavor`	`GENERIC`	`ACTIVE_DIRECTORY`, `OPENLDAP` or `GENERIC`: fills in the user filter, the subject and the group strategy
`transport`	`LDAPS`	`LDAPS`, `STARTTLS`, or `PLAIN` - accepted only with `allowInsecureForLab`, and every sign-in over it writes `DIRECTORY_INSECURE`
`servers`		URLs, tried in order: the next one takes over when one refuses connections
`trustedCertificates`	`instance/pki/ldap/` + id + `.pem`	The PEM file the directory's certificate must chain to; always under `instance/`
`bindDn`, `bindPassword`		The service account, and `env:NAME` for its password. Anonymous binds are not supported
`userBaseDn`, `userFilter`	by flavor	Where users are searched, and how. The filter must contain `{login}`, which is escaped before it is put in
`subjectAttribute`	by flavor	The stable id - `objectGUID`, `entryUUID`; required for `GENERIC`
`displayNameAttribute`, `emailAttribute`	`displayName`, `mail`	What fills in the user's name and email
`groupStrategy`	by flavor	`AD_NESTED`, `GROUP_SEARCH` or `MEMBER_OF`
`groupBaseDn`, `groupFilter`	`userBaseDn`, `(member={dn})`	Where `AD_NESTED` and `GROUP_SEARCH` look for groups, and what `GROUP_SEARCH` asks
`entryGroups`	empty	The groups whose members may sign in - never a source of roles
`connectTimeoutMs`, `responseTimeoutMs`	`5000`	For each server
`poolSize`	`4`	Service connections kept open
`revalidateMinutes`	`15`	How often open sessions are checked against the directory

The service account's password

The record holds `env:LDAP_BIND_PASSWORD`; the password reaches Data Orchester through the service's environment:

Installation	Where the password goes
Docker, deployed with Terraform	`TF_VAR_ldap_bind_password`, which the `ldap_bind_password` variable passes on as `LDAP_BIND_PASSWORD`
Docker Compose	`LDAP_BIND_PASSWORD: ${LDAP_BIND_PASSWORD}` under `environment:`, from an `.env` file kept out of version control
Linux	A line `LDAP_BIND_PASSWORD=...` in `/etc/data-orchester/data-orchester.env`, then a restart of the service
macOS	`LDAP_BIND_PASSWORD` in the `EnvironmentVariables` of the daemon's plist, with `PlistBuddy`, then the daemon loaded again
Windows	`LDAP_BIND_PASSWORD=...` in the service's `Environment` registry value, then `install.ps1 -Unattended` again

On Linux, macOS and Windows the password goes where the install scripts keep the activation code, which they keep across an upgrade with every variable they did not write. The commands are those for the key store password in the configuration reference, with `LDAP_BIND_PASSWORD` in place of `DATAORCHESTER_TLS_PASSWORD`. `--uninstall` (`-Uninstall` on Windows) removes these secrets with the service, while the instance home and its provider records stay, so a reinstall needs them added again.

It is read when the provider first reaches the directory, so a changed password needs a restart. While the variable is unset, the provider is left out at start - `PROVIDER_DISABLED` with `SECRET_UNSET` - and its users cannot sign in.

Checking the configuration

In the provider's editor, **Test** binds as the service account and says **Bound as** its DN, or why it could not. With a login in **Login to look up**, it also finds that login's entry - without any password - and says whether the entry groups admit it. Neither needs anybody to sign in.

Directory sessions

A session granted through a directory lasts only while the directory still admits its user. Every `revalidateMinutes` each open session is checked, and a user who was deleted, disabled or taken out of every entry group has the session closed: `SESSION_REVOKED`, with the reason `GONE`, `DISABLED` or `NOT_ADMITTED`. The provider's administrators are not held to the entry groups here either, and roles are never looked at - they are Data Orchester's. An unreachable directory closes nothing: `DIRECTORY_UNAVAILABLE` is written once per outage, and sign-ins it refuses meanwhile do not count towards a lockout.

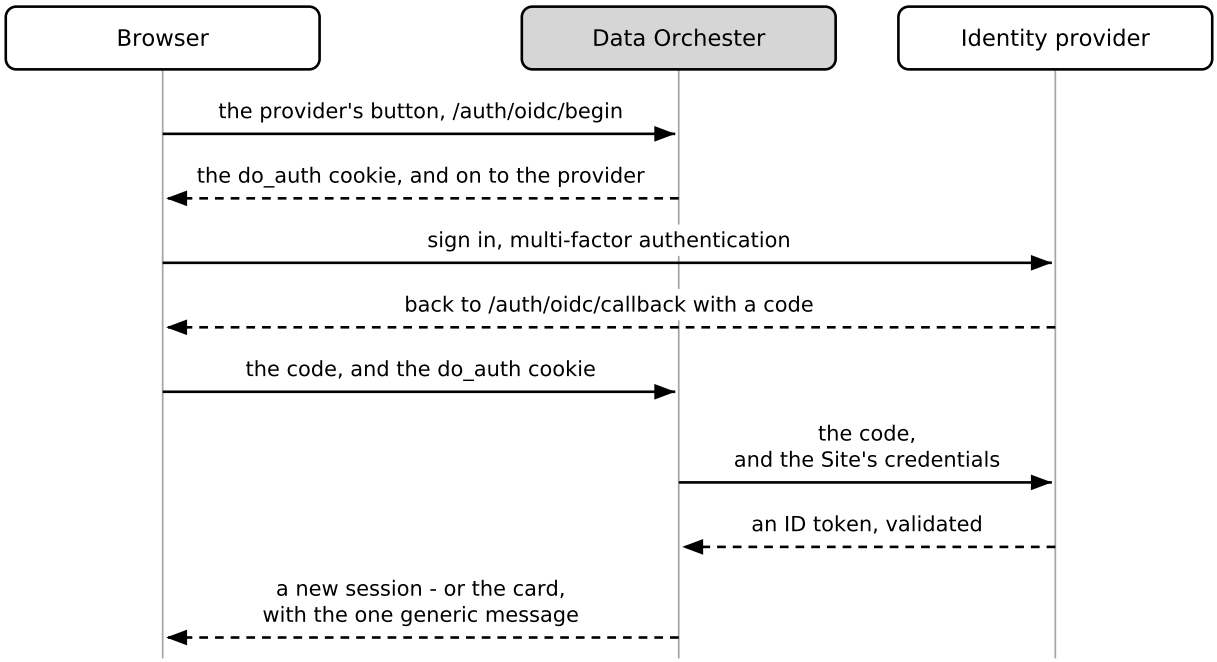
When it does not work

What Test or the log says	What to check
`connect error` with `Connection refused`, or a timeout	The URLs in `servers`, and any firewall between Data Orchester and the directory
`connect error` with `nor any of its issuers was found in the PEM file trust store`	`trustedCertificates` must hold the CA that issued the directory's certificate
`connect error` with `Hostname verification failed`	`servers` must use a name the directory's certificate carries, not an address it does not list
`invalid credentials`, binding the service account	`bindDn`, and the password in the variable `bindPassword` names
`parameter error` with `which is not set in the environment`	Set that variable for the service, then restart it
`no such object`	`userBaseDn` or `groupBaseDn` names an entry the directory does not have
`matches no entry under`, or `UNKNOWN_ACCOUNT`	The login is not under `userBaseDn`, or `userFilter` does not match it
`matches more than one entry`, or `AMBIGUOUS_ACCOUNT`	`userFilter` is too loose: tighten it until a login finds one entry
Entry groups: in none of them, or `NOT_IN_ENTRY_GROUP`	The group's DN in `entryGroups`, and the strategy: `MEMBER_OF` sees direct groups only
`ACCOUNT_DISABLED`, `PASSWORD_CHANGE_REQUIRED`	The directory's own state: the account is disabled, or its password must be changed at the domain

Active Directory adds a code to `invalid credentials`: `data 52e` is a wrong password, `data 532` an expired one, `data 533` a disabled account and `data 775` a locked one.

OpenID Connect: single sign-on

An `OIDC` provider signs operators in at their organization's identity provider - Microsoft Entra ID, Google Workspace, Keycloak, Okta or AD FS - so its passwords, multi-factor authentication and policies apply. The sign-in card shows a button for each one; the operator is sent there and back, and never types a password into Data Orchester. LDAP, Active Directory and single sign-on are in every edition.



What to set up first

- **`SITE_PUBLIC_BASE_URL`**: the address browsers reach Data Orchester at, such as `https://orchester.example.cl`. The provider sends them back to `/auth/oidc/callback` under it - the **redirect URI** to register at the provider, which the provider's editor shows. It is configured, never read from a request: the `application.public-base-url` setting holds it, and the variable overrides the setting. Without either, OIDC providers are left out at start with `PROVIDER_DISABLED`.
- **A client registered at the provider** for Data Orchester: its client id, and a client secret held in an environment variable - `env:OIDC_CLIENT_SECRET` - or, with `PRIVATE_KEY_JWT`, a signing key under `instance/pki/oidc/`.
- **To sign out at the provider too**: register `SITE_PUBLIC_BASE_URL` followed by `/` as a post-logout redirect URI, and set `rpInitiatedLogout`.

```
{
  "id": "entra",
  "kind": "OIDC",
  "name": "Microsoft Entra ID",
  "enabled": true,
  "buttonLabel": "Sign in with Microsoft",
  "issuer": "https://login.microsoftonline.com/00000000-0000-0000-0000-000000000000/v2.0",
  "clientId": "11111111-1111-1111-1111-111111111111",
  "clientSecret": "env:OIDC_CLIENT_SECRET",
  "entryClaim": "roles",
  "entryClaimValues": ["DataOrchester.User"],
  "rpInitiatedLogout": true,
  "administrators": ["JPEREZ@CORP.EXAMPLE.CL"]
}
```

Field	Default	Meaning
`issuer`		The provider's issuer, exactly as its discovery document states it; https unless `allowInsecureForLab`
`clientId`, `clientSecret`		The client registered for Data Orchester, and `env:NAME` for its secret
`clientAuthentication`	`CLIENT_SECRET_BASIC`	Or `PRIVATE_KEY_JWT`: the client proves itself with a signing key instead of a secret
`clientKeyFile`, `clientKeyId`	`instance/pki/oidc/` + id + `.key`	`PRIVATE_KEY_JWT` only: a PKCS#8 PEM key, RSA or EC P-256, and the key id the provider knows it by
`scopes`	`openid profile email`	What is asked for; `openid` is required
`loginNameClaim`	`preferred_username`	The claim giving the login name that `administrators` lists
`displayNameClaim`, `emailClaim`	`name`, `email`	What fills in the user's name and email
`entryClaim`, `entryClaimValues`		A string or array claim - a dotted path such as `realm_access.roles` reaches a nested one - and the values that admit an identity. Never a source of roles
`allowedEmailDomains`		When set, only an email the provider verified (`email_verified`) in one of these domains may enter
`buttonLabel`	the provider's name	The text of the button on the sign-in card
`rpInitiatedLogout`	off	Signing out also ends the session at the provider, through its end-session endpoint
`connectTimeoutMs`, `readTimeoutMs`	`5000`	For each call to the provider

An OIDC user is identified by the issuer and `sub` together - never by an email or a login name, which can be reassigned. A claim missing from the ID token is read from the provider's userinfo endpoint.

Recipes

Provider	Issuer	Who may enter	Notes
Microsoft Entra ID	`https://login.microsoftonline.com/` + tenant id + `/v2.0`	**Assignment required** on the enterprise application, or an app role as `entryClaim` `roles`	A web application with the redirect URI. Entra ID sends no `email_verified`, so `allowedEmailDomains` refuses everyone. Prefer app roles to `groups`: a user in more than 200 groups gets no `groups` claim
Google Workspace	`https://accounts.google.com`	No group claim: `allowedEmailDomains` with the Workspace's domain, and no entry values	Set `loginNameClaim` to `email`: Google sends no `preferred_username`
Keycloak	`https://` + host + `/realms/` + realm	A realm role as `entryClaim` `realm_access.roles`, put in the ID token by a realm-role mapper on the client	Also brokers SAML and LDAP, and runs on-premises for plants without internet
Okta	`https://` + organization + `.okta.com/oauth2/default`	A `groups` claim configured on the authorization server, as `entryClaim`	
AD FS 2016 and later	`https://` + host + `/adfs`	A claim issuance rule sending groups or roles, as `entryClaim`	An application group with a server application

The client secret

Installation	Where it goes
Docker, deployed with Terraform	`TF_VAR_oidc_client_secret`, passed on as `OIDC_CLIENT_SECRET`, and `site_public_base_url`, passed on as `SITE_PUBLIC_BASE_URL`
Docker Compose	`OIDC_CLIENT_SECRET: ${OIDC_CLIENT_SECRET}` and `SITE_PUBLIC_BASE_URL` under `environment:`, the secret from an `.env` file kept out of version control
Linux, macOS, Windows	`OIDC_CLIENT_SECRET` as the directory's password - see *The service account's password* above - and the address in `application.public-base-url`, in the instance home's `application.properties`

Checking the configuration

Test in the provider's editor fetches the discovery document and the keys, and shows the issuer, the endpoints, the key ids and the redirect URI to register. It warns when the page is open at another address than `SITE_PUBLIC_BASE_URL` - the usual sign that it is wrong behind a reverse proxy.

How a sign-in is protected

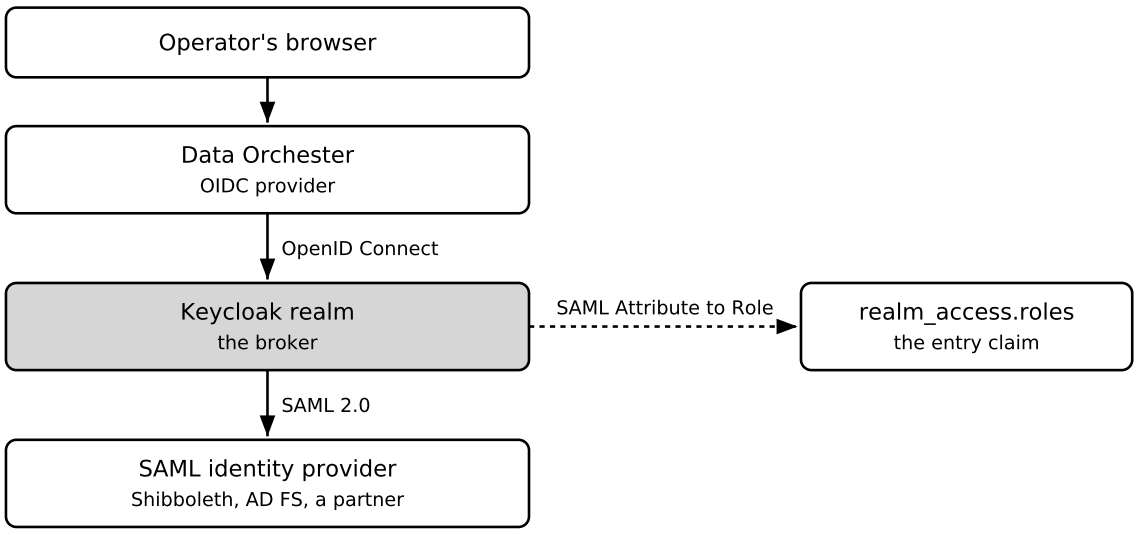
- The authorization code flow with PKCE (`S256`). A sign-in must come back within ten minutes, once, and to the browser that started it - the `do_auth` cookie, `HttpOnly` and `SameSite=Lax`, is how Data Orchester knows. A callback link somebody else completed cannot sign anyone in.
- The ID token must be signed - `RS256`, `PS256` or `ES256` - with a key the provider publishes, for this client, by this issuer, with this sign-in's nonce, and within its lifetime give or take 60 seconds. Keep the host's clock synchronised.
- Whatever goes wrong, the card says **Signing in through the provider did not complete**; the reason is in the security log.

When it does not work

What Test or the log says	What to check
`Discovery failed`	`issuer` exactly as the provider states it - the tenant, a trailing slash - and whether the host can reach the provider
The provider says the redirect URI does not match	Register exactly the redirect URI the editor shows, and check `SITE_PUBLIC_BASE_URL`
`STATE_UNKNOWN`	The sign-in took more than ten minutes, came back twice, or was never started here
`BROWSER_MISMATCH`	The browser came back without its `do_auth` cookie: cookies blocked, or `SITE_PUBLIC_BASE_URL` at another address than the one the operator started from
`TOKEN_REFUSED`	The client id, secret or key is wrong; `detail` names the provider's error
`TOKEN_INVALID`	The issuer, audience or signature do not match - or the host's clock is off by more than a minute
`LOGIN_NAME_MISSING`	`loginNameClaim` names a claim the provider does not send - `email` for Google
`NOT_IN_ENTRY_GROUP`	The entry claim is missing, or holds none of `entryClaimValues`
`EMAIL_NOT_ALLOWED`	The email is unverified, or outside `allowedEmailDomains`
`PROVIDER_ERROR`	The provider refused the user, or they cancelled; `detail` names the error
`PROVIDER_UNAVAILABLE`	The provider could not be reached

SAML through a broker

Data Orchester has no SAML code of its own. An identity provider that speaks only SAML 2.0 - Shibboleth, an older AD FS, a partner company's provider - is reached through a **broker**: a Keycloak realm that signs operators in at the SAML provider, then signs them in to Data Orchester over OpenID Connect, as any `OIDC` provider does. Many SAML providers speak OpenID Connect too - AD FS 2016 and later does - and are then configured directly, as in *OpenID Connect: single sign-on*.



In Keycloak

1. A realm for Data Orchester, with an OpenID Connect client for it and a realm-role mapper that puts the realm roles in the ID token - the Keycloak recipe above.

2. Under **Identity providers**, a **SAML v2.0** provider, from the SAML provider's metadata. Turn on **Use metadata descriptor URL**, so its signing keys are refreshed, and set:

   - **Principal type** `Subject NameID`, with the NameID policy format `persistent`: the brokered user is linked by the SAML provider's persistent NameID, never by email, which a provider may reassign or let its users change;
   - **Validate signature** on, and **Trust email** off;
   - **Sync mode** `force`, so what the SAML provider says is applied at every sign-in.

   Register Keycloak at the SAML provider with the metadata Keycloak publishes for it, under `/realms/` + realm + `/broker/` + alias + `/endpoint/descriptor`.

3. On that provider, two kinds of mapper:

   - **SAML Attribute to Role**: the group attribute - `groups`, or for AD FS `http://schemas.microsoft.com/ws/2008/06/identity/claims/role` - with the group that may enter, turned into a realm role such as `dataorchester-user`;
   - **Attribute Importer** for `email`, `firstName` and `lastName`.

4. The **first broker login** flow creates each user at their first sign-in. Keep it from linking existing accounts by email.

To send operators straight to the SAML provider instead of Keycloak's own login page, set the realm's browser flow's **Identity Provider Redirector** to the provider's alias.

In Data Orchester

The realm is an `OIDC` provider. A brokered user's Keycloak username is the SAML provider's NameID - an opaque id - so the login name comes from the email:

```
{
  "id": "partner",
  "kind": "OIDC",
  "name": "Partner company",
  "enabled": true,
  "buttonLabel": "Sign in with your company account",
  "issuer": "https://sso.example.cl/realms/dataorchester",
  "clientId": "dataorchester-site",
  "clientSecret": "env:OIDC_CLIENT_SECRET",
  "loginNameClaim": "email",
  "entryClaim": "realm_access.roles",
  "entryClaimValues": ["dataorchester-user"],
  "rpInitiatedLogout": true,
  "administrators": []
}
```

The group decides who may enter, and nothing more: roles are still assigned in the Users pane.

When the broker does not work

What you see	What to check
Keycloak reports an invalid signature or an unknown requester	The SAML provider's metadata in Keycloak, and Keycloak's broker metadata registered at the SAML provider
`NOT_IN_ENTRY_GROUP` for everybody from the SAML provider	The attribute name in **SAML Attribute to Role**, exactly as the SAML provider sends it - a URI for AD FS
`LOGIN_NAME_MISSING`	The **Attribute Importer** for `email`, and `loginNameClaim` set to `email`
A user renamed at the SAML provider gets a second user in Data Orchester	The NameID format: only a persistent NameID is a stable link, not a transient or an email one

Managing providers and users

Both live in the **Security** section of the workspace, and need `SECURITY`.

Identity providers lists every provider with its kind. The editor holds what every kind shares - id, name, enabled, administrators (one per line) - and the kind's own section below. **Test** checks what can be checked without anybody signing in: for the local provider, whether each administrator's account exists and can sign in; for every kind, whether each environment variable a secret names is set.

Saving is refused when the id is not letters, digits, `.`, `_` and `-`; when a second local provider would exist; when a second provider would be `defaultForPassword`; when a secret is not `env:NAME`; and when the administrator guard described under *Administrators* would be broken.

Users shows each user's name with their source, and marks administrators, disabled users and those awaiting roles; the list above can be filtered to **Awaiting roles**, to local accounts, or to one provider's users. In the editor:

- an external user's account and password belong to their provider and cannot be changed;
- roles and **Disabled** can be changed for every user - disabling a user also closes their open sessions;
- **Sign out everywhere** closes every session the user has open;
- a local account may not contain `:`.

Identity providers never travel with a configuration export: access control is not configuration.

Sessions

A session is named by a 128-bit random id, and holding it is all it takes to act as whoever signed in. The page sends it in a header, `X-J2W-Session`, and keeps it out of the address bar, so it ends up in no history, copied link or proxy access log. The address carries it only for a moment - on the page load right after a sign-in, and when a reload takes the tab back to its session - and the page takes it off at once. It is replaced when somebody signs in, so an address handed out before the sign-in grants nothing after it. A download opens with a one-time id of its own, never with the session.

A session has no time limit: it lasts as long as its page stays connected, whether or not anybody uses it. It ends two minutes after its page stops reaching Data Orchester - closed, or cut off from the network - at sign-out, and when its user is disabled or signed out everywhere. A page whose session has ended returns to the sign-in card on its next request.

Behind a reverse proxy: SITE_TRUSTED_PROXIES

When a reverse proxy terminates TLS in front of Data Orchester, it tells Data Orchester who the client is in the `X-Forwarded-For` and `X-Forwarded-Proto` headers. A client can write those headers too, so Data Orchester believes them **only from the proxies it trusts**: loopback, plus whatever `SITE_TRUSTED_PROXIES` lists - IPv4 and IPv6 addresses, and IPv4 CIDR blocks, separated by commas.

```
SITE_TRUSTED_PROXIES=10.0.0.5
SITE_TRUSTED_PROXIES=10.0.0.5, 172.18.0.0/16
```

The header is read from the right, and the first address that is not a trusted proxy is the client. An entry that is neither an address nor a block is logged and ignored: a typo trusts less, never more.

> **Warning**
>
> A proxy that is **not** listed is taken for the client: every operator appears to come from it, so five failed attempts from anyone lock everyone out for fifteen minutes, and `https` is not seen. Leave the variable empty only when nothing stands in front of Data Orchester.

The security log

Every sign-in, refusal, throttle, sign-out, provisioning, role change and provider change is written to `logs/security.log`, one line each, with the time in UTC:

```
2026-09-23T08:14:03.512Z WARN  event=AUTH_FAILURE provider="corp-ad" login="JPEREZ" address="10.2.0.17" reason=
```

Event	When
`AUTH_SUCCESS`, `AUTH_FAILURE`, `AUTH_THROTTLED`	A sign-in succeeded, was refused - with the `reason` - or was throttled
`ADMINISTRATOR_SIGNED_IN`	An administrator signed in; a warning when the provider is local
`USER_PROVISIONED`	A user was created: first sign-in, or install-time set-up
`ROLES_CHANGED`, `USER_DISABLED`	An administrator changed a user's roles, or whether they may sign in
`SESSION_ROTATED`, `LOGOUT`, `SIGN_OUT_EVERYWHERE`	A session was granted, was signed out, or was closed with its user's others
`PROVIDER_CHANGED`, `PROVIDER_DISABLED`	A provider was installed, saved or deleted; or left out at start, with the reason
`SESSION_REVOKED`, `DIRECTORY_UNAVAILABLE`, `DIRECTORY_INSECURE`	A directory no longer admits a session's user; a directory could not be reached while sessions were checked, once per outage; a sign-in went over a connection without TLS

The reasons a sign-in is refused include `UNKNOWN_ACCOUNT`, `WRONG_PASSWORD`, `NO_PROVIDER`, `NOT_IN_ENTRY_GROUP`, `USER_DISABLED` and `AWAITING_ROLES`, and from a directory `ACCOUNT_DISABLED`, `AMBIGUOUS_ACCOUNT`, `PASSWORD_CHANGE_REQUIRED`, `INVALID_LOGIN` and `DIRECTORY_UNAVAILABLE`, and from a single sign-on provider

`STATE_UNKNOWN`, `BROWSER_MISMATCH`, `PROVIDER_ERROR`, `TOKEN_REFUSED`, `TOKEN_INVALID`, `LOGIN_NAME_MISSING`, `EMAIL_NOT_ALLOWED` and `PROVIDER_UNAVAILABLE`, with what the provider said in `detail`. No password, secret, token or session id is ever written, and every value is sanitised: whatever is typed in the account field stays on one line.

Troubleshooting

What you see	What to check
No authenticator configured	No provider can take a sign-in. On a new installation, configure the first one - see *Setting up the first authenticator* above. If one is configured, look for `PROVIDER_DISABLED` in `logs/security.log`: a secret's variable may be unset
Your account has no access yet	The user signed in correctly and has no roles. Assign them in the Users pane (filter **Awaiting roles**)
Wrong account or password with the right password	Find the attempt's `AUTH_FAILURE` line: `NO_PROVIDER` means no realm, local account or default took the account; `NOT_IN_ENTRY_GROUP`, `USER_DISABLED` say why the provider's answer was not enough
Signing in through the provider did not complete	Find the attempt's `AUTH_FAILURE` line: its `reason` and `detail` - see *When it does not work* under *OpenID Connect*
Your directory password must be changed before you can sign in	The directory wants a new password first: the user changes it at the domain - on a domain-joined PC, or the organization's password page - and signs in again
Too many failed attempts	Wait fifteen minutes. If every operator is locked at once, a proxy is missing from `SITE_TRUSTED_PROXIES`
A provider's button or users stopped working after a restart	`PROVIDER_DISABLED` with `SECRET_UNSET` names the environment variable to set
Nobody who can administer can sign in	Stop Data Orchester, add a local account to `administrators` in `entities/identityProvider/local` - or, on an installation with no provider left, run `identity-setup --local ACCOUNT` from the instance home - and start it again

References

- After installing
- Install with Docker · Install on Linux
- RFC 4515 - LDAP: String Representation of Search Filters
- OpenID Connect Core 1.0
- RFC 7636 - Proof Key for Code Exchange
- RFC 9700 - OAuth 2.0 Security Best Current Practice
- Keycloak - Identity brokering
- NIST SP 800-63B-4 - Digital Identity Guidelines: Authentication and Authenticator Management