sawtooth-next-directory Documentation

Release 0.0

Hyperledger Contributors

Jan 04, 2019

Contents:

Identity and Access Management :: SOX Compliance Ledger :: Org Change Monitoring

NEXT is an approval platform for the enterprise incorporating aspects of identity management, role-based access control, and SOX compliance recording. Its apis are used to orchestrate changes to an organization (promotions, terminations, team changes, etc), management of roles (customization of and members in), and tasks (capabilities) granted. In daily operations, these changes are carried out using a proposal/approval process. In addition, the system can intake organizational data via one or more connectors to other identity and access providers such as Active Directory.

The main goal of the project is to supply an immutable log of organizational change events to SOX auditors.

Additional goals of NEXT include:

- Customizable, team-defined-and-managed roles
- Rapid new hire registration into multiple systems and applications
- Retroactive integration and bi-directional sync with other identity providers
- Configurable proposal escalations (scheduled, delegation-based)
- Customizable approvals (one or more required)
- Approval delegation
- Batch approvals

NEXT's underlying blockchain is Hyperledger SawTooth. Since Sawtooth is a permissioned blockchain maintained by a private cluster of nodes, ledger continuity is supported by disseminating block id hashes over time to external systems of record (eg: sent in an email). Because new hashes are constructed from the hashes of earlier blocks, and because the hashes also reflect of contents of the merkle tree content within their blocks, any manipulation of data stored in the blockchain/ledger will result in a new chain of hash ids. Identical block ids between the current blockchain and those of the external system is proof that no alterations have been made to the events stored within the ledger.

The domain model is a graph-like representation of an organizational structure comprising of users, tasks, roles, and packs (sets of roles). This representation is stored in both a blockchain for immutability, and a nosql database for access. Running processes manage the data sync between data stores as well as (if active) delta sync between NEXT and third party identity providers.

CHAPTER 1

Providers

NEXT Directory Platform has the ability to incorporate various identity providers.

Here are the providers that we currently support:

- *Active Directory Provider*
- *Azure Active Directory Provider*

Provider modules are housed within the providers package. NEXT is currently using tables in RethinkDb to act as buffers for Pub/Sub messaging between itself and external systems. Because of this, there are offsets to be reconciled when a sync job is kicked off. At a high level, there are initial inbound and outbound syncs that transfer the majority of records between systems followed by scheduled jobs that sync new changes that come in. Refer to the following diagram:

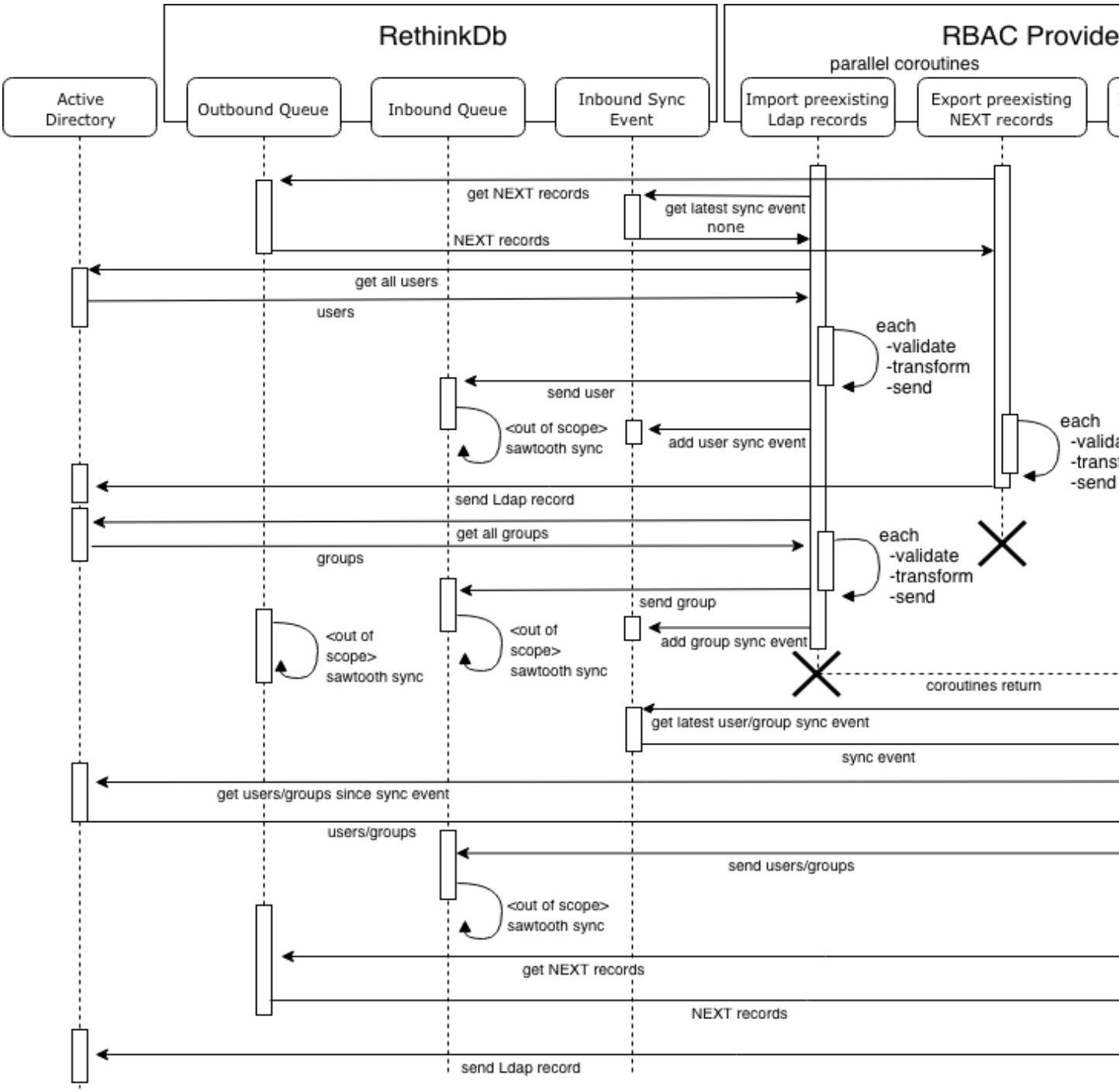

Fig. 1: Ldap inbound and outbound sync

CHAPTER 2

Active Directory Provider

The NEXT Identity platform interacts with Active Directory Provider through three syncs:

Initial Inbound Sync The initial inbound sync pulls all user and group records from Active Directory and stores it in a queue to be imported to the NEXT platform. This sync will begin when the LDAP daemon is first initialized. After a successful initial inbound sync, you would be able to see all your Active Directory users and groups on the NEXT platform.

Delta Inbound Sync The delta inbound sync pulls all user or group records from Active Directory that have been modified since the last inbound sync occurred. The records will be queued and users and groups will be updated accordingly on the NEXT platform. To check for users or groups that have been modified since the last sync occurred, we utilized the whenChanged field on each Active Directory record. A search query will look for all records that has a whenChanged timestamp at a later date than the last sync time. By default, delta inbound sync will occur by default every hour, but can be configured in the environment variables (refer to the Setting up Environment Variables below for more details).

Delta Outbound Sync The delta outbound sync will record all changes to users and groups as they occur on the NEXT platform by storing the changes in a queue. From the queue, each change will be processed and Active Directory will be updated accordingly. The delta outbound sync will help keep your Active Directory stay up to date with the changes that are being performed on NEXT.

2.1 Integrating Active Directory Provider and NEXT

2.1.1 Setting up Environment Variables

To connect your organization's Active Directory provider to the NEXT platform, you must configure the following environment variables in the .env file located in the root directory.

- **LDAP_DC**: The unique domain controller for your Active Directory instance
- **LDAP_SERVER**: The IP address of the Active Directory instance containing the prefix "ldap://"
- **LDAP_USER**: The username of a user that has access to search and udpate users and groups in the Active Directory domain controller

- **LDAP_PASS**: The password used to authenticate as the LDAP_USER
- **DELTA_SYNC_INTERVAL_SECONDS**: The interval (in seconds) of when the outbound delta sync would be performed

Example list of environment variables in .env: LDAP_DC=DC=test,DC=example,DC=com

LDAP_SERVER=ldap://10.123.123.123

LDAP_USER=ad_admin

LDAP_PASS=ad_admin_pw

DELTA_SYNC_INTERVAL_SECONDS=3600

2.1.2 Active Directory Data Field Mapping

When data is imported into the NEXT platform from Active Directory, the data field names for Active Directory users and groups will be mapped to new field names to allow our platform to interact with the incoming data. Active Directory data fields are currently being mapped according to the tables below.

User Fields

AD Field	NEXT Field
objectGUID	user_id
telephoneNumber	business_phones
whenCreated	created_date
company	company_name
countryCode	country
department	department
memberOf	member_of
displayName	name
employeeID	employee_id
givenName	given_name
title	job_title
mail	email
cn	user_nickname
manager	manager
mobilePhone	mobile_phone
distinguishedName	distinguished_name
postalCode	postal_code
preferredLanguage	preferred_language
streetAddress	street_address
userPrincipalName	user_principal_name

Group Fields

AD Field	NEXT Field
objectGUID	role_id
whenChanged	created_date
description	description
name	name
groupType	group_types
member	members
managedBy	owners

CHAPTER 3

Azure Active Directory Provider

The NEXT Identity platform interacts with Azure Active Directory Provider through three syncs:

Initial Inbound Sync The initial inbound sync pulls all user and group records from Active Directory and stores it in a queue to be imported to the NEXT platform. This sync will begin when the Azure daemon is first initialized. After a successful initial inbound sync, you would be able to see all your Azure Active Directory users and groups on the NEXT platform, including a user's manager and a group's members and owner.

Delta Inbound Sync The delta inbound sync pulls all user or group records from Azure Active Directory that have been modified since the last inbound sync occurred. The records will be queued in the inbound queue and users and groups will be updated accordingly on the NEXT platform. To check for users or groups that have been modified since the last sync occurred, we utilized a changelog event hub in Azure. A statistics query set up on the event hub will look for changes to users and groups to be ingested into the inbound_queue. By default, delta inbound sync will occur by default 50 seconds.

Delta Outbound Sync The delta outbound sync will record all changes to users and groups as they occur on the NEXT platform by storing the changes in a queue called 'outbound_queue'. From the queue, each change will be processed and Azure Active Directory will be updated accordingly. The delta outbound sync will help keep your Active Directory stay up to date with the changes that are being performed on NEXT.

3.1 Integrating Active Directory Provider and NEXT

3.1.1 Setting up Environment Variables

To connect your organization's Azure Active Directory provider to the NEXT platform, you must have an application registered with application level credentials. You must also have an event hub that is ingesting the audit log for your Azure instance. Once that has been established you must configure the following environment variables in the .env file located in the root directory.

Azure Application variables:

- **CLIENT_ID**: The client id for your Azure application.
- **TENANT_ID**: The tenant id for your Azure application.

- **AUTH_TYPE**: The type of authorization your application is using for it's Azure credentails. This is either 'CERT' or 'SECRET'.

If AUTH TYPE is 'SECRET':

- **CLIENT_SECRET**: The secret for your application's authorization credentials to Azure.

If AUTH TYPE is 'CERT':

- **CLIENT_ASSERTION**: The certificate for your application's authorization credentials to Azure.

Event Hub Variables:

- **AAD_EH_SAS_POLICY**: The SAS policy for your Azure event hub
- **AAD_EH_SAS_KEY**: The SAS key for your Azure event hub
- **AAD_EH_CONSUMER_GROUP**: The consumer group for your Azure event hub
- **AAD_EH_NAMESPACE**: The namespace of your Azure event hub
- **AAD_EH_NAME**: The name of your Azure event hub

Examples of these can be found in .env.example

3.1.2 Active Directory Data Field Mapping

When data is imported into the NEXT platform from Azure Active Directory, the data field names for users and groups will be mapped to new field names to allow our platform to interact with the incoming data. Active Directory data fields are currently being mapped according to the tables below.

User Fields

Azure AD Field	NEXT Field
id	user_id
deletedDateTime	deleted_date
accountEnabled	account_enabled
businessPhones	business_phones
city	city
createdDateTime	created_date
companyName	company_name
country	country
department	department
displayName	name
employeeId	employee_id
givenName	given_name
jobTitle	job_title
mail	email
mailNickname	user_nickname
manager	manager
mobilePhone	mobile_phone
onPremisesDistinguishedName	distinguished_name
officeLocation	office_location
postalCode	postal_code
preferredLanguage	preferred_language
state	state
streetAddress	street_address
surname	surname
usageLocation	usage_location
userPrincipalName	user_principal_name
userType	user_type

Group Fields

Azure AD Field	NEXT Field
id	role_id
createdDateTime	created_date
deletedDateTime	deleted_date
description	description
displayName	name
groupTypes	group_types
mailNickname	group_nickname
members	members
classification	classification
mailEnabled	mail_enabled
securityEnabled	security_enabled
owners	owners
visibility	visibility

CHAPTER 4

Support

The easiest way to get help with the project is to join the `#sawtooth-next-directory` channel on RocketChat. We hang out there and you can get real-time help with your projects. The other good way is to open an issue on Github.

CHAPTER 5

Developer Setup

The purpose of this document is to get all developers on the same page for development when beginning so all dependencies install properly in a virtual environment and all IDE interpreters run off the same environment.

Windows users: The files in this repo use unix-style line wraps. Ensure you have git configured such that it checks out/commits with newlines as-is:

```
git config core.autocrlf
```

5.1 Prerequisites

Python 3.7 and pip are installed: https://www.python.org/downloads/

Docker and docker-compose are installed: https://docs.docker.com/docker-for-windows/install/

5.2 Operating System Setups

- Windows
- Mac/OSX
- Linux

5.2.1 Windows

Docker: Ensure all required drives are shared with Docker

A. Right click Docker on Task Tray

B. Select “Settings”

C. Select “Shared Drives”

D. Check "D" (as well as any others that show up)

E. Click "Apply"

Other Tools:

1. Git-bash must be installed: https://git-scm.com/downloads
2. Git-bash will conflict with the 64-bit version of Cyqwin. Windows users must uninstall if present and instead install the 32-bit version of Cygwin: https://cygwin.com/install.html
3. Run the Cygwin installer again and install the 64-bit versions of libtool, automake, and pkgconfig.
4. Run `echo 'export PATH="$PATH:/c/cygwin/bin"' > ~/.bashrc` in git-bash to access those tools from git-bash.
5. Add `C:\cygwin\bin` to your Windows PATH variable as well.
6. Add `COMPOSE_CONVERT_WINDOWS_PATHS=1` to .env to allow docker-compose to run as per: https://github.com/docker/for-win/issues/1829
7. Run `echo 'export PATH="$PATH:/c/Python37"' > ~/.bashrc` in git-bash to run Python3.7 and pip3 from git-bash.
8. Restart git-bash to allow the changes to take effect.

Set-up your virtual env:

1. Navigate to the top level of the project (sawtooth-next-directory)
2. Run the following:

```
python -m venv ENV
```

3. To Activate your virtual environment, run:

```
source ./ENV/scripts/activate
```

You should now see '(ENV)' at the beginning of your terminal command line. This indicates you are in your ENV. To Deactivate, run the following: `deactivate`

Tar File Dependencies:

1. Navigate to the "windows-dependencies" folder from the home folder.
2. Untar "secp256k1-0.13.2-py3.7.egg-info.tar" and "secp256k1.tar"
3. Move both untarred folders into "sawtooth-next-directoryENVLibsite-packages"

Setting up Dependencies: Prior to setting up dependencies you should activate your virtual env and have automake installed. Automake is to help with the installation of secp256k1 as it will error without during installation.

1. Follow this solution to get a C++ support for a couple of dependencies. 2. Run the following:

```
pip3 install -r requirements.txt
```

You should now have all dependencies installed within your virtualenv. You can tell by running:

```
pip freeze -l
```

This will list all of your installed dependencies. You can then deactivate your env, run the same command, and see the difference.

5.2.2 Mac/OSX

Prerequisites: Ensure brew is up to date Ensure XCode is installed and up to date

Getting Tools:

1. Run in home directory:
   A. `brew install automake`
   B. `brew install libtool`
   C. `brew install pkg-config`
   D. `python3.7 -m venv ENV`
2. Activate virtual env

```
source ENV/bin/activate
```

You should now see '(ENV)' at the beginning of your terminal command line. This indicates you are in your ENV. To Deactivate, run the following: `deactivate`

3. Run the following:

```
pip3 install -r requirements.txt
```

If you receive errors with a permission denied error with directory creation run with sudo:

```
sudo pip install -r requirements.txt
```

You should now have all dependencies installed within your virtualenv. You can tell by running

```
pip freeze -l
```

This will list all of your installed dependencies. You can then deactivate your env, run the same command, and see the difference.

5.2.3 Linux

Getting Tools:

1. Run in home directory:
   A. `apt-get install automake`
   B. `apt-get install libtool`
   C. `apt-get install pkg-config`
   D. `python3.7 -m venv ENV`
2. Activate virtual env

```
source ENV/bin/activate
```

You should now see '(ENV)' at the beginning of your terminal command line. This indicates you are in your ENV. To Deactivate, run the following: `deactivate`

3. Run the following:

```
pip3 install -r requirements.txt
```

If you receive errors with a permission denied error with directory creation run with sudo:

```
sudo pip install -r requirements.txt
```

You should now have all dependencies installed within your virtualenv. You can tell by running

```
pip freeze -l
```

This will list all of your installed dependencies. You can then deactivate your env, run the same command, and see the difference.

5.3 IDE Interpreter setup

This will of course change between each IDE for the example process I will be using PyCharm. You need to set your Interpreter path to point at the python in the ENV folder in the top level directory. If you are using PyCharm the following steps will work for you; if not please adapt to your IDE. Go to http://pylint.pycqa.org/en/latest/user_guide/ide-integration.html#pylint-in-pycharm for easy integration with different interpreters.

1. Select Preferences > Project: saw-tooth-next-directory > Project Interpreter
2. Click the 'gear' image in the top corner and click 'Add..'
3. Select 'Existing Environment' and in 'Interpreter:' from the drop down select the path that leads to your 'ENV'. This will automatically be detected and be in the dropdown.
4. Select 'OK'
5. You will see your new interpreter with all the dependencies listed.

5.4 IDE pylint setup

Once again this will be using PyCharm as an example. This is to unite all pep8 pylint errors under the same standards across IDEs. Our common standards and what to ignore and pass is defined in setup.cfg. That means that when running pylint you need to add the argument '–rcfile=setup.cfg'. Lets set this up now.

1. In PyCharm go to Preferences > Plugins and search for Pylint
2. Click on the plugin named Pylint, select Download and Install and click on restart PyCharm when prompted (You may have to click ok afterward and leave before it will restart)
3. Select Preferences > Pylint
4. For path to executable put the path to you virtual env's pylint (i.e. ENV/bin/pylint)
5. For Arguments put –rcfile=setup.cfg

You are now done. If you want it easily available for selection under External tools go to http://pylint.pycqa.org/en/latest/user_guide/ide-integration.html#pylint-in-pycharm and follow the section 'Using External Tools'

5.5 Client Setup

1. Run the following:

```
# Install Yarn and Gulp globally
npm install -g yarn gulp
```

2. Install NPM packages:

```
# Install NPM packages and create yarn.lock
cd client && yarn
```

3. Build Semantic UI (https://react.semantic-ui.com/usage/):

```
# Build Semantic UI
yarn build:semantic
```

Client Development # Watch for changes to Semantic source yarn watch:semantic