Auth Workflow

Christo

Communications between the local Ncube installation and any remote Ncube installation is restricted. Any request has to be authorized and authenticated. Operators of a remote Ncube installation can create accounts for a workspace and pass the connection details for those accounts to a user. Accounts are created using a special one time password (OTP) which can be used to update the account password upon creation. Those one time passwords expire after 48 hours.

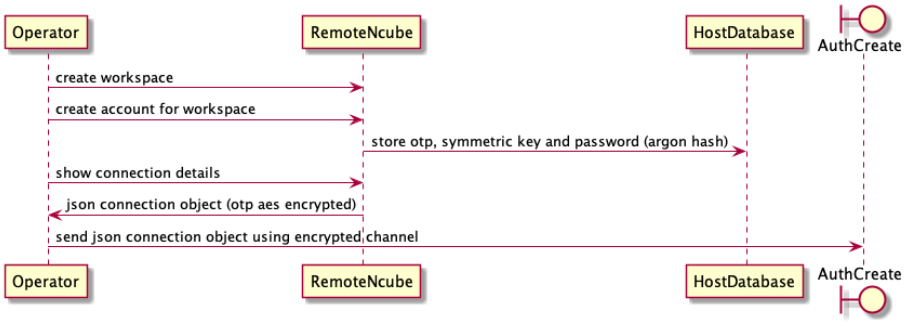
Passwords are stored on the remote Ncube installation using Argon2 password hashing. Each account has an additional key associated, which is used to encrypt the password using AES256CBC for storage on the local Ncube installation.

The JSON connection object

The operator can provide special files that contain all needed data to connect a local Ncube installation to a remote Ncube installation. These connection files can be imported using the UI and simplify linking to a remote workspace. Connection objects contain the following fields:

```
{
    "endpoint": "https://example.org",
    "workspace": "some-workspace",
    "description": "optionally a description of the workspace",
    "email": "me@example.org",
    "otp": "aes256$6DKBQtkjfXFvZnrbhozOUQ==$9wX5/XoLbCiN7fZhHu0qJPfsQsELZ9qn4+VJ+yIWkxo="
}
```

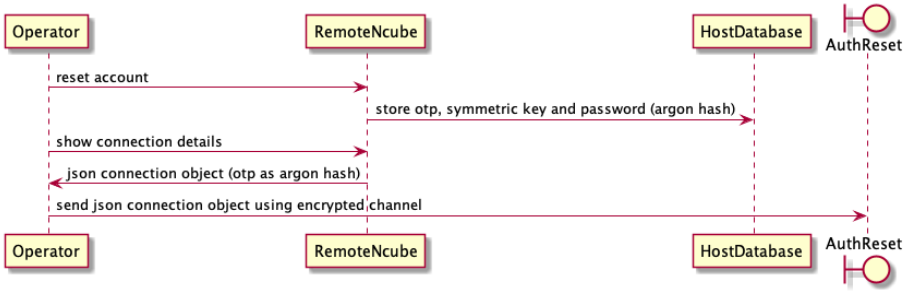
Account Create



Account creation is a purely CLI process. The `ncubectl` command line tools is used to create accounts on server setups.

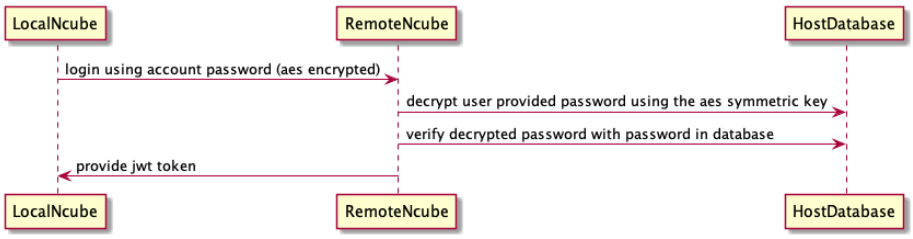
```
ncubectl set endpoint https://example.org
ncubectl workspace syrian-archive
ncubectl account syrian-archive me@example.org
ncubectl connection syrian-archive me@example.org
```

Account Reset

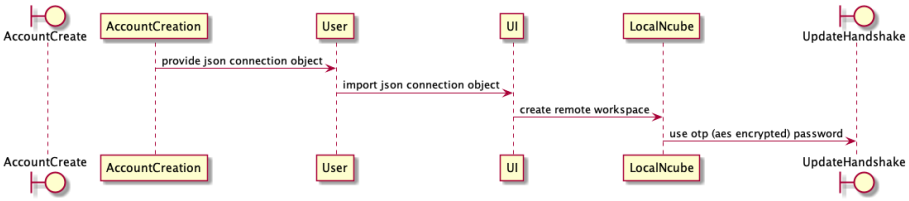



```
ncubectl reset account syrian-archive me@example.org
ncubectl connection syrian-archive me@example.org
```

Auth Login



Auth Create



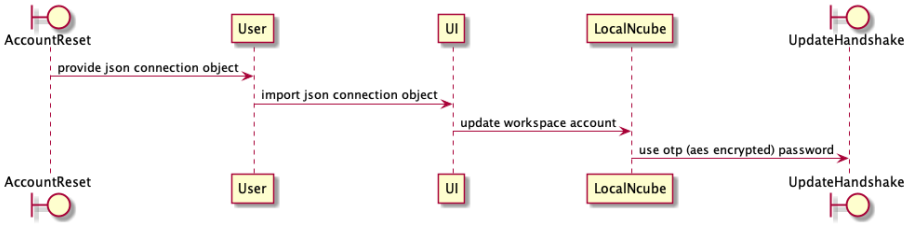

```
BROWSE: http://127.0.0.1/link
- Import ~connection.json~
- Fill in name (req)
- Fill in password & password-again (req)
- Press 'Link to remote workspace'
```

The `UI` sends a HTTP call to `LocalNcube`.

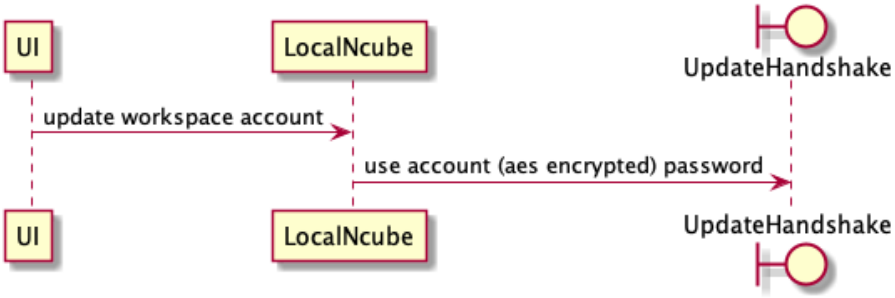
```
POST http://127.0.0.1:4066/workspaces
Content-Type: application/json

{
  "name": "Syrian Archive",
  "description": "A longer description of your workspace.",
  "kind": "remote",
  "endpoint": "https://example.org/workspaces/syrian-archive",
  "database": "http",
  "account": {
    "email": "me@example.org",
    "password": "new-pass",
    "password_again": "new-pass",
    "otp": "aes256cbc$otp-password"
  }
}
```

With this in place `LocalNcube` initiates a `Reset Workflow` and uses the `otp` password as the account password.

Auth Reset

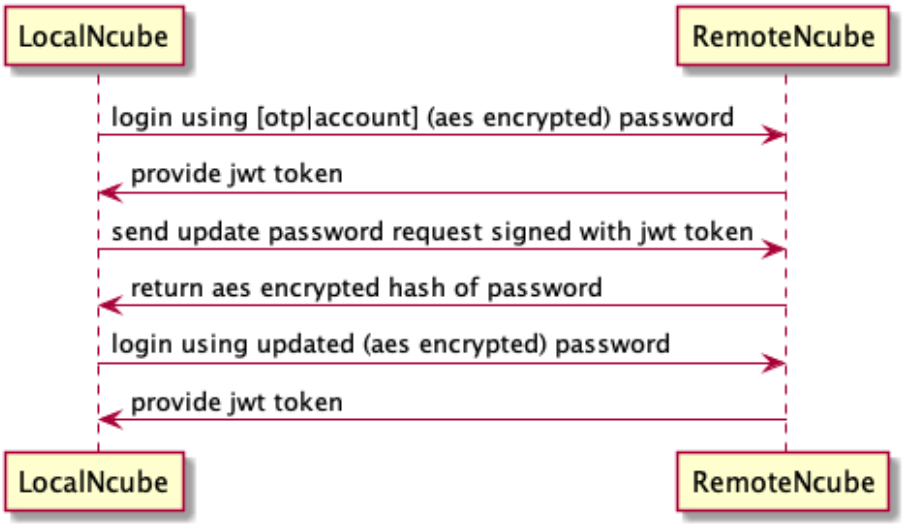


Auth Update




```
PUT http://127.0.0.1:4066/workspaces
Content-Type: application/json

{
  "account": {
    "email": "me@example.org",
    "password": "new-pass",
    "password_again": "new-pass",
    "otp": "otp-password"
  }
}
```

Update Handshake



`LocalNcube` attempts to login to `RemoteNcube`:

```
POST https://example.org/api/workspaces/syrian-archive/account
Content-Type: application/json

{
    "email": "me@example.org",
    "password": "aes256cbc$current-password"
}
```

If the login fails the `Update Handshake` fails as a whole. No resources have been yet created on `LocalNcube`.

Otherwise the response contains a valid JWT token for this account.

```
{
    "status": "success",
    "data": {
        "token": "<jwt-token>"
    }
}
```

Using the token, `LocalNcube` sends a account update request to `RemoteNcube`:

```
PUT https://example.org/api/workspaces/syrian-archive/account
Content-Type: application/json
Authorization: Bearer <jwt-token>

{
    "email": "me@example.org",
    "name": "Nomen Nescio",
    "password": "new-password",
    "password-again": "new-password"
}
```

Upon successful update `LocalNcube` updates the JWT token by repeating the login using the newly updated password.

```
POST https://example.org/api/workspaces/syrian-archive/account
Content-Type: application/json

{
    "email": "me@example.org",
    "password": "aes256cbc$hash-from-new-password"
}
```