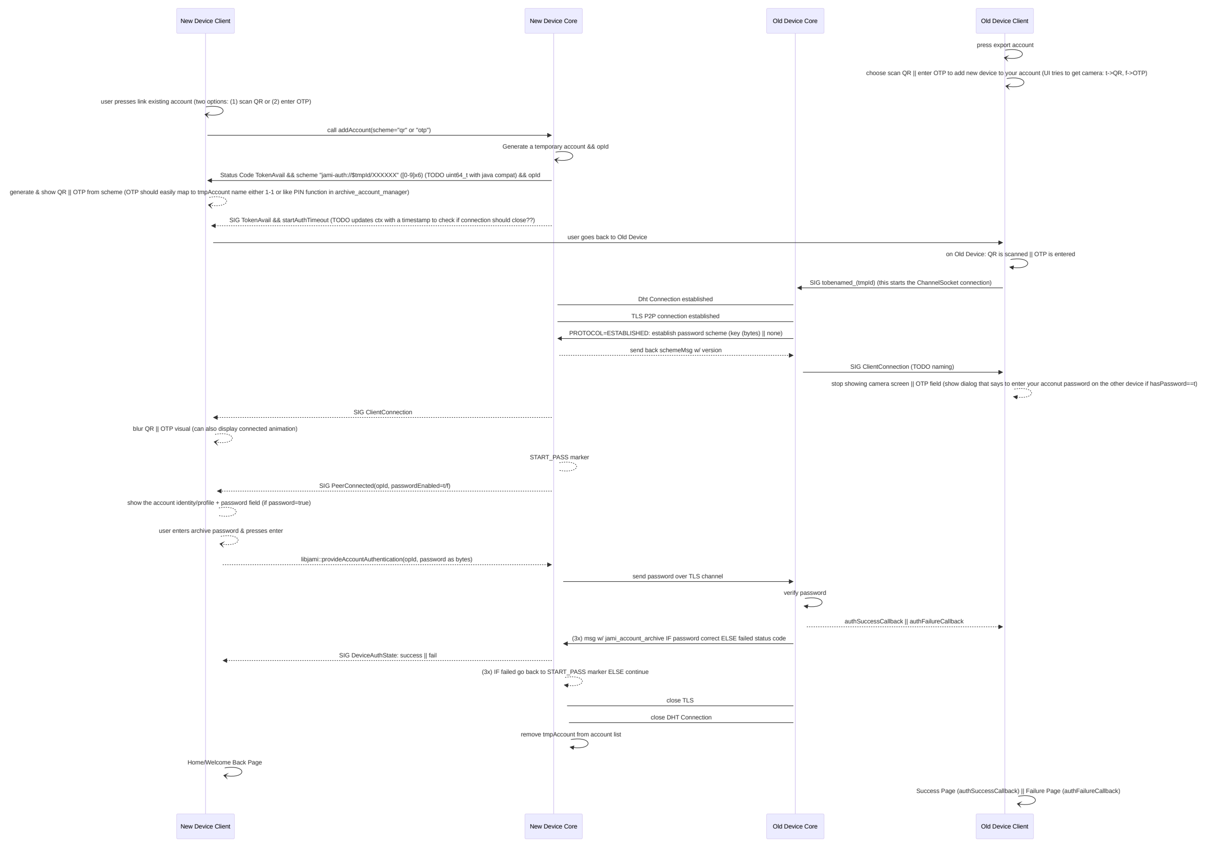
New Link Device Documentation (WIP+TODO)


New Device Client
New Device Core
Old Device Core
Old Device Client
press export account
choose scan QR || enter OTP to add new device to your account (UI tries to get camera: t->QR, f->OTP)
user presses link existing account (two options: (1) scan QR or (2) enter OTP)
call addAccount(scheme="qr" or "otp")
Generate a temporary account && opId
Status Code TokenAvail && scheme "jami-auth://$tmpId/XXXXXX" ([0-9]x6) (TODO uint64_t with java compat) && opId
generate & show QR || OTP from scheme (OTP should easily map to tmpAccount name either 1-1 or like PIN function in archive_account_manager)
SIG TokenAvail && startAuthTimeout (TODO updates ctx with a timestamp to check if connection should close??)
user goes back to Old Device
on Old Device: QR is scanned || OTP is entered
SIG tobenamed_(tmpId) (this starts the ChannelSocket connection)
Dht Connection established
TLS P2P connection established
PROTOCOL=ESTABLISHED: establish password scheme (key (bytes) || none)
send back schemeMsg w/ version
SIG ClientConnection (TODO naming)
stop showing camera screen || OTP field (show dialog that says to enter your acconut password on the other device if hasPassword==t)
SIG ClientConnection
blur QR || OTP visual (can also display connected animation)
START_PASS marker
SIG PeerConnected(opId, passwordEnabled=t/f)
show the account identity/profile + password field (if password=true)
user enters archive password & presses enter
libjami::provideAccountAuthentication(opId, password as bytes)
send password over TLS channel
verify password
authSuccessCallback || authFailureCallback
(3x) msg w/ jami_account_archive IF password correct ELSE failed status code
SIG DeviceAuthState: success || fail
(3x) IF failed go back to START_PASS marker ELSE continue
close TLS
close DHT Connection
remove tmpAccount from account list
Home/Welcome Back Page
Success Page (authSuccessCallback) || Failure Page (authFailureCallback)
New Device Client
New Device Core
Old Device Core
Old Device Client

Notes

- `AccountId` is an interaction that involves the user scanning a QR code displayed on the **New Device** using the camera on the **Old Device**
- `Dht Connection` is an abstract representation of a lower-level set of operations performed by DhtNet
- `TLS Connection` is an abstract representation of a lower-level set of operations performed by TLS
- all other interactions are using the secure TLS protocol to transmit data
- `jami_account_archive` will be a default account config OR empty if **Old Device** encounters an *error*

---

more notes on states:

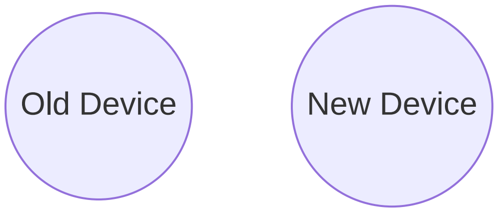
- TokenAvail
    - client: need to show QR code on current screen (think it's add account screen TODO)
- ...

todos:

- ponder JAMS integration in future (2FA for authenticateAccount, etc.)

Communication Channel Implementation

The two devices have to balance user interaction and TLS communication. The devices will use the DHT to initiate a peer-to-peer connection and communicate over a single TLS channel. Both devices send and receive messages throughout the process of transferring the archive from the old device to the new device, so the channel is bidirectional.



Future Plans for the Protocol

- the protocol should be reversible because not all devices have cameras and it may be more convenient to do the reverse OOB operation of scanning qr code from old device with new device (new device scans qr code from old device)
  - need to verify that this does not pose any security threats due to modifications to the order of presented certificate chains (ask Adrien)
  - need to rework the UI in order to accomodate the accessibility of being scan a qr from either end
- generalize protocol to request loading any account address via proxy account TLS connection with an API that accepts username (username or other identifier of the account that is associated with the requested archive), password (password associated with opening this account), sender_id (tmp account that will send the archive over TLS)
  - need to make sure that connection is initiated first before password is sent and have some sort of verification code that the user must verify before the channel communicates the password or the archive to prevent mitm/faking/phishing

Archive Download Communication Scheme v1.0

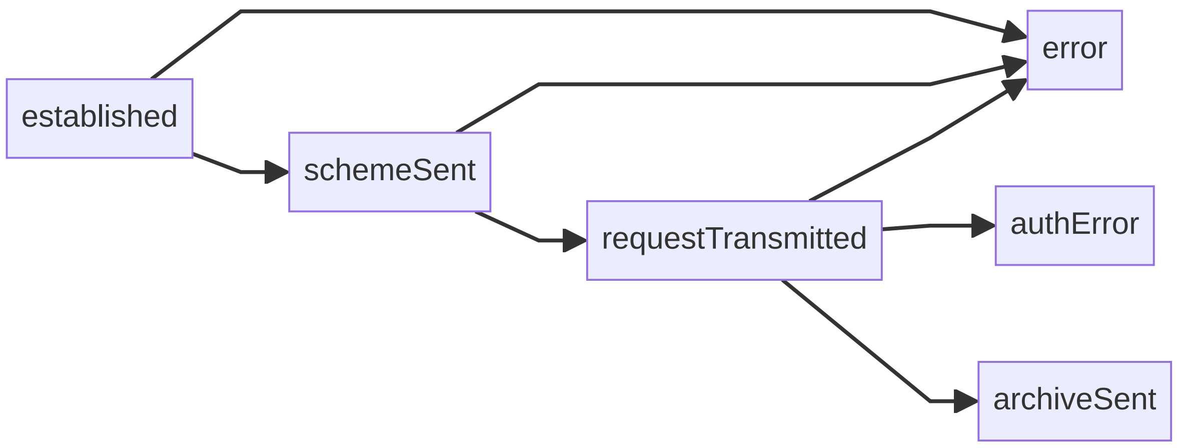
Developed after partner programming with Adrien on 2023-11-20.

Summary

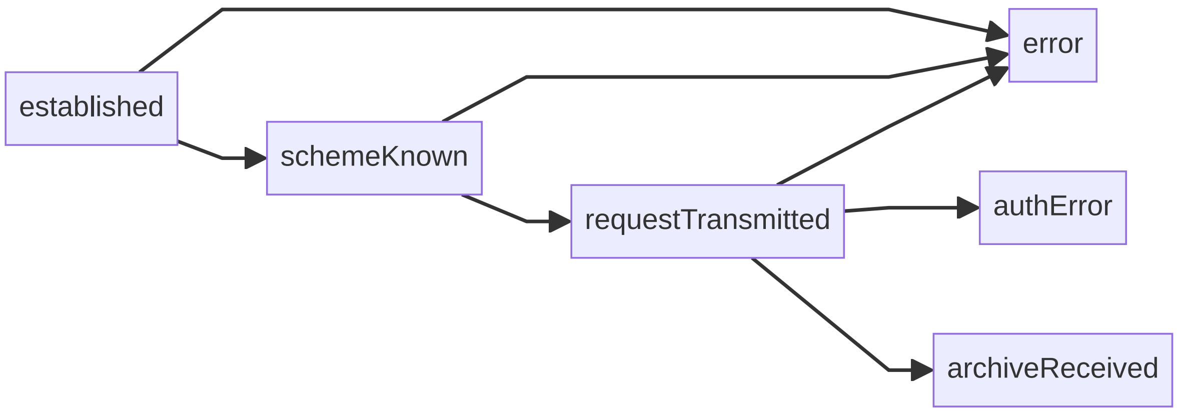
This protocol is for validating the account transfer by requiring a password or key that only the account owner can provide. This lock and key mechanism ensures that the archive cannot be exported by attackers.

Diagrams

Old Device (client):



New Device (server):



State Descriptions

- Established
    - communication channel is opened
    - scheme version needs to be communicated before any further interaction can occur
        - ensures backwards compatibility
        - web-like behavior is well-understood and can be built upon easily
- SchemeSent / SchemeKnown
    - ...
- GenericError
    - can be reached from any state if invalid message received
    - currently error states are terminal and will close the communication channel
- AuthError
    - if password is incorrect send this
    - currently error statses are terminal and will close the communication channel

State Machine for Status Transitions

Two Signals:

1. `AddDeviceStateChanged`
    1.
2. `DeviceAuthStateChanged`
    1.

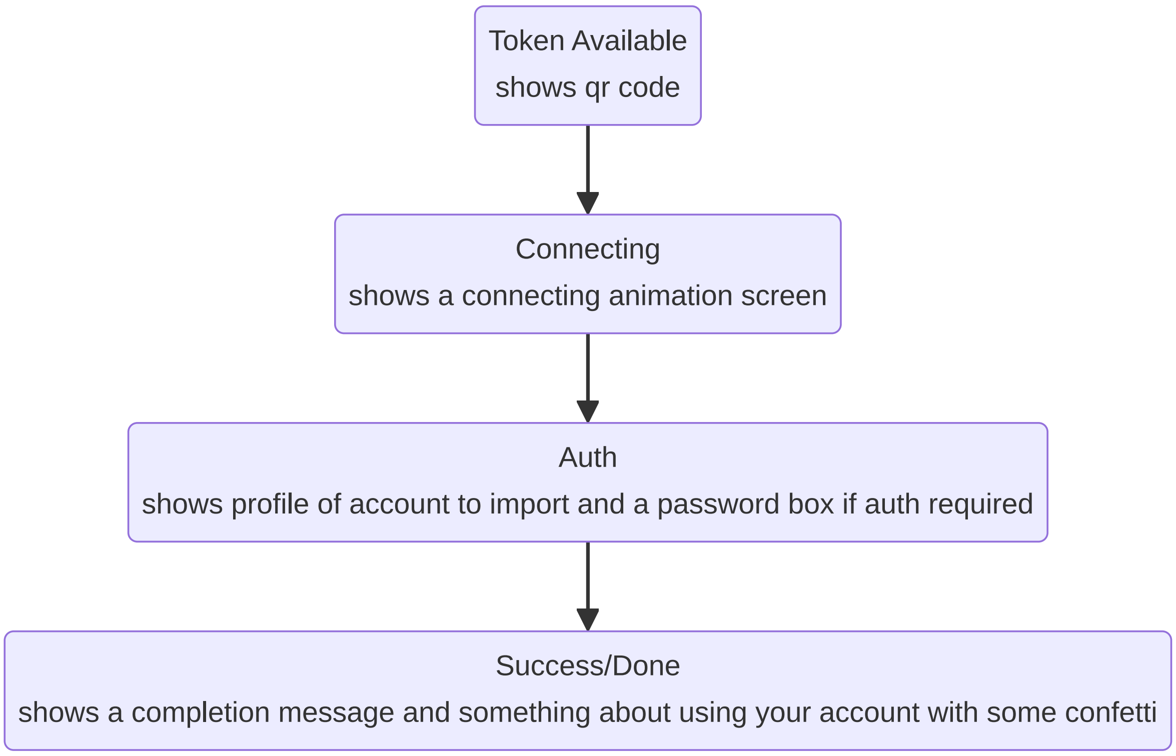
Documentation of Status Codes

- 0XX
    - general description: TODO
    - list of codes
        - TokenAvailabe: TokenAvail, 1
          -
- 1XX
    - some sort of continuation
- 2XX

Android UI Demo

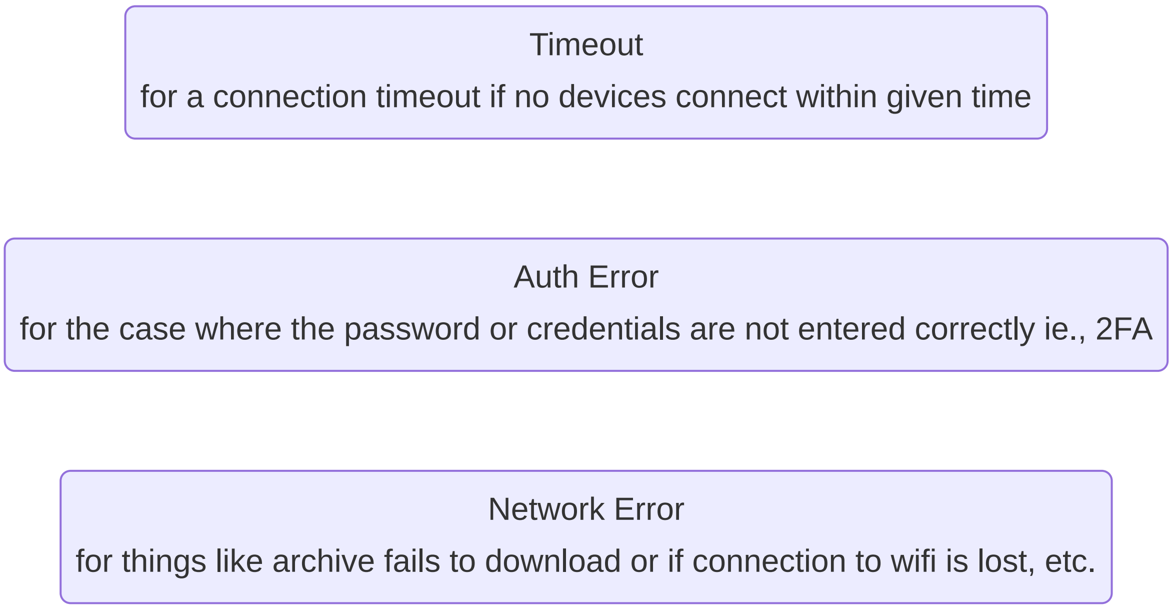
New Device

Flow


Token Available
shows qr code
Connecting
shows a connecting animation screen
Auth
shows profile of account to import and a password box if auth required
Success/Done
shows a completion message and something about using your account with some confetti

Error States

These error states can occur at any time and modify the above flow of UI changes.



Timeout

for a connection timeout if no devices connect within given time

Auth Error

for the case where the password or credentials are not entered correctly ie., 2FA

Network Error

for things like archive fails to download or if connection to wifi is lost, etc.