

# Intermission

We will return shortly!



<https://bit.ly/3rFNDfA>

# Intro to Hyperledger Aries

Daniel Bluhm  
Char Howland



**HYPERLEDGER**  
FOUNDATION

**Indicio**



# Daniel Bluhm

Indicio Software Engineer

 [daniel@indicio.tech](mailto:daniel@indicio.tech)

 [@dbluhm](#)

Co-chair of DIF Interoperability Working Group



# Char Howland

Indicio Software Engineer

 char@indicio.tech

 @cjhowland



Co-chair of Hyperledger Identity Implementers Working Group

Co-chair of Hyperledger Indy Contributors Working Group



# Agenda

1. Agents
2. Aries Cloud Agent Python and Aries Toolbox
3. Protocols and messaging
4. Starting up the agents and toolbox
5. Creating a connection
6. Issuing a credential
7. Verifying a credential

# Handout

<https://bit.ly/3rFNDfA>



# 1. Agents





# Identity Agents

## Definition

- Acts on behalf of a single identity owner
- Holds and uses cryptographic keys to securely perform its responsibilities
- Interacts with other agents through DID Communication Protocols

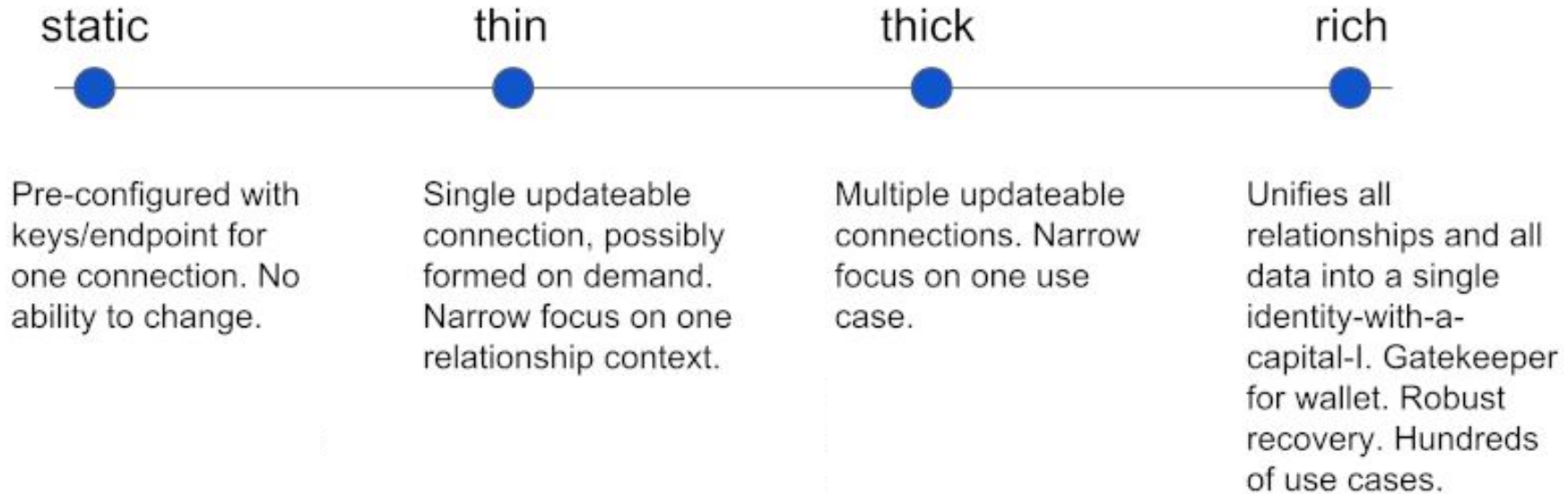
# Identity Agents

Categorization, definitions

- **Cloud Agent** - An agent in the “cloud.” Does not imply trust or lack of trust, a particular transport, ownership, etc
- **Edge Agent** - An agent located at the “edge”
- **Some other examples** - Mobile, Workstation, Server, Embedded, Browser, Blockchain (embodied in smart contract), Mesh (IoT, etc.), Paper?

# Identity Agents

Gradient of complexity







# Identity Agents

Categorization, definitions

- No hard and fast definition of an agent or a category of an agent
- These categories serve to help succinctly describe **interoperability goals**

## 2. Aries Cloud Agent Python and Aries Toolbox



# Aries Cloud Agent - Python (ACA-Py)

A foundation for building decentralized identity applications and services running in non-mobile environments

## Aries Toolbox

A tool for interacting with Aries Agents

Open-source building blocks for your own custom software to issue, hold, and verify credentials



# Aries Cloud Agent - Python (ACA-Py)

A foundation for building decentralized identity applications and services running in non-mobile environments

- Server; not intended for use on mobile or browser
- Designed for horizontal scalability
- Can act as Issuer, Verifier, and Holder of:
  - AnonCreds
  - W3C Credentials in JSON-LD Format using ED25519 or BBS Signatures
- Revocation of AnonCreds Credentials
- Multi-tenancy

# Aries Toolbox

A tool for interacting with Aries Agents

## Development and Testing Tool

- Not intended as an end-user product
- But still an effective tool for training, debugging, and testing

## Building blocks

- Electron
- Vue.JS
- ElementUI

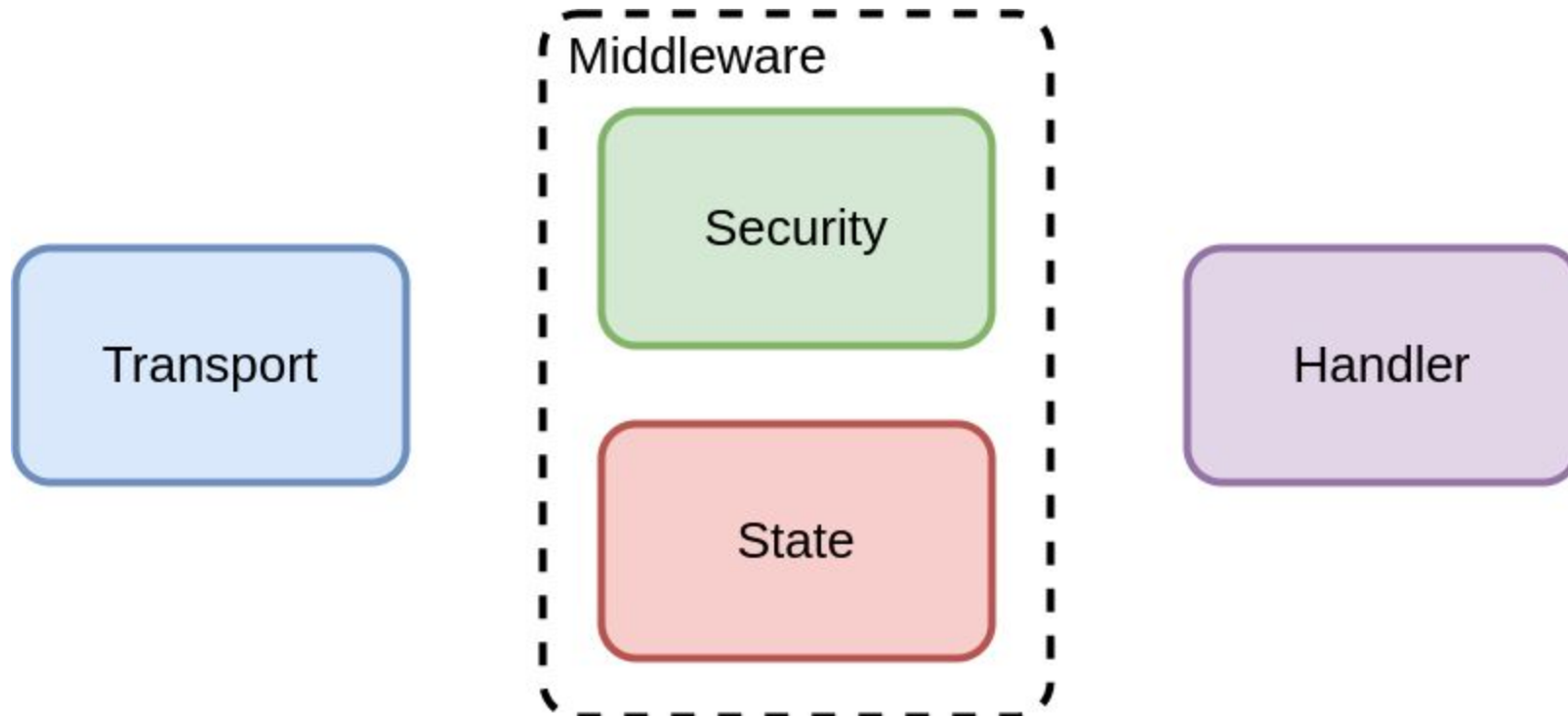
# Aries Toolbox

A tool for interacting with Aries Agents

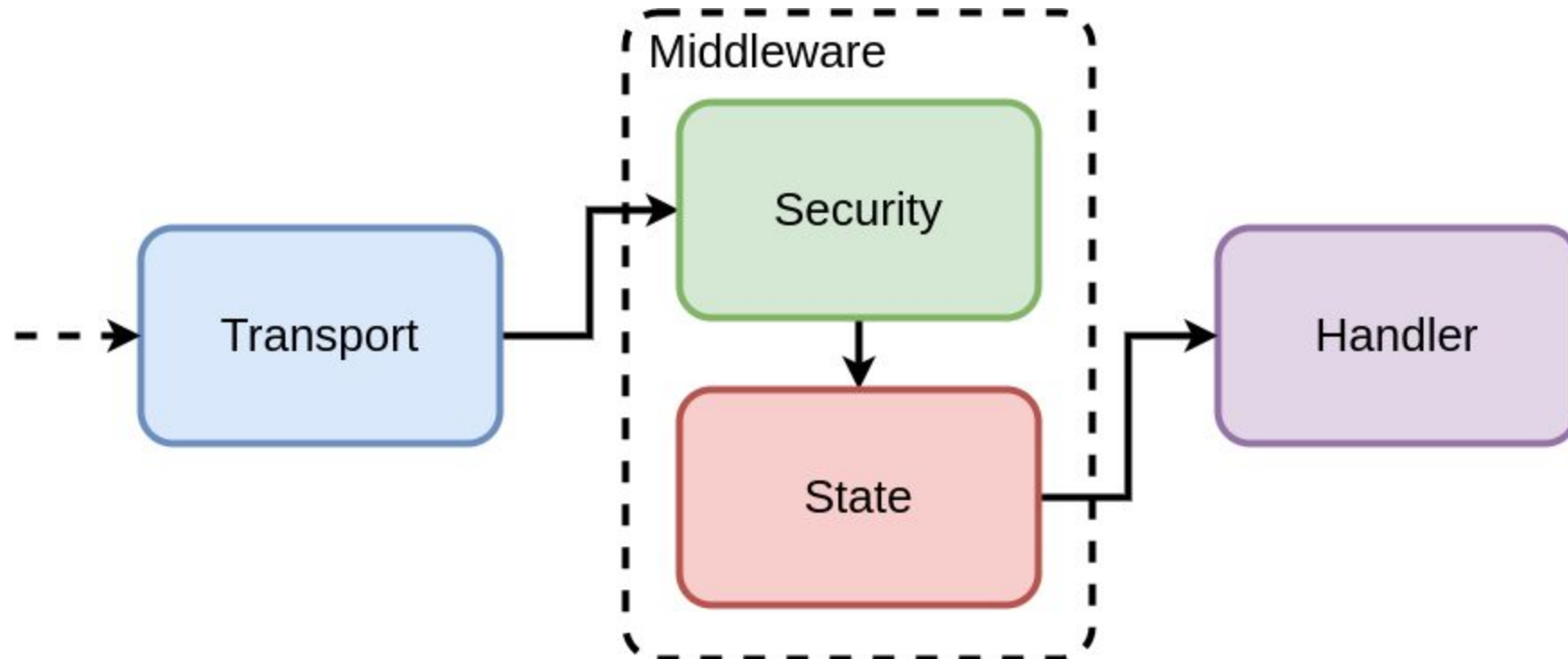
- Implemented as a minimal (thick) agent
- Protocol implementations encapsulated in their own components
- Feature detection protocol to show matching UI components
- Direct protocols
- Indirect protocols (puppet stringing)



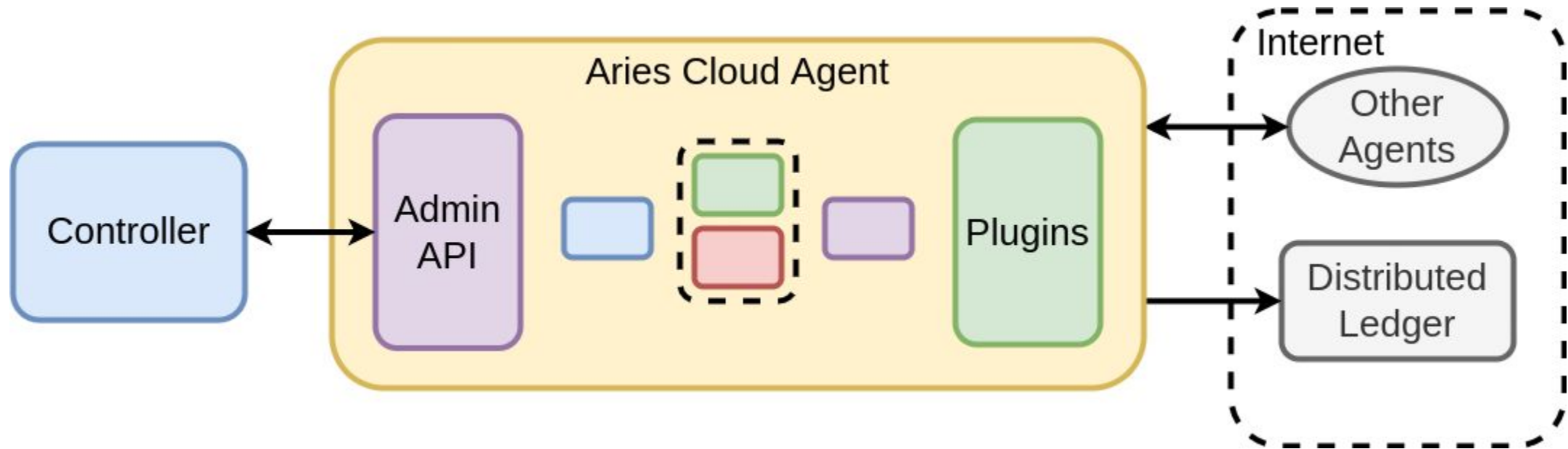
# High Level Agent Architecture



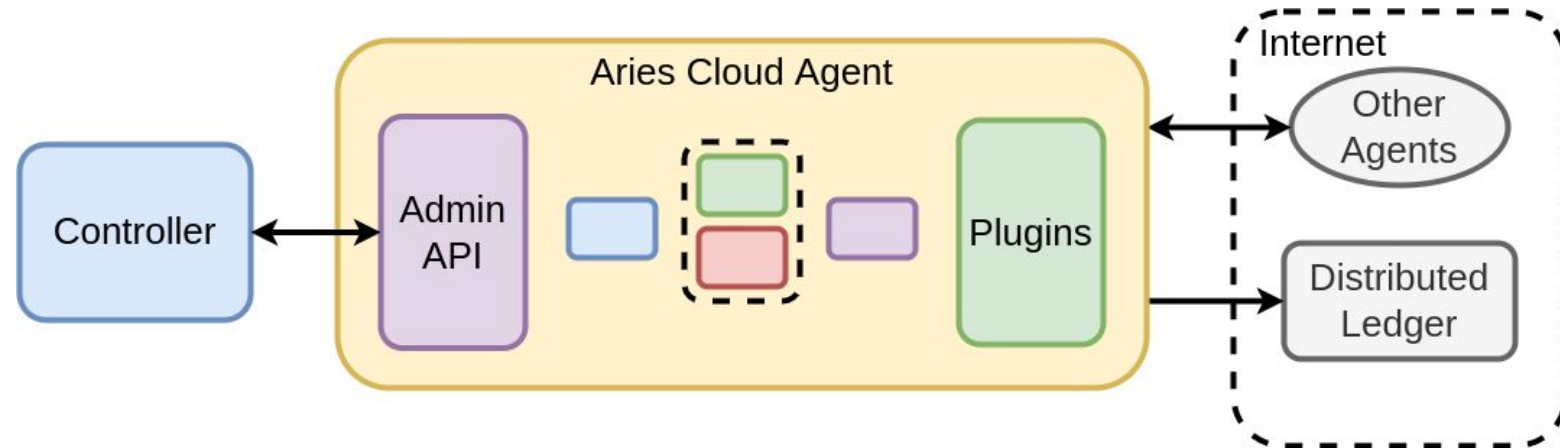
# High Level Agent Architecture



# ACA-Py Architecture



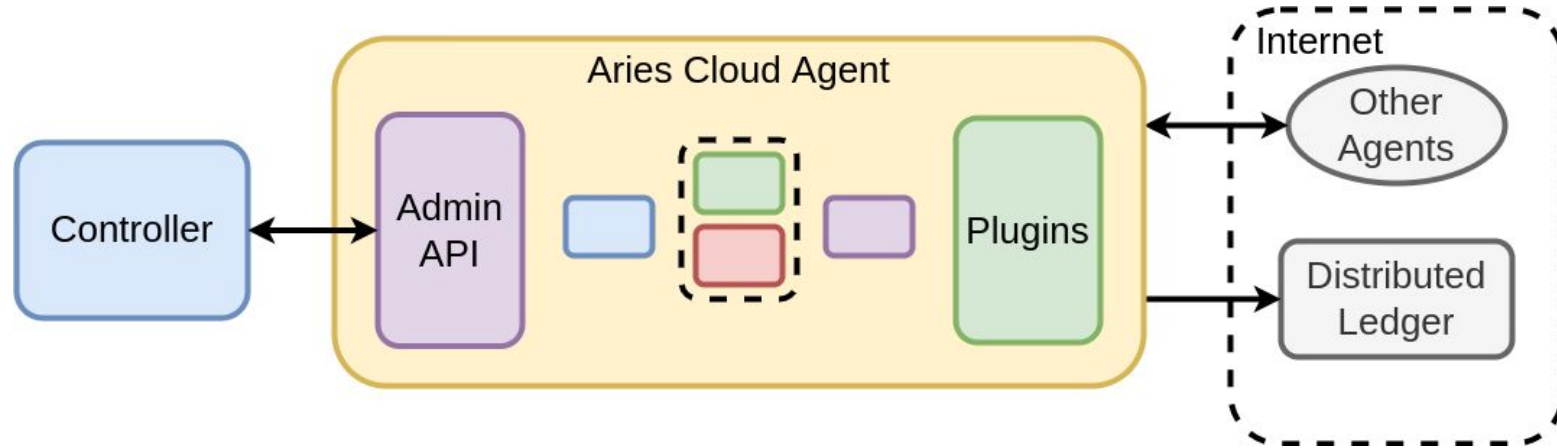
# ACA-Py Architecture



- Controller: application business logic
  - Receives events from agent
  - Sends requests to agent using HTTP JSON administrative API



# ACA-Py Architecture



- Agent responsibilities:
  - Secret storage and management
  - Connecting to other agents
  - Interacting with distributed ledgers

# ACA-Py Plugins

Extending your Agent

ACA-Py has an increasingly mature plugin system for adding:

- Protocol Implementations
- Admin API Endpoints
- Custom event handlers
- DID Resolvers
- Message Queuing Systems



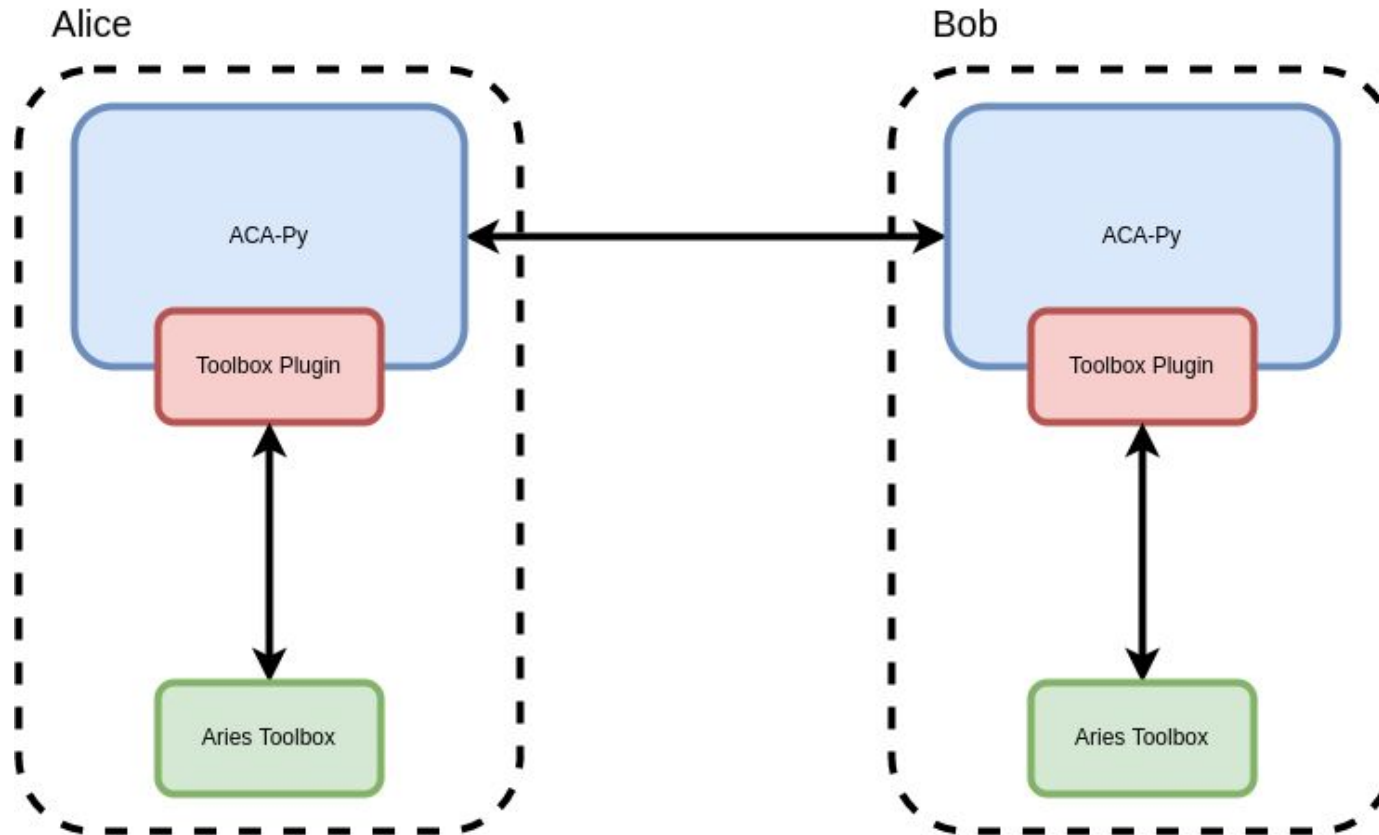
# ACA-Py Plugin for Aries Toolbox

Enabling ACA-Py to work with the Toolbox

- Indirect Protocol Implementation
- Allows for operating Aries Cloud Agent-Python through the toolbox

# ACA-Py Plugin for Aries Toolbox

Enabling ACA-Py to work with the Toolbox



# Aries Toolbox 2.0 in the works

- Built on top of actively maintained Agent Framework (AFJ)
- Will be built using React
- Will resemble its relatives in the Aries umbrella
- More maintainable path forward for DIDComm v2

# Other Aries Codebases

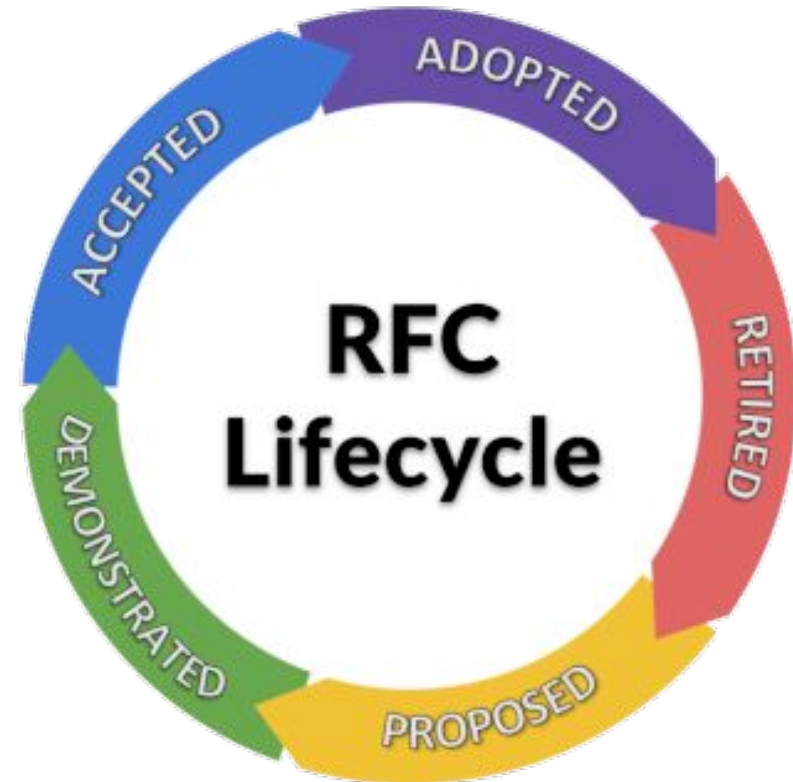
- Agents and Frameworks
  - Aries Framework JavaScript (AFJ)
  - Aries Bifold
  - Aries Framework Go (AFGO)
  - Aries Framework .Net
- Aries Agent Test Harness
- Other Agents and Services
  - Static Agent Python
  - Aries Mediator Service
  - Aries Askar



# Aries RFCs

RFCs (Request for Comments) are the source of much of the content of this workshop

<https://github.com/hyperledger/aries-rfcs>

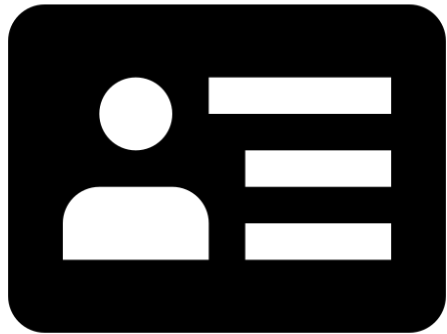


# 3. Protocols and Messaging

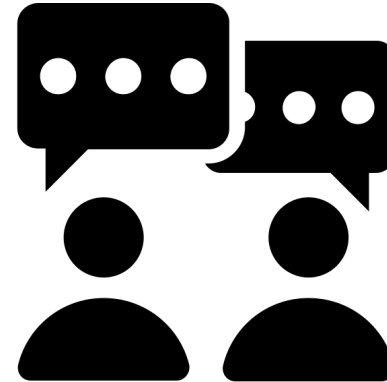
# What is DID Communication?

Secure, private peer-to-peer messaging

Verifiable Credentials  
are **about** the subject



DIDComm is communication  
**with** the subject





# Why DIDComm?

- **The** way for DID and VC-capable entities to communicate securely and privately
  - Used for credentials and presentations
  - Not limited to verifiable credentials
    - Instant messaging
    - Relationship-based user authentication (login)
    - Buying and selling

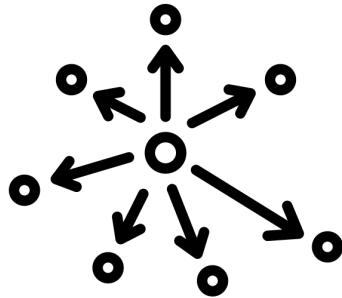
# Properties of DIDComm



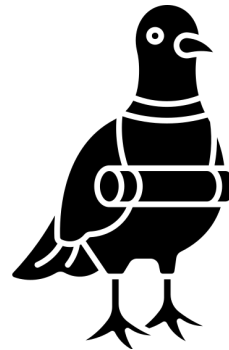
Secure



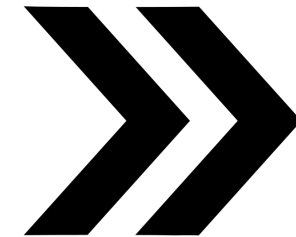
Private



Interoperable



Transport-agnostic

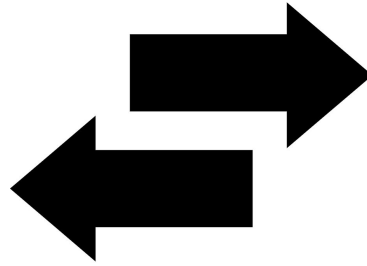


Extensible

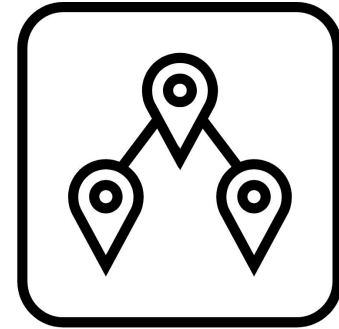
# Additional properties of DIDComm



Message-oriented



Asynchronous



Routable



# DIDComm

Interactions Between Transports, Security, and Interoperability

- Security of DIDComm is not dependent on a given transport
- Transports and routing can augment security and usability
  - Using HTTPS, many current agent implementations benefit from Perfect Forward Secrecy
    - PFS is a long-term goal of native DIDComm
- Requiring specific transports can damage interoperability
  - Take advantage of technologies through protocols

# DIDComm Layers

DID Communication Protocols

DIDs (especially Peer DIDs)

Transport (any transport)

# DIDComm Protocols

## Definition

- “A recipe for a stateful interaction.” (Daniel Hardman)
- Many of the layers underpinning DID Communication have their own protocols
  - TCP, HTTP, managing DIDs, Diffie-Hellman, etc.
- **DID Communication Protocols build on top of the lower layers to define protocols with real world social value:**
  - Forming connections, requesting and issuing credentials, proving things, buying and selling, etc.

# DIDComm Protocols

## Comparison to REST

### REST

- **R**epresentational **S**tate **T**ransfer
- Client and Server
- One request, one response
- Uses HTTP POST, GET, HEAD, DELETE, etc.
- Uses paths such as `/users/{user_id}/messages`
- Paths routed to handlers

### DIDComm

- Message-based stateful interactions
- Participants are peers with roles defined by protocol
- Many messages may be exchanged to complete an interaction
- Transport agnostic; action determined by message type
- Uses message types such as `https://didcomm.org/my-proto/1.0/my-type`
- Message types routed to handlers

# DIDComm Protocols

## Components of a Protocol

- Name and Version
- Unique URI
- Message type name
- Roles
- State and sequencing rules
- Events
- Constraints that provide trust and incentives

# DIDComm Protocols: Basic Message

## Components of a Protocol

- Name and Version: basicmessage, 1.0
- Unique URI: <https://didcomm.org/basicmessage/1.0>
- Message type name: Exactly one with name “message”:  
<https://didcomm.org/basicmessage/1.0/message>
- Roles: Sender, Receiver
- State and sequencing rules: No real state changes
- Events: Send or receive
- Constraints that provide trust and incentives: None





# Example Agent Message

Message types

```
{  
  "@id": "123456780",  
  "@type": "https://didcomm.org/basicmessage/1.0/message",  
  "~l10n": { "locale": "en" },  
  "sent_time": "2022-01-15 18:42:01Z",  
  "content": "Your hovercraft is full of eels."  
}
```

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

Message types

```
{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}
```

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

Message types

```
                                Document URI
{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}
```

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

Message types

```

{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}

```

Protocol name

|

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

Message types

```
{  
  "@id": "123456780",  
  "@type": "https://didcomm.org/basicmessage/1.0/message",  
  "~l10n": { "locale": "en" },  
  "sent_time": "2022-01-15 18:42:01Z",  
  "content": "Your hovercraft is full of eels."  
}
```

**Protocol version**

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

Message types

```
{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}
```

Message type name

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

Message types

```

{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}

```

Protocol identifier URI

|

See [RFC 0095: BasicMessage](#) for protocol details



# Example Agent Message

Message types

```

{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}

```

Message type URI

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

Message types

```
{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}
```

See [RFC 0095: BasicMessage](#) for protocol details

# A More Complex Example

Message types

```

{
  "@type": "https://didcomm.org/issue-credential/1.0/offer-credential",
  "@id": "<uuid-of-offer-message>",
  "comment": "some comment",
  "credential_preview": <json-ld object>,
  "offers~attach": [
    {
      "@id": "libindy-cred-offer-0",
      "mime-type": "application/json",
      "data": {
        "base64": "<bytes for base64>"
      }
    }
  ]
}

```



# DIDComm Message Layers

There are two layers of messages that combine to enable interoperable DID Communication

- Agent Messages (typically what is meant when “message” is used)
- Encryption Envelope or “Packed Message”



# Agent Messages

Messages sent between identities to accomplish some shared goal

- Establishing connections between identities
- Issuing and presenting verifiable credentials
- Instant messages



# Packed Messages

A wrapper around an agent message enabling messages to be securely sealed while in transport from one agent to the other

- Authenticated or anonymous encryption
- End-to-end encrypted
- Derivative of JSON Web Encryption specification

# Example Agent Message

```
{  
  "@id": "123456780",  
  "@type": "https://didcomm.org/basicmessage/1.0/message",  
  "~l10n": { "locale": "en" },  
  "sent_time": "2022-01-15 18:42:01Z",  
  "content": "Your hovercraft is full of eels."  
}
```

See [RFC 0095: BasicMessage](#) for protocol details

# Example Agent Message

```
{  
  "@id": "123456780",  
  "@type": "https://didcomm.org/basicmessage/1.0/message",  
  "~l10n": { "locale": "en" },  
  "sent_time": "2022-01-15 18:42:01Z",  
  "content": "Your hovercraft is full of eels."  
}
```

↓

pack(msg, their\_vk, my\_sk)

↓



# Example Agent Message

```

{
  "@id": "123456780",
  "@type": "https://didcomm.org/basicmessage/1.0/message",
  "~l10n": { "locale": "en" },
  "sent_time": "2022-01-15 18:42:01Z",
  "content": "Your hovercraft is full of eels."
}

```

pack(msg, their\_vk, my\_sk)

vk: verification key (verkey), Ed25519 Public Key

sk: signing key (sigkey), Ed25519 Private Key

# Example Packed Message

```
{  
  "protected": "eyJlbnMiOiJ4Y2hhY2hhMjBwb2x5MTMwNV9pZ...",  
  "iv": "Zq0rBZiA-RdFMhy2",  
  "ciphertext": "K7KxkeYGtQpbi-gNuL0bS8w724mIDP7IyGV_a...",  
  "tag": "kAuP18mwb0FFVyip1omEhQ=="  
}
```

For a detailed description of each field, see [RFC 0019 Encryption Envelope](#).

# Example Packed Message

```
{  
  "protected": "<base64 encoded headers> ",  
  "iv": "<initial vector> ",  
  "ciphertext": "<encrypted basicmessage/1.0/message> ",  
  "tag": "<hmaced headers> "  
}
```

For a detailed description of each field, see [RFC 0019 Encryption Envelope](#).

# DIDComm v2

- Ratified, rapid adoption
- Most protocols work in both v1 and v2 contexts
- Significant differences
  - Message structure split between 'headers' and body
  - Special Handling of Peer DIDs eliminated
  - DID Exchange not needed

DIDComm v1  
trust ping:

```
{  
  "@type": "https://didcomm.org/trust_ping/1.0/ping",  
  "@id": "518be002-de8e-456e-b3d5-8fe472477a86",  
  "response_requested": true  
}
```

DIDComm v2  
trust ping:

```
{  
  "type": "https://didcomm.org/trust-ping/2.0/ping",  
  "id": "518be002-de8e-456e-b3d5-8fe472477a86",  
  "from": "did:example:123456",  
  "body": {  
    "response_requested": true  
  }  
}
```

# Intermission

We will return shortly!



<https://bit.ly/3rFNDfA>

# 4. Starting up the agents and toolbox

The background of the slide is a blurred image of a person's hands typing on a laptop keyboard. In the top-left corner, there is a white network diagram consisting of several circular nodes connected by thin lines, forming a web-like structure. The overall color palette is a soft, muted blue.

# Demo

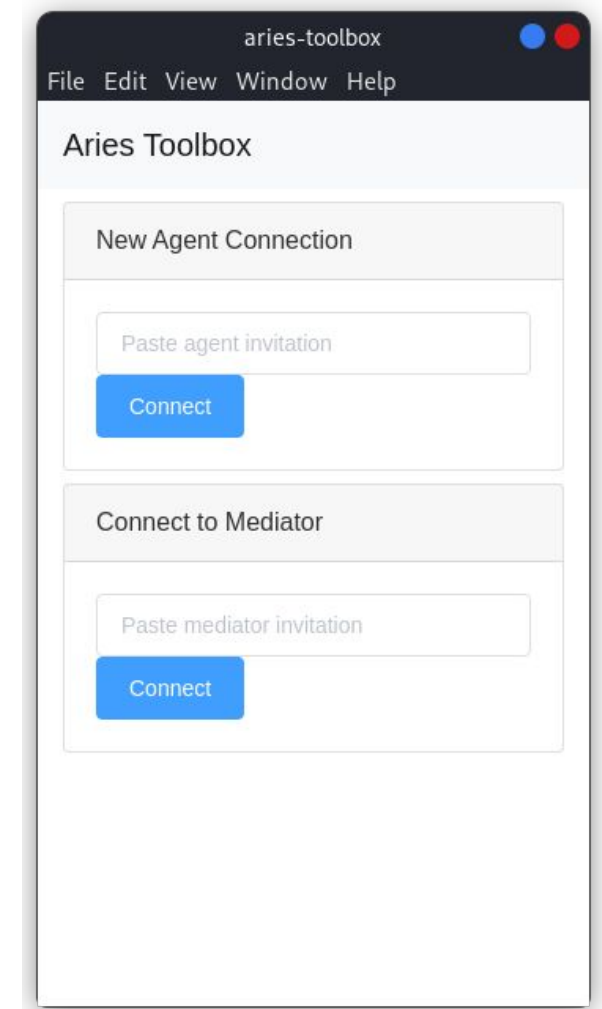
Starting up the agents and  
toolbox



# Starting up the Aries Toolbox

In one terminal, run the following:

```
cd ~/git-hltraining/aries-toolbox  
git pull  
npm install  
npm run dev
```



# Starting up the Aries Agents

From another terminal, run the following:

```
cd ~/git-hltraining/aries-acapy-plugin-toolbox/demo
git pull
git checkout hl-workshop
docker-compose -f docker-compose.alice-bob.yml up --build
```

## Permissions

When running docker-compose (or other docker commands), you may need to prefix the command with `sudo` if it initially fails due to permissions. The above command would then look like:

```
sudo docker-compose -f docker-compose.alice-bob.yml up --build
```

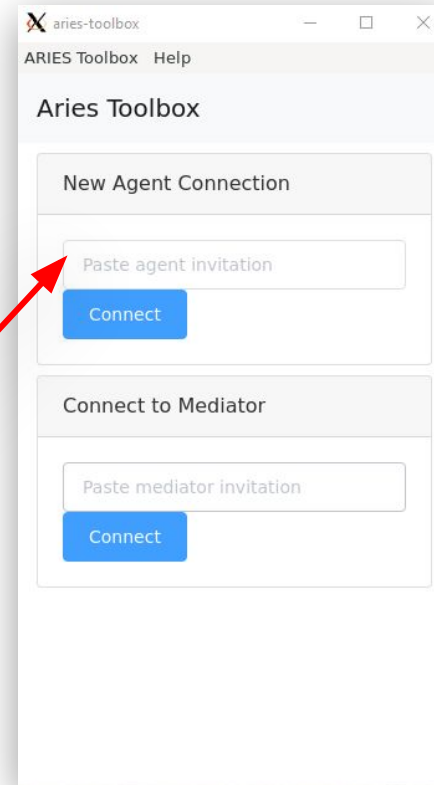
# Connecting the Agents to the Toolbox

Copy Alice's Invitation URL into the toolbox New Agent Connection bar.

```

agent-alice_1 | ::::::::::::::::::::::::::::::::::::::::::::
agent-alice_1 | :: Alice ::
agent-alice_1 | :: ::
agent-alice_1 | :: Inbound Transports: ::
agent-alice_1 | :: - http+ws://0.0.0.0:3000 ::
agent-alice_1 | :: ::
agent-alice_1 | :: Outbound Transports: ::
agent-alice_1 | :: - http ::
agent-alice_1 | :: - https ::
agent-alice_1 | :: ::
agent-alice_1 | :: Administration API: ::
agent-alice_1 | :: - http://0.0.0.0:3001 ::
agent-alice_1 | :: ::
agent-alice_1 | :: ver: 0.7.3-rc0 ::
agent-alice_1 | ::::::::::::::::::::::::::::::::::::::::::::
agent-alice_1 | Listening...
agent-alice_1 | Created new invitation
agent-alice_1 | connection: {'connection_id': 'ab2a342d-ddb0-4feb-b
e47-548ee64c2c8a', 'connection_protocol': 'connections/1.0', 'routing_sta
te': 'none', 'state': 'invitation', 'their_role': 'invitee', 'accept': 'a
uto', 'rfc23_state': 'invitation-sent', 'invitation_key': '74mLSRaHtjPCoi
XRY5Uz4URGnQwXh7hP8BoNBrZ1VoET', 'updated_at': '2022-01-17T16:18:44.93514
3Z', 'created_at': '2022-01-17T16:18:44.935143Z', 'invitation_mode': 'onc
e'}
agent-alice_1 | Invitation URL (Connections protocol):
agent-alice_1 | http://localhost:3000?c_i=eyJAdHlwZSI6ICJkaWQ6c2920k7J6Q
2JzT1loTXJqSG1xwKRUVUFTSGc7c3B1Yy9jb25uZWN0aw9ucy8xLjAvaw52aXRhdGlvbiIsIC
JAaWQiOiAiNzcyNTk1MTgtY2FmMC00NjllLWZlZGUtNmM4YzBlY2RjMG4IiwgInNlcnZpY2V
FbmRwb2ludCI6ICJodHRwOi8vbG9jYXRob3N00jMwMMDAilCAicmVjaXBpZW50S2V5cyI6IFsi
NzRtTFNSYUhoalBDb2lYU1k1VXo0VVJHb1F3Wg3aFA4Qm90QnJaMvZvRVQIXSwgImxhYmVsI

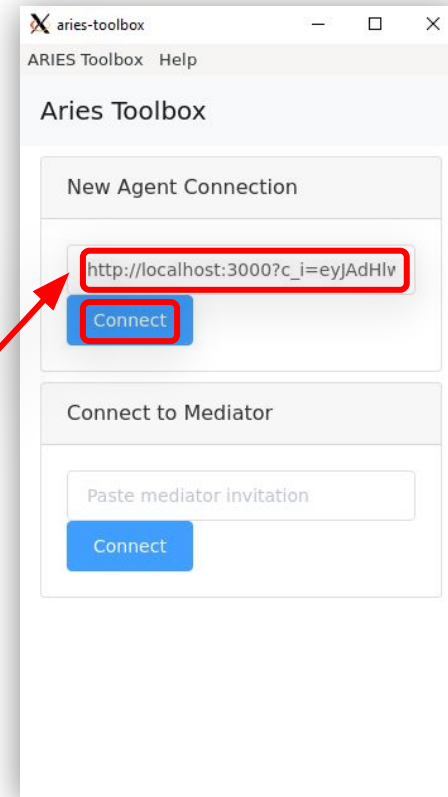
```



# Connecting the Agents to the Toolbox

Copy Alice's Invitation URL into the toolbox New Agent Connection bar... and click **connect**.

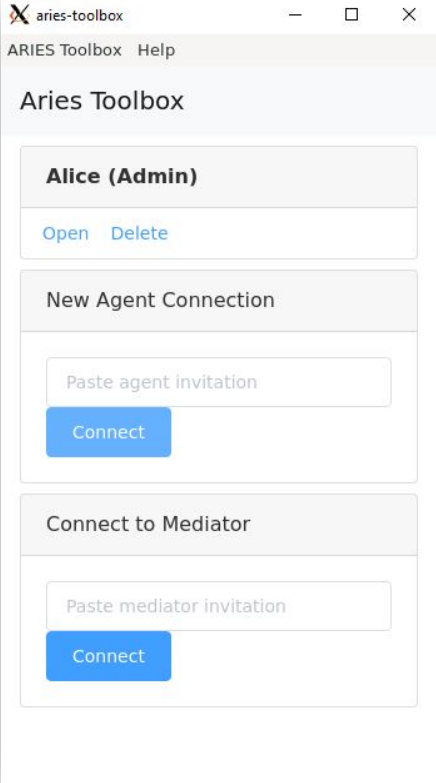
```
agent-alice_1 :: Alice ::  
agent-alice_1 :: Inbound Transports: ::  
agent-alice_1 :: - http+ws://0.0.0.0:3000 ::  
agent-alice_1 :: Outbound Transports: ::  
agent-alice_1 :: - http ::  
agent-alice_1 :: - https ::  
agent-alice_1 :: Administration API: ::  
agent-alice_1 :: - http://0.0.0.0:3001 ::  
agent-alice_1 :: ver: 0.7.3-rc0 ::  
agent-alice_1 :: Listening... ::  
agent-alice_1 Created new invitation ::  
agent-alice_1 connection: {'connection_id': 'ab2a342d-ddb0-4feb-b  
e47-548ee64c2c8a', 'connection_protocol': 'connections/1.0', 'routing_sta  
te': 'none', 'state': 'invitation', 'their_role': 'invitee', 'accept': 'a  
uto', 'rfc23_state': 'invitation-sent', 'invitation_key': '74mLSRaHtjPCoi  
XRY5Uz4URGnQwXh7hP8BoNBrZ1VoET', 'updated_at': '2022-01-17T16:18:44.93514  
3Z', 'created_at': '2022-01-17T16:18:44.935143Z', 'invitation_mode': 'onc  
e'}  
agent-alice_1 | Invitation URL (Connections protocol):  
agent-alice_1 http://localhost:3000?c_i=eyJAdHlwZSI6ICJkaWQ6c2920kY6Q  
2JzT1loTXJqSGlxlwRUVUFTSGc7c3B1Yy9jb25uZW50aW9ucy8xLjAvaW52aXRhdGlvbiIsIC  
JAaWQiOiAiNzcyNTk1MTgtY2FmMC00NjllLWZlZGUtNmM4YzBlY2RjMG4IiwgInNlcnZpY2V  
FbmRwb2ludCI6ICJodHRwOi8vbG9jYXRob3N00jMwMMDAilCAicmVjaXBpZW50S2V5cyI6IFsi  
NzRtTFNSYUhoalBDb2lYU1k1VXo0VVJHb1F3WGg3aFA4Qm90QnJaMvZvRVQlXSgwImxhYmVsI
```



# Connecting the Agents to the Toolbox

Alice is connected to the toolbox.

```
agent-alice_1 | ::::::::::::::::::::::::::::::::::::::::::::  
agent-alice_1 | :: Alice ::  
agent-alice_1 | :: ::  
agent-alice_1 | :: Inbound Transports: ::  
agent-alice_1 | :: - http+ws://0.0.0.0:3000 ::  
agent-alice_1 | :: ::  
agent-alice_1 | :: Outbound Transports: ::  
agent-alice_1 | :: - http ::  
agent-alice_1 | :: - https ::  
agent-alice_1 | :: ::  
agent-alice_1 | :: Administration API: ::  
agent-alice_1 | :: - http://0.0.0.0:3001 ::  
agent-alice_1 | :: ::  
agent-alice_1 | :: ver: 0.7.3-rc0 ::  
agent-alice_1 | ::::::::::::::::::::::::::::::::::::::::::::  
agent-alice_1 | Listening...  
agent-alice_1 | Created new invitation  
agent-alice_1 | connection: {'connection_id': 'ab2a342d-ddb0-4feb-b  
e47-548ee64c2c8a', 'connection_protocol': 'connections/1.0', 'routing_sta  
te': 'none', 'state': 'invitation', 'their_role': 'invitee', 'accept': 'a  
uto', 'rfc23_state': 'invitation-sent', 'invitation_key': '74mLSRaHtjPCoi  
XRY5Uz4URGnQwXh7hP8BoNBrZ1VoET', 'updated_at': '2022-01-17T16:18:44.93514  
3Z', 'created_at': '2022-01-17T16:18:44.935143Z', 'invitation_mode': 'onc  
e'}  
agent-alice_1 | Invitation URL (Connections protocol):  
agent-alice_1 | http://localhost:3000?c_i=eyJAdHlwZSI6ICJkaWQ6c2920kV6Q  
2JzT1loTXJqSGlxwRUVUFTSGc7c3B1Yy9jb25uZW50aW9ucy8xLjAvaw52aXRhdGlvbiIsIC  
JAaWQiOiAiNzcyNTk1MTgtY2FmMC00NjllLWZzZGUtNmM4YzBlY2RjMG4IiwgInNlcnZpY2V  
FmRwb2ludCI6ICJodHRwOi8vbG9jYXRob3N00jMwMMDAilLCAicmVjaXBpZW50S2V5cyI6IFsi  
NzRtTFN5UHoalBDb2lYUlk1VXo0VVJHb1F3WGg3aFA4Qm90QnJaMVZvRVV0iXSgImxhYmVsi
```



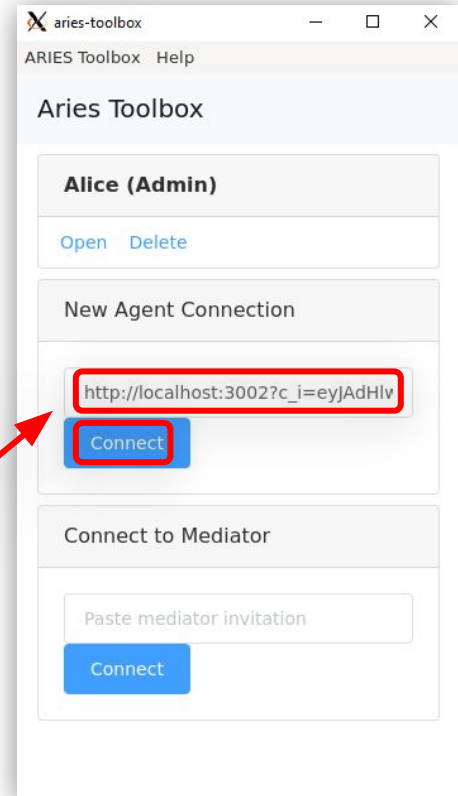




# Connecting the Agents to the Toolbox

Copy Bob's Invitation URL into the toolbox New Agent Connection bar... click to **connect**.

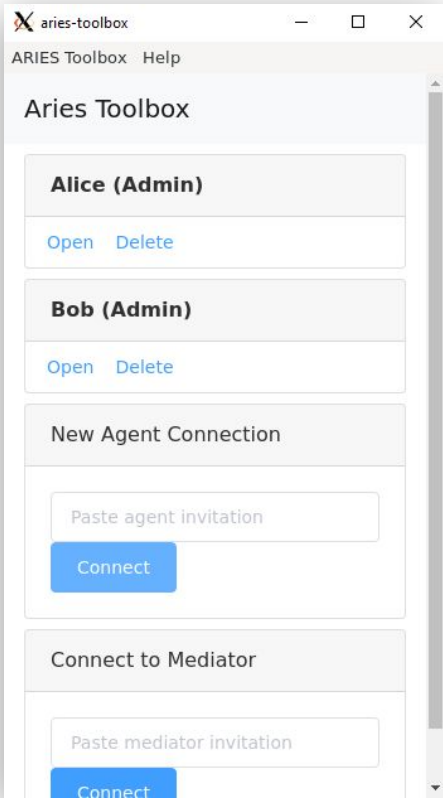
```
agent-bob_1 ::::::::::::::::::::::::::::::::::::::::::::::::::::  
agent-bob_1 :: Bob ::  
agent-bob_1 ::  
agent-bob_1 :: Inbound Transports: ::  
agent-bob_1 :: - http+ws://0.0.0.0:3002 ::  
agent-bob_1 ::  
agent-bob_1 :: Outbound Transports: ::  
agent-bob_1 :: - http ::  
agent-bob_1 :: - https ::  
agent-bob_1 ::  
agent-bob_1 :: Administration API: ::  
agent-bob_1 :: - http://0.0.0.0:3003 ::  
agent-bob_1 ::  
agent-bob_1 :: ver: 0.7.3-rc0 ::  
agent-bob_1 ::::::::::::::::::::::::::::::::::::::::::::::::::::  
agent-bob_1 Listening...  
agent-bob_1 Created new invitation  
agent-bob_1 connection: {'updated_at': '2022-01-17T16:19:05.176  
427Z', 'created_at': '2022-01-17T16:19:05.176427Z', 'their_role': 'invite  
e', 'accept': 'auto', 'state': 'invitation', 'invitation_key': '9P1wfHbpz  
FLWSVHu7vg9M2v7z92FQ5dMs6Qpmj4M7cSy', 'routing_state': 'none', 'connectio  
n_id': '1d92c573-20ec-4a7a-ad4a-a6e92051037d', 'invitation_mode': 'once',  
'rfc23_state': 'invitation-sent', 'connection_protocol': 'connections/1.  
0'}  
agent-bob_1 Invitation URL (Connections protocol):  
agent-bob_1 http://localhost:3002?c_i=eyJAdHlwZSI6ICJkaWQ6c2920k16Q  
ZJzI1I01XJQSG1xwKRUVUFTSGc7c3B1Yy9jb25uZW50aw9ucy8xLjAvalW52aXRhdG1vbiIsIC  
JAawQioiAiOGJmOTFizMteYTNjNC00NDQzLWExZDUtNDlmNWlhYTYzZTZQzIiwgImxhYmVsIjo  
gIkjvYiAoQRtaW4pIiwgInJlY2lwaWVudEtleXMiOiBBIj1QaXdmSGJwekZMV1NWSHU3dkc5  
TTJ2N3o5MkZRNWRnczZRCG1qNE03Y1N5I10sICJzZXJ2aWN1RW5kG9pbmQioiAiaHR0cDovL  
2xvY2FsaG9zdDozMDAyIn0=
```



# Connecting the Agents to the Toolbox

Bob is connected to the toolbox.

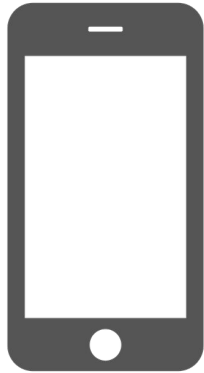
```
agent-bob_1 ::::::::::::::::::::::::::::::::::::::::::::::::::::
agent-bob_1 :: Bob ::
agent-bob_1 ::
agent-bob_1 ::
agent-bob_1 :: Inbound Transports: ::
agent-bob_1 :: - http+ws://0.0.0.0:3002 ::
agent-bob_1 ::
agent-bob_1 :: Outbound Transports: ::
agent-bob_1 :: - http ::
agent-bob_1 :: - https ::
agent-bob_1 ::
agent-bob_1 :: Administration API: ::
agent-bob_1 :: - http://0.0.0.0:3003 ::
agent-bob_1 ::
agent-bob_1 :: ver: 0.7.3-rc0 ::
agent-bob_1 ::::::::::::::::::::::::::::::::::::::::::::::::::::
agent-bob_1 Listening...
agent-bob_1 Created new invitation
agent-bob_1 connection: {'updated_at': '2022-01-17T16:19:05.176
427Z', 'created_at': '2022-01-17T16:19:05.176427Z', 'their_role': 'invite
e', 'accept': 'auto', 'state': 'invitation', 'invitation_key': '9PiwfHbpz
FLWSVHu7vg9M2v7z92FQ5dMs6Qpmj4M7cSy', 'routing_state': 'none', 'connectio
n_id': '1d92c573-20ec-4a7a-ad4a-a6e92051037d', 'invitation_mode': 'once',
'rfc23_state': 'invitation-sent', 'connection_protocol': 'connections/1.
0'}
agent-bob_1 Invitation URL (Connections protocol):
agent-bob_1 http://localhost:3002?c_i=eyJAdHlwZSI6ICJkaWQ6c2920k16Q
2JzT1loTXJqSG1xwkrUVUFTSGc7c3B1Yy9jb25uZWNoaw9ucy8xLjAvaW52aXRhdGlvbiIsIC
JAAwQioiAiOGJmOTFizMteYTNjNC00NDQzLWExZDUtNDlmNWNhYTYzZTZQzIiwgImxhYmVsIjo
gIkjvYiAoQWRtaW4pIiwgInJlY2lwaWVudEtleXMiOiBBIj1QaXdmSGJwekZMV1NWSHU3dkc5
TTJ2N3o5MkZRNWRnczZRCG1qNE03Y1N5I10sICJzZXJ2aWN1RW5kcG9pbmQioiAiaHR0cDovL
2xvY2FsaG9zdDozMDAyIn0=
```





# 5. Creating a Connection

Alice

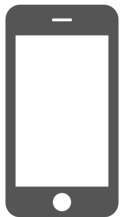


Bob





Mediator  
Agent



Edge  
Agent

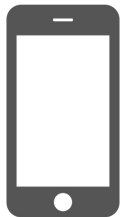
Alice's Domain



Mediator  
Agent

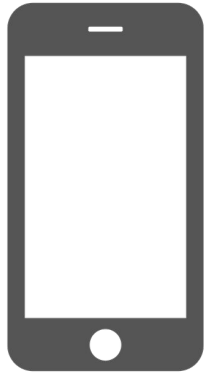


Edge  
Agent



Bob's Domain

Alice



Bob



# Invitation

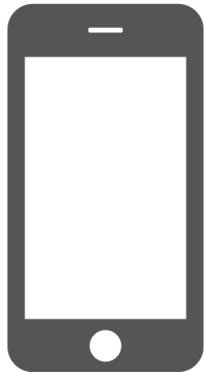
Alice



Bob

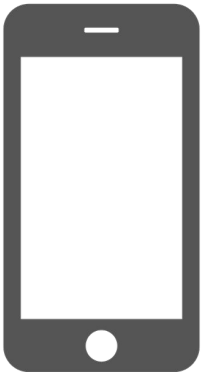


# Invitation



Alice

- Initial Connection Key
- Endpoint Info
- Routing Keys
- Label



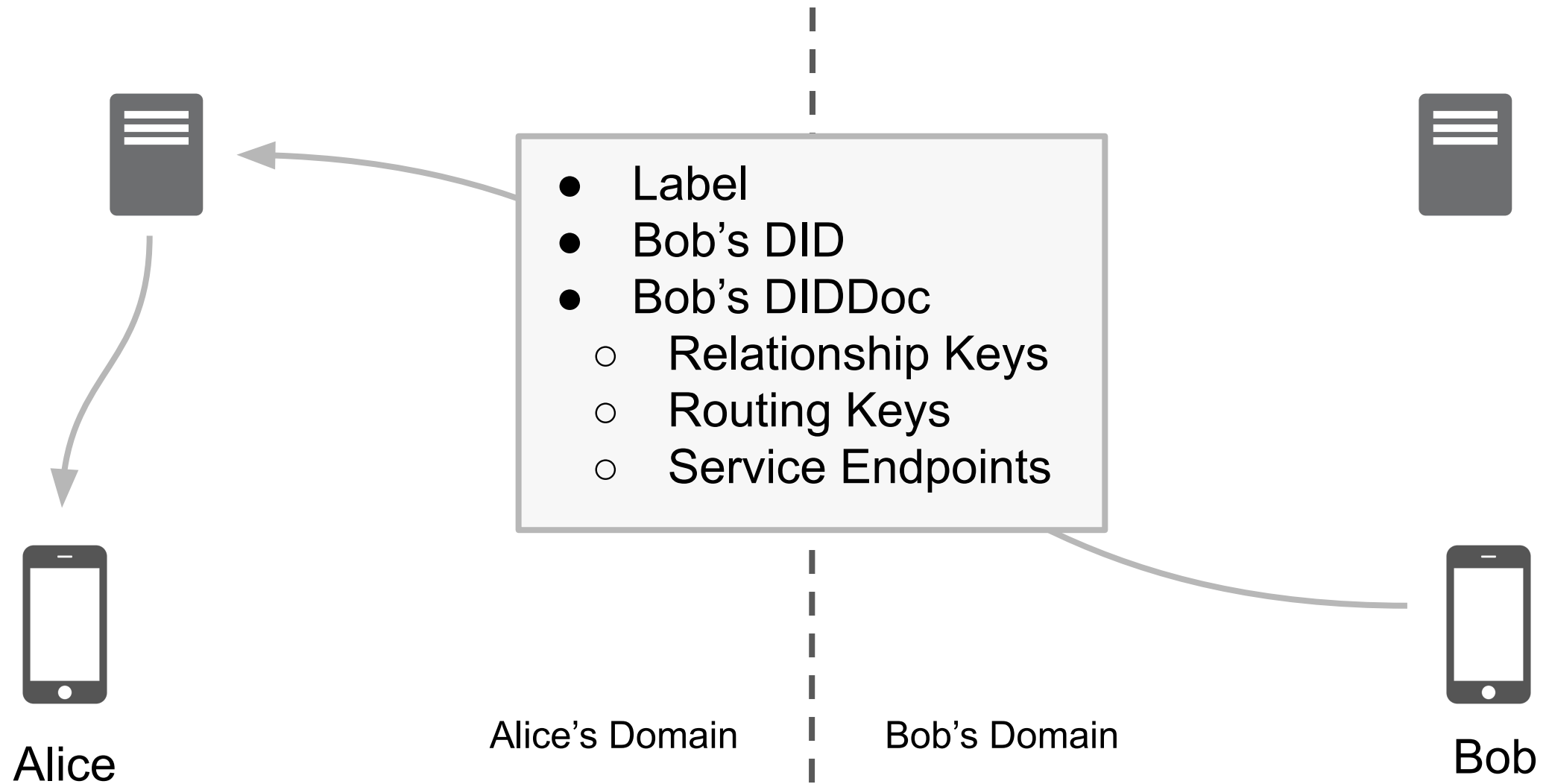
Bob

# Invitation

```
{  
  "@type": "https://didcomm.org/connections/1.0/invitation",  
  "@id": "12345678900987654321",  
  "label": "Alice",  
  "recipientKeys": [ "8HH5gYEeNc3z7PYXmd54d4x6qAfCNrqQqEB3nS7Zfu7K" ],  
  "serviceEndpoint": "https://example.com/endpoint",  
  "routingKeys": [ "8HH5gYEeNc3z7PYXmd54d4x6qAfCNrqQqEB3nS7Zfu7K" ]  
}
```

See [Connection Protocol, Section 0](#).

# Request



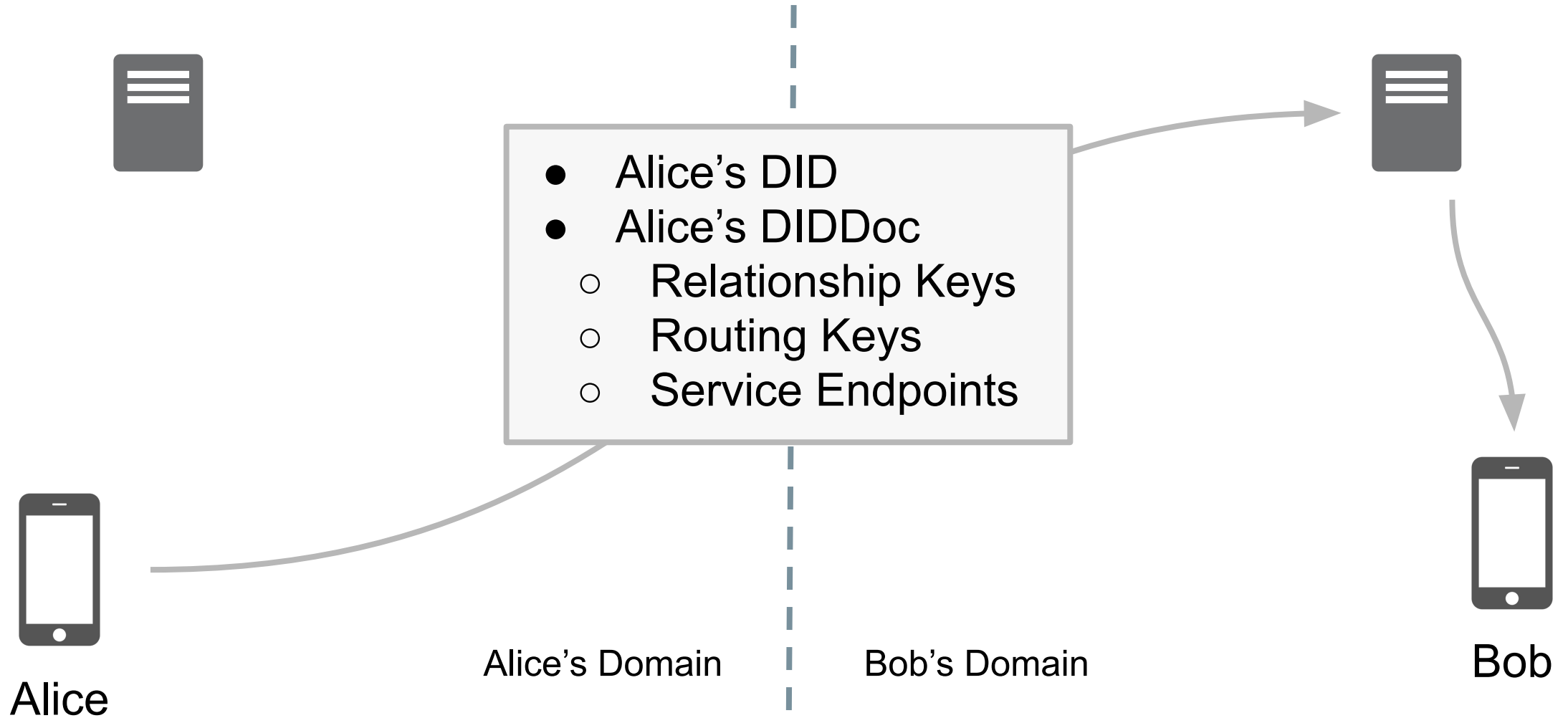


# Request

```
{
  "@id": "5678876542345",
  "@type": "https://didcomm.org/connections/1.0/request",
  "label": "Bob",
  "connection": {
    "did": "B.did@B:A",
    "did_doc": {
      "@context": "https://w3id.org/did/v1"
      // DID Doc contents here.
    }
  }
}
```

See [Connection Protocol, Section 1.](#)

# Response

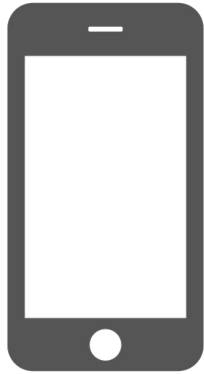


# Response

```
{
  "@type": "https://didcomm.org/connections/1.0/response",
  "@id": "12345678900987654321",
  "~thread": {
    "thid": "<@id of request message>"
  },
  "connection": {
    "did": "A.did@B:A",
    "did_doc": {
      "@context": "https://w3id.org/did/v1"
      // DID Doc contents here.
    }
  }
}
```

See [Connection Protocol, Section 2](#).

Alice



Connection  
Established



Bob



# DIDComm

## Forming Connections

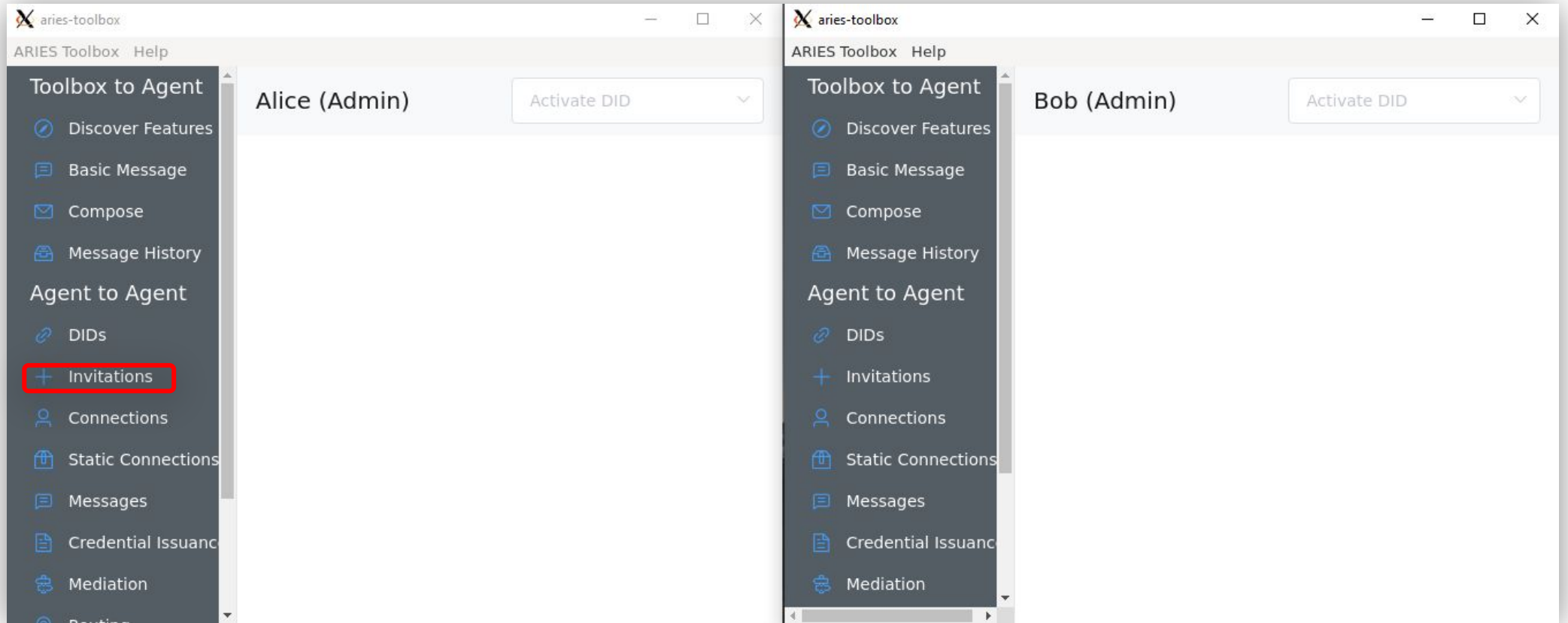
- Peer-to-peer connections
- Not necessarily direct
- Connection is not a transport layer connection (socket, open http request, TCP, UDP, etc.)
- A connection is considered completed when both parties have securely exchanged DID Documents (DIDs, keys, and service endpoints) for future communication

The background of the slide is a blurred image of a person's hands typing on a laptop keyboard. In the top-left corner, there is a white network diagram consisting of several circular nodes connected by thin lines, forming a web-like structure. The overall color palette is a soft, muted blue.

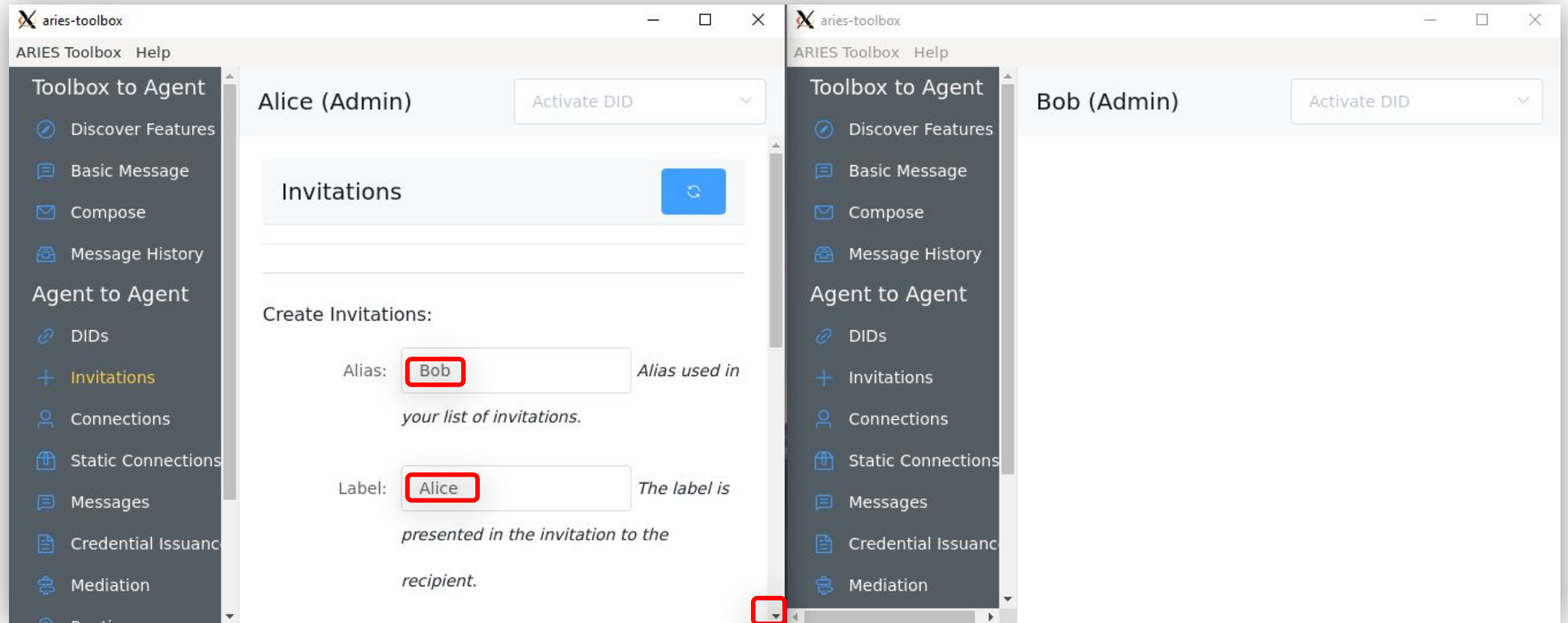
# Demo

Creating a connection

# Demo: Creating a Connection

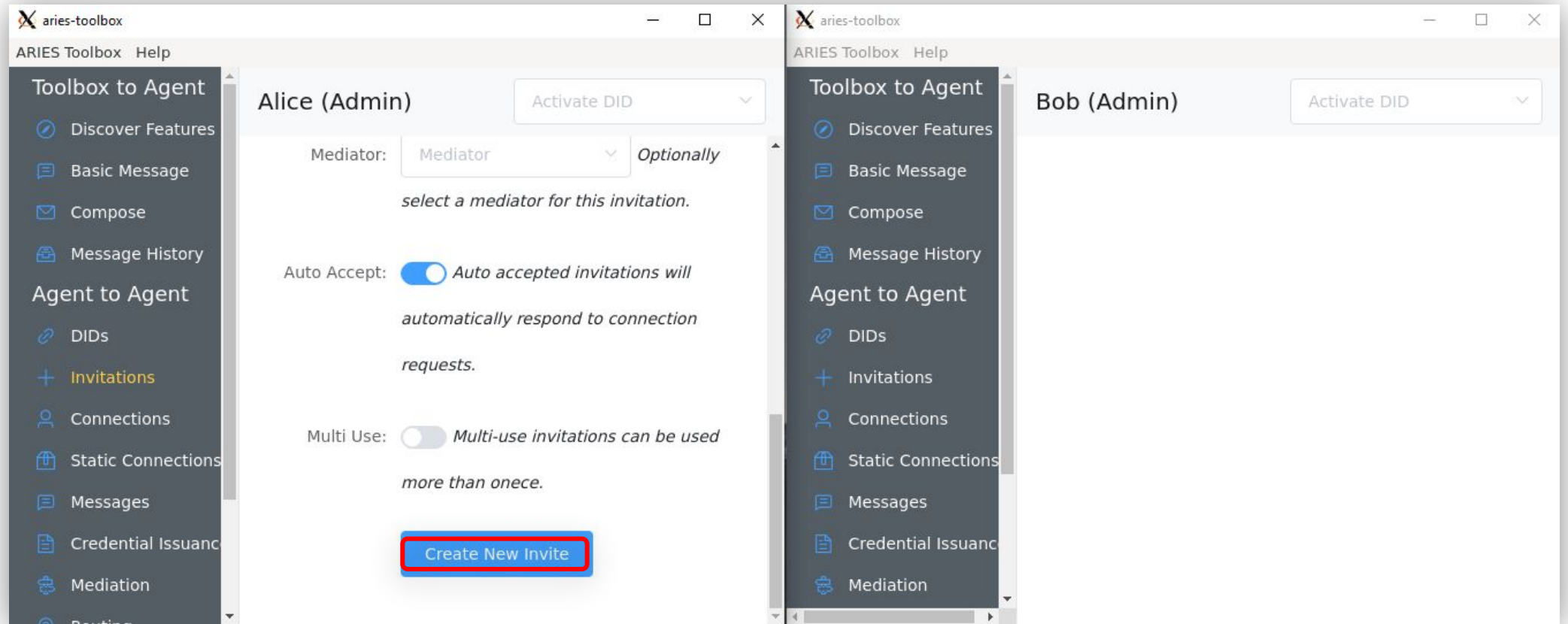


# Demo: Creating a Connection

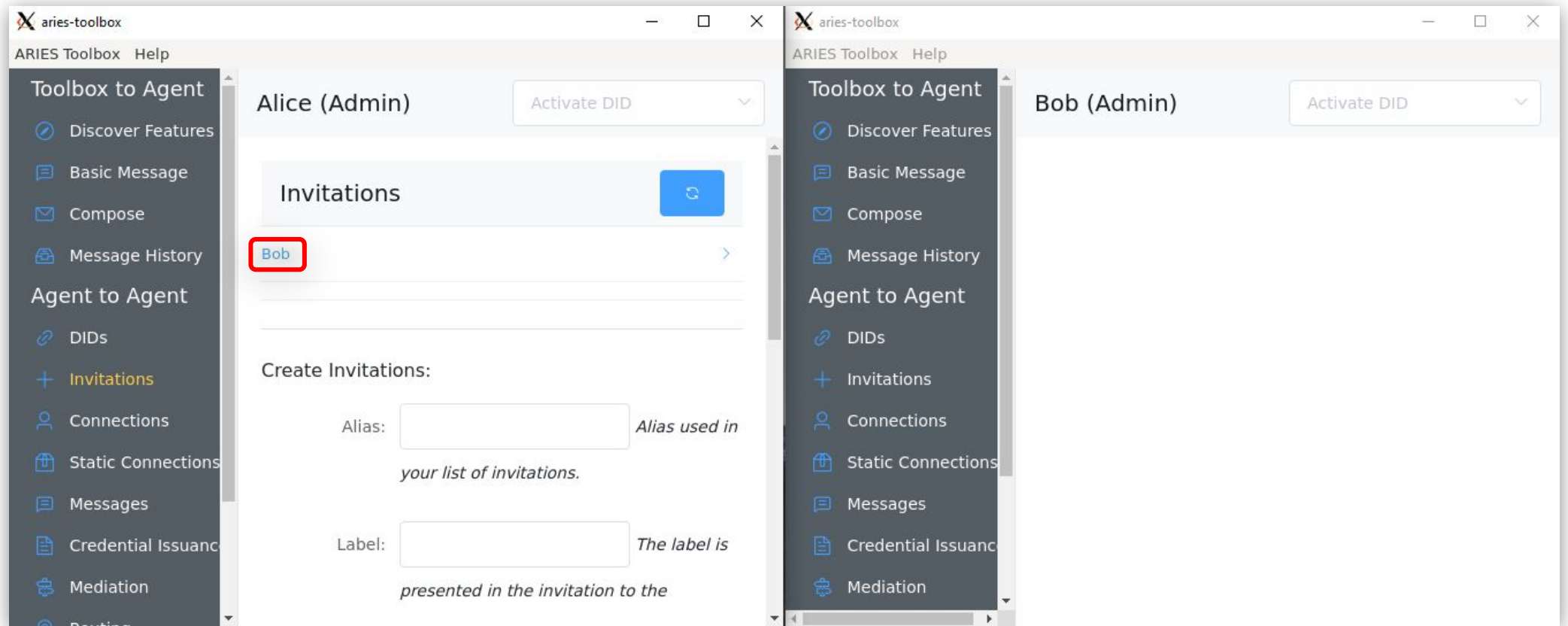




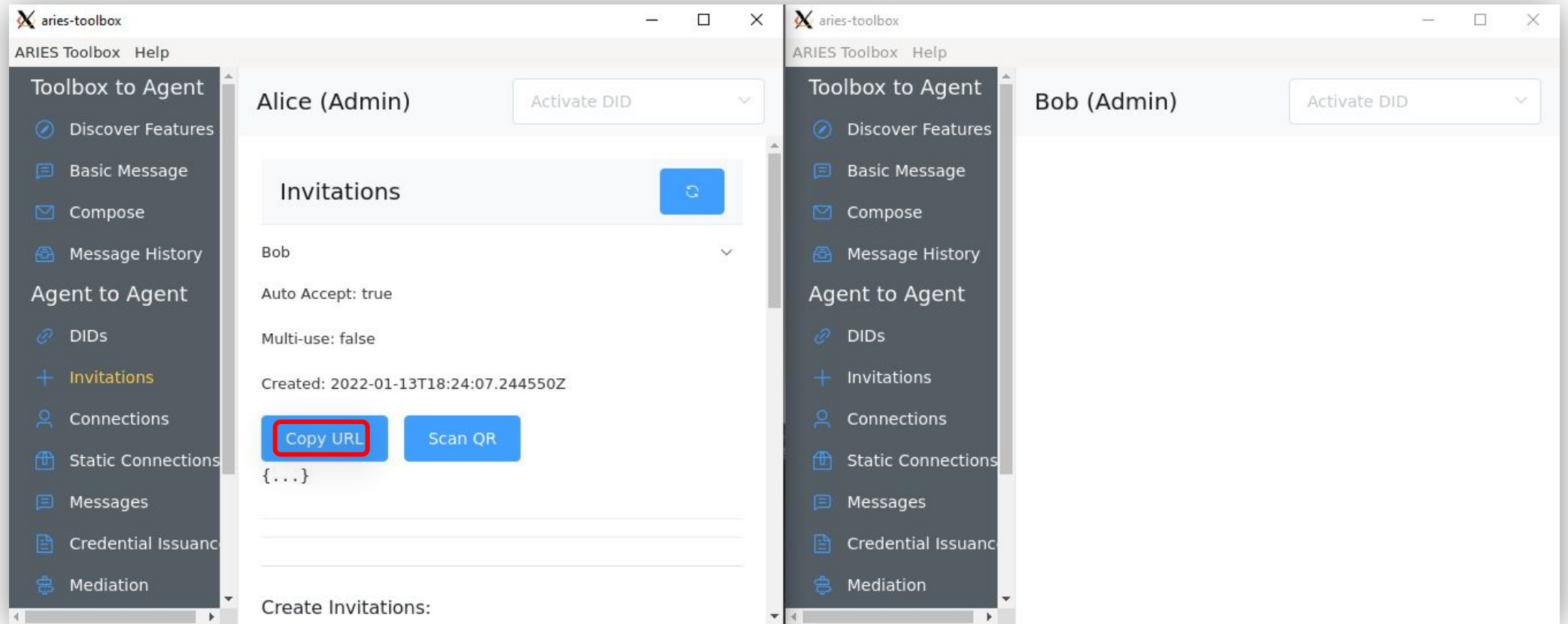
# Demo: Creating a Connection



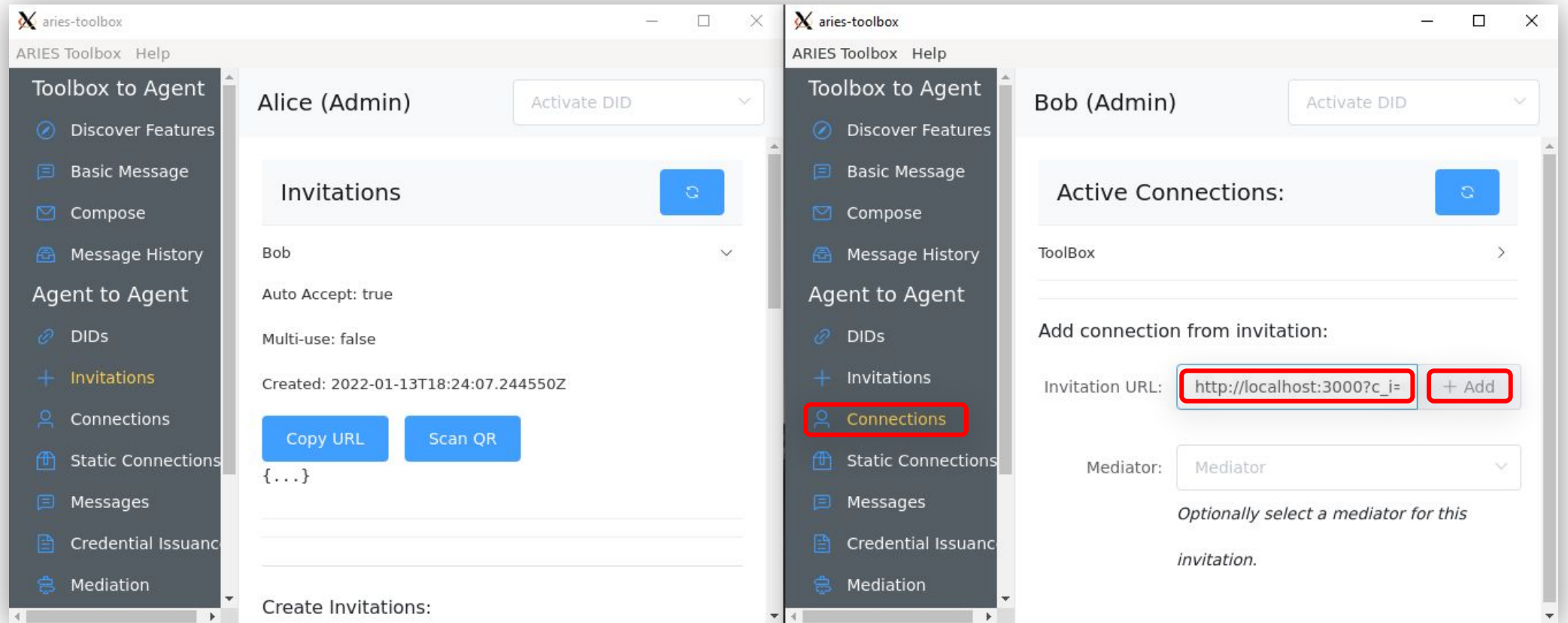
# Demo: Creating a Connection



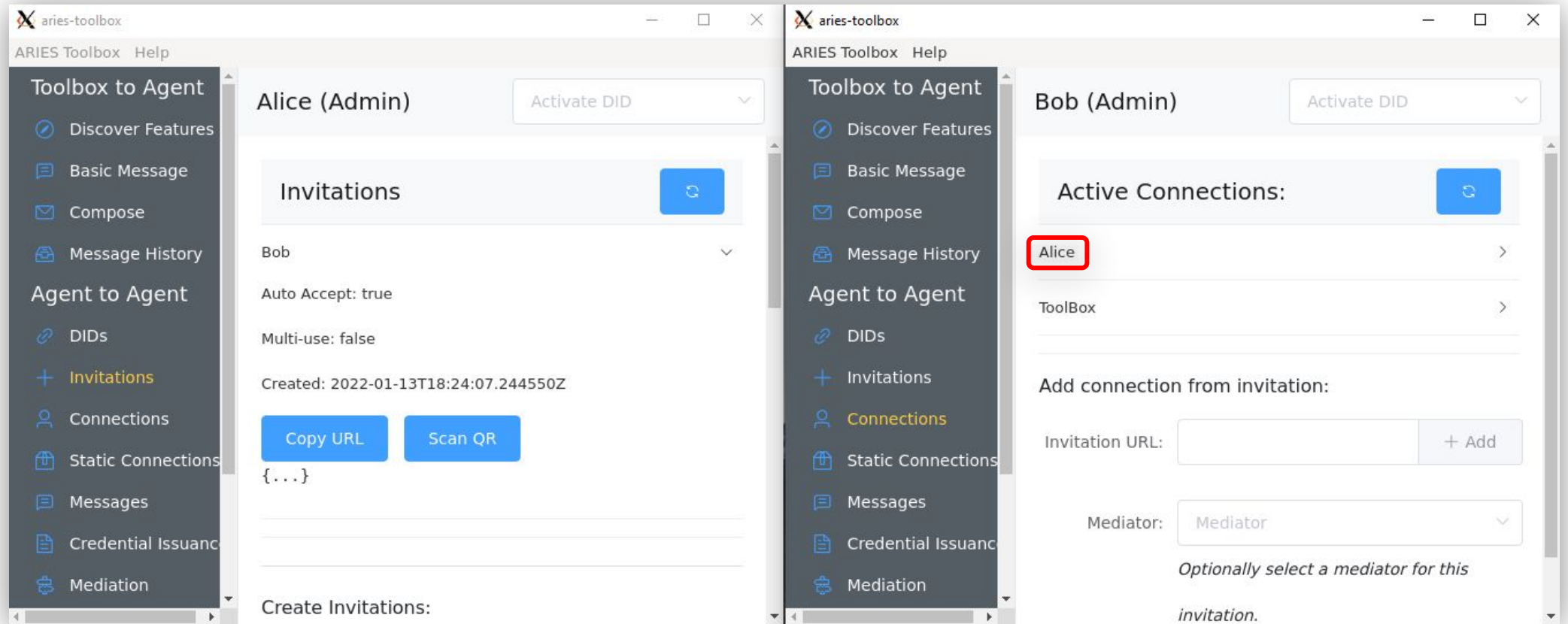
# Demo: Creating a Connection



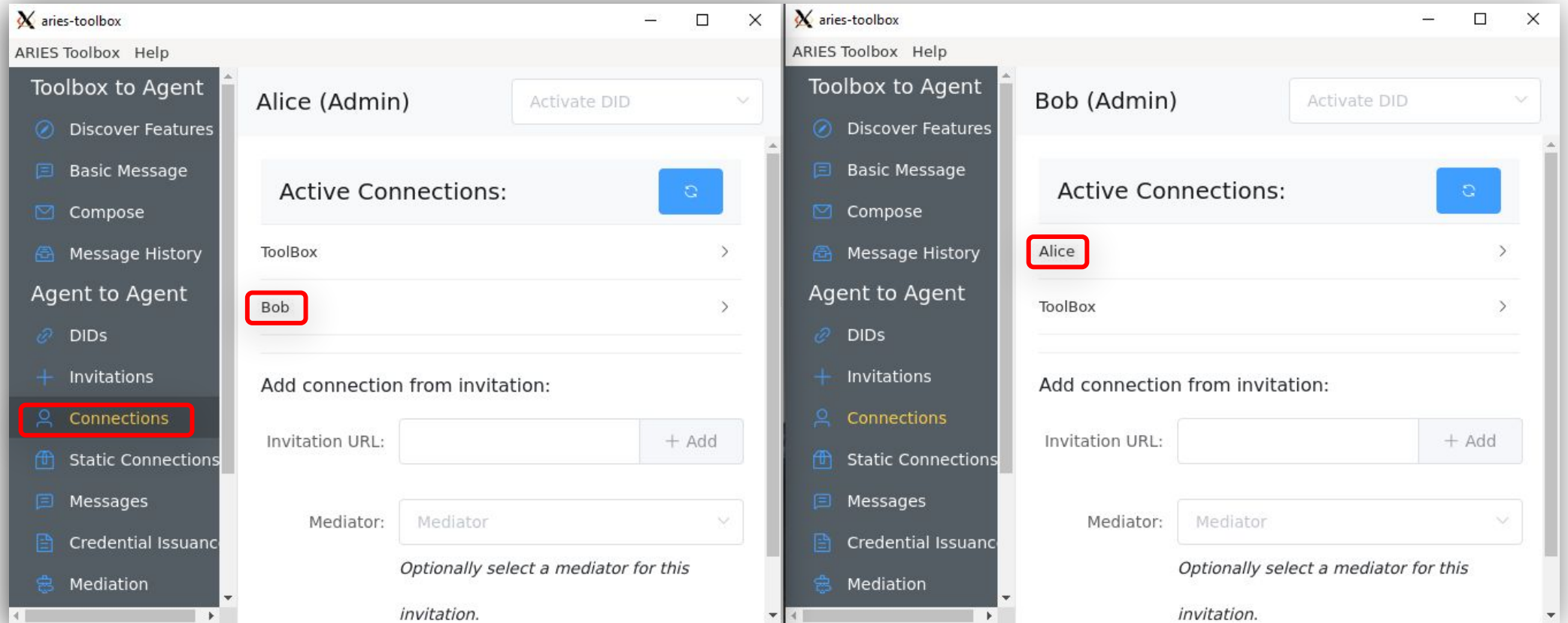
# Demo: Creating a Connection



# Demo: Creating a Connection



# Demo: Creating a Connection





# 6. Issuing a Credential

# Anonymous Credentials

- Zero-knowledge proofs
- Selective disclosure
- Revocation



Indicio Meetup:

How to use ACA-Py to issue and verify W3C JSON-LD credentials

Tuesday, November 29 | 12pm EST / 5pm GMT







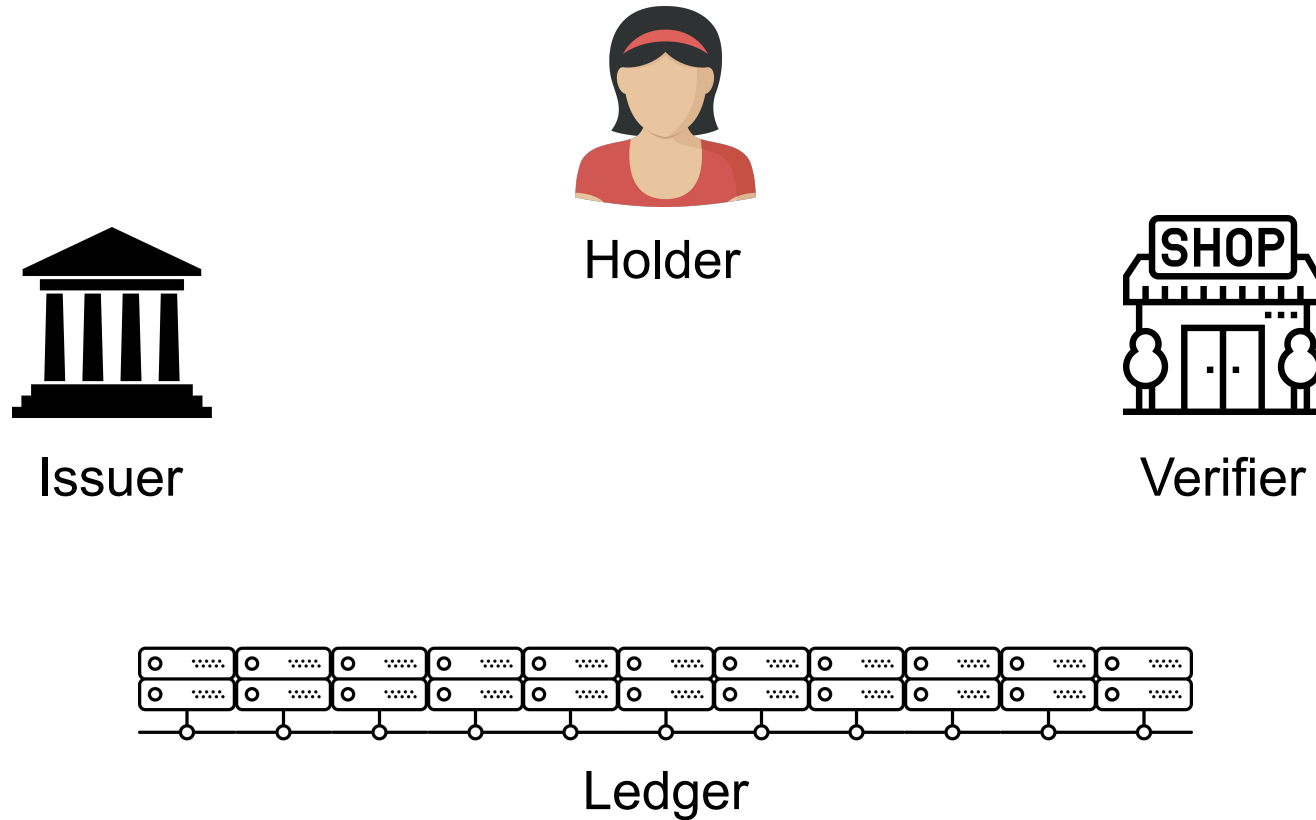
# The Issuer Role

## Issuer responsibilities

- Decide which credentials to issue and to whom
- Interact with the holder
- Write to the ledger
- Perform cryptographic computations
- Issue the credential

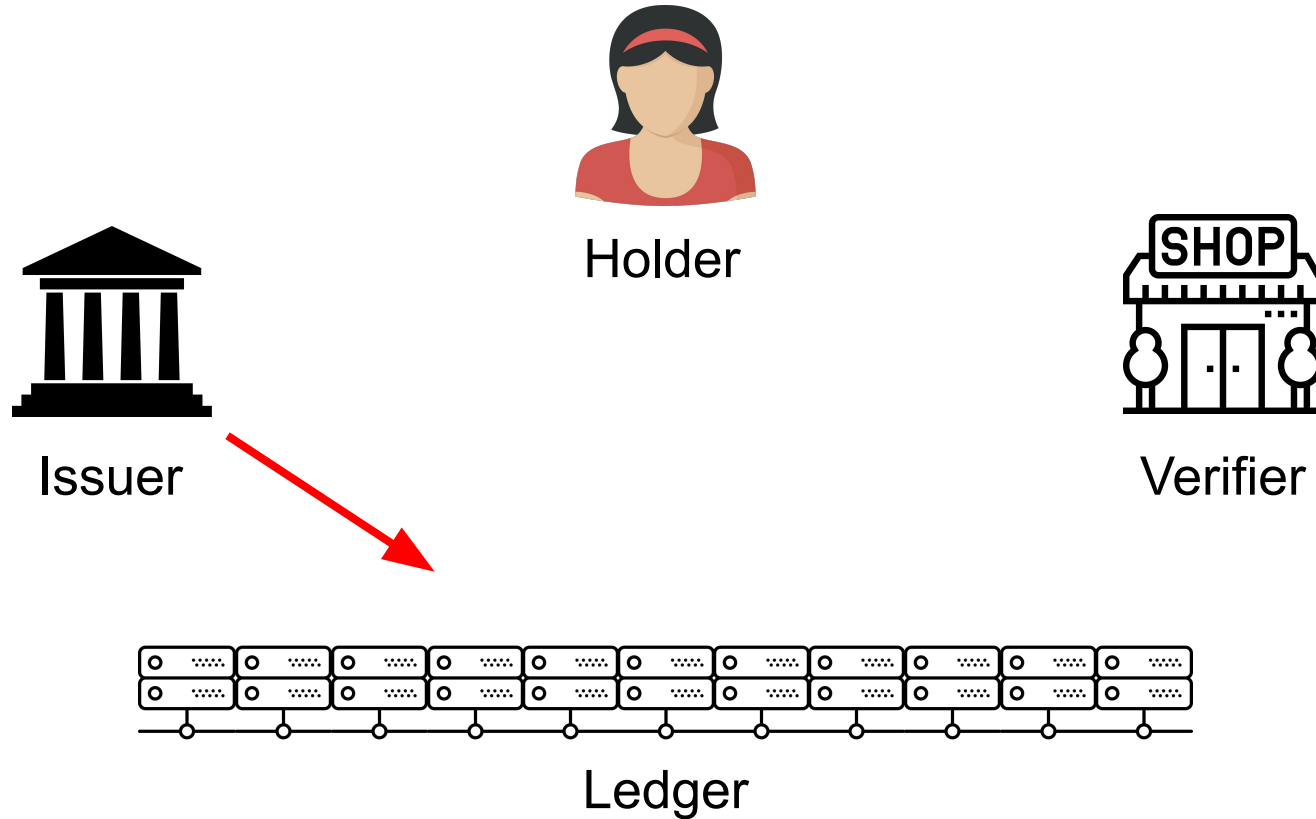
The issuer does **not** write personally identifiable information to the ledger

# The Verifiable Credential Implementation



# The Verifiable Credential Implementation

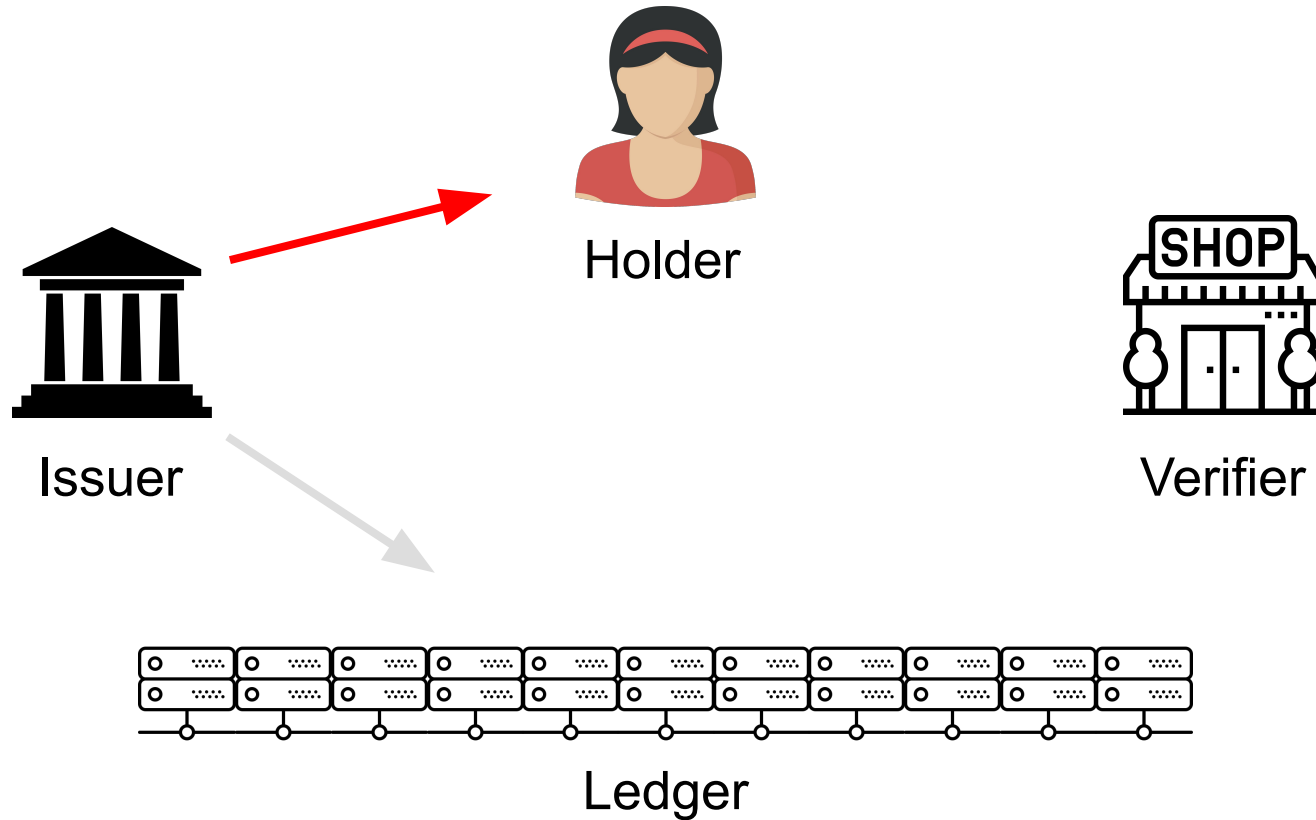
1. Write DID, schema, and credential definition to the ledger



# The Verifiable Credential Implementation

2. Sign and issue a verifiable credential

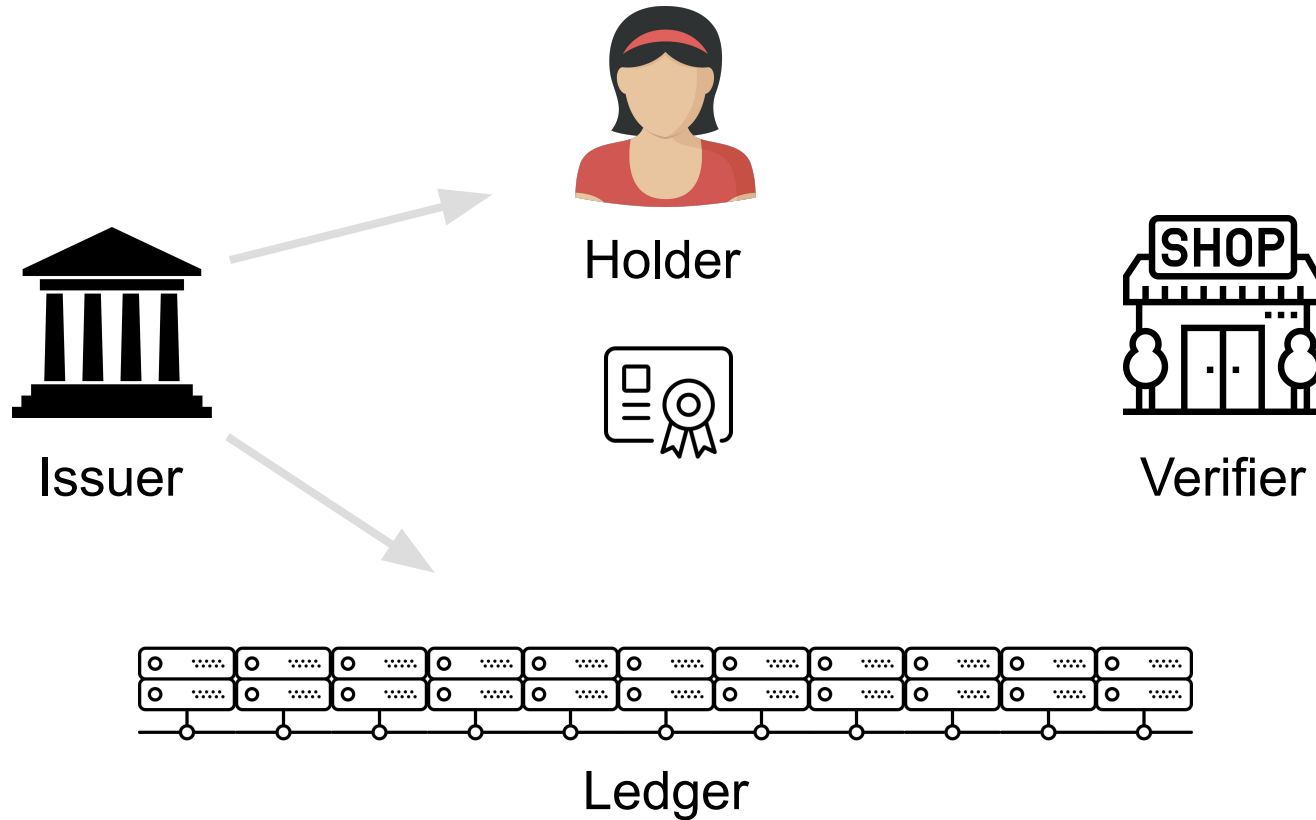
1. Write DID, schema, and credential definition to the ledger



# The Verifiable Credential Implementation

2. Sign and issue a verifiable credential

1. Write DID, schema, and credential definition to the ledger



# The Issuer Role

Write a DID, schema, and credential definition to the Ledger

- a. Anchor a public DID to a ledger
- b. Write a new schema to the ledger or retrieve a pre-existing schema from the ledger
- c. Create a credential definition and write it to the ledger
- d. Issue a verifiable credential

## a. Anchor a DID

### **Author**

- Has privileges to create a transaction and issue credentials, but needs an endorser to endorse their transaction
- DID must be on the ledger

### **Endorser**

- Has privileges to write to the ledger
- DID must be on the ledger with endorser status

### **Transaction Author Agreement**

- The text of agreement between network users and government
- Must be agreed to before anchoring a DID

The background of the slide is a blurred image of a person's hands typing on a laptop keyboard. Overlaid on the top left is a white network diagram consisting of several circular nodes connected by thin lines, forming a web-like structure. The overall color palette is a muted, light blue-grey.

# Demo

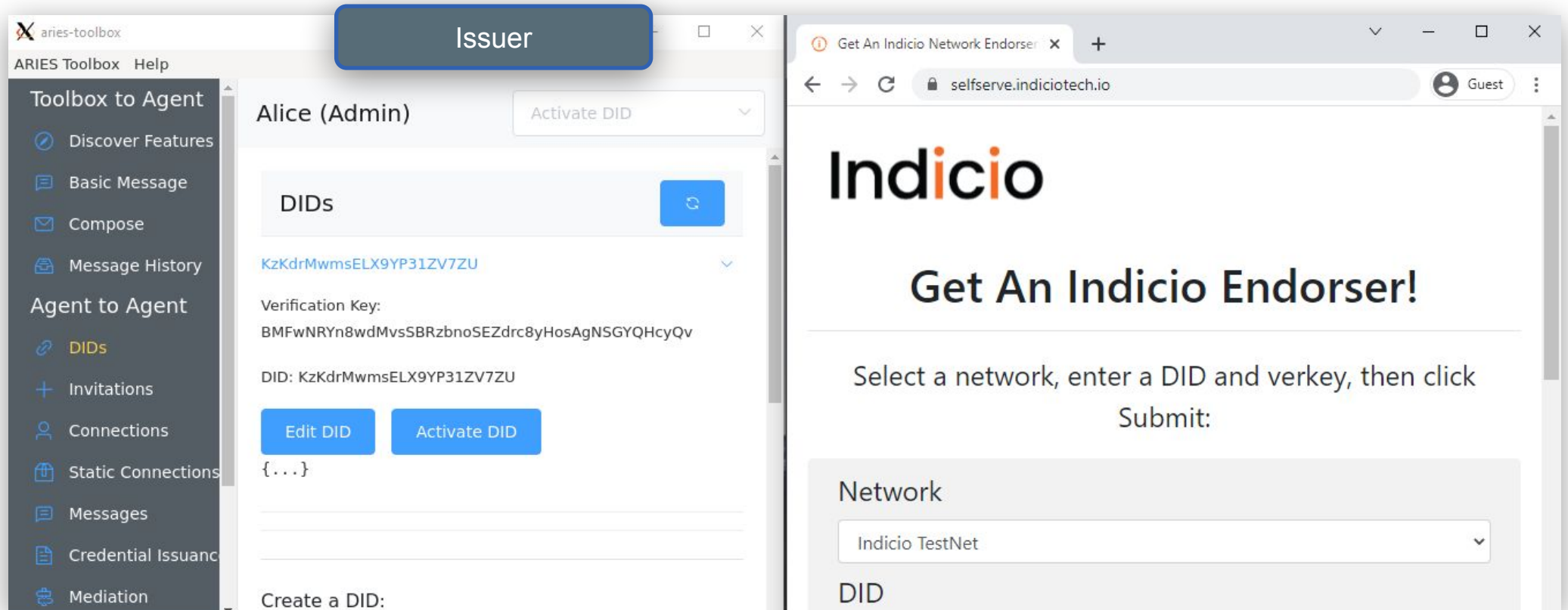
Anchoring a DID



# Demo: Anchoring a DID

The image displays two side-by-side screenshots of the ARIES Toolbox web application. The left window, titled "Issuer", shows the "DIDs" section for "Alice (Admin)". The sidebar on the left has "DIDs" highlighted in red. The main content area shows a "DIDs" section with a refresh button, a dropdown menu with the value "KzKdrMwmsELX9YP31ZV7ZU", a "Verification Key" field containing "BMFwNRYn8wdMvsSBRzbn0SEZdrc8yHosAgNSGYQHcyQv", and a "DID" field containing "KzKdrMwmsELX9YP31ZV7ZU". Below these are "Edit DID" and "Activate DID" buttons. The right window, titled "Holder", shows the "Active Connections" section for "Bob (Admin)". The sidebar on the left has "Connections" highlighted in yellow. The main content area shows an "Active Connections" section with a refresh button, a list of connections including "Alice" and "ToolBox", and an "Add connection from invitation" section with fields for "Invitation URL" and "Mediator".

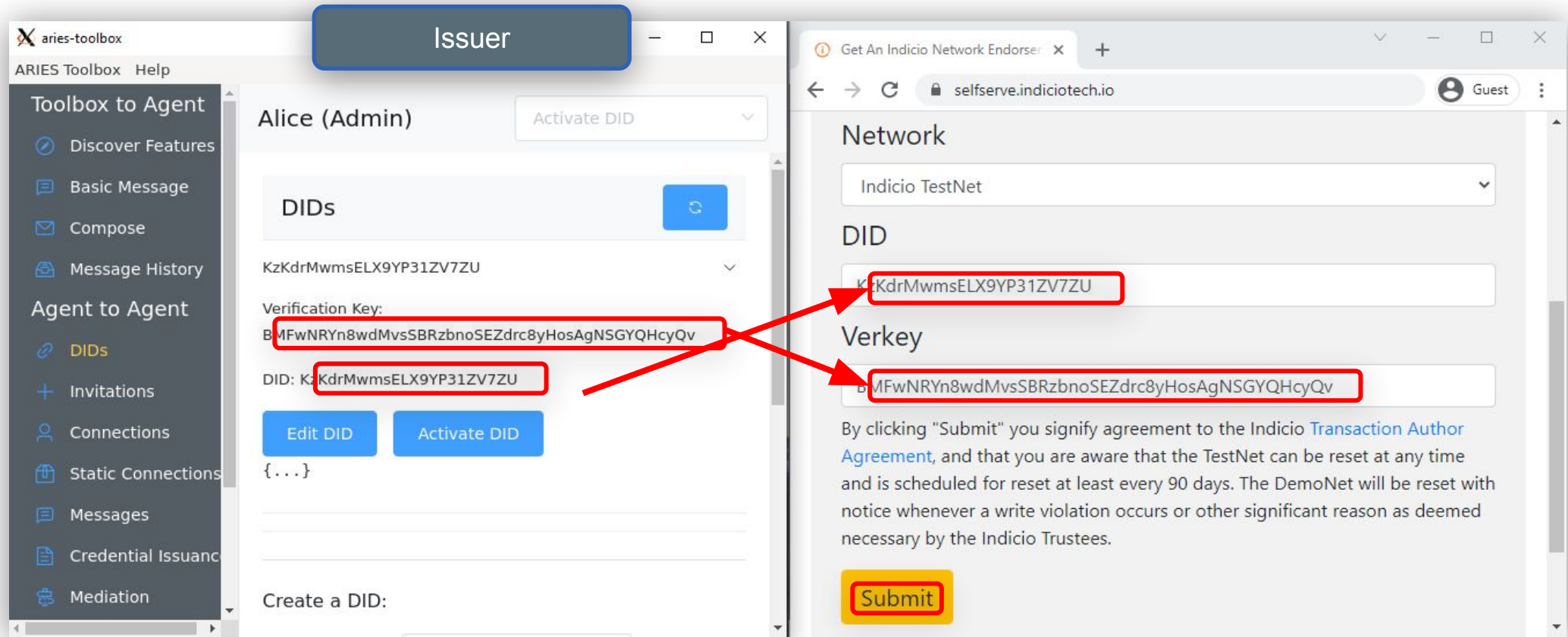
# Demo: Anchoring a DID



The image shows two side-by-side screenshots. The left screenshot is from the 'aries-toolbox' application, displaying the 'Issuer' interface for 'Alice (Admin)'. It shows a list of DIDs with the selected one being 'KzKdrMwmsELX9YP31ZV7ZU'. Below the DID, the verification key is shown as 'BMFwNRYn8wdMvsSBRzbnoseZdrc8yHosAgNSGYQHcyQv'. There are 'Edit DID' and 'Activate DID' buttons. The right screenshot is from the 'selfserve.indiciotech.io' website, titled 'Get An Indicio Endorser!'. It prompts the user to 'Select a network, enter a DID and verkey, then click Submit:'. A dropdown menu for 'Network' is set to 'Indicio TestNet', and there is a 'DID' input field below it.

Copy DID and Verkey into <https://selfserve.indiciotech.io/>

# Demo: Anchoring a DID



Copy DID and Verkey using <https://selfserve.indiciotech.io/>

# Demo: Anchoring a DID

The image shows two side-by-side screenshots. The left screenshot is of the 'aries-toolbox' application, specifically the 'Issuer' window for 'Alice (Admin)'. A red box highlights the 'Activate DID' button. The interface shows a list of DIDs with the first one selected: 'KzKdrMwmsELX9YP31ZV7ZU'. Below the list, there are 'Edit DID' and 'Activate DID' buttons. The 'Activate DID' button is highlighted with a blue glow. The right screenshot is a browser window at 'selfserve.indiciotech.io' showing a 'Verkey' endpoint. The input field contains the DID 'BMFwNRyN8wdMvsSBRzbn0SEZdrc8yHosAgNSGYQHcyQv'. Below the input field, there is a paragraph of text explaining the agreement. At the bottom, there is a 'Submit' button. A black box displays the JSON response from the endpoint:

```

{"statusCode": 200, "headers": {"Access-Control-Allow-Origin": "*"}, "body": {"statusCode": 200, "KzKdrMwmsELX9YP31ZV7ZU": {"status": "Success", "statusCode": 200, "reason": "Successfully wrote NYM identified by KzKdrMwmsELX9YP31ZV7ZU to the ledger with role ENDORSER"}}}
    
```

# Demo: Anchoring a DID

The image displays two side-by-side windows of the Aries Toolbox interface. The left window is titled 'Issuer' and shows the configuration for 'Alice (Admin)'. It features a sidebar with navigation options like 'Discover Features', 'Basic Message', 'Compose', 'Message History', 'DIDs', 'Invitations', 'Connections', 'Static Connections', 'Messages', 'Credential Issuance', and 'Mediation'. The main content area shows the DID 'KzKdrMwmsELX9YP31ZV7ZU' selected in a dropdown menu, with a red box highlighting the text. Below the dropdown, there are buttons for 'Edit DID' and 'Activate DID'. The right window is titled 'Holder' and shows the configuration for 'Bob (Admin)'. It features a sidebar with similar navigation options. The main content area shows the 'Active Connections' section with a refresh button and a list of connections including 'Alice' and 'ToolBox'. Below this, there is a section for 'Add connection from invitation' with fields for 'Invitation URL' and 'Mediator'.



# Demo: Anchoring a DID

The image displays two side-by-side windows of the Aries Toolbox application. The left window is titled 'Issuer' and shows the configuration for 'Alice (Admin)'. It features a dropdown menu with the DID 'KzKdrMwmsELX9YP31ZV'. Below this, there is a 'DIDs' section with a refresh button and a list of DIDs: '2xPZzRwnho7aMFuw1PtMgZ' and 'KzKdrMwmsELX9YP31ZV7ZU'. A 'Verification Key' is displayed as 'BMFwNRYn8wdMvsSBRzbnSEZdrc8yHosAgNSGYQHcyQv'. The 'DID' is 'KzKdrMwmsELX9YP31ZV7ZU'. At the bottom, there are 'Edit DID' and 'Activate DID' buttons. The right window is titled 'Holder' and shows the configuration for 'Bob (Admin)'. It features a dropdown menu with 'Activate DID'. Below this, there is an 'Active Connections' section with a refresh button and a list of connections: 'Alice' and 'ToolBox'. At the bottom, there is an 'Add connection from invitation' section with an 'Invitation URL' field, a '+ Add' button, and a 'Mediator' dropdown menu set to 'Mediator'. A note below the mediator dropdown reads: 'Optionally select a mediator for this invitation.'

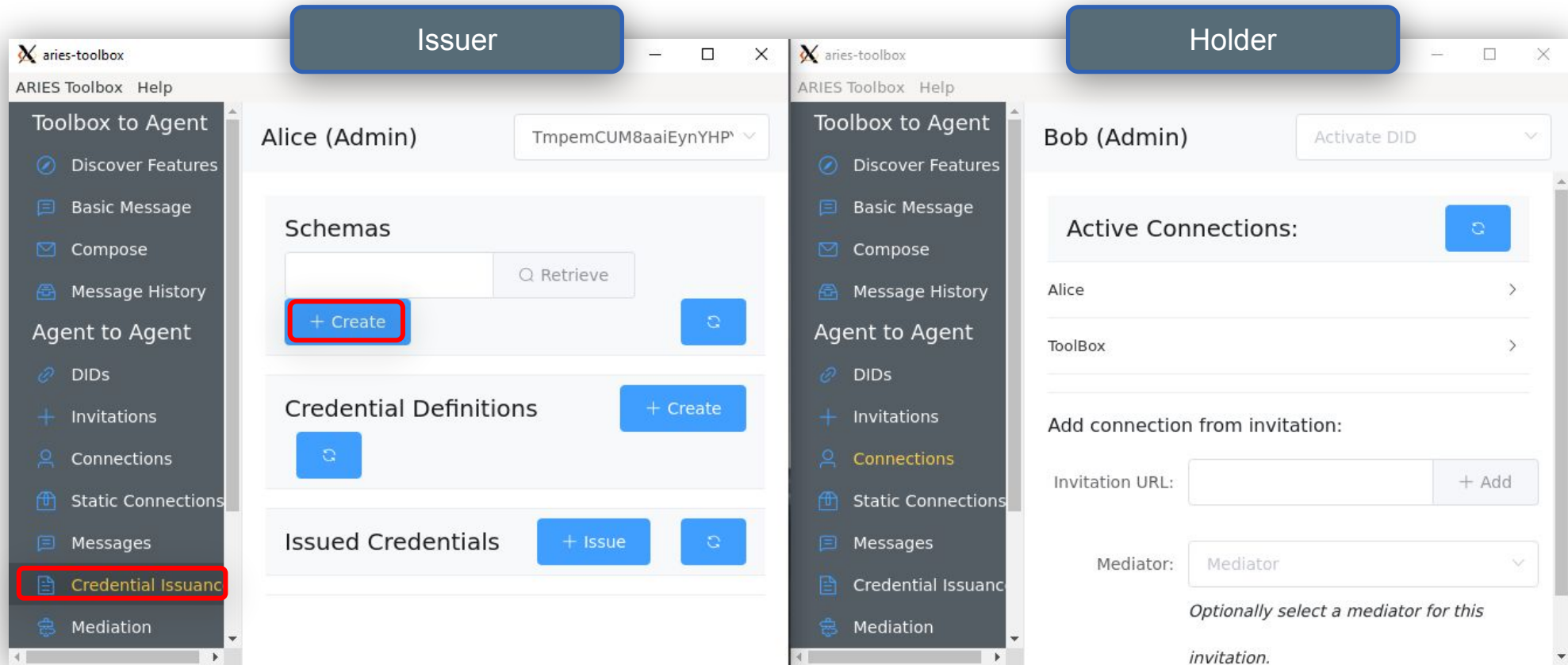
## b. Write or retrieve a schema

Schema: the data organization structure that defines the attributes in a credential

```
{
  "author": "self",
  "state": "written",
  "schema_name": "Test Schema",
  "created_at": "2022-01-14T19:11:13.094763Z",
  "schema_version": "1.1",
  "schema_id": "UZukR8GpvEz3Fi1HbRv5qY:2:Test
               Schema:1.1",
  "updated_at": "2022-01-14T19:11:13.094763Z",
  "attributes": [
    "first_name",
    "last_name",
    "age"
  ]
}
```

See schema on ledger using [IndyScan.io](https://IndyScan.io)

# Demo: Writing a schema to the ledger

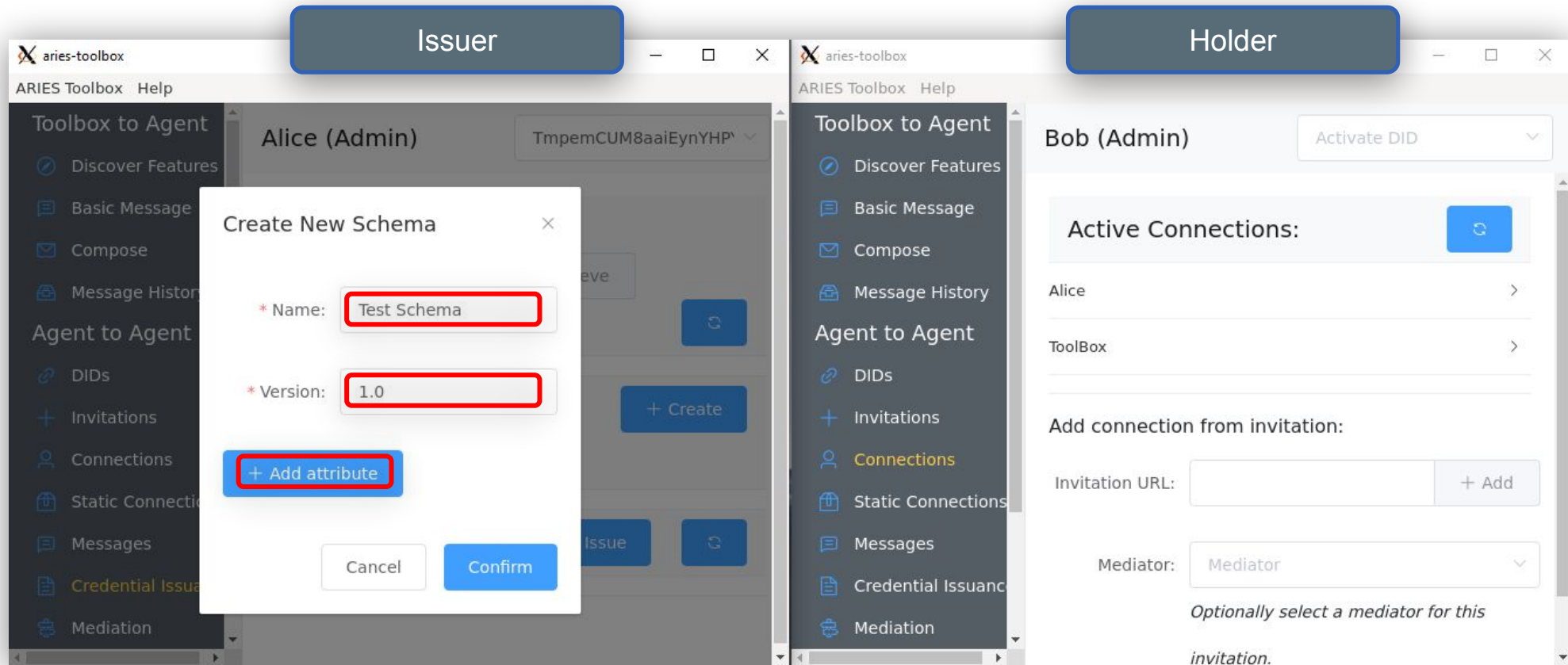




# Demo: Writing a schema to the ledger

The image displays two side-by-side windows of the ARIES Toolbox application. The left window, titled 'Issuer', shows the 'Alice (Admin)' interface with a 'Create New Schema' dialog box open. The dialog box has two input fields: '\* Name:' and '\* Version:'. Below these fields is a blue button labeled '+ Add attribute'. At the bottom of the dialog are 'Cancel' and 'Confirm' buttons. The right window, titled 'Holder', shows the 'Bob (Admin)' interface. It features a dropdown menu for 'Activate DID' and a section for 'Active Connections' with a refresh button. Below this is a section for 'Add connection from invitation' with an 'Invitation URL' input field and a '+ Add' button. A 'Mediator' dropdown menu is also present, with the text 'Optionally select a mediator for this invitation.' below it.

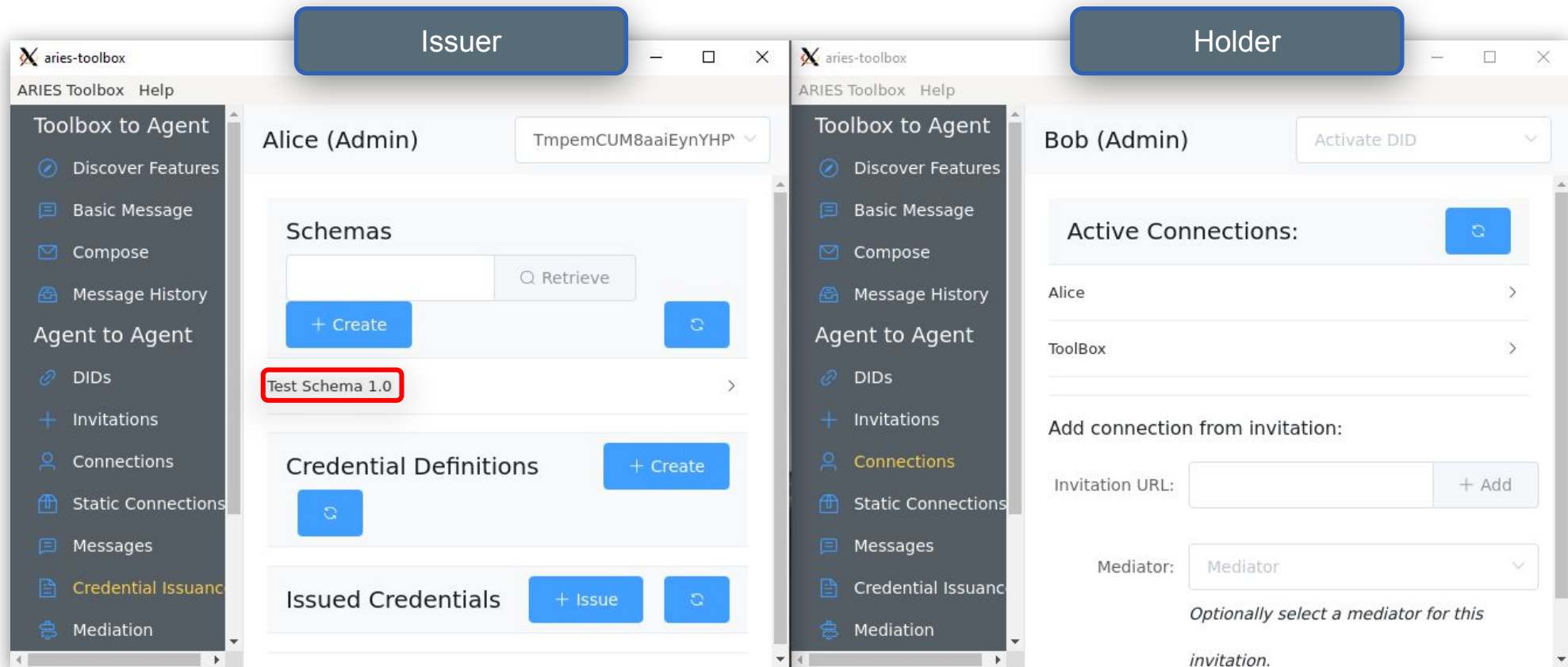
# Demo: Writing a schema to the ledger



# Demo: Writing a schema to the ledger

The image displays two side-by-side windows of the Aries Toolbox application. The left window, titled 'Issuer', shows a modal dialog for creating a schema. The dialog has the following fields: Name (Test Schema), Version (1.0), Attribute 0 (name), and Attribute 1 (age). The 'name' and 'age' fields are highlighted with red boxes. At the bottom of the dialog are 'Cancel' and 'Confirm' buttons, with 'Confirm' also highlighted in red. The right window, titled 'Holder', shows the user interface for 'Bob (Admin)'. It includes an 'Activate DID' button, a list of 'Active Connections' (Alice and ToolBox), and a section for 'Add connection from invitation' with an 'Invitation URL' field and a 'Mediator' dropdown menu.

# Demo: Writing a schema to the ledger



See the transaction on the ledger at:

[https://indyscan.indiciotech.io/home/INDICIO\\_TESTNET](https://indyscan.indiciotech.io/home/INDICIO_TESTNET)

## c. Write a credential definition to the ledger

Credential definition: creates the issuer-specific cryptographic keys necessary for issuing a credential

```
{
  "author": "self",
  "support_revocation": false,
  "state": "written",
  "created_at": "2022-01-14T19:11:17.699780Z",
  "schema_id": "UZukR8GpvEz3Fi1HbRv5qY:2:Test
                Schema:1.1",
  "updated_at": "2022-01-14T19:11:17.699780Z",
  "attributes": [
    "first_name",
    "last_name",
    "age"
  ]
  "cred_def_id": "UZukR8GpvEz3Fi1HbRv5qY:3:CL:463]
                Schema_1.1"
}
```

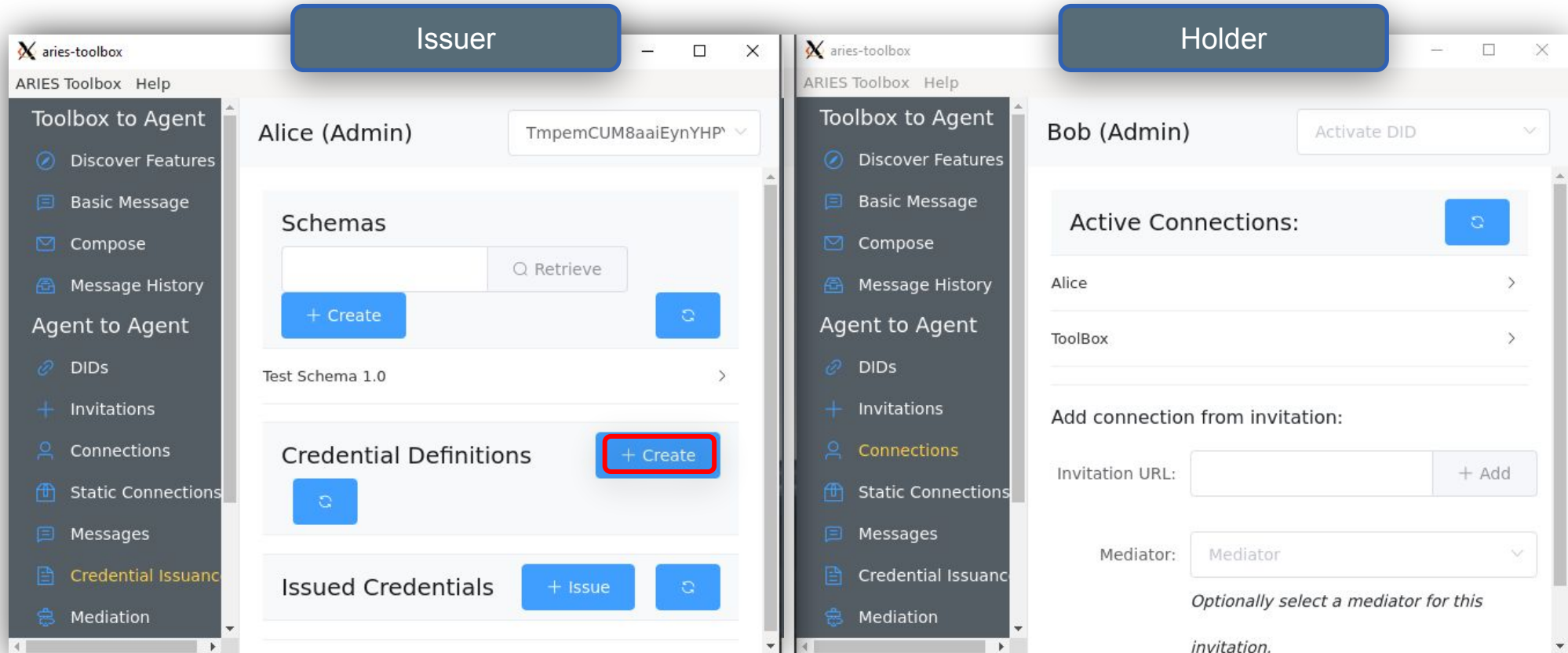
The background of the slide is a blurred image of a person's hands typing on a laptop keyboard. Overlaid on the top left is a white network diagram consisting of several circular nodes connected by thin lines, forming a web-like structure. The overall color palette is a muted, light blue-grey.

# Demo

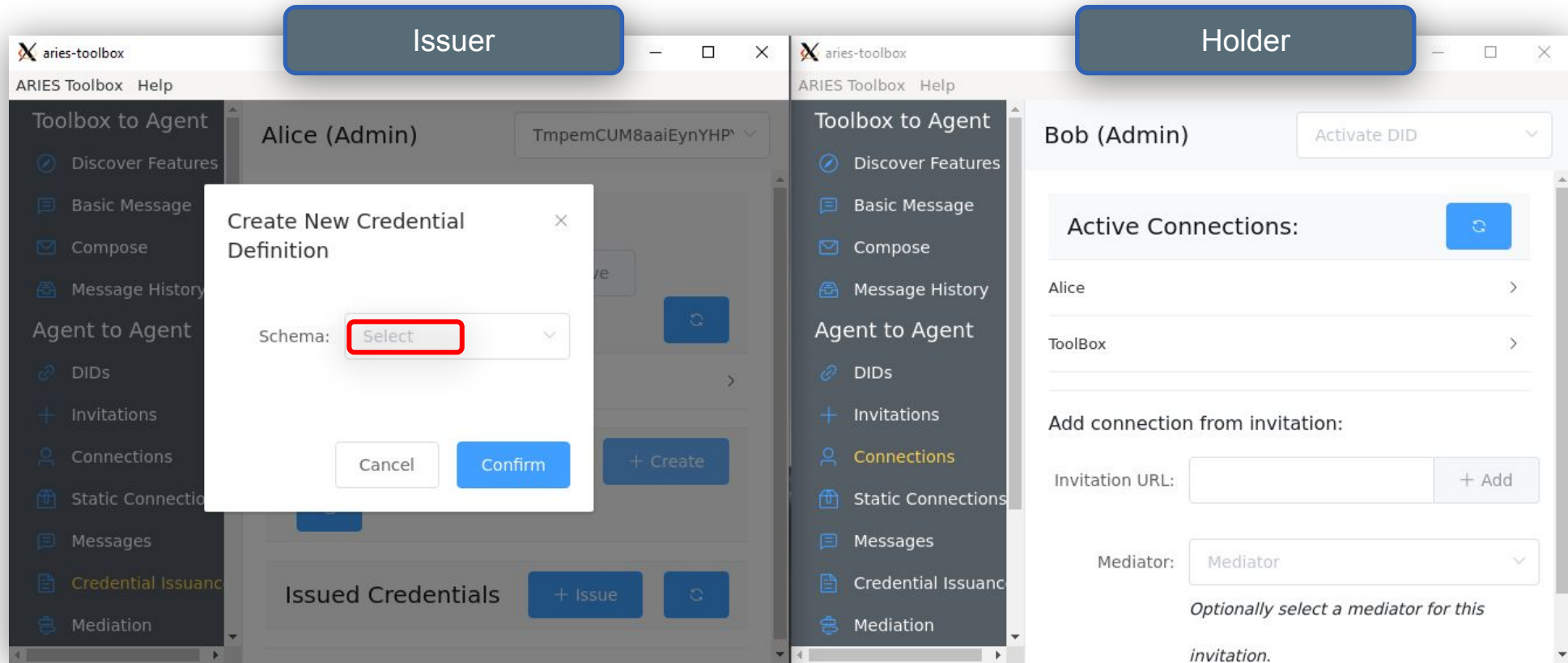
Creating a schema and  
credential definition



# Demo: Creating a Credential Definition

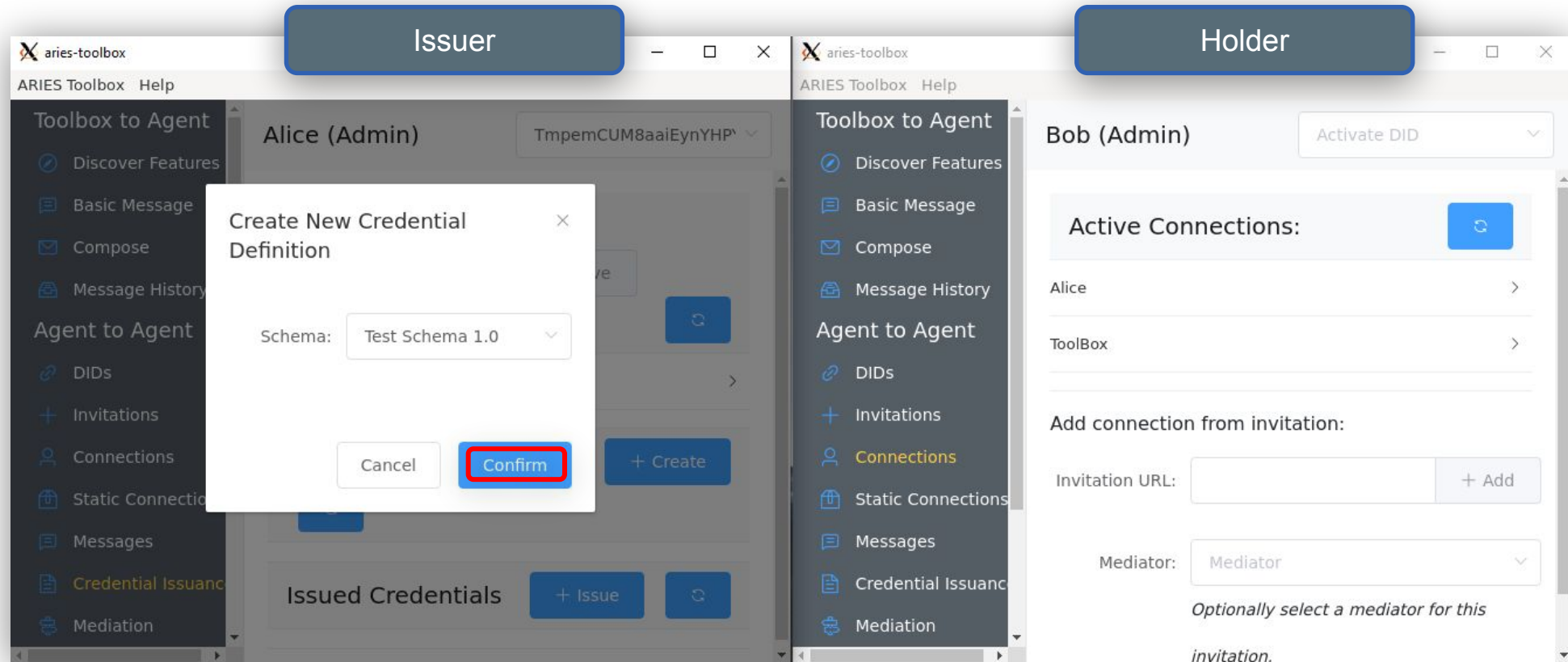


# Demo: Creating a Credential Definition

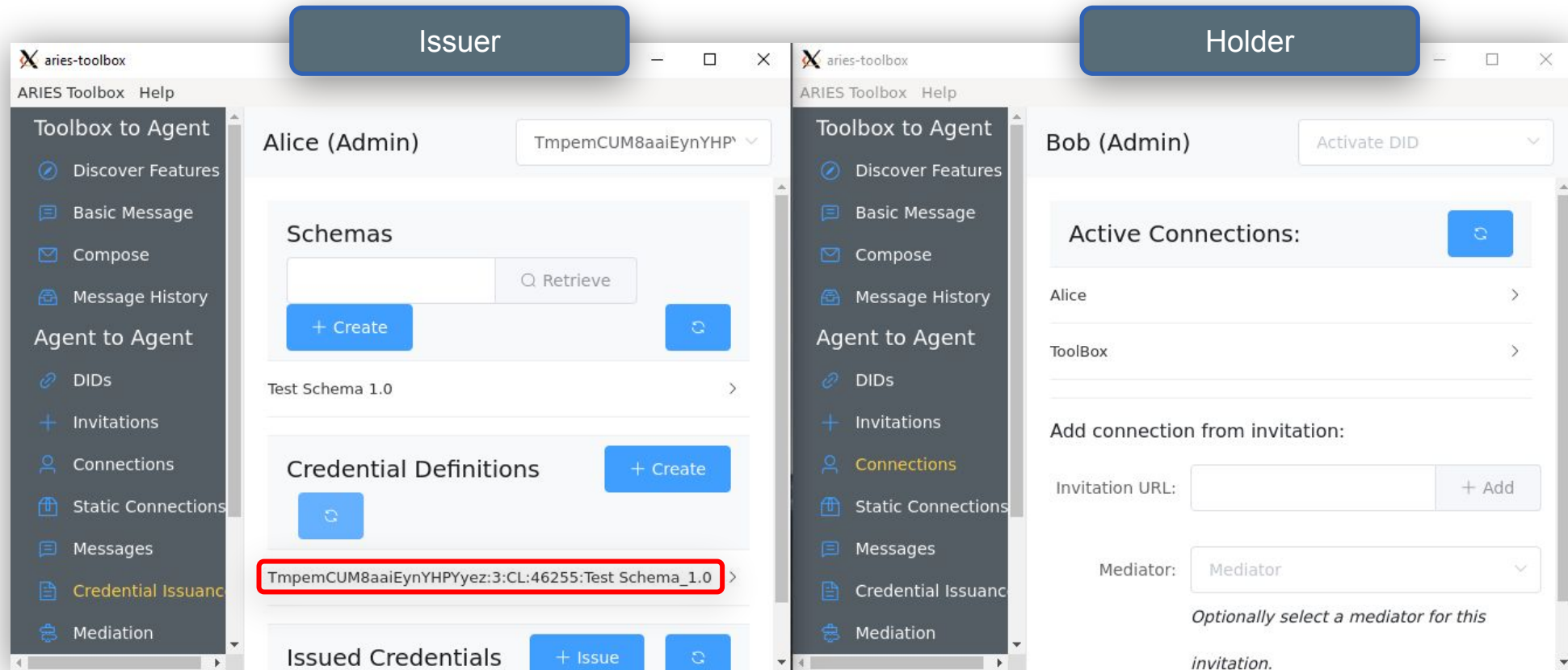




# Demo: Creating a Credential Definition



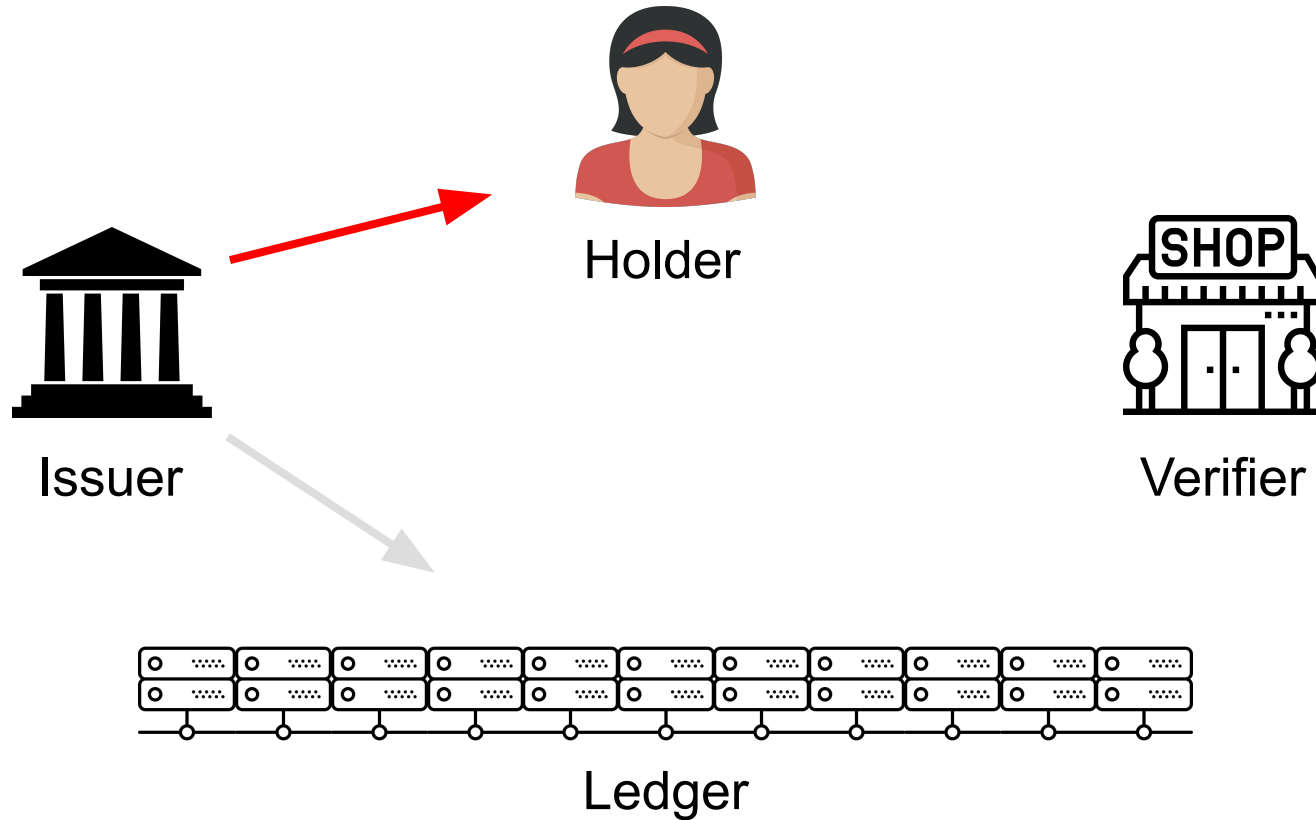
# Demo: Creating a Credential Definition



# The Verifiable Credential Implementation

2. Sign and issue a verifiable credential

1. Write DID, schema, and credential definition to the ledger



## d. Issue a verifiable credential

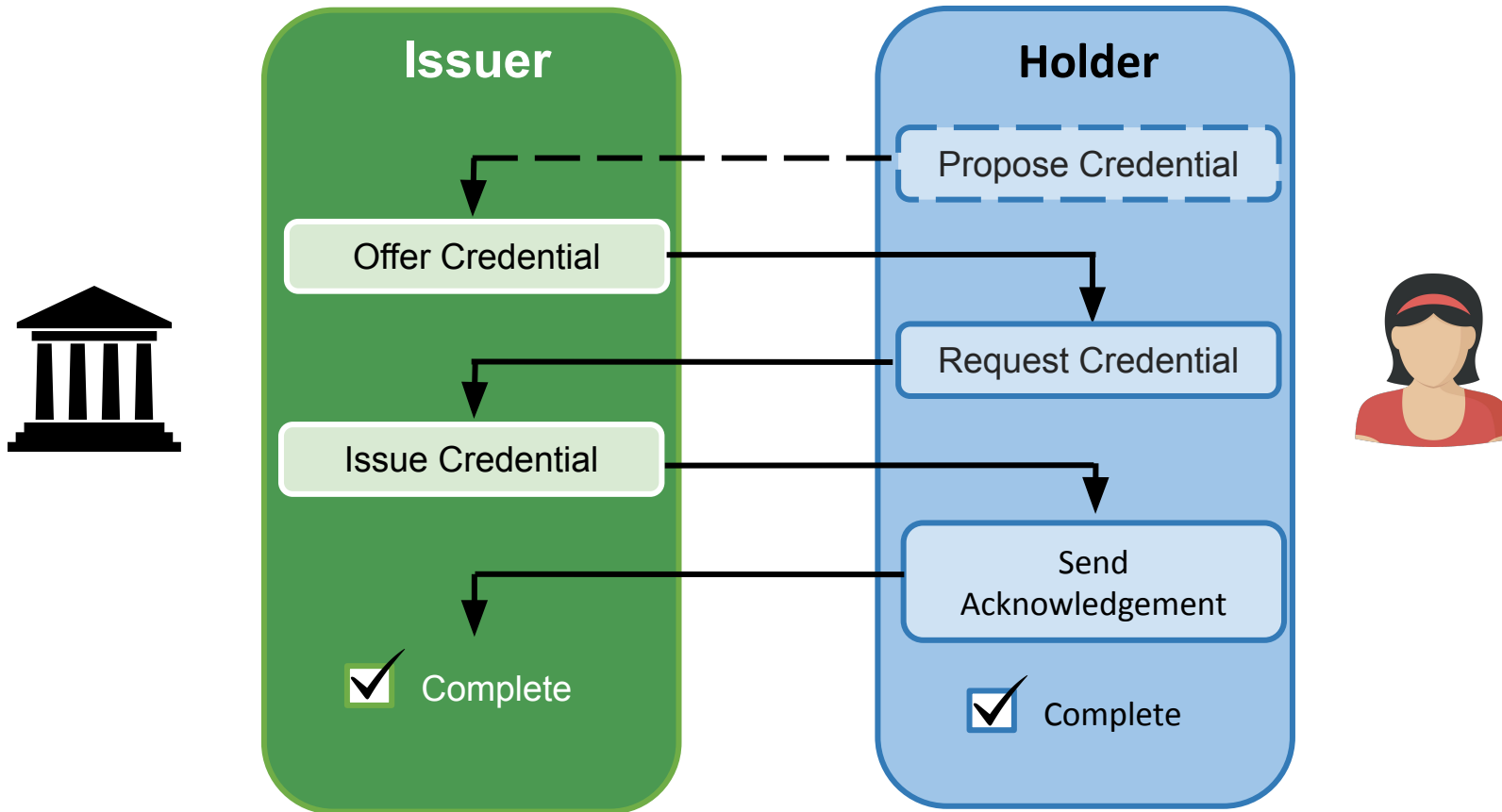
A signed statement with attributes about a subject.

### Contents

- Metadata about the credential
- Attributes about the subject
- Proof (signature)



# Credential Issuance



# Demo: Issuing a Credential

The image displays two side-by-side windows of the ARIES Toolbox interface. The left window is titled "Issuer" and shows the "Alice (Admin)" view. The right window is titled "Holder" and shows the "Bob (Admin)" view.

**Issuer View (Alice (Admin)):**

- Toolbox to Agent: Discover Features, Basic Message, Compose, Message History
- Agent to Agent: DIDs, Invitations, Connections, Static Connections, Messages, Credential Issuance, Mediation
- Current DID: TmpemCUM8aaiEynYHP
- Search: Retrieve
- Buttons: + Create, Refresh
- Test Schema 1.0
- Credential Definitions: + Create, Refresh
- Current Credential: TmpemCUM8aaiEynYHPyez:3:CL:46255:Test Schema\_1.0
- Issued Credentials: + Issue (highlighted with a red box), Refresh

**Holder View (Bob (Admin)):**

- Toolbox to Agent: Discover Features, Basic Message, Compose, Message History
- Agent to Agent: DIDs, Invitations, Connections, Static Connections, Messages, Credential Issuance, Mediation
- Current DID: Activate DID
- Active Connections: Refresh
- Connections: Alice, ToolBox
- Add connection from invitation: Invitation URL: + Add, Mediator: Mediator
- Note: Optionally select a mediator for this invitation.

# Demo: Issuing a Credential

The image displays two side-by-side screenshots of the ARIES Toolbox interface, labeled 'Issuer' and 'Holder'.

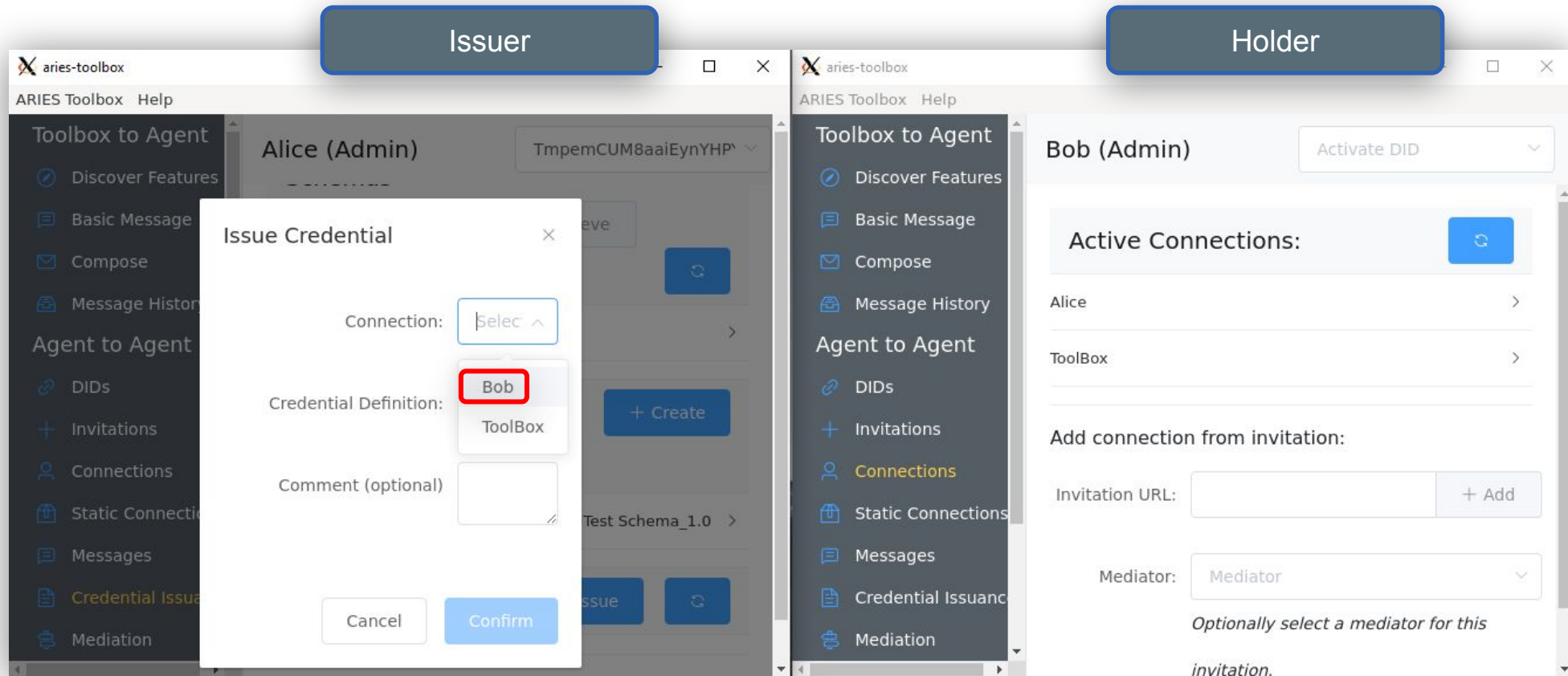
**Issuer View:** The interface shows the 'Alice (Admin)' user. A modal dialog titled 'Issue Credential' is open, containing the following fields:

- Connection: A dropdown menu with 'Select' visible.
- Credential Definition: A dropdown menu with 'Select' visible.
- Comment (optional): A text input field.

Buttons for 'Cancel' and 'Confirm' are at the bottom of the dialog. The background interface includes a sidebar with options like 'Discover Features', 'Basic Message', 'Compose', 'Message History', 'DIDs', 'Invitations', 'Connections', 'Static Connections', 'Messages', 'Credential Issuance', and 'Mediation'. A '+ Create' button is also visible.

**Holder View:** The interface shows the 'Bob (Admin)' user. It features an 'Active Connections' section with a refresh button and a list of connections including 'Alice' and 'ToolBox'. Below this is the 'Add connection from invitation' section, which includes an 'Invitation URL' field with a '+ Add' button and a 'Mediator' dropdown menu. A note at the bottom states: 'Optionally select a mediator for this invitation.'

# Demo: Issuing a Credential

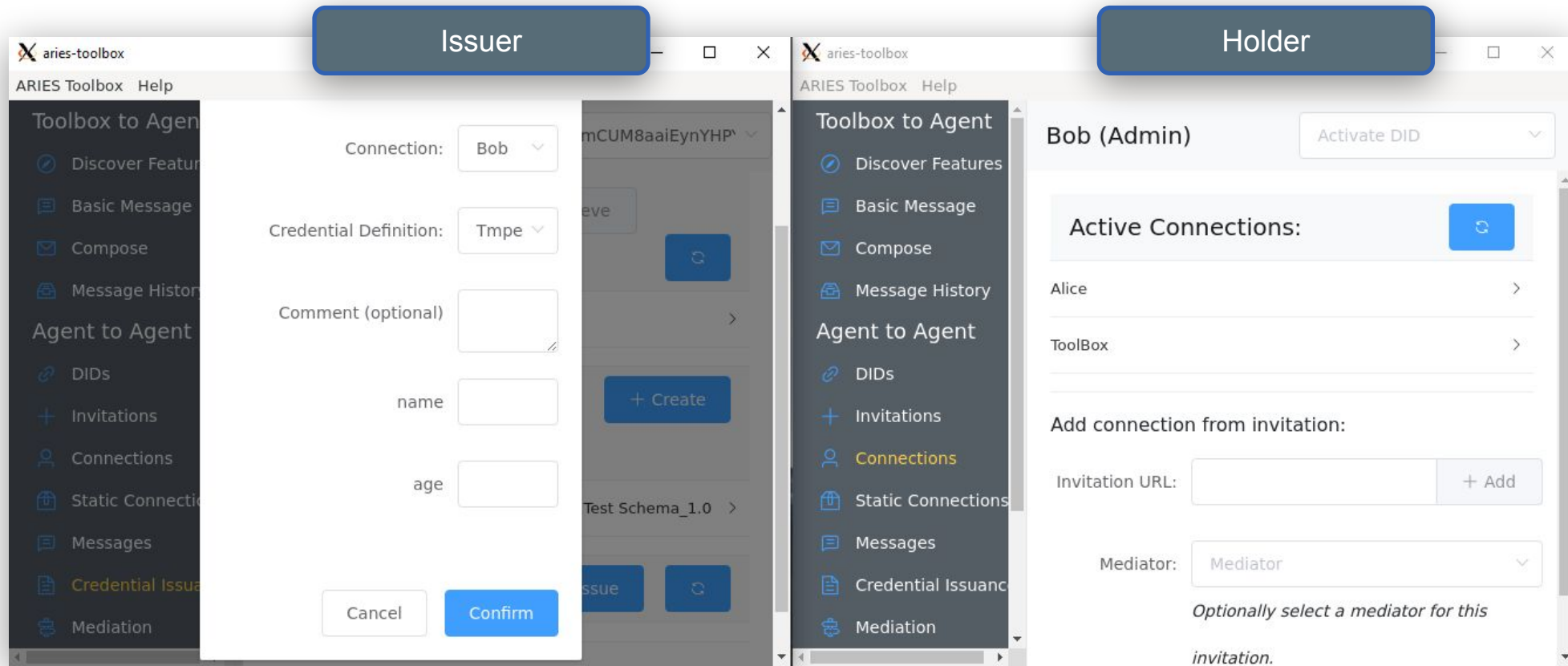




# Demo: Issuing a Credential

The image displays two side-by-side windows from the Aries Toolbox application. The left window, labeled 'Issuer', shows the interface for Alice (Admin). A modal dialog titled 'Issue Credential' is open, showing a 'Connection' dropdown set to 'Bob' and a 'Credential Definition' dropdown set to 'Select...'. Below the dropdowns, the credential definition 'TmpemCUM8aaEynYHPYyez:3:CL:46255:Test Schema\_1.0' is displayed and highlighted with a red box. The right window, labeled 'Holder', shows the interface for Bob (Admin). The 'Active Connections' section is visible, listing 'Alice' and 'ToolBox' as active connections. Below this, there is a section for 'Add connection from invitation' with an 'Invitation URL' field and a '+ Add' button, and a 'Mediator' dropdown set to 'Mediator'.

# Demo: Issuing a Credential



# Demo: Issuing a Credential

The image displays two side-by-side screenshots of the ARIES Toolbox web application. The left window, titled 'aries-toolbox', is labeled 'Issuer' and shows a modal form for issuing a credential. The form includes a 'Connection' dropdown set to 'Bob', a 'Credential Definition' dropdown set to 'Tmpe', and a 'Comment (optional)' text area. The 'name' field contains 'Robert' and the 'age' field contains '50', both highlighted with red boxes. At the bottom of the form are 'Cancel' and 'Confirm' buttons. The right window, also titled 'aries-toolbox', is labeled 'Holder' and shows the 'Bob (Admin)' profile page. It features an 'Activate DID' button, a list of 'Active Connections' (Alice and ToolBox), and a section for 'Add connection from invitation' with an 'Invitation URL' field and a '+ Add' button. A 'Mediator' dropdown is set to 'Mediator'.

# Demo: Issuing a Credential

The image displays two side-by-side browser windows of the ARIES Toolbox interface. The left window is titled "Issuer" and shows the "Alice (Admin)" view. The right window is titled "Holder" and shows the "Bob (Admin)" view.

**Issuer View (Alice Admin):**

- Toolbox to Agent: Discover Features, Basic Message, Compose, Message History.
- Agent to Agent: DIDs, Invitations, Connections, Static Connections, Messages, **Credential Issuance**, Mediation.
- Current view: Alice (Admin) with a dropdown menu showing "TmpemCUM8aaiEynYHP".
- Buttons: "+ Create" and a refresh icon.
- Section: "Test Schema 1.0" with a right arrow.
- Section: "Credential Definitions" with a "+ Create" button and a refresh icon.
- Text: "TmpemCUM8aaiEynYHPyez:3:CL:46255:Test Schema\_1.0" with a right arrow.
- Section: "Issued Credentials" with a "+ Issue" button and a refresh icon.
- Text: "Test Schema v1.0 issued to Bob" (highlighted with a red box).

**Holder View (Bob Admin):**

- Toolbox to Agent: Discover Features, Basic Message, Compose, Message History.
- Agent to Agent: DIDs, Invitations, **Connections**, Static Connections, Messages, Credential Issuance, Mediation.
- Current view: Bob (Admin) with a dropdown menu showing "Activate DID".
- Section: "Active Connections:" with a refresh icon.
- List: Alice, Toolbox (with right arrows).
- Section: "Add connection from invitation:"
- Form: "Invitation URL:" with an empty input field and a "+ Add" button.
- Form: "Mediator:" with a dropdown menu showing "Mediator".
- Text: "Optionally select a mediator for this invitation."

# Demo: Issuing a Credential

The image displays two side-by-side screenshots of the ARIES Toolbox interface, labeled 'Issuer' and 'Holder'.

**Issuer Interface:**

- Header: Alice (Admin), TmpemCUM8aaieYnYHP
- Buttons: + Create, Refresh
- Section: Test Schema 1.0
- Section: Credential Definitions (+ Create, Refresh)
- Section: TmpemCUM8aaieYnYHPyez:3:CL:46255:Test Schema\_1.0
- Section: Issued Credentials (+ Issue, Refresh)
- Item: Test Schema v1.0 issued to Bob (highlighted with a red box)

**Holder Interface:**

- Header: Bob (Admin), Activate DID
- Section: Retrieved Credential Definitions (Retrieve, Refresh)
- Section: Credentials (+ Send Credential Proposal, Refresh)
- Item: Test Schema v1.0 received from Alice (highlighted with a red box)

Both interfaces have a sidebar menu with options like Discover Features, Basic Message, Compose, Message History, DIDs, Invitations, Connections, Static Connections, Messages, Credential Issuance, and Mediation. The 'My Credentials' option in the Issuer sidebar is also highlighted with a red box.

The background of the slide is a blurred image of a person's hands typing on a laptop keyboard. In the upper-left corner, there is a white network diagram consisting of several circular nodes connected by thin lines, forming a web-like structure. The overall color palette is a soft, muted blue.

# Demo

Issuing a credential



# 7. Verifying a Credential



# The Verifier Role

Verifier responsibilities

- Decide who and what to trust
- Interact with the holder
- Read from the ledger
- Perform cryptographic computations

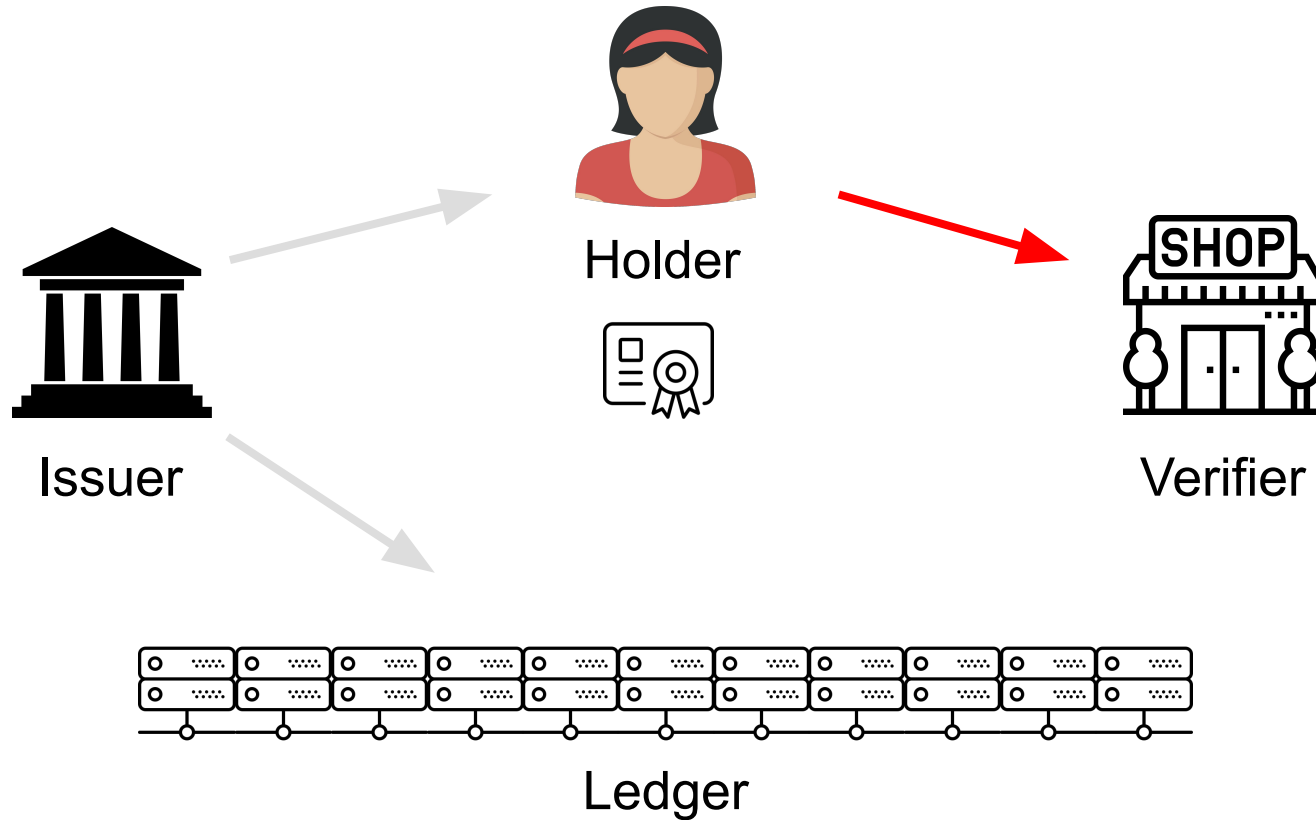
The verifier **does not** interact with the issuer



# The Verifiable Credential Implementation

2. Sign and issue a verifiable credential

1. Write DID, schema, and credential definition to the ledger

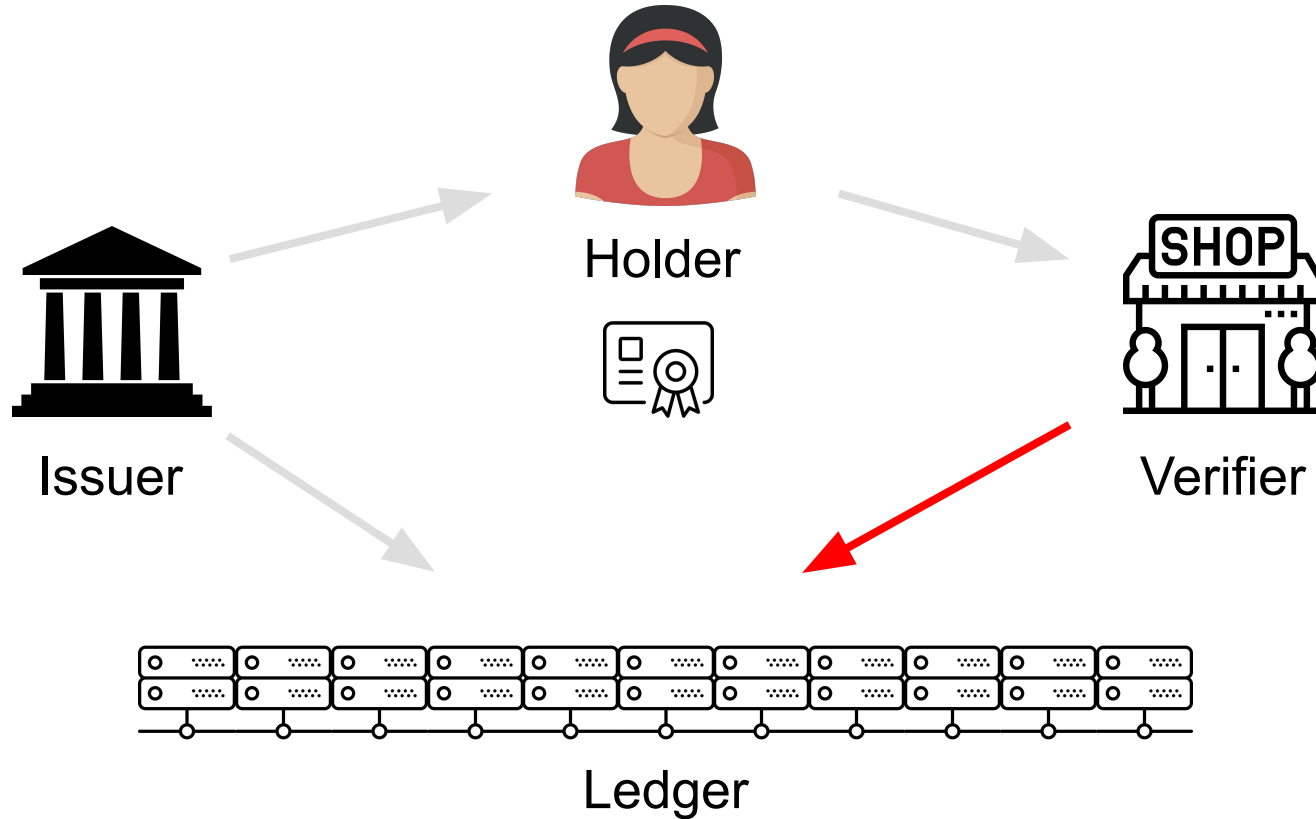


3. Create a presentation

# The Verifiable Credential Implementation

2. Sign and issue a verifiable credential

1. Write DID, schema, and credential definition to the ledger



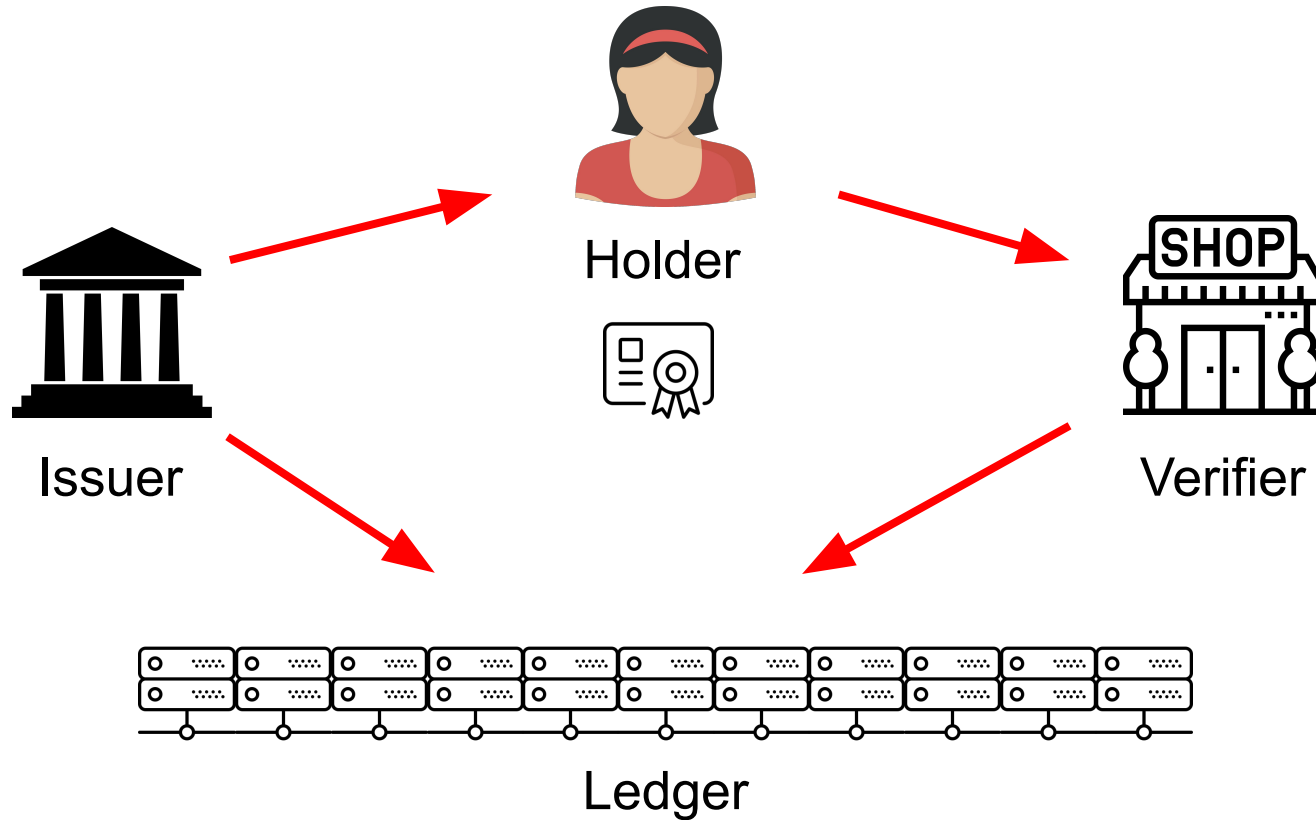
3. Create a presentation

4. Read DID, schema, and credential definition from the ledger and verify attestations

# The Verifiable Credential Implementation

2. Sign and issue a verifiable credential

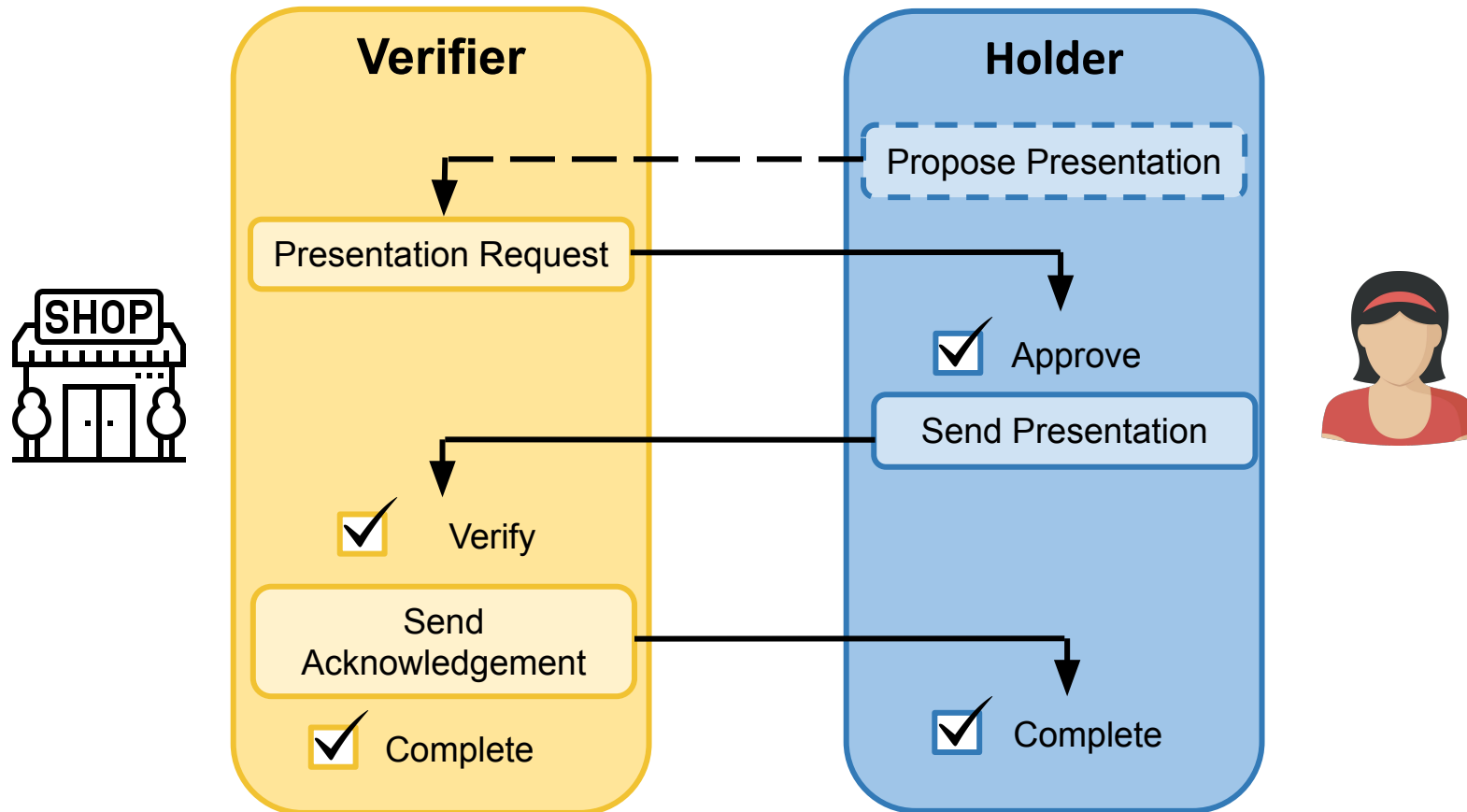
1. Write DID, schema, and credential definition to the ledger



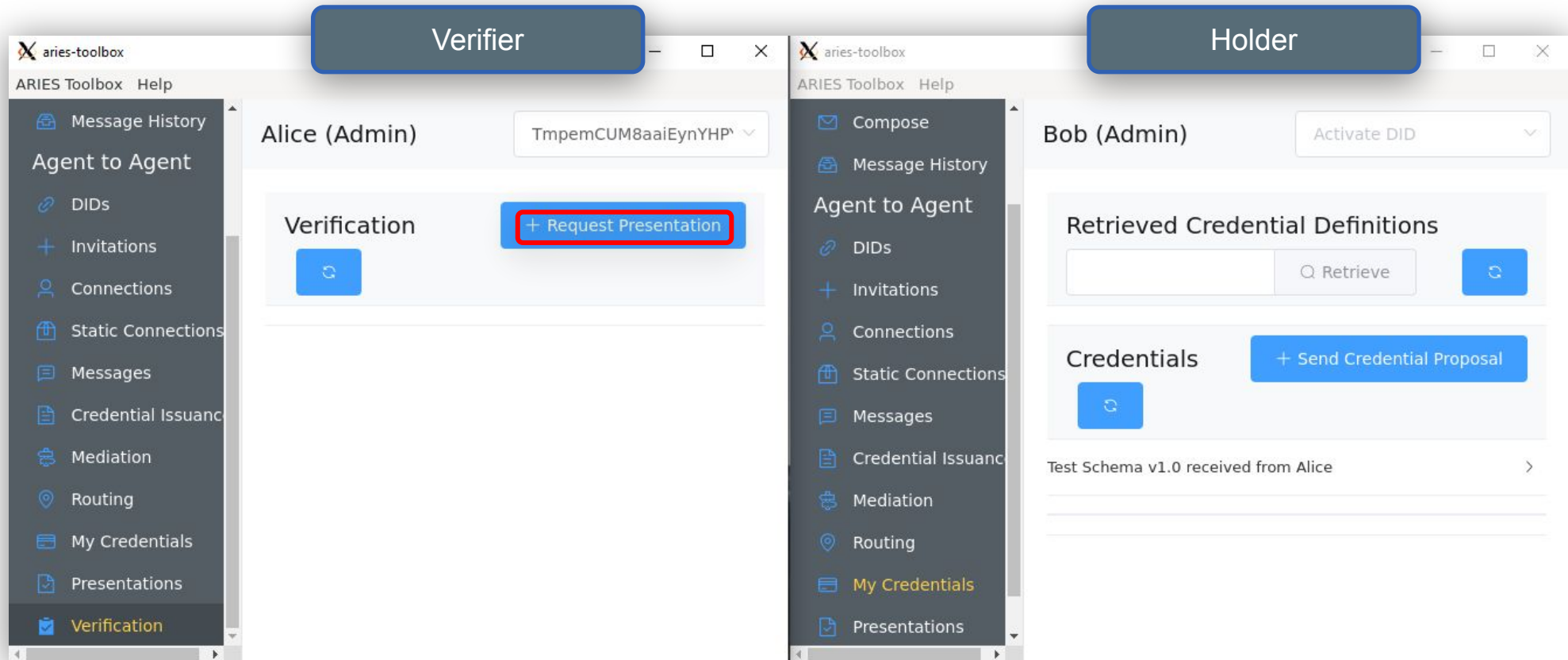
3. Create a presentation

4. Read DID, schema, and credential definition from the ledger and verify attestations

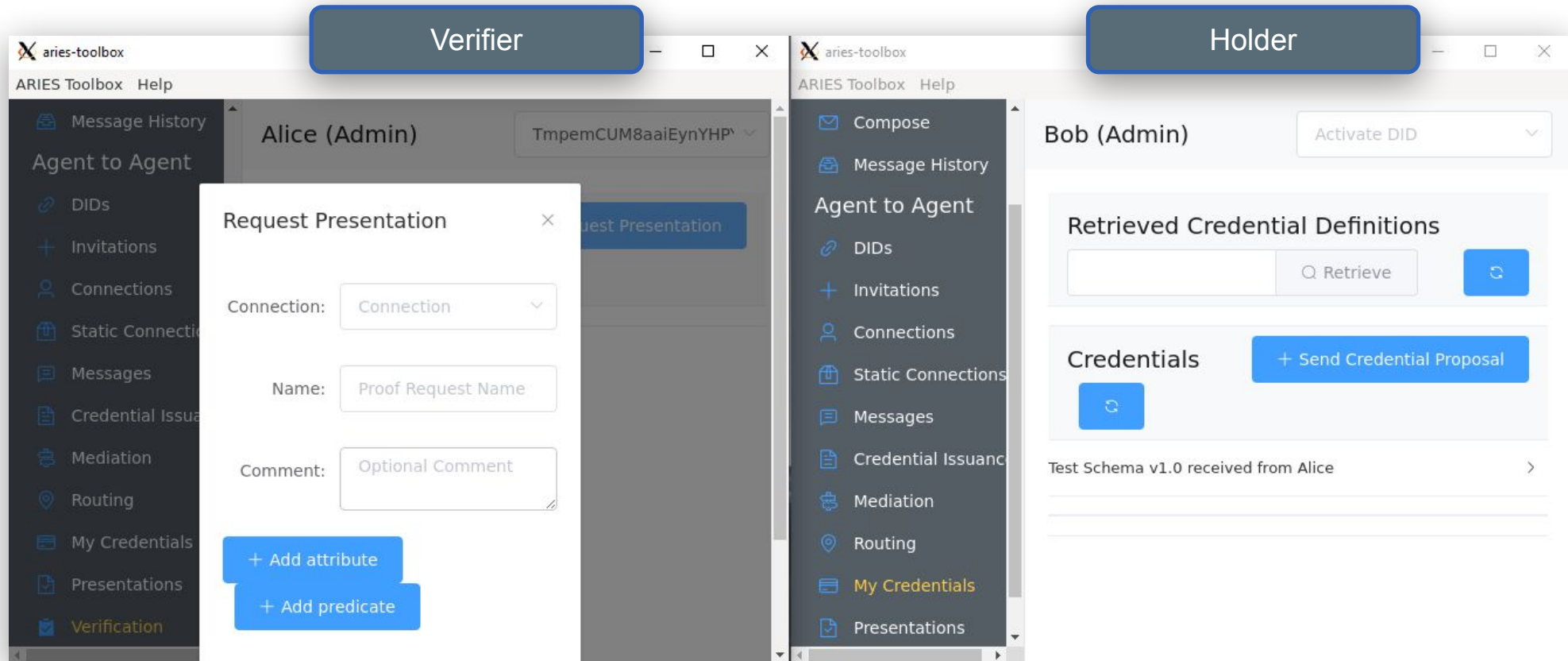
# Basic Presentation



# Demo: Verifying a Credential



# Demo: Verifying a Credential



The image displays two side-by-side windows from the Aries Toolbox application, labeled 'Verifier' and 'Holder'.

**Verifier Window:** The interface shows 'Alice (Admin)' with a connection ID 'TmpeM8aaiEynYHP'. A 'Request Presentation' dialog is open, containing the following fields and buttons:

- Connection:
- Name:
- Comment:
- + Add attribute
- + Add predicate

**Holder Window:** The interface shows 'Bob (Admin)' with an 'Activate DID' dropdown. It features two main sections:

- Retrieved Credential Definitions:** Includes a search bar, a 'Retrieve' button, and a refresh button.
- Credentials:** Includes a '+ Send Credential Proposal' button and a refresh button. Below this, it shows 'Test Schema v1.0 received from Alice' with a right-pointing arrow.

# Demo: Verifying a Credential

The image displays two side-by-side windows of the ARIES Toolbox interface. The left window is labeled 'Verifier' and shows the 'Alice (Admin)' user interface. A 'Request Presentation' dialog box is open, with the 'Connection' dropdown set to 'Bob' and the 'Name' dropdown set to 'Bob' (highlighted with a red box). Below the dialog are two blue buttons: '+ Add attribute' and '+ Add predicate'. The right window is labeled 'Holder' and shows the 'Bob (Admin)' user interface. It features a 'Retrieve' button in the 'Retrieved Credential Definitions' section and a '+ Send Credential Proposal' button in the 'Credentials' section. Below these sections, a message reads 'Test Schema v1.0 received from Alice'.

# Demo: Verifying a Credential

The image displays two side-by-side windows from the ARIES Toolbox application, labeled 'Verifier' and 'Holder'.

**Verifier Window:** The main interface shows 'Alice (Admin)' with a connection ID 'TmpeM8aaiEynYHP'. A 'Request Presentation' dialog is open, showing a 'Connection' dropdown set to 'Bob', a 'Name' field containing 'Test Presentation', and a 'Comment' field with 'Optional Comment'. Below the dialog are two buttons: '+ Add attribute' and '+ Add predicate'.

**Holder Window:** The main interface shows 'Bob (Admin)' with an 'Activate DID' dropdown. Below this are two sections: 'Retrieved Credential Definitions' with a search bar and a 'Retrieve' button, and 'Credentials' with a '+ Send Credential Proposal' button. A notification at the bottom states 'Test Schema v1.0 received from Alice'.



# Demo: Verifying a Credential

The image displays two side-by-side windows from the ARIES Toolbox application. The left window, labeled 'Verifier', shows the 'Alice (Admin)' interface with a 'Request Presentation' dialog box open. The dialog contains the following fields: 'Connection' set to 'Bob', 'Name' set to 'Test Presentation', 'Comment' set to 'Optional Comment', and 'Attribute 0' set to 'Attribute nam'. The right window, labeled 'Holder', shows the 'Bob (Admin)' interface. It features a 'Retrieved Credential Definitions' section with a search bar and a 'Retrieve' button. Below this is a 'Credentials' section with a '+ Send Credential Proposal' button. A message at the bottom of the Holder window reads 'Test Schema v1.0 received from Alice'.

# Demo: Verifying a Credential

The image displays two side-by-side windows from the Aries Toolbox application. The left window, titled 'Verifier', shows a form for creating a presentation. The 'Connection' is set to 'Bob', the 'Name' is 'Test Presentation', and the 'Comment' is 'Optional Comment'. Under 'Attribute 0', the value 'name' is entered and highlighted with a red box. Below this, a 'Restrictions' field contains the identifier 'TmpeM8aaiEynYHPYez:3:CL:46255:Test Schema 1.0', which is also highlighted with a red box. A '+ Add predicate' button is visible at the bottom. The right window, titled 'Holder', shows the user 'Bob (Admin)' with an 'Activate DID' dropdown. It features a 'Retrieved Credential Definitions' section with a search bar and a 'Retrieve' button. Below that is a 'Credentials' section with a '+ Send Credential Proposal' button and a refresh icon. A notification at the bottom reads 'Test Schema v1.0 received from Alice'.

# Demo: Verifying a Credential

The image displays two side-by-side browser windows of the ARIES Toolbox interface. The left window is titled 'Verifier' and shows a configuration screen for verifying a credential. It features two attribute input fields: 'Attribute 0' with the value 'name' and 'Attribute 1' with the value 'Attribute nam'. Below each attribute field is a 'Restrictions' dropdown menu, with the first set to 'TmpemCUM8aaieYr' and the second to 'Credential Definitior'. At the bottom of the Verifier window are two blue buttons: '+ Add attribute' and '+ Add predicate'. The right window is titled 'Holder' and shows the user interface for Bob (Admin). It includes an 'Activate DID' dropdown menu, a 'Retrieved Credential Definitions' section with a search bar and a 'Retrieve' button, and a 'Credentials' section with a '+ Send Credential Proposal' button. A notification at the bottom of the Holder window reads 'Test Schema v1.0 received from Alice'.

# Demo: Verifying a Credential

The image displays two side-by-side browser windows of the ARIES Toolbox interface. The left window is titled 'Verifier' and shows a configuration screen for a credential verification request. It features two attribute fields: 'Attribute 0' with the value 'name' and 'Attribute 1' with the value 'age'. Both fields have a red box around them. Below these fields are two 'Restrictions' dropdown menus, both set to 'TmpemCUM8aaieYr', with the second one also highlighted by a red box. At the bottom of the configuration area are two blue buttons: '+ Add attribute' and '+ Add predicate'. The right window is titled 'Holder' and shows a user profile for 'Bob (Admin)' with an 'Activate DID' dropdown. Below this is a 'Retrieved Credential Definitions' section with a search bar and a 'Retrieve' button. Underneath is a 'Credentials' section with a '+ Send Credential Proposal' button and a refresh icon. At the bottom, it shows a notification: 'Test Schema v1.0 received from Alice' with a right-pointing arrow.

# Demo: Verifying a Credential

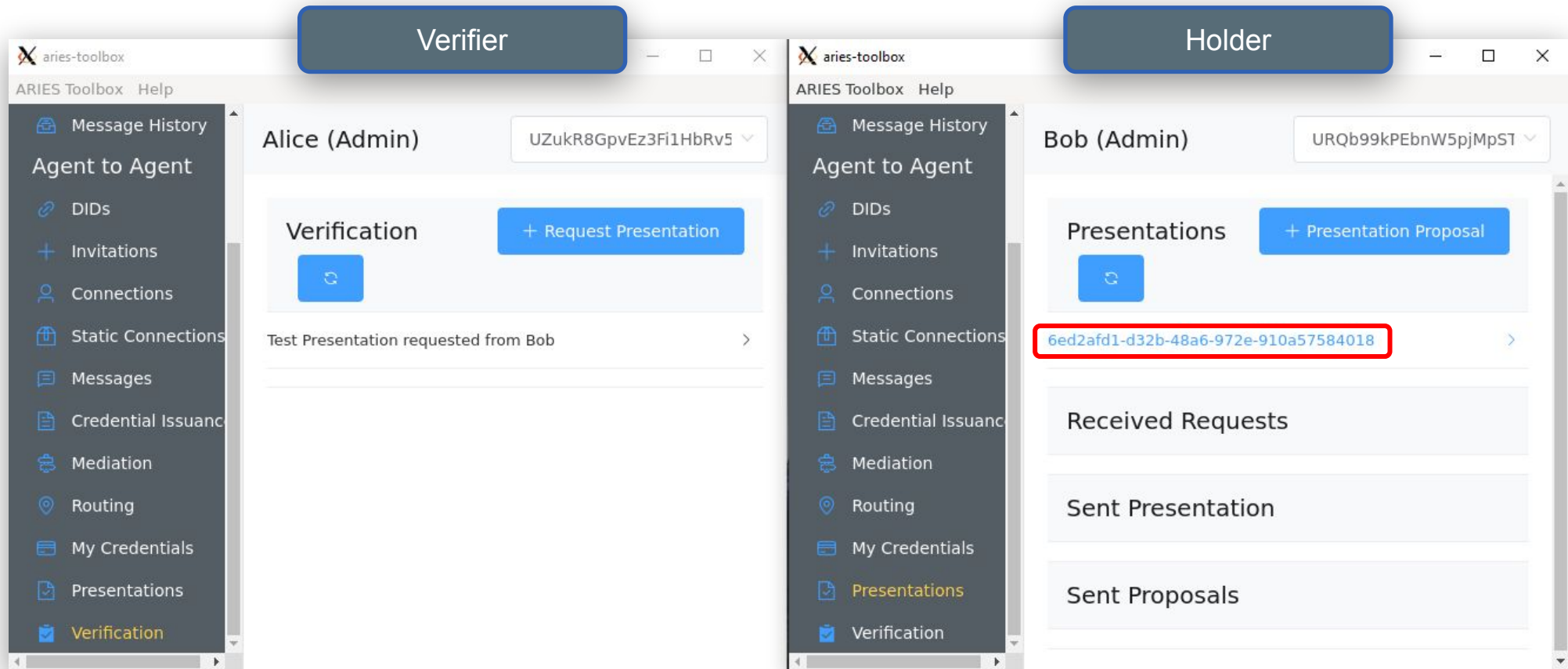
The image displays two side-by-side windows of the ARIES Toolbox application. The left window is titled "Verifier" and shows a configuration dialog for verifying a credential. It includes a dropdown menu for the credential ID (TmpeCUM8aaEyr), a text input for "Attribute 1" containing the value "age", and a "Restrictions" section with another dropdown menu (TmpeCUM8aaEyr). At the bottom of the dialog are buttons for "+ Add attribute", "+ Add predicate", "Cancel", and "Confirm" (which is highlighted with a red border). The right window is titled "Holder" and shows the user interface for a credential holder named "Bob (Admin)". It features a "Retrieve" button and a "Send Credential Proposal" button. Below these, a message states "Test Schema v1.0 received from Alice".

# Demo: Verifying a Credential

The image displays two side-by-side browser windows of the ARIES Toolbox interface. The left window is titled 'Verifier' and shows the user 'Alice (Admin)' with a dropdown menu set to 'TmpemCUM8aaiEynYHP'. The 'Verification' section is active, featuring a '+ Request Presentation' button and a refresh icon. A red box highlights the text 'Test Presentation requested from Bob' in the message area. The left sidebar menu has 'Verification' highlighted. The right window is titled 'Holder' and shows the user 'Bob (Admin)' with a dropdown menu set to 'Activate DID'. The 'Retrieved Credential Definitions' section is active, featuring a search bar, a 'Retrieve' button, and a refresh icon. The 'Credentials' section below shows a '+ Send Credential Proposal' button and a refresh icon. A message 'Test Schema v1.0 received from Alice' is visible. The left sidebar menu has 'Presentations' highlighted.



# Demo: Verifying a Credential



# Types of Presentations

## Full Disclosure Presentation

- All attributes are revealed from one verifiable credential

## Multiple Credentials Presentation

- All attributes are revealed from multiple verifiable credentials
- A single credential does not contain all of the required attributes



# Types of Presentations

## Selective Disclosure

- Only the minimum required attributes are revealed from one or more verifiable credentials

## Predicate Proof Presentation

- Reveals information about attribute value without revealing the value (e.g., reveals “true” if age  $\geq$  21)

**Revocation:** the ability to revoke a credential

The background of the slide is a blurred image of a person's hands typing on a laptop keyboard. In the top-left corner, there is a white network diagram consisting of several circular nodes connected by thin lines, forming a web-like structure. The overall color palette is a light, muted blue.

# Demo

Verifying a credential

# Conclusion

## We have:

- Discussed Aries Cloud Agent Python, the Aries toolbox and other codebases, agents, and DIDComm messaging
- Started up and connected agents
- Issued and verified credentials



# Join Us and Get Started!

- Follow the links in the handout to look at the code.
- Join or listen to community meetings.
- Try stuff and ask questions.
- Contribute if you are able.
- Reach out to any of us if we can help.