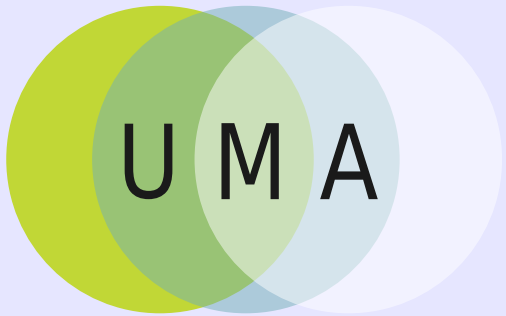
UMA protocol flow deep-dive

Paul C. Bryan
email@pbryan.net

January 29, 2010

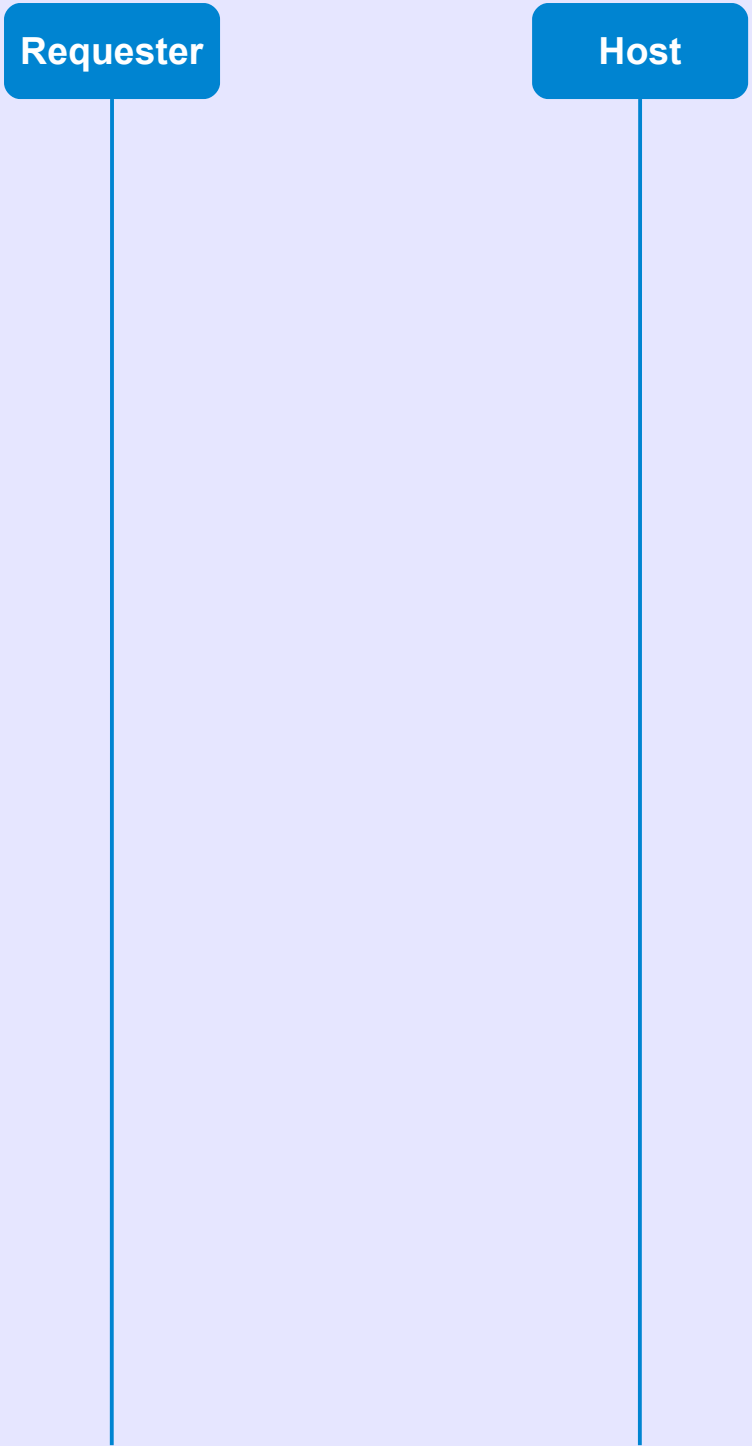


Introducing the actors

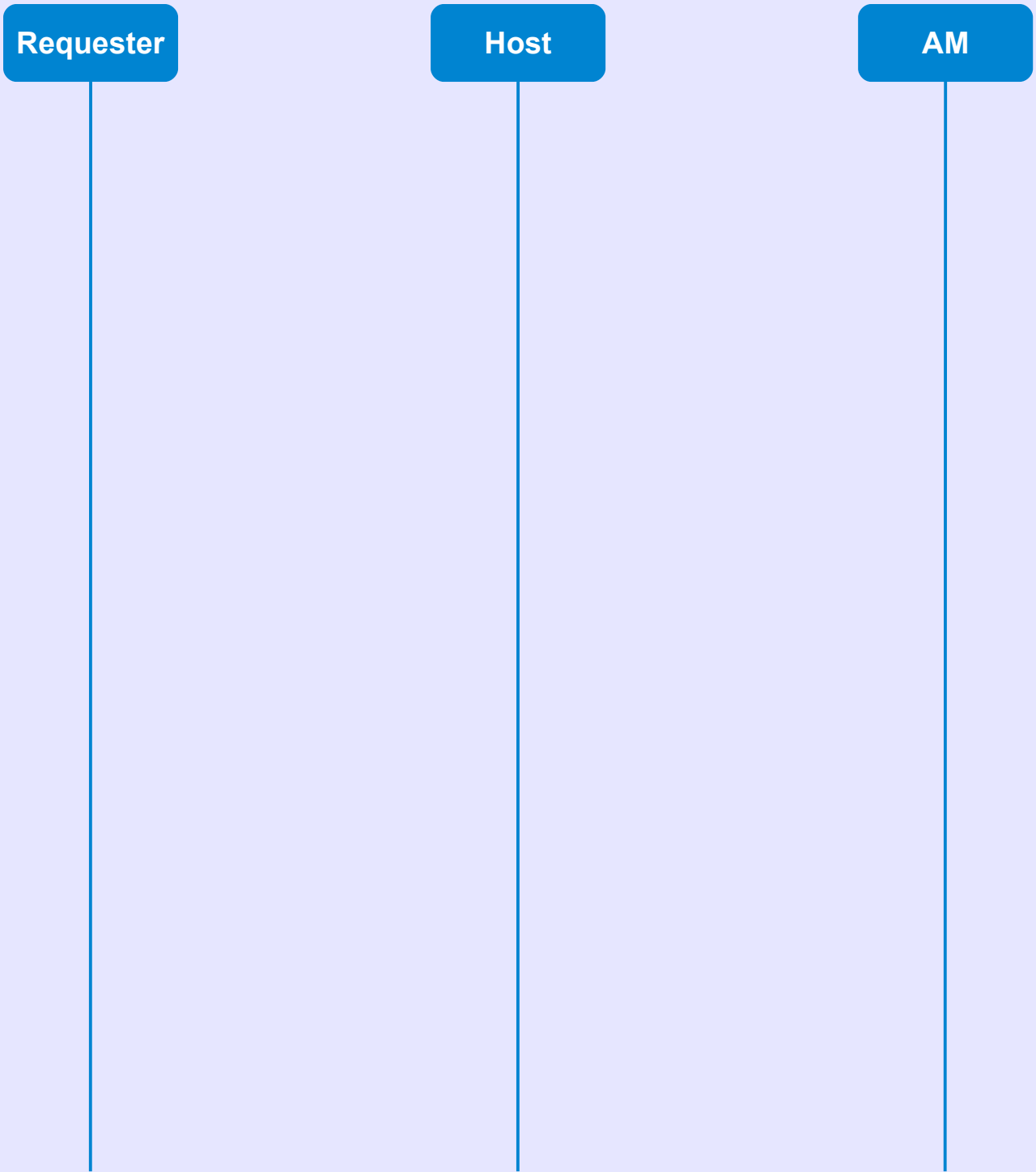
Introducing the actors

Requester

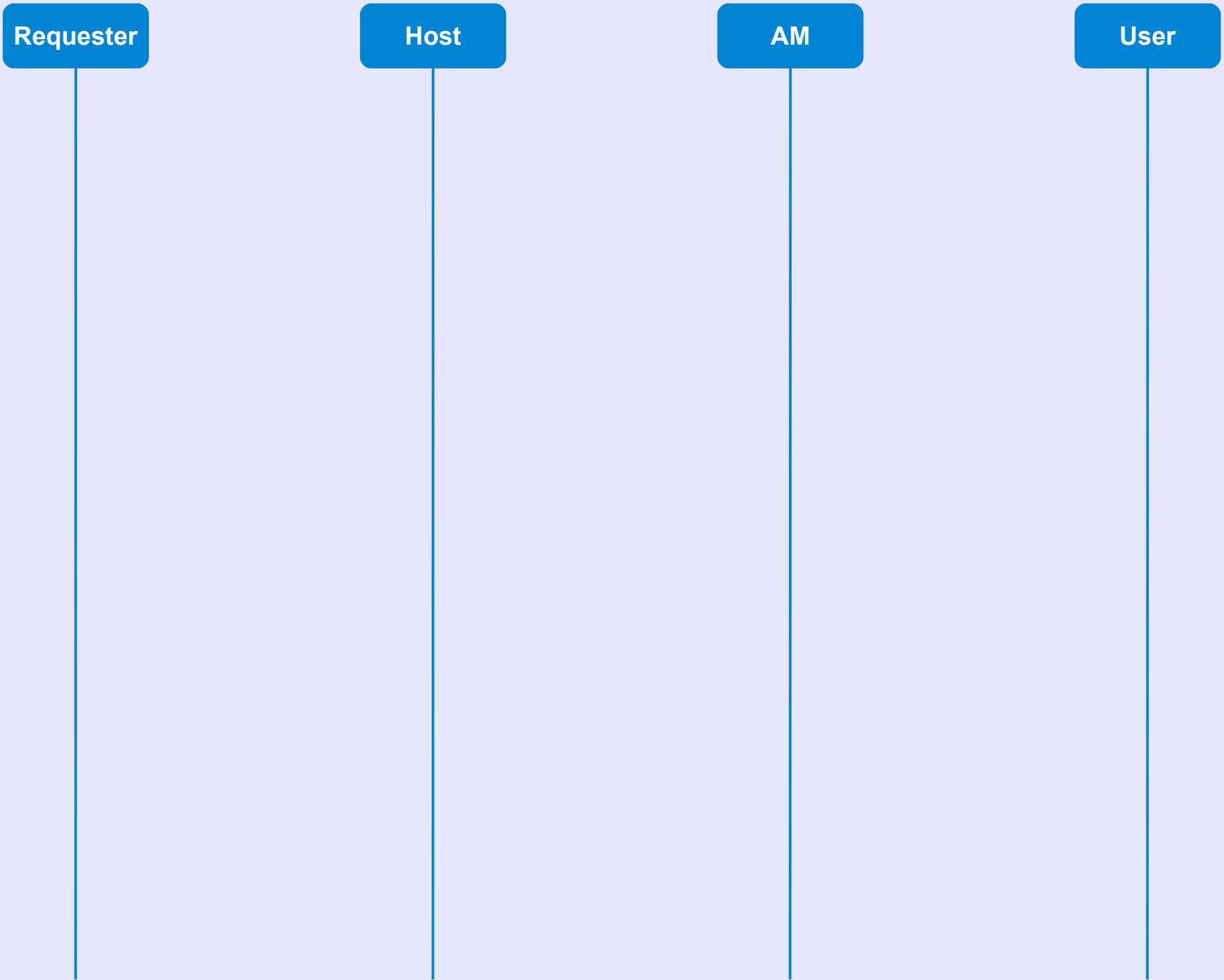
Introducing the actors



Introducing the actors

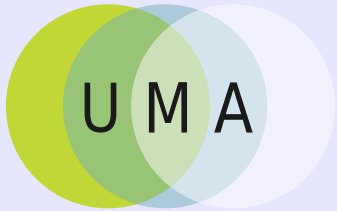


Introducing the actors

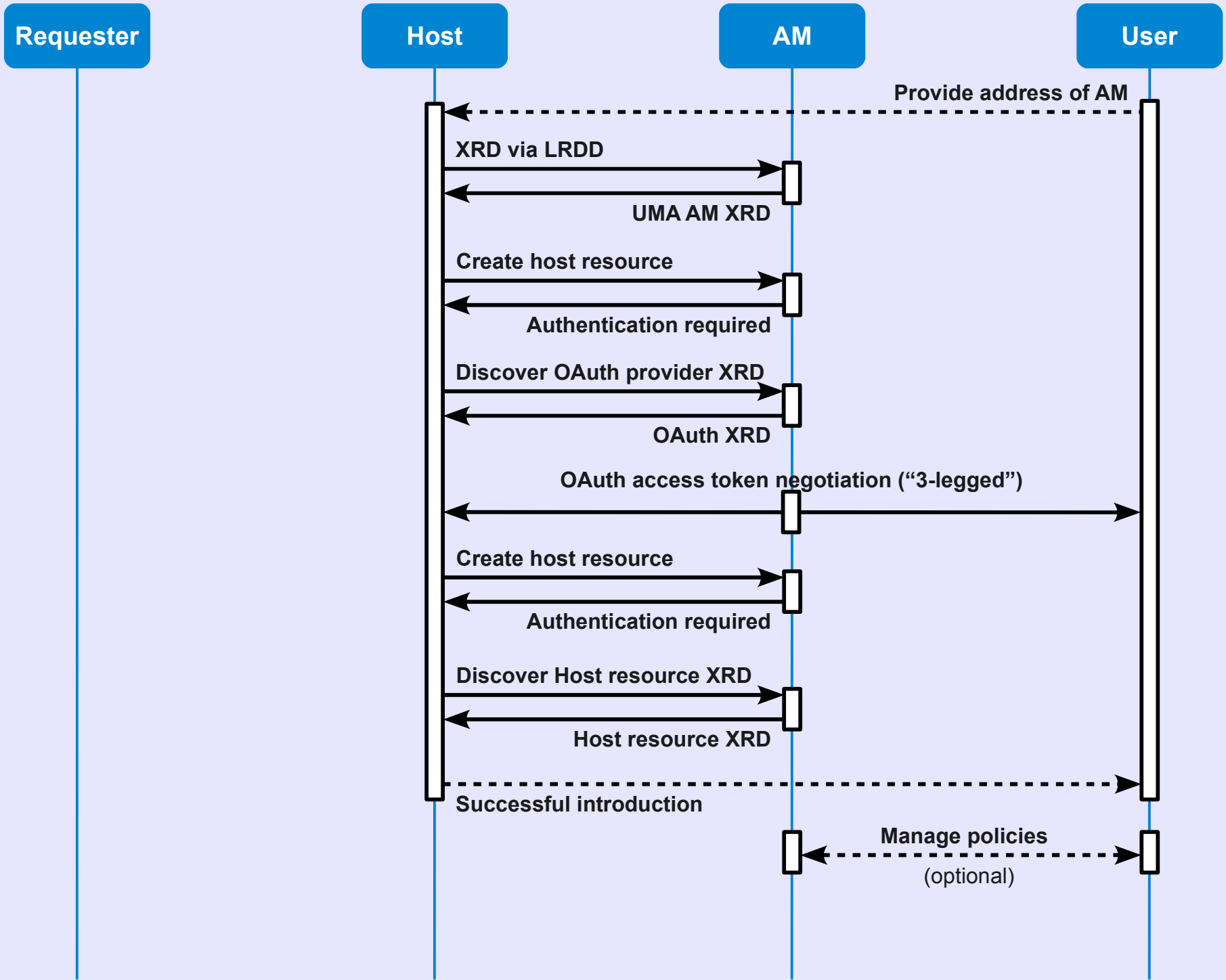


Step 1. User introduces Host to Authorization Manager

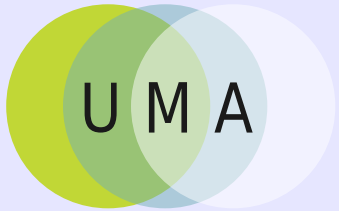
One time per user–authorization manager



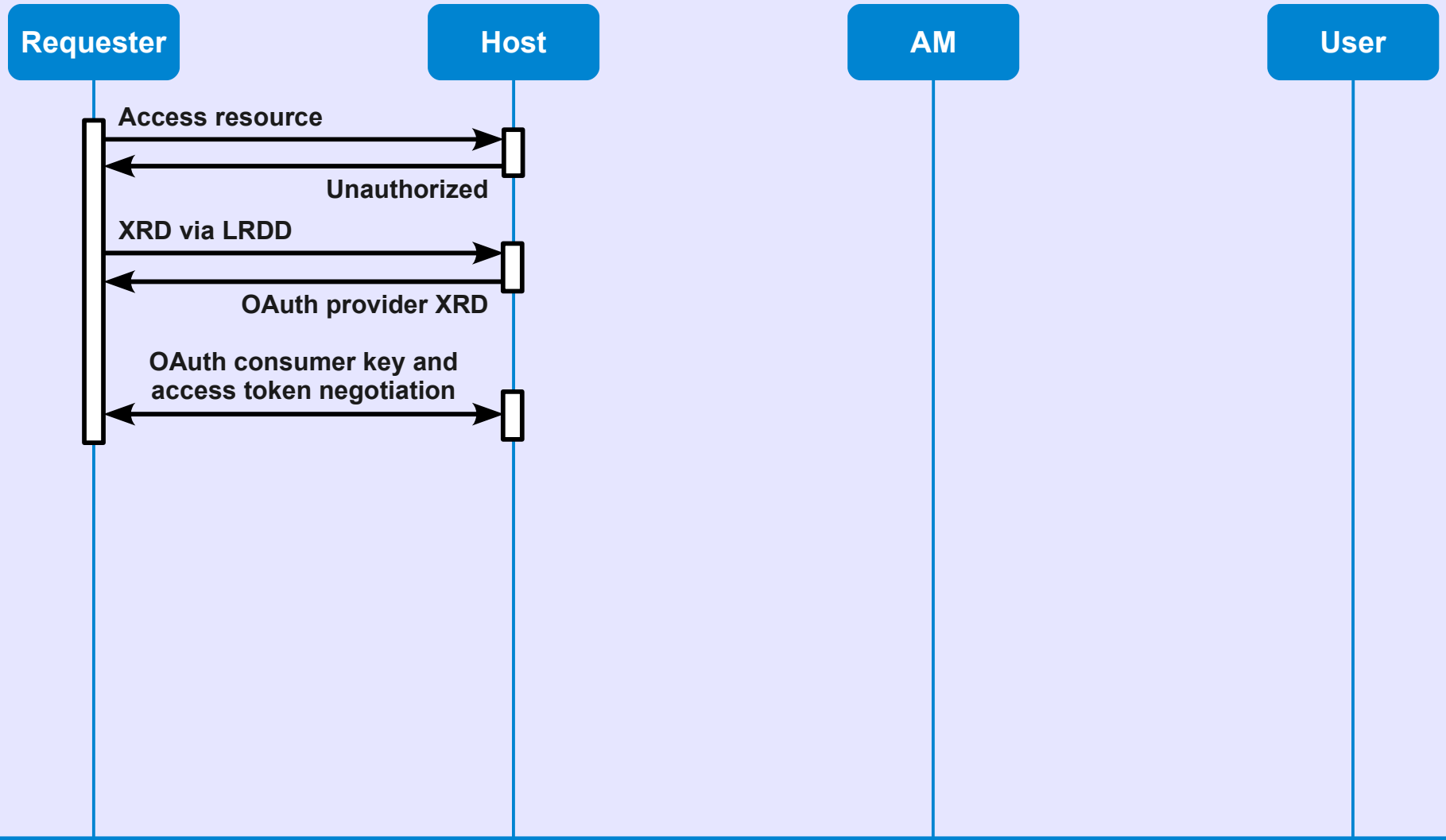
Step 1. User introduces Host to Authorization Manager



Step 2. Requester obtains Host access token



Step 2. Requester obtains Host access token

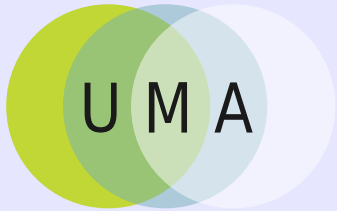


Summary

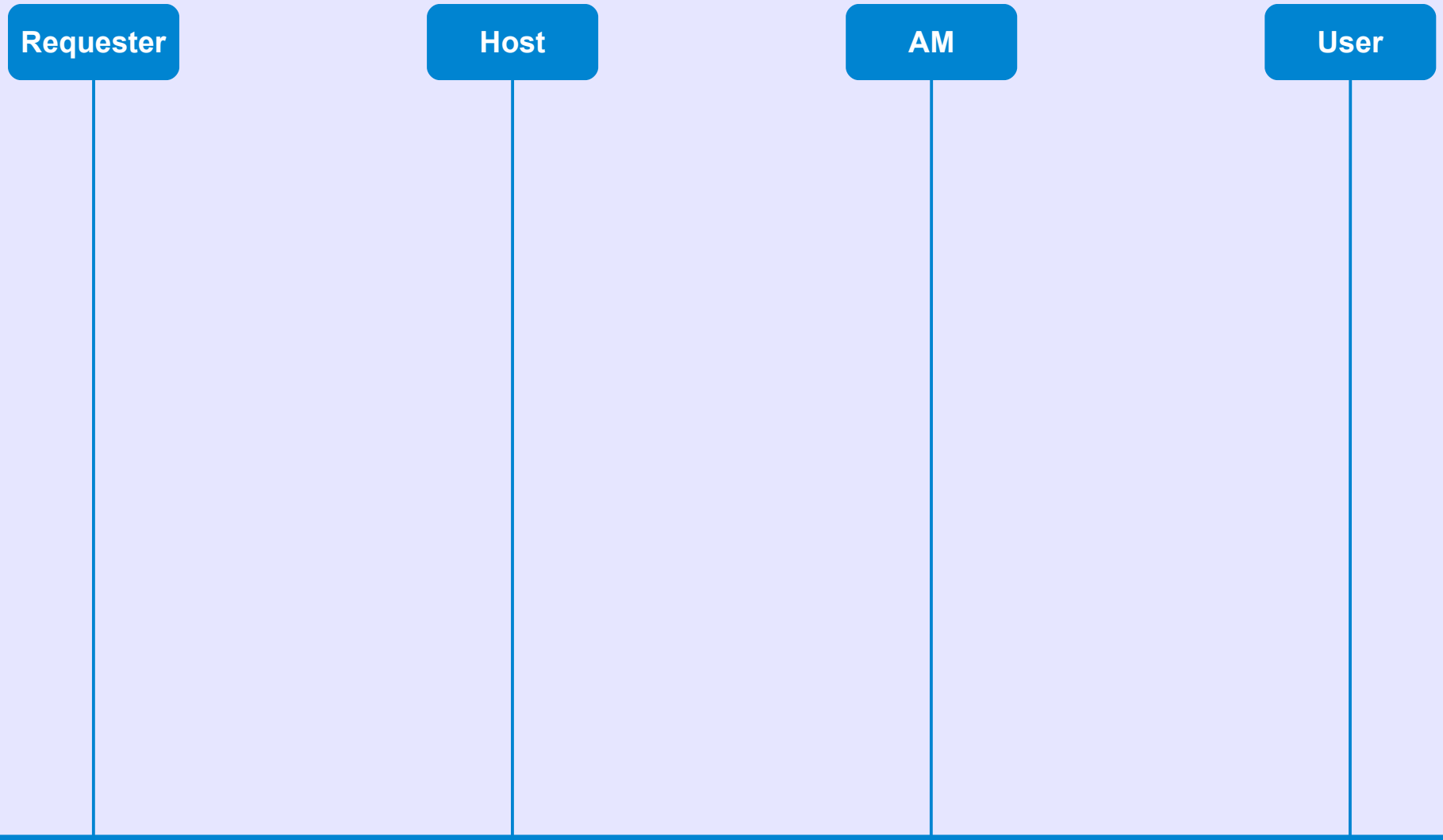
- OAuth case seen here is for illustrative purposes only.
- Host determines—and can change!—its own its authentication methods.
- Host allocates and manages its own identifiers.

Step 3a. Host refers Requester to Authorization Manager

One-time per requester per user



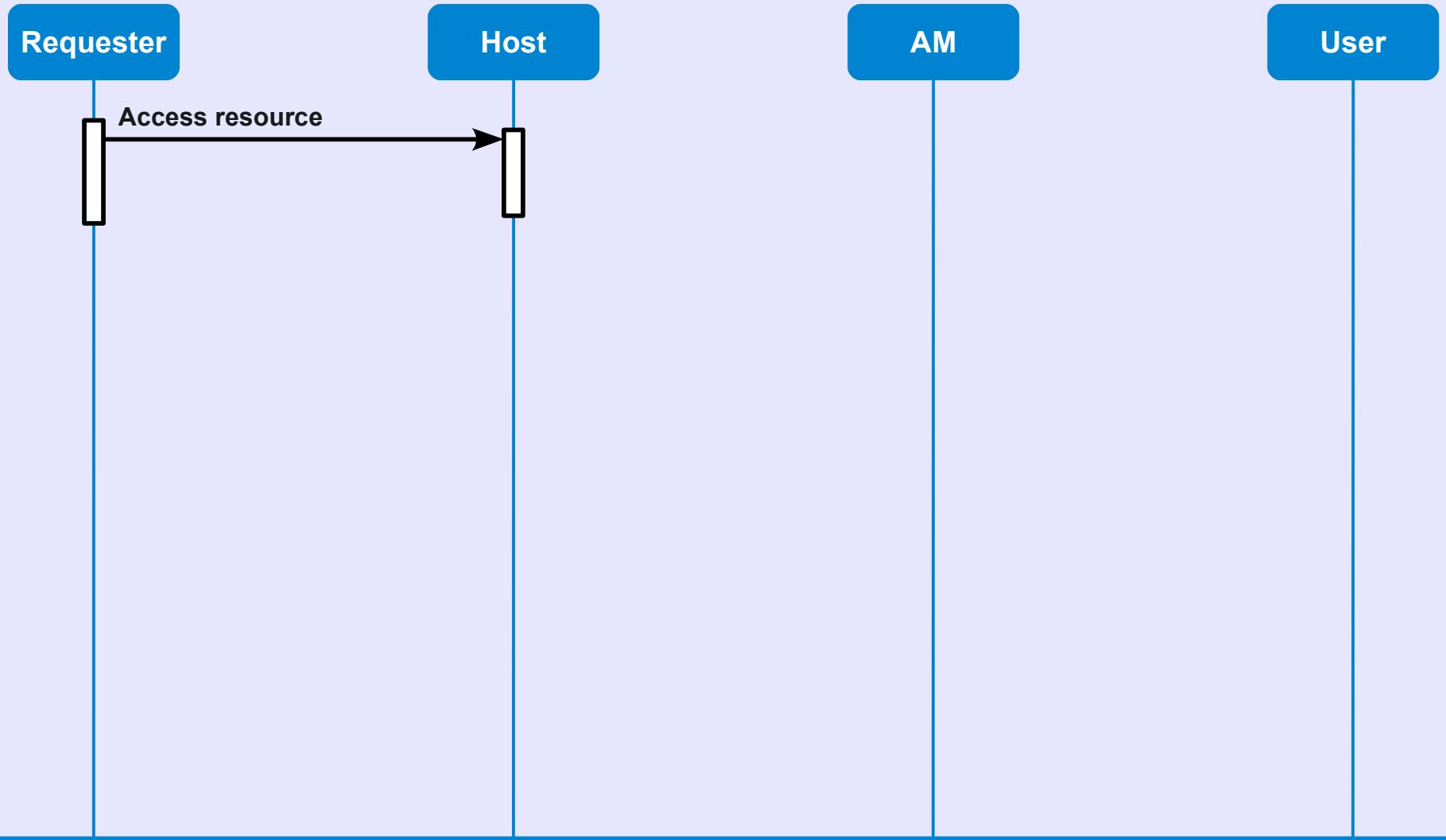
Step 3a. Host refers Requester to Authorization Manager



Background

- Requester has established some access token with host.
- Requester wants to access a resource on host.

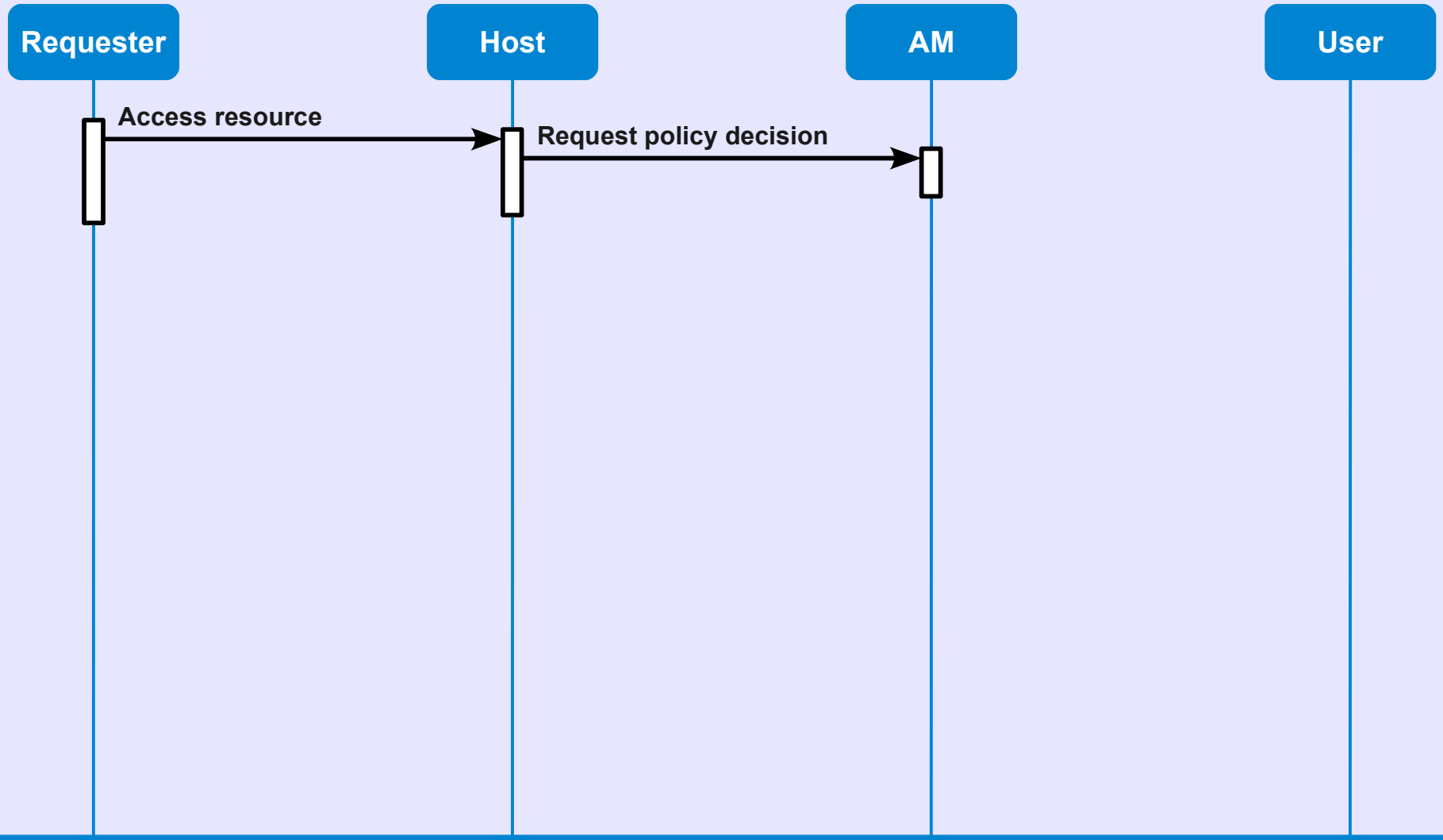
Step 3a. Host refers Requester to Authorization Manager



HTTP request from Requester to Host (schedewl.com:80)

```
GET /calendar/ical/alice/public/travel.ics
Authorization: OAuth realm="schedewl", oauth_consumer_key="86d2e3ae50f249c0",
 oauth_token="5cdd7b5c68e24908", oauth_signature_method="HMAC-SHA1", …
…
```

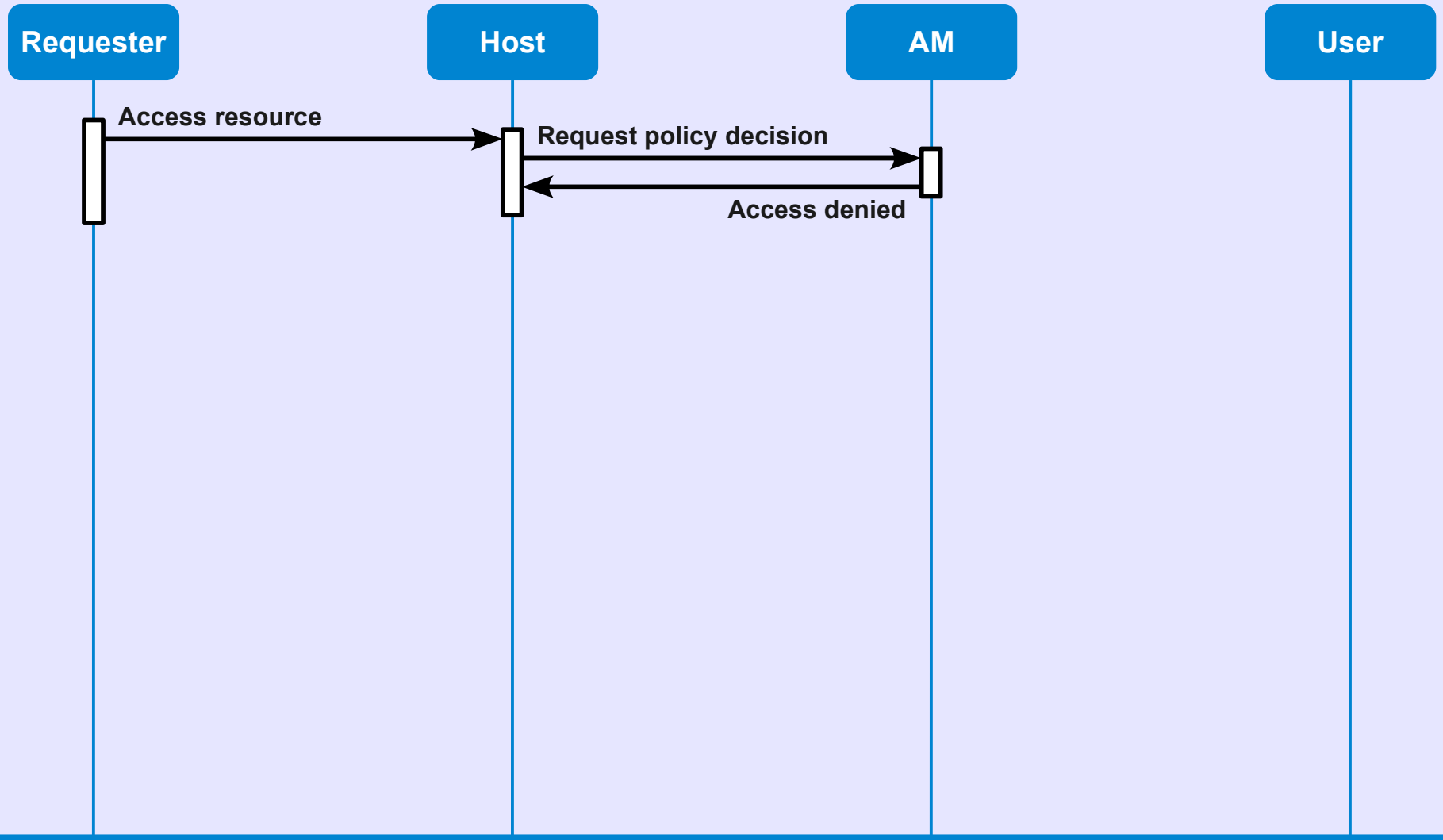
Step 3a. Host refers Requester to Authorization Manager



HTTPS request from Host to Authorization Manager (copmonkey.com:443)

```
GET /host/75284056/decision?requester_id=5cdd7b5c68e24908&method=GET&resource=http://schedewl.com/⏎
 calendar/ical/alice/public/travel.ics
Authorization: OAuth realm="copmonkey-host", oauth_consumer_key="53032297b44847ed",
 oauth_token="2f5fa6f0613942d9", oauth_signature_method="HMAC-SHA1", …
…
```

Step 3a. Host refers Requester to Authorization Manager

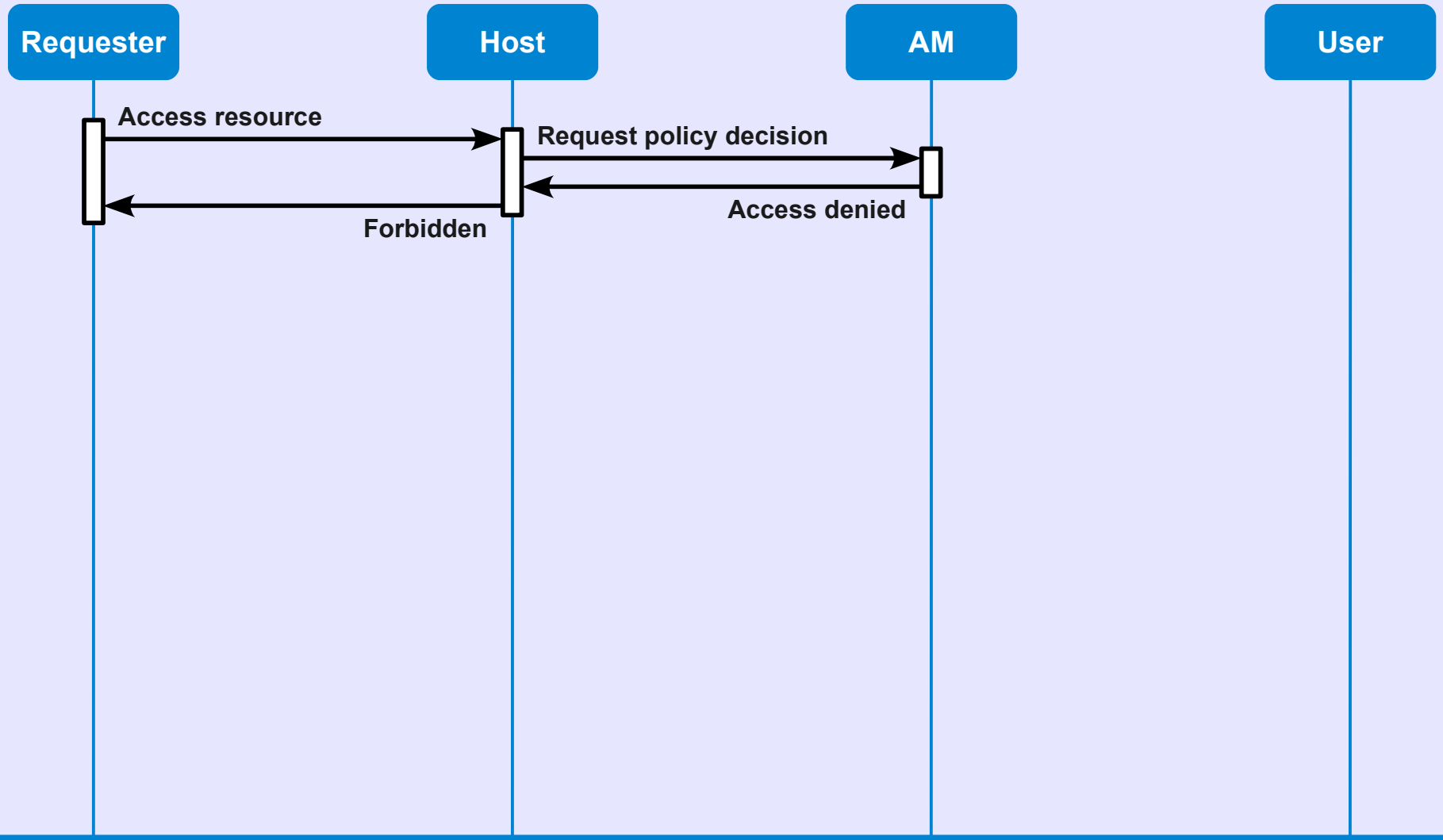


HTTPS response from Authorization Manager to Host

```
HTTP/1.1 200 OK
Content-Type: application/json
…

{"access": "denied"}
```

Step 3a. Host refers Requester to Authorization Manager

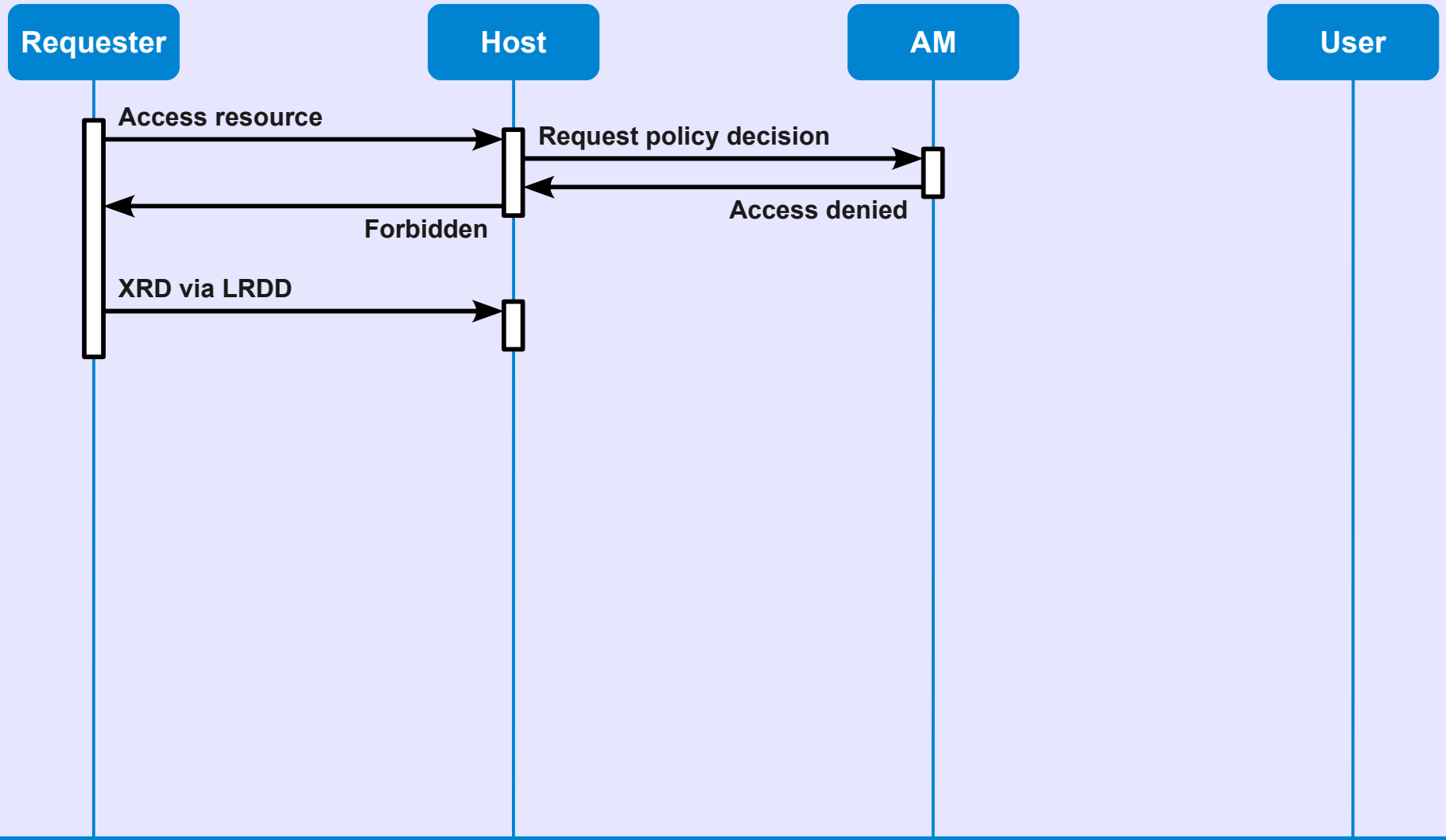


HTTPS response from Host to Requester

```
HTTP/1.1 403 Forbidden
...
```

Entity contains human-readable page describing authorization prerequisite.

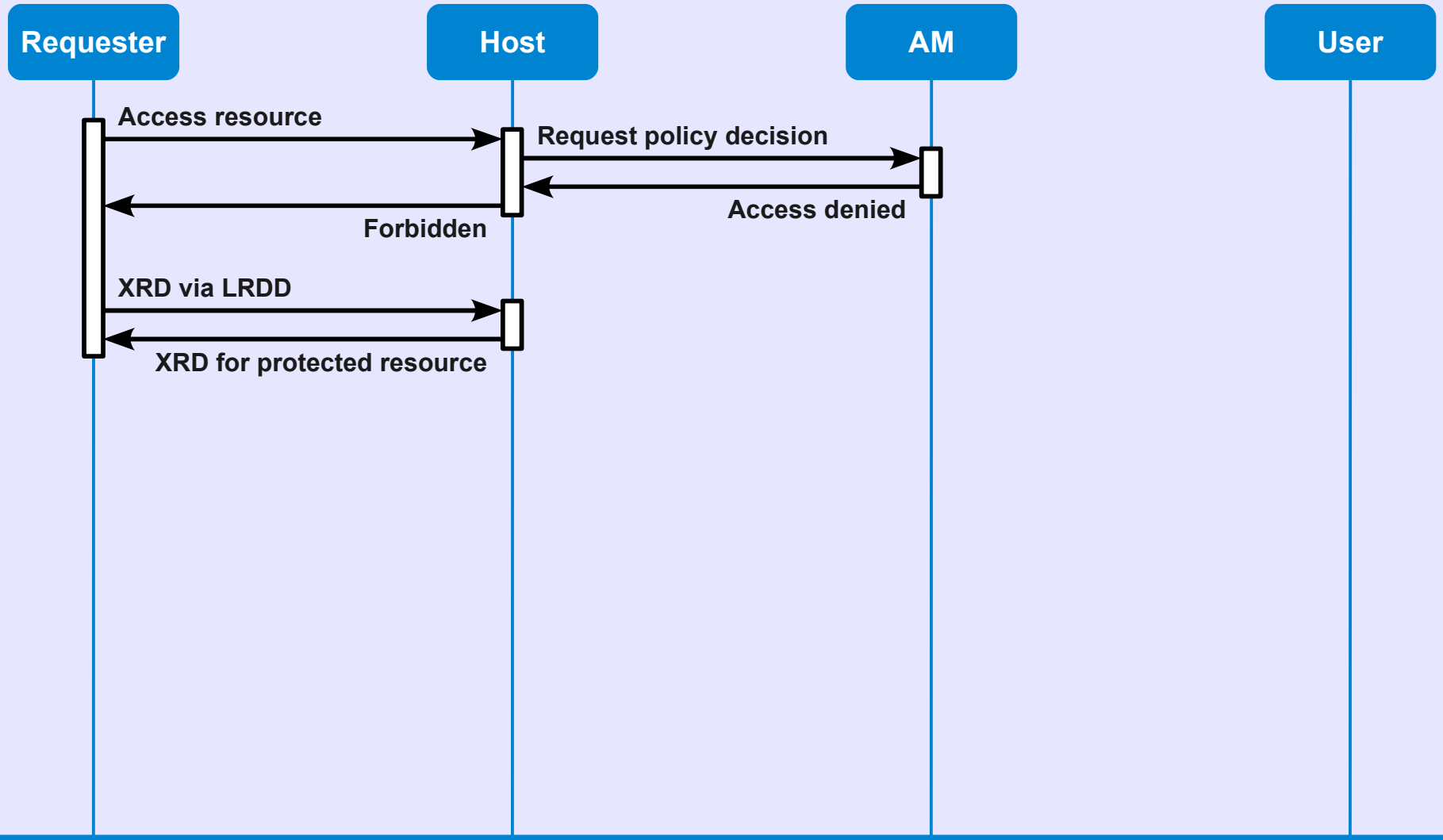
Step 3a. Host refers Requester to Authorization Manager



And now for a bit of resource descriptor discovery magic

- **Access was forbidden; no machine-readable reason is necessarily provided.**
- **Information about obtaining authorization provided through a resource's XRD.**

Step 3a. Host refers Requester to Authorization Manager

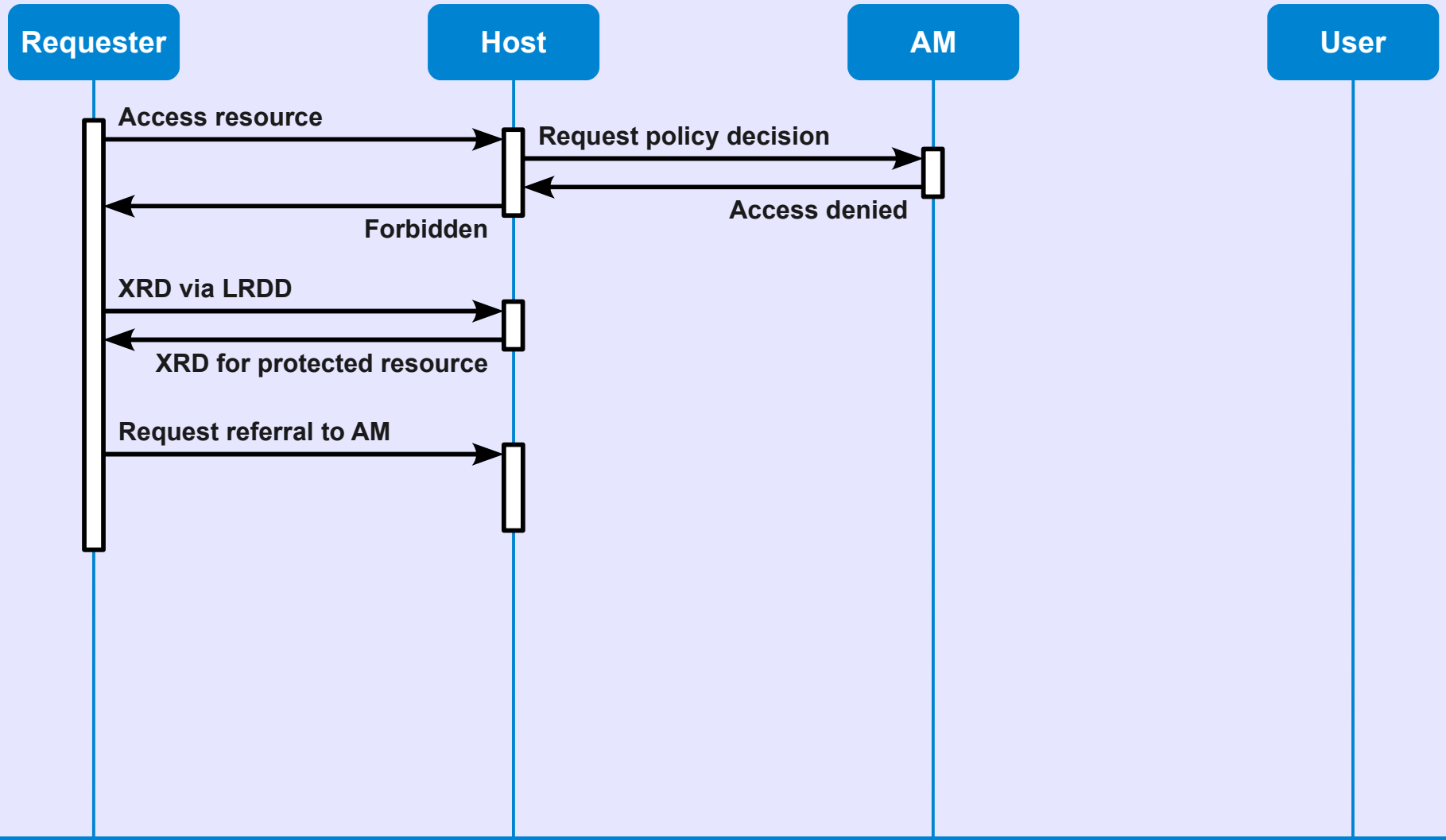


Resource descriptor for the protected resource

```
<XRD>
  ...
  <Link>
    <Rel>http://uma-wg.net/core/1.0/requester/referral/resource</Rel>
    <URI>http://schedewl.com/uma/referral</URI>
  </Link>
</XRD>
```

AM-protected resource

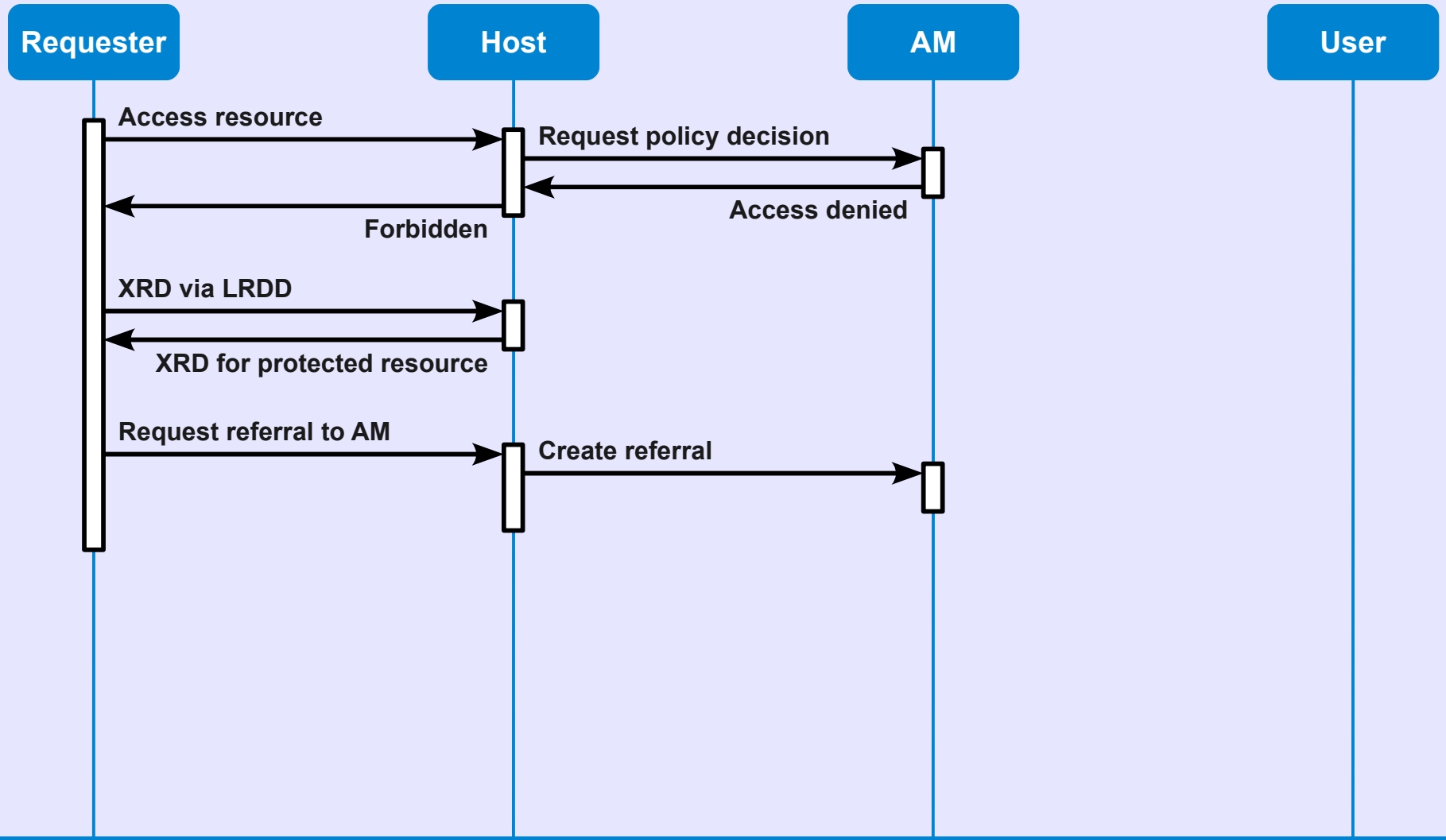
Step 3a. Host refers Requester to Authorization Manager



HTTP request from Requester to Host (schedewl.com:80)

```
POST /uma/referral
Authorization: OAuth realm="schedewl", oauth_consumer_key="86d2e3ae50f249c0",
 oauth_token="5cdd7b5c68e24908", oauth_signature_method="HMAC-SHA1", …
…
```

Step 3a. Host refers Requester to Authorization Manager

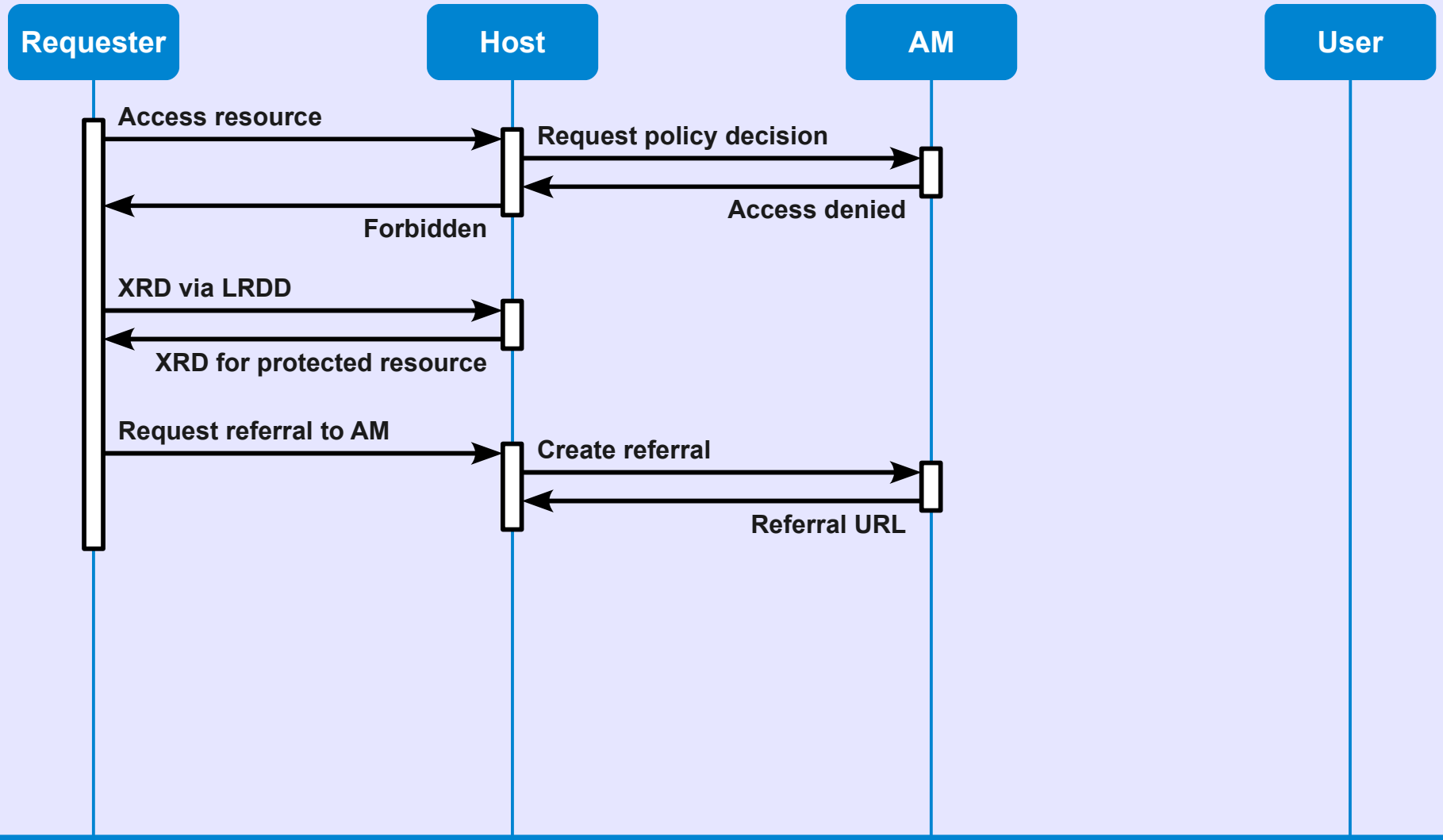


HTTPS request from Host to Authorization Manager (copmonkey.com:443)

```
POST /host/75284056/referral
Authorization: OAuth realm="copmonkey-host", oauth_consumer_key="53032297b44847ed",
 oauth_token="2f5fa6f0613942d9", oauth_signature_method="HMAC-SHA1", …
Content-Type: x-www-form-urlencoded
…

requester_id=5cdd7b5c68e24908
```

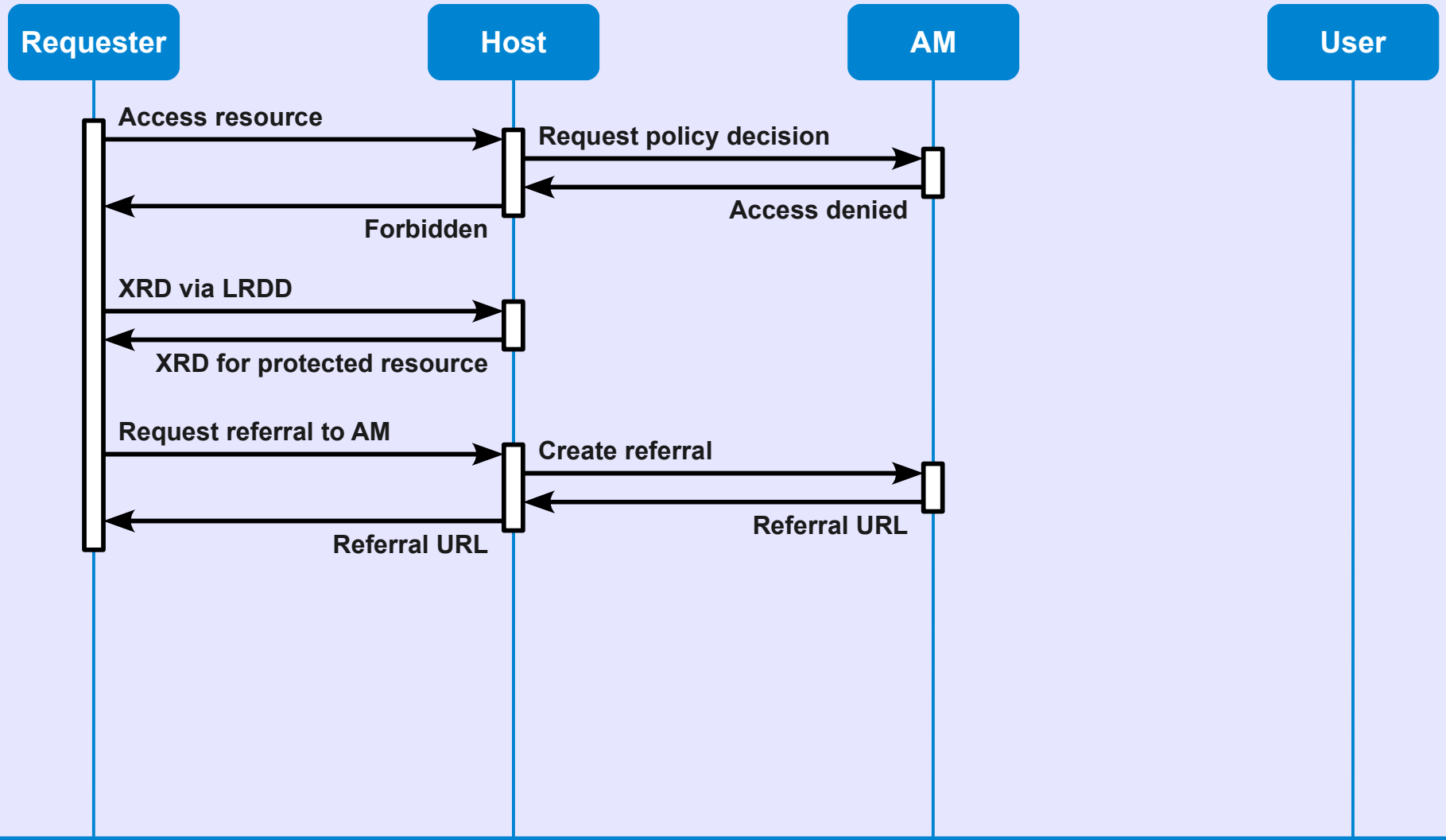
Step 3a. Host refers Requester to Authorization Manager



HTTPS response from Authorization Manager to Host

```
HTTP/1.1 201 Created
Location: https://copmonkey.com/referral/08449224
…
```

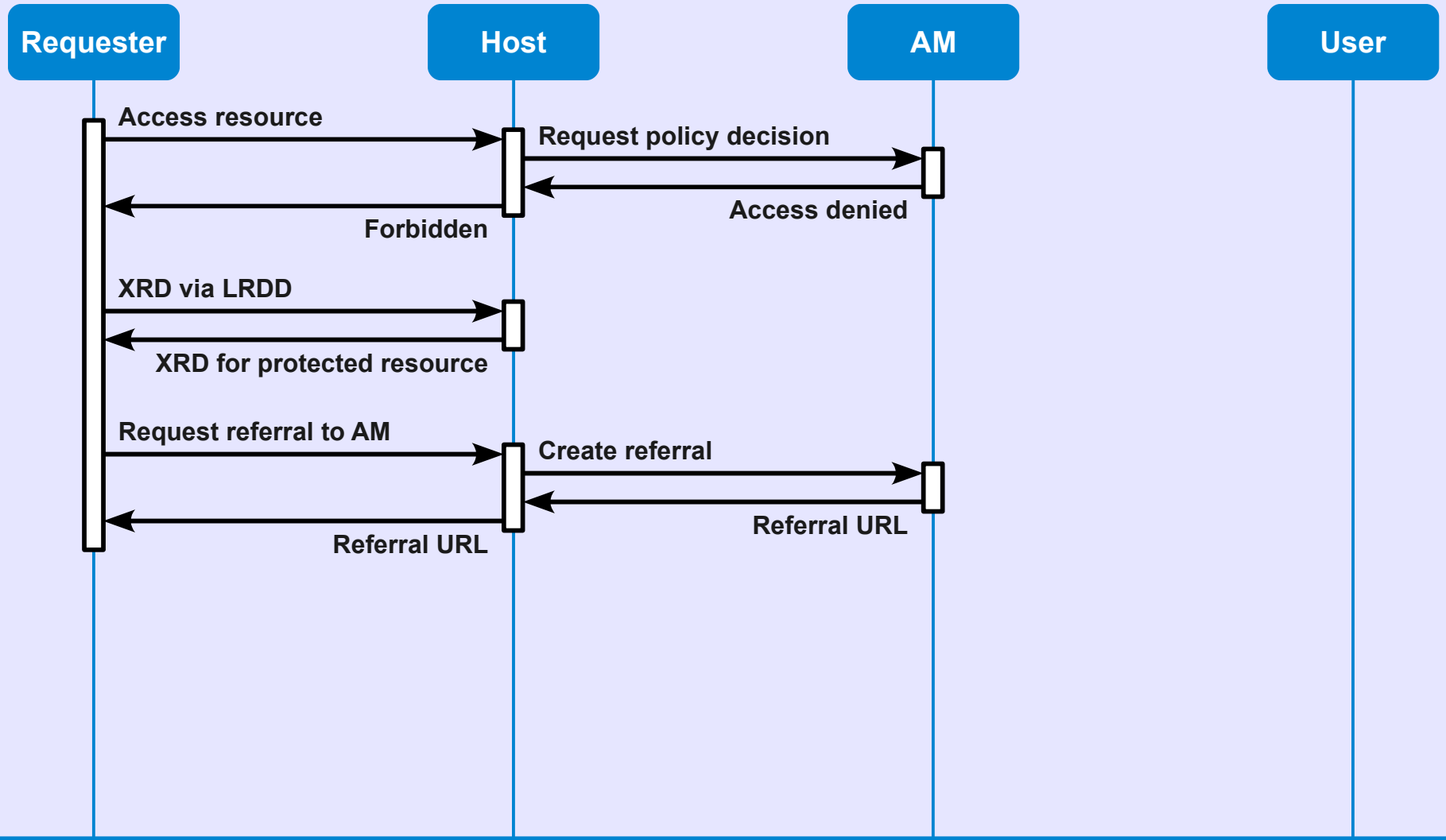
Step 3a. Host refers Requester to Authorization Manager



HTTP response from Host to Requester

```
HTTP/1.1 201 Created
Location: https://copmonkey.com/referral/08449224
…
```

Step 3a. Host refers Requester to Authorization Manager

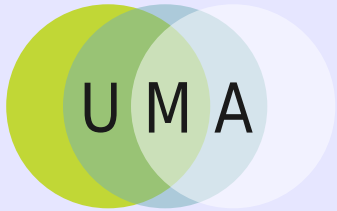


Summary

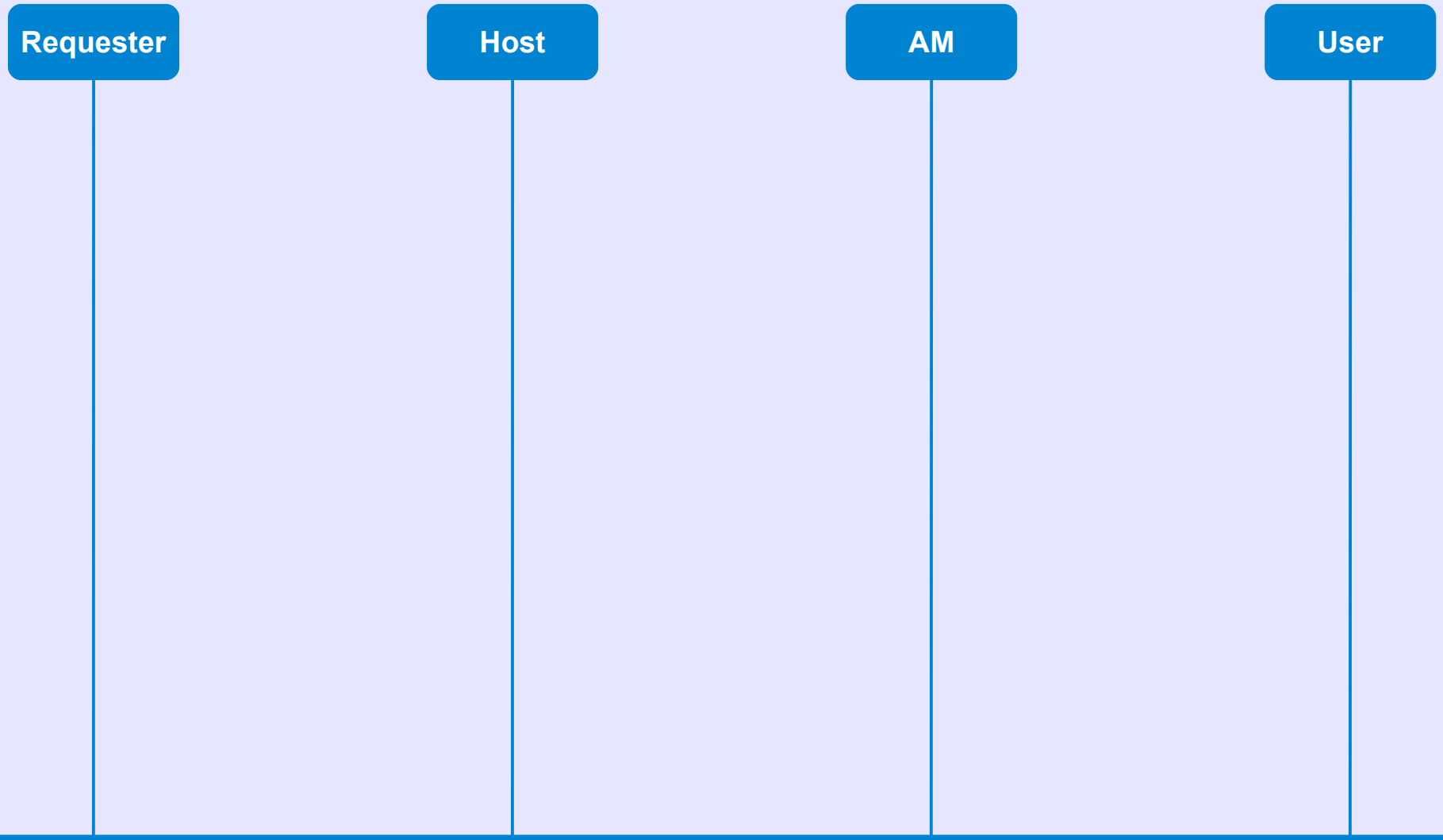
- Requester now has referral URL to establish correlation on AM.

Step 3b. Requester follows referral to Authorization Manager

One-time per requester per user



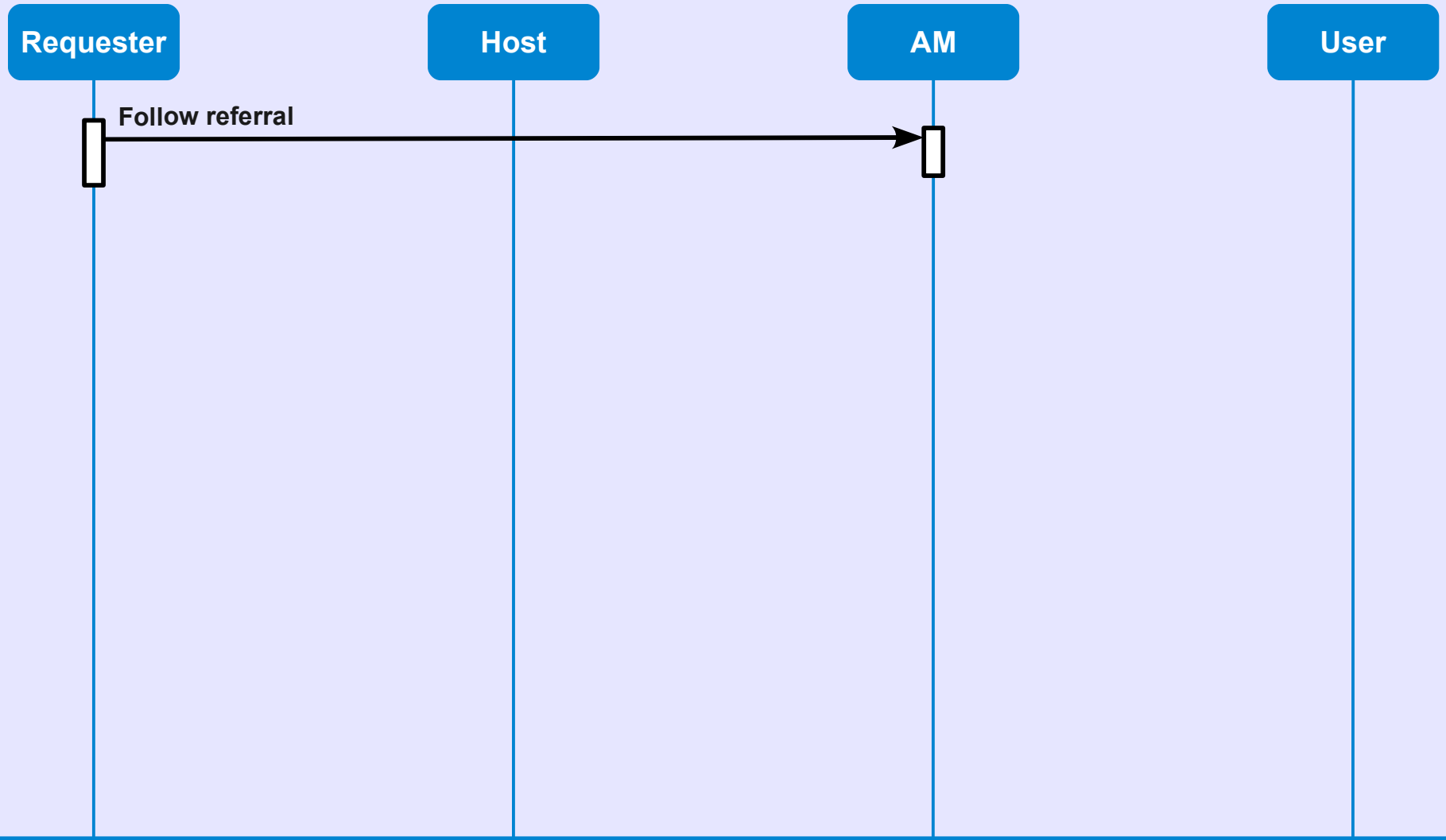
Step 3b. Requester follows referral to Authorization Manager



Background

- **Requester follows referral to begin negotiation for authorization to access protected resource.**

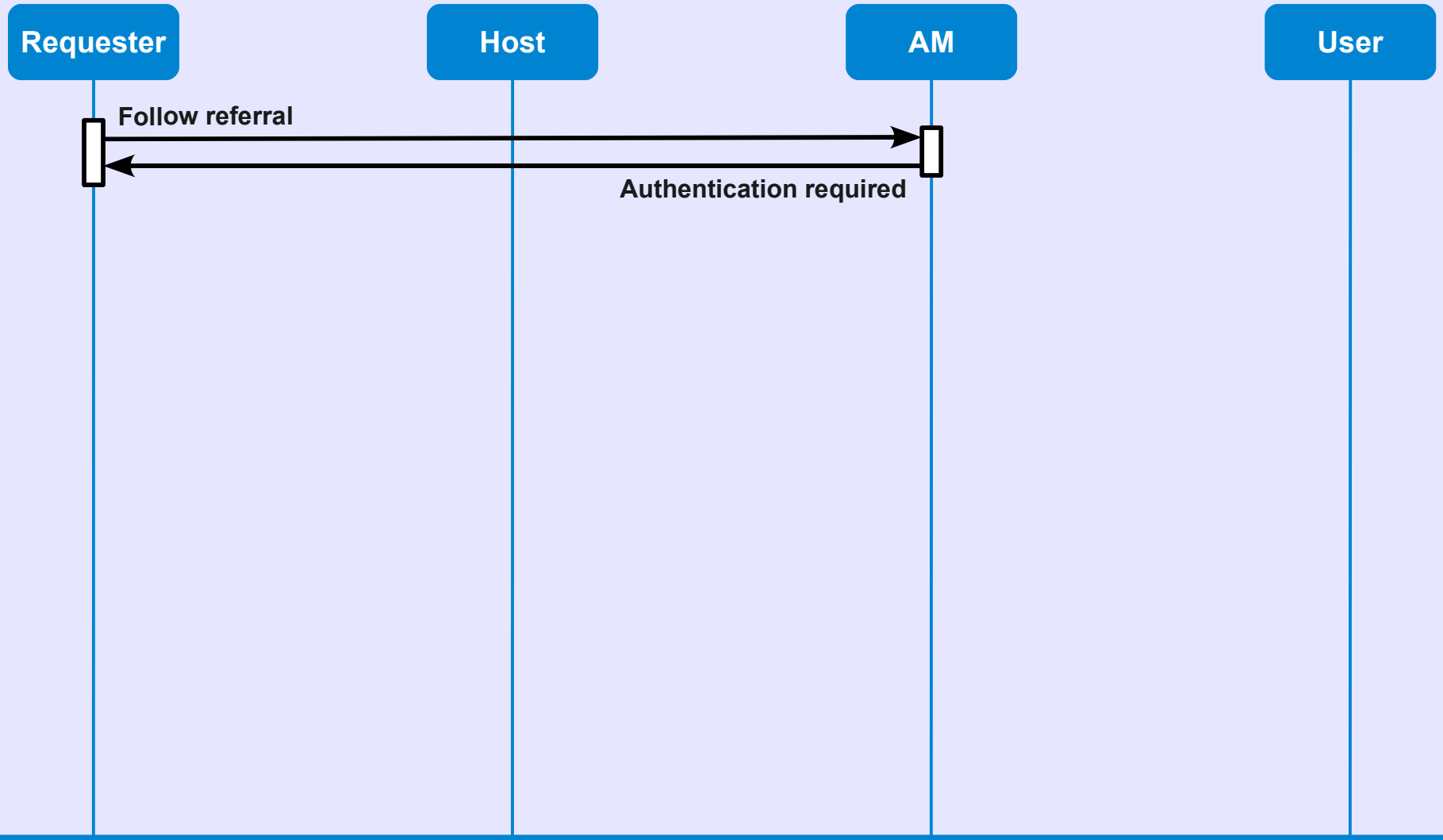
Step 3b. Requester follows referral to Authorization Manager



HTTPS request from Requester to Authorization Manager (copmonkey:443)

```
POST /referral/08449224
Content-Length: 0
…
```

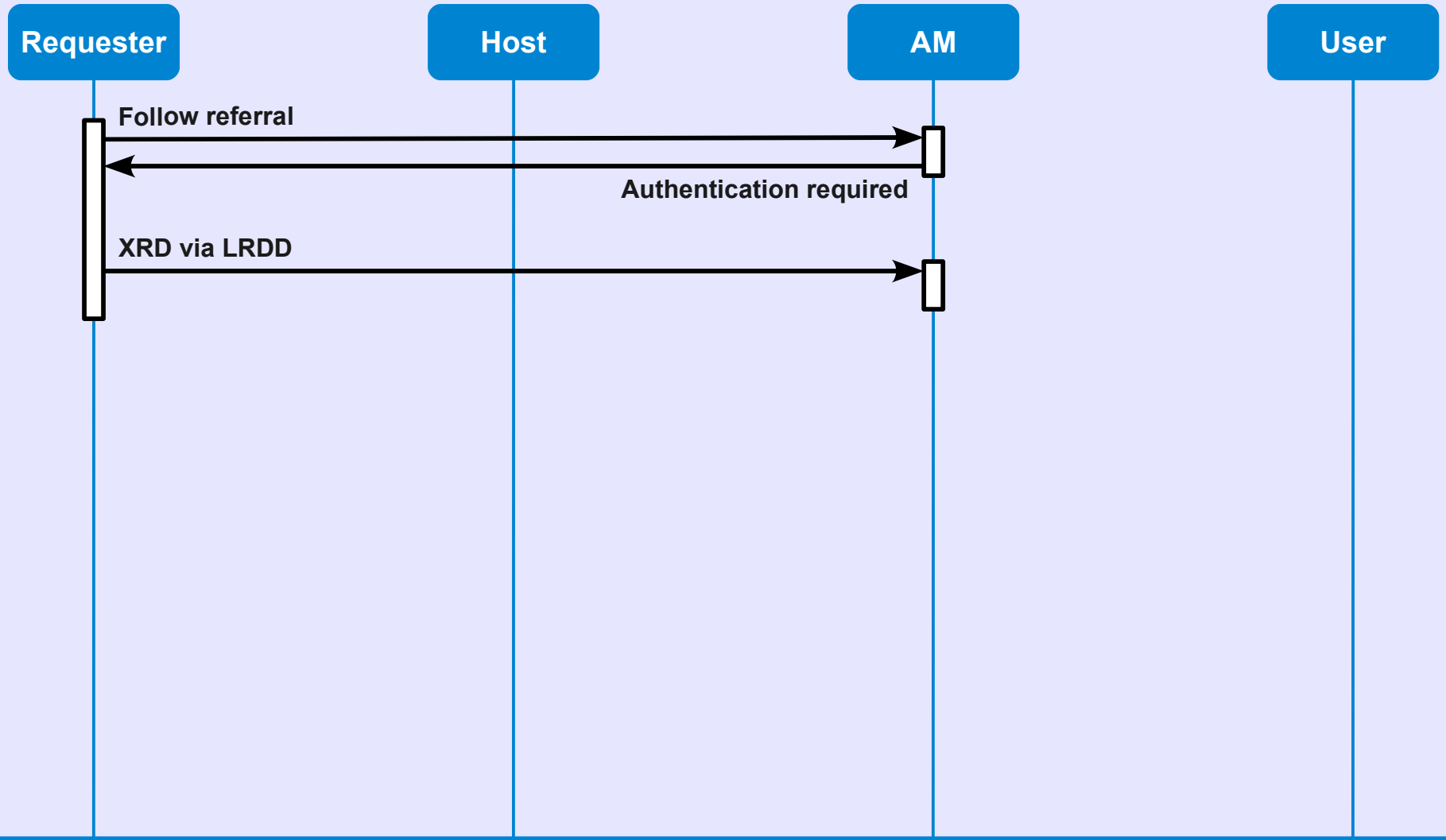
Step 3b. Requester follows referral to Authorization Manager



HTTPS response from Authorization Manager to Requester

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: OAuth realm="copmonkey-requester", provider="https://copmonkey.com/oauth/requester"
…
```

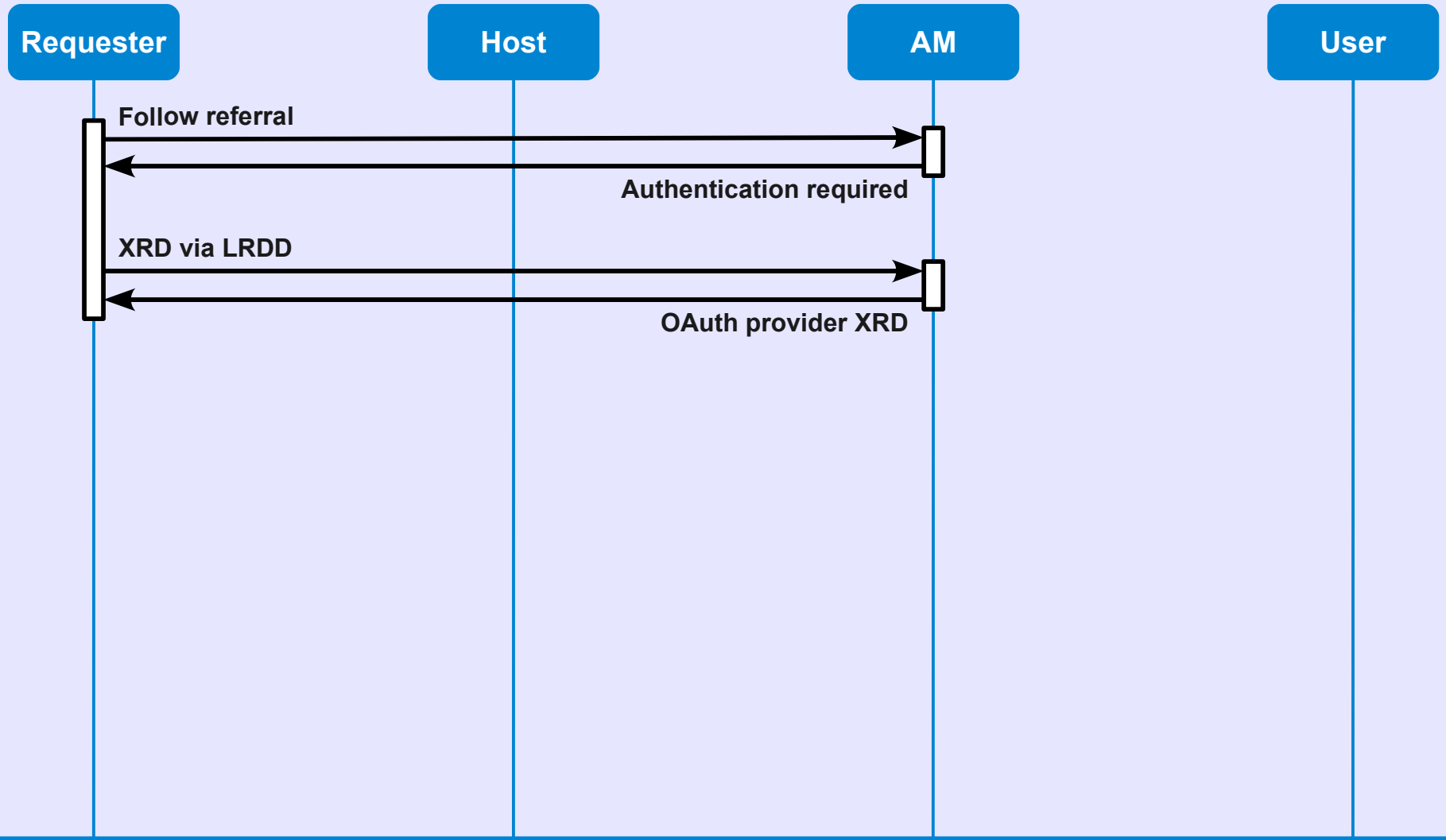
Step 3b. Requester follows referral to Authorization Manager



Discovery of OAuth provider resource to determine authentication requirements

- **Eran Hammer-Lahav's proposal for OAuth provider discovery through LRDD/XRD.**

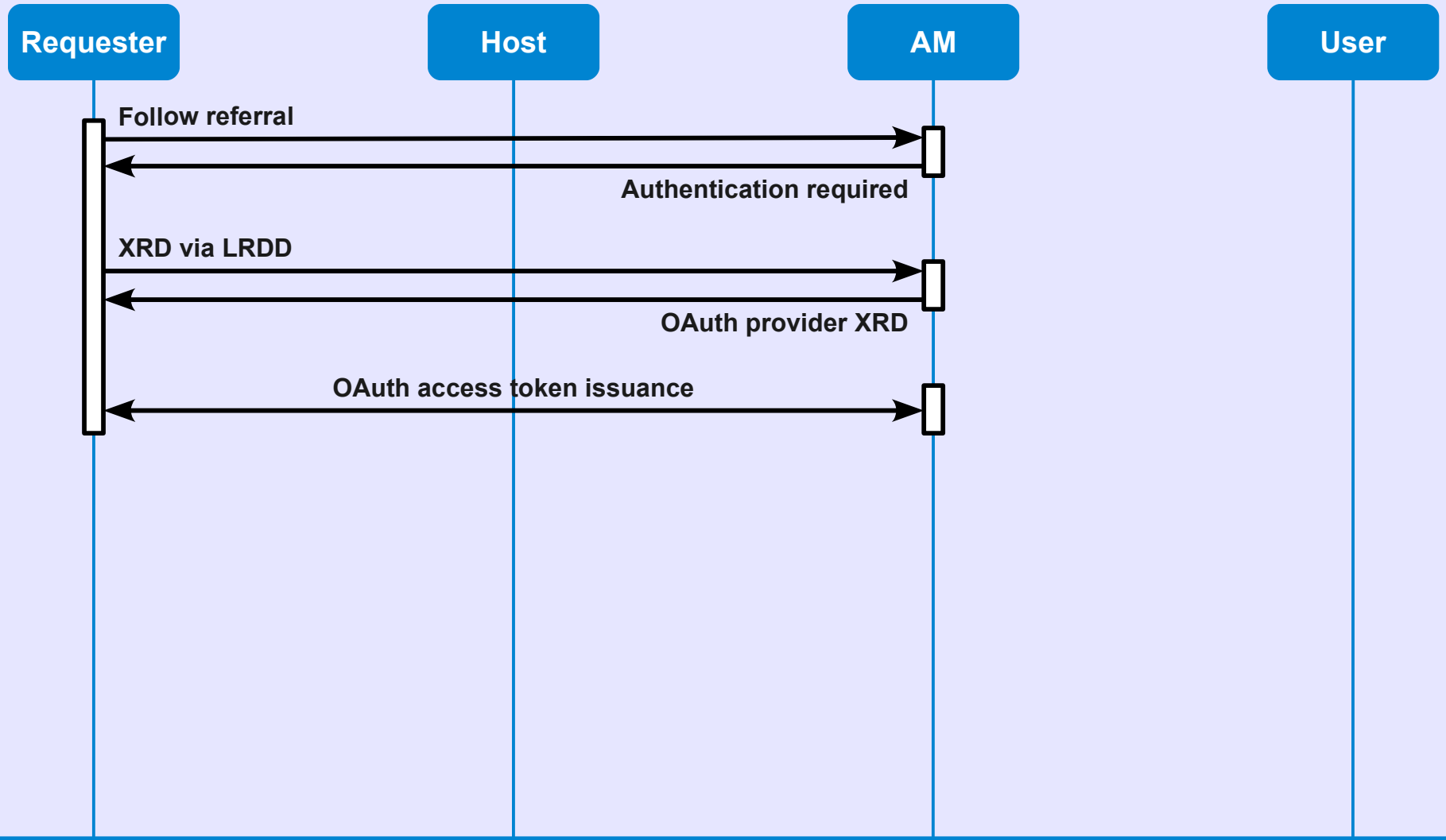
Step 3b. Requester follows referral to Authorization Manager



Resource descriptor for OAuth provider

- OAuth signature algorithm
- Static consumer key and secret to allow unregistered consumer access
- Endpoints for OAuth protocol initiation, authorization and access token issuance

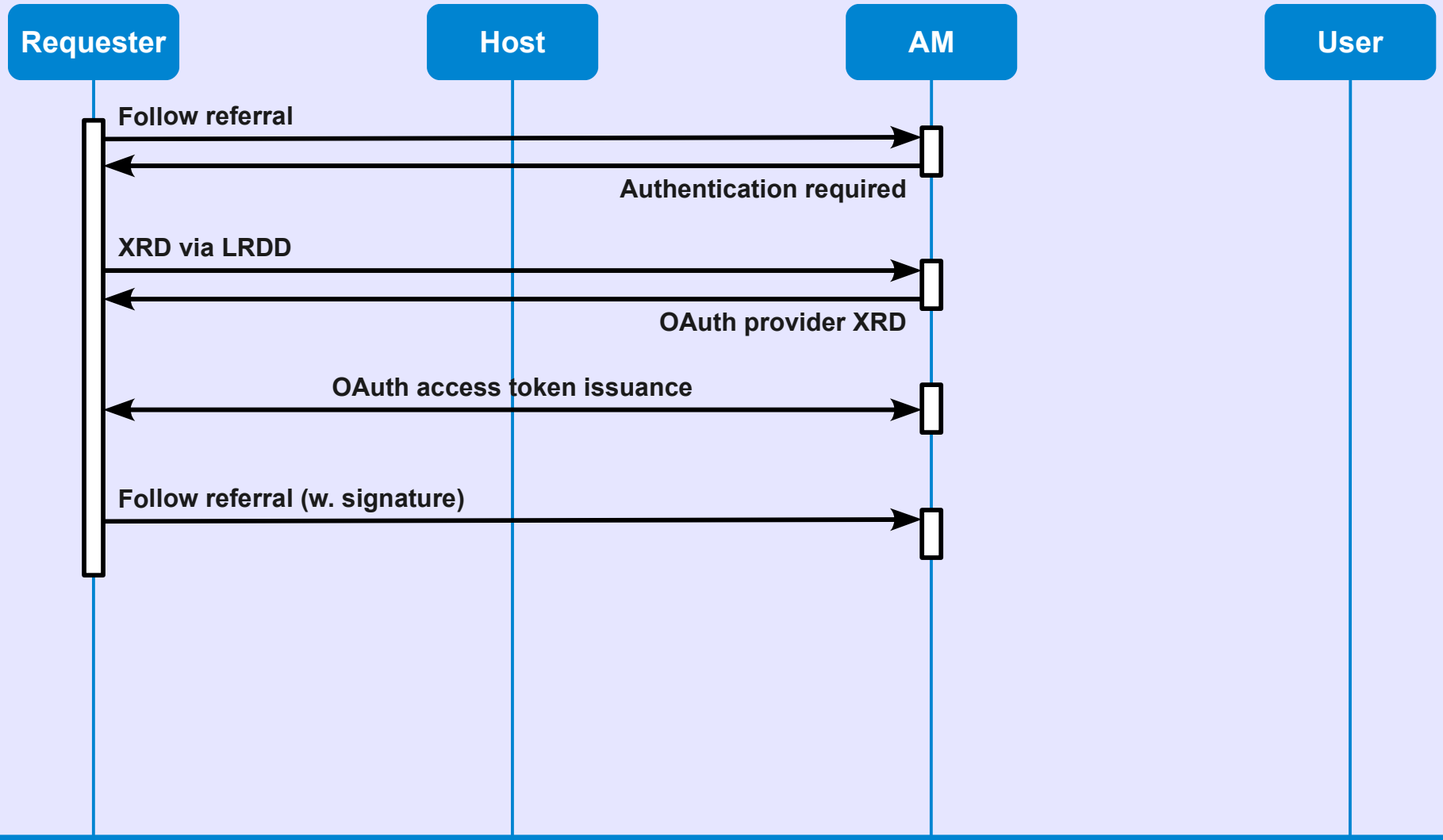
Step 3b. Requester follows referral to Authorization Manager



OAuth consumer key and access token negotiation

- **“2-legged” profile of OAuth access token issuance: request token automatically authorized**

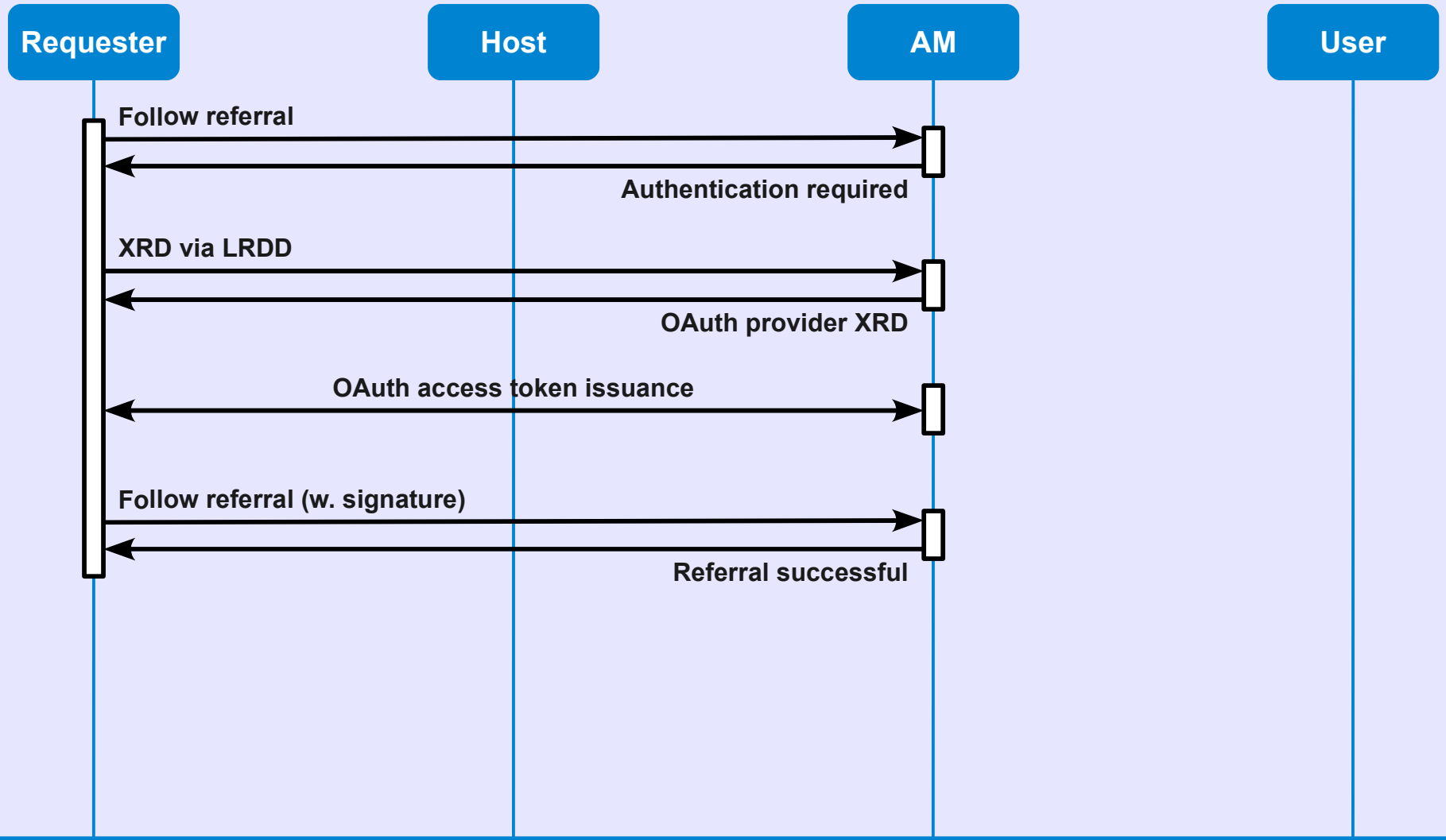
Step 3b. Requester follows referral to Authorization Manager



HTTPS request from Requester to Authorization Manager (copmonkey:443)

```
POST /referral/08449224
Authorization: OAuth realm="copmonkey-requester", oauth_consumer_key="3972c639fb72476f",
 oauth_token="4f30db8d0117464e", oauth_signature_method="HMAC-SHA1", …
Content-Length: 0
…
```

Step 3b. Requester follows referral to Authorization Manager

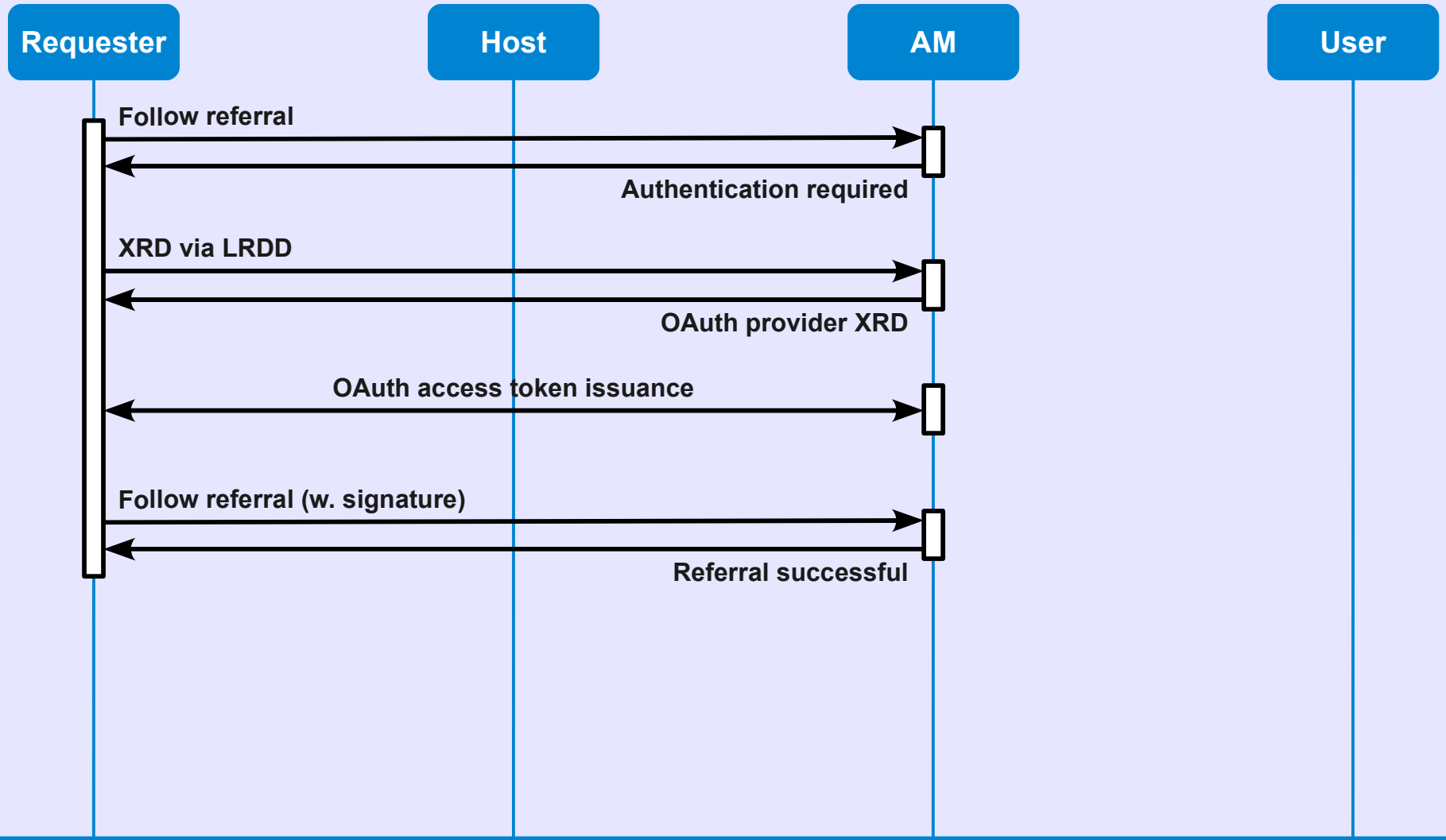


HTTPS response from Authorization Manager to Requester

```
HTTP/1.1 200 OK
...
```

Entity contains human-readable page describing success.

Step 3b. Requester follows referral to Authorization Manager

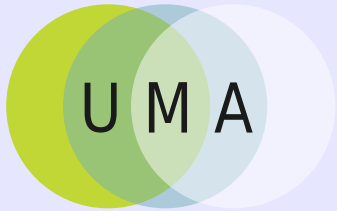


Summary

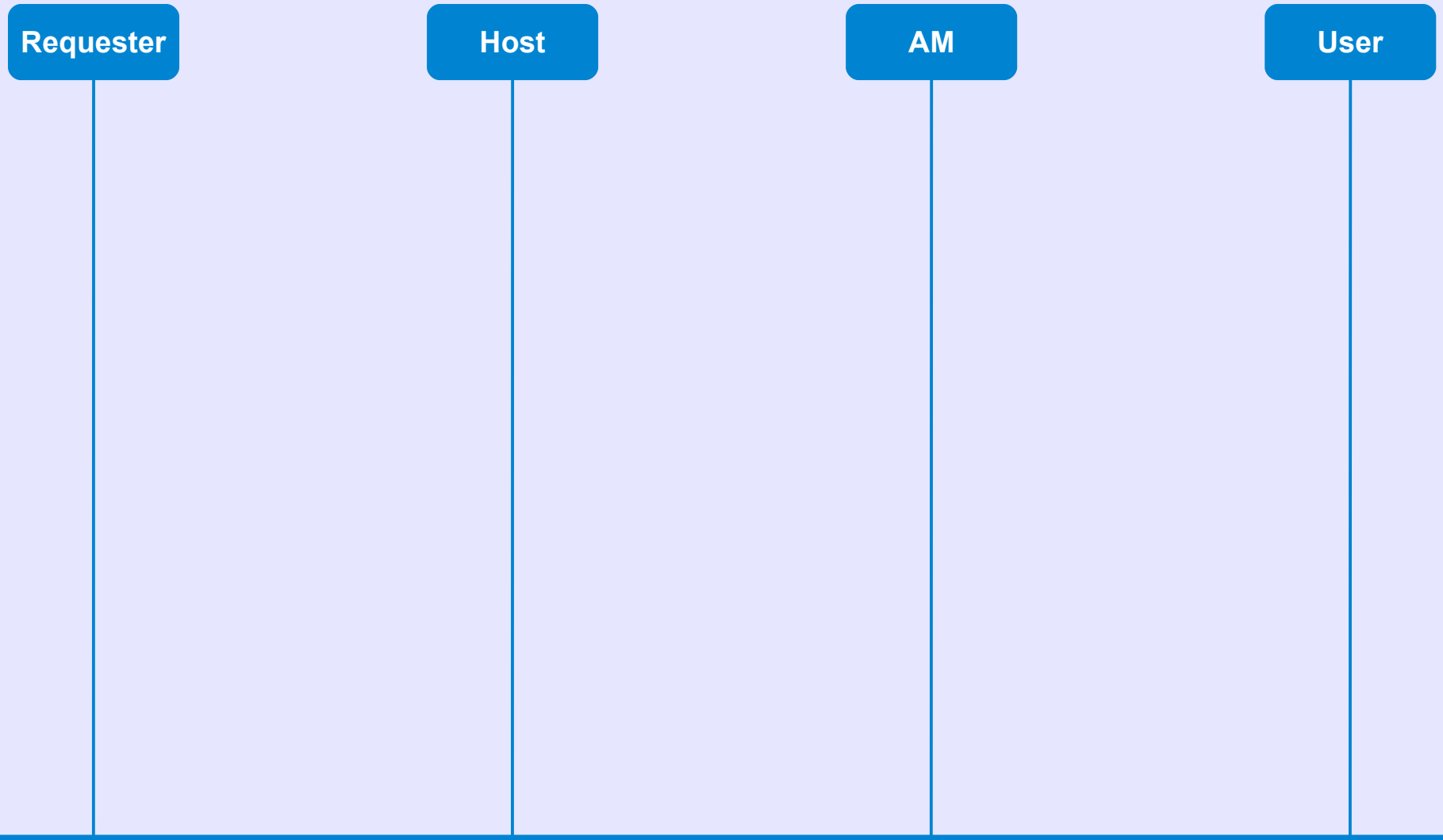
- Requester has been successfully referred to Authorization Manager by Host.
- Authorization Manager issued OAuth access token to requester.
- Requester's identifier at Host has been associated with Authorization Manager token (correlation).
- Requester can reuse its AM access token for other resources protected by the same AM.
- No authorization to access resources has yet been established.

Step 4. Requester negotiates with Authorization Manager

One-time per requester per user per authorization



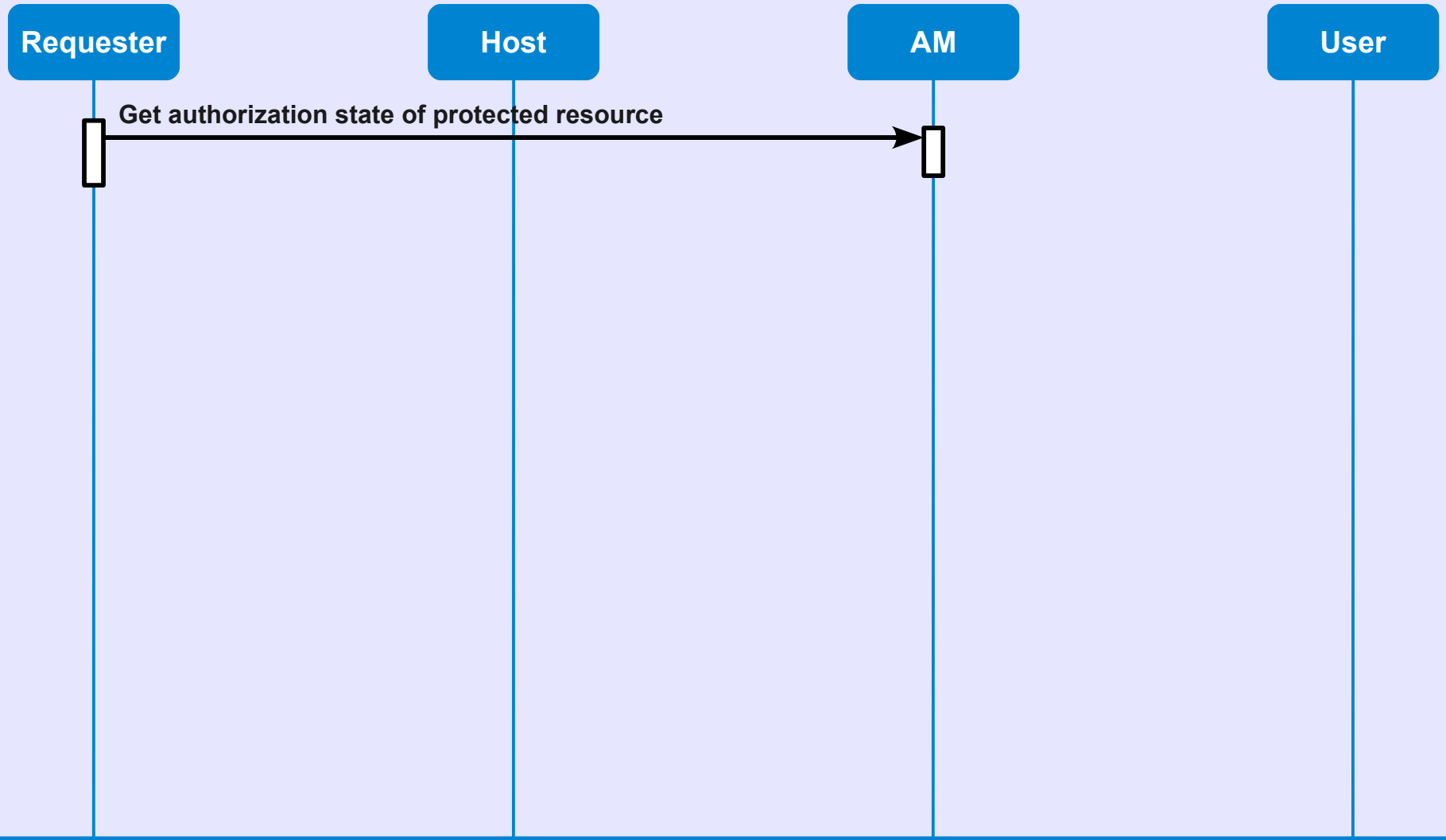
Step 4. Requester negotiates with Authorization Manager



Background

- Requester has OAuth access token on Authorization Manager.
- AM has correlated its own access token with the identifier provided by Host when referral was created.
- Requester now seeks to determine what is required to obtain authorization to access the resource.
- This is where it gets interesting!

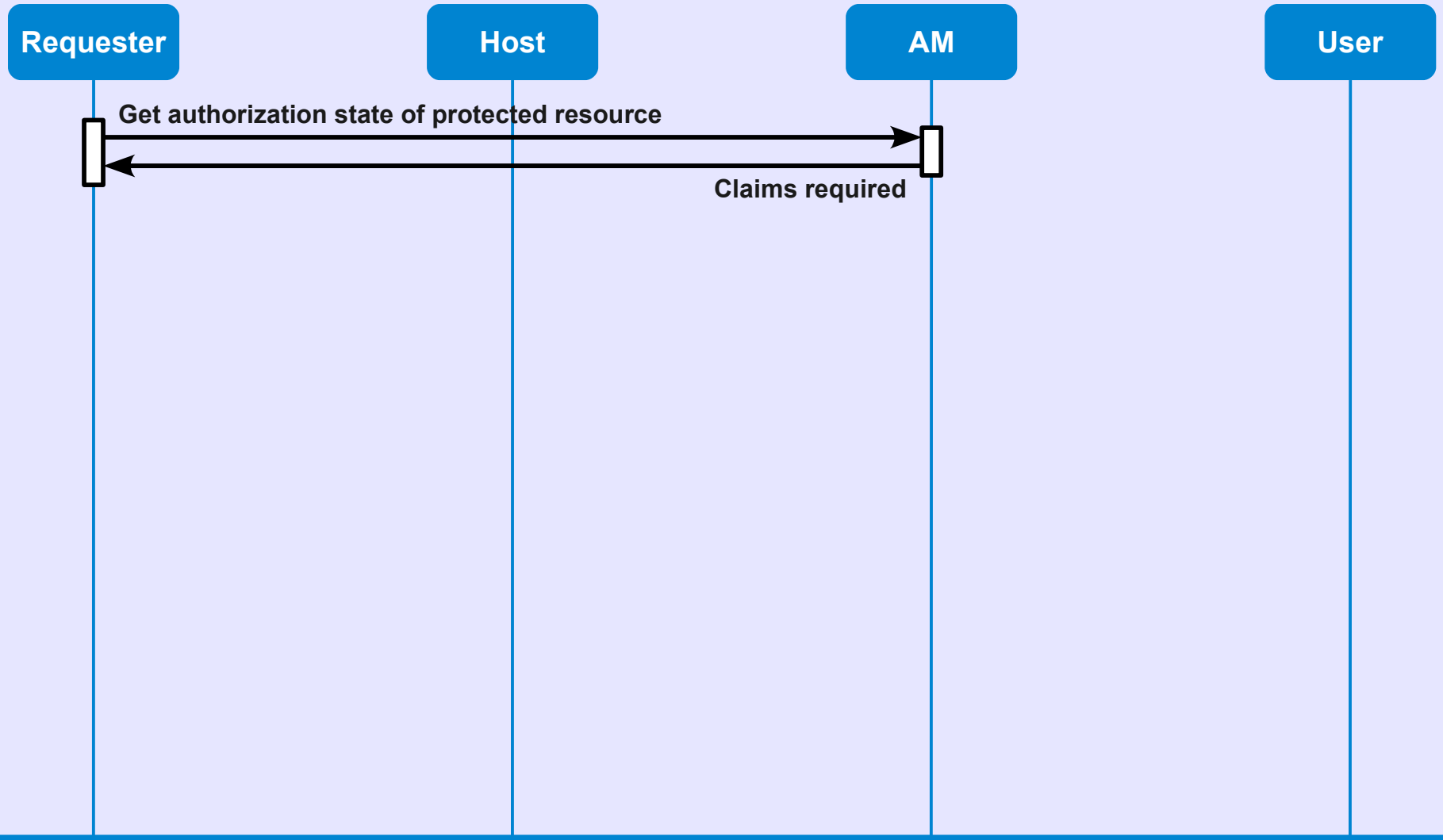
Step 4. Requester negotiates with Authorization Manager



HTTPS request from Requester to Authorization Manager (copmonkey.com:443)

```
GET /requester/authorization/status?method=GET&resource=http://schedewl.com/…/travel.ics
Authorization: OAuth realm="copmonkey-requester", oauth_consumer_key="3972c639fb72476f",
 oauth_token="4f30db8d0117464e", oauth_signature_method="HMAC-SHA1", …
…
```

Step 4. Requester negotiates with Authorization Manager

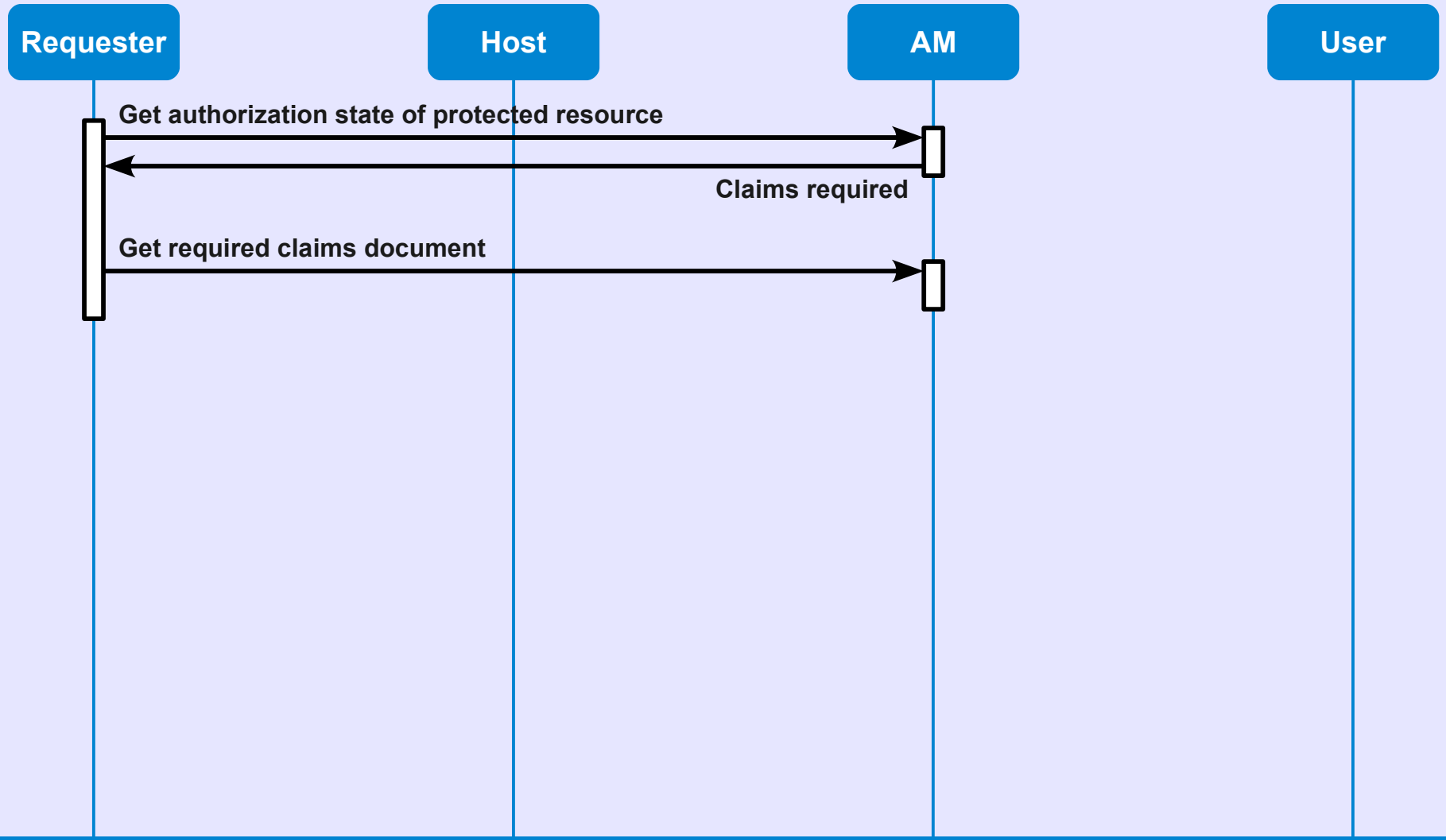


HTTPS response from Authorization Manager to Requester

```
HTTP/1.1 200 OK
Content-Type: application/json
…

{ "authorization": "claims-required", "claims-required":
  "https://copmonkey.com/requester/authorization/claims?method=GET&res=http://schedewl.com/…/travel.ics"
}
```

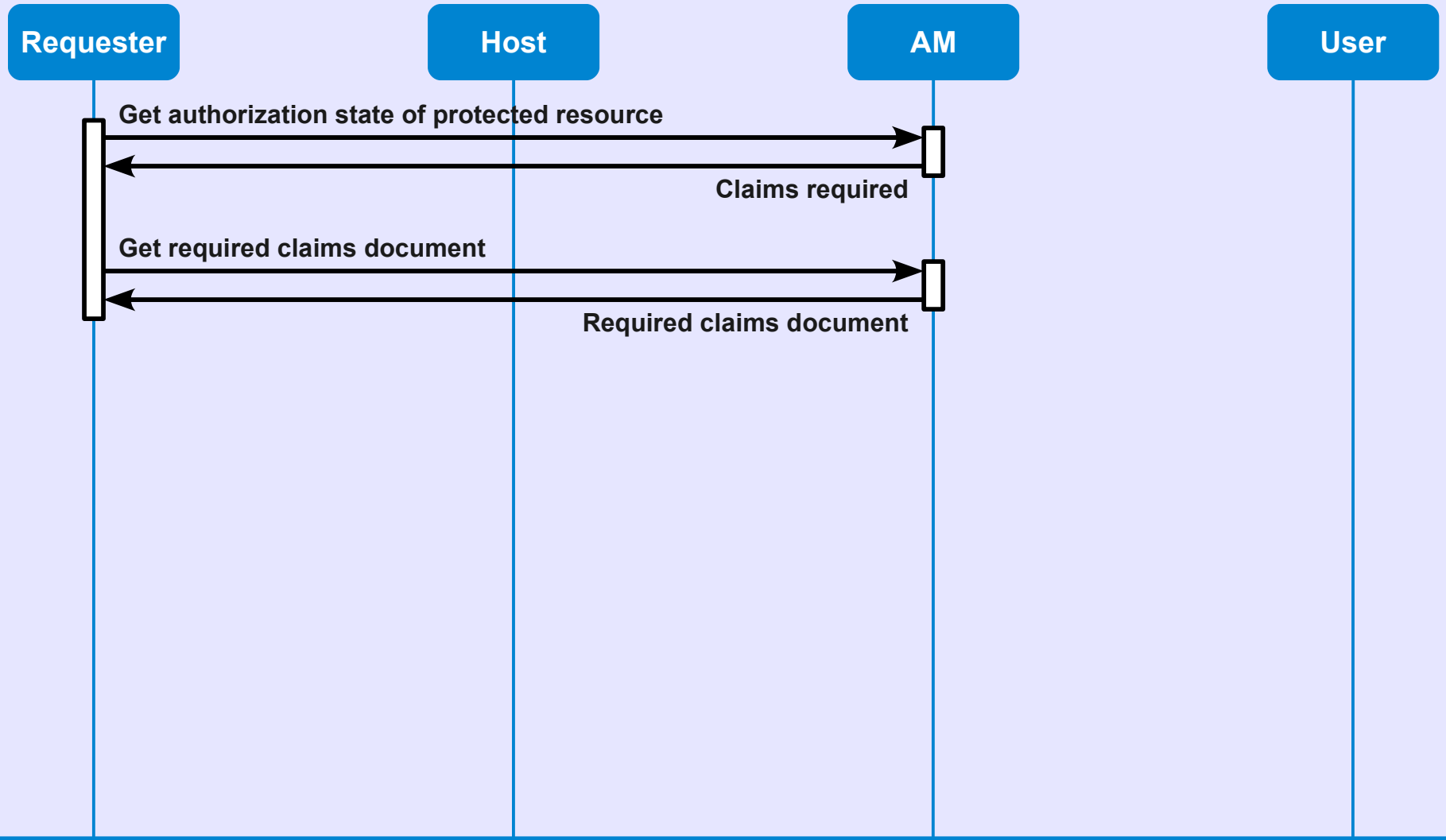
Step 4. Requester negotiates with Authorization Manager



HTTPS request from Requester to Authorization Manager (copmonkey.com:443)

```
GET /requester/authorization/claims?method=GET&resource=http://schedewl.com/…/travel.ics
Accept: application/x-claims-format-v2, application/x-claims-format-v1;q=0.9
Authorization: OAuth realm="copmonkey-requester", oauth_consumer_key="3972c639fb72476f",
  oauth_token="4f30db8d0117464e", oauth_signature_method="HMAC-SHA1",…
…
```

Step 4. Requester negotiates with Authorization Manager

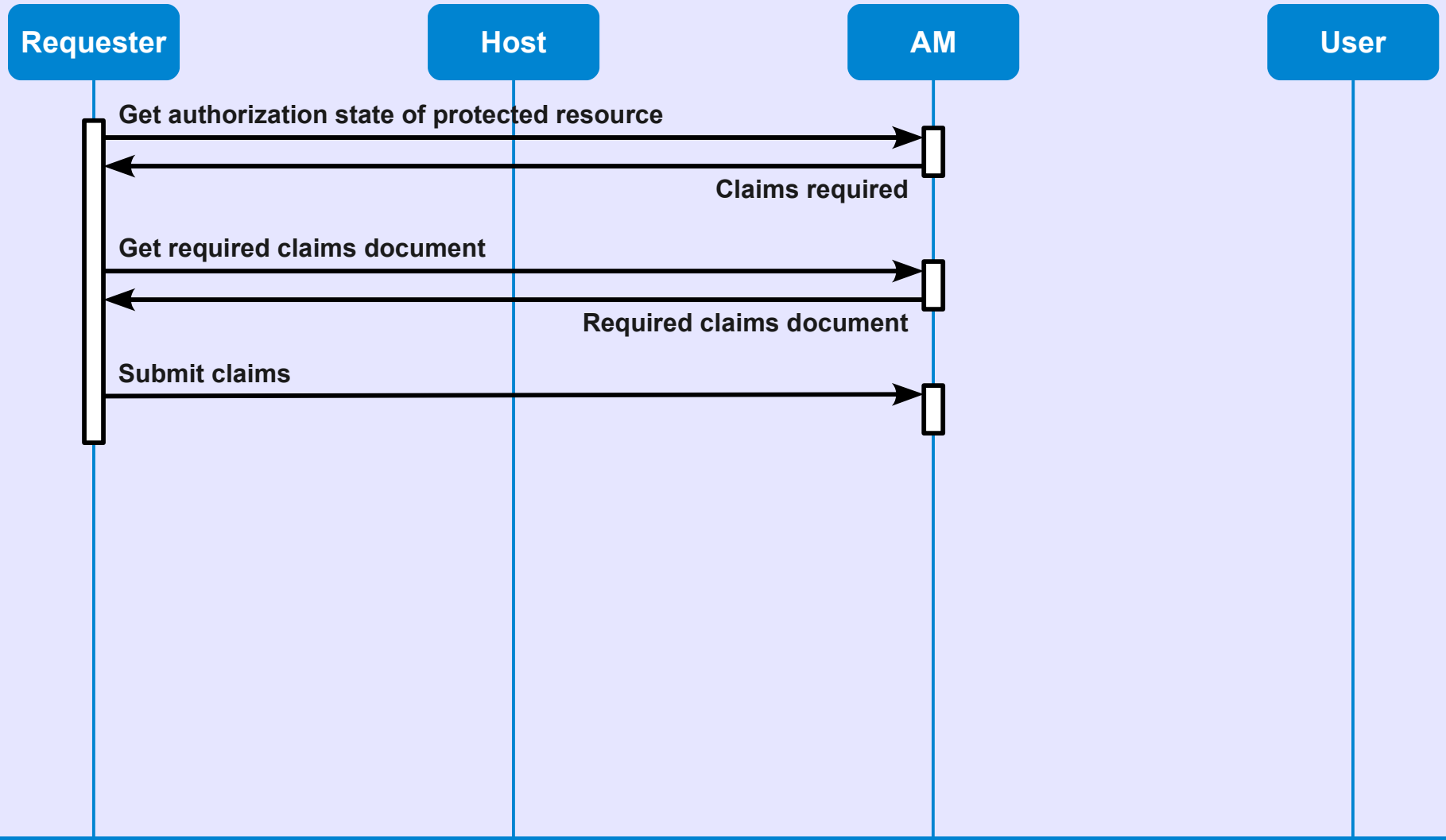


HTTPS response from Authorization Manager to Requester

```
HTTP/1.1 200 OK
Content-Type: application/x-claims-format-v2
…
```

Entity contains required-claims document.

Step 4. Requester negotiates with Authorization Manager

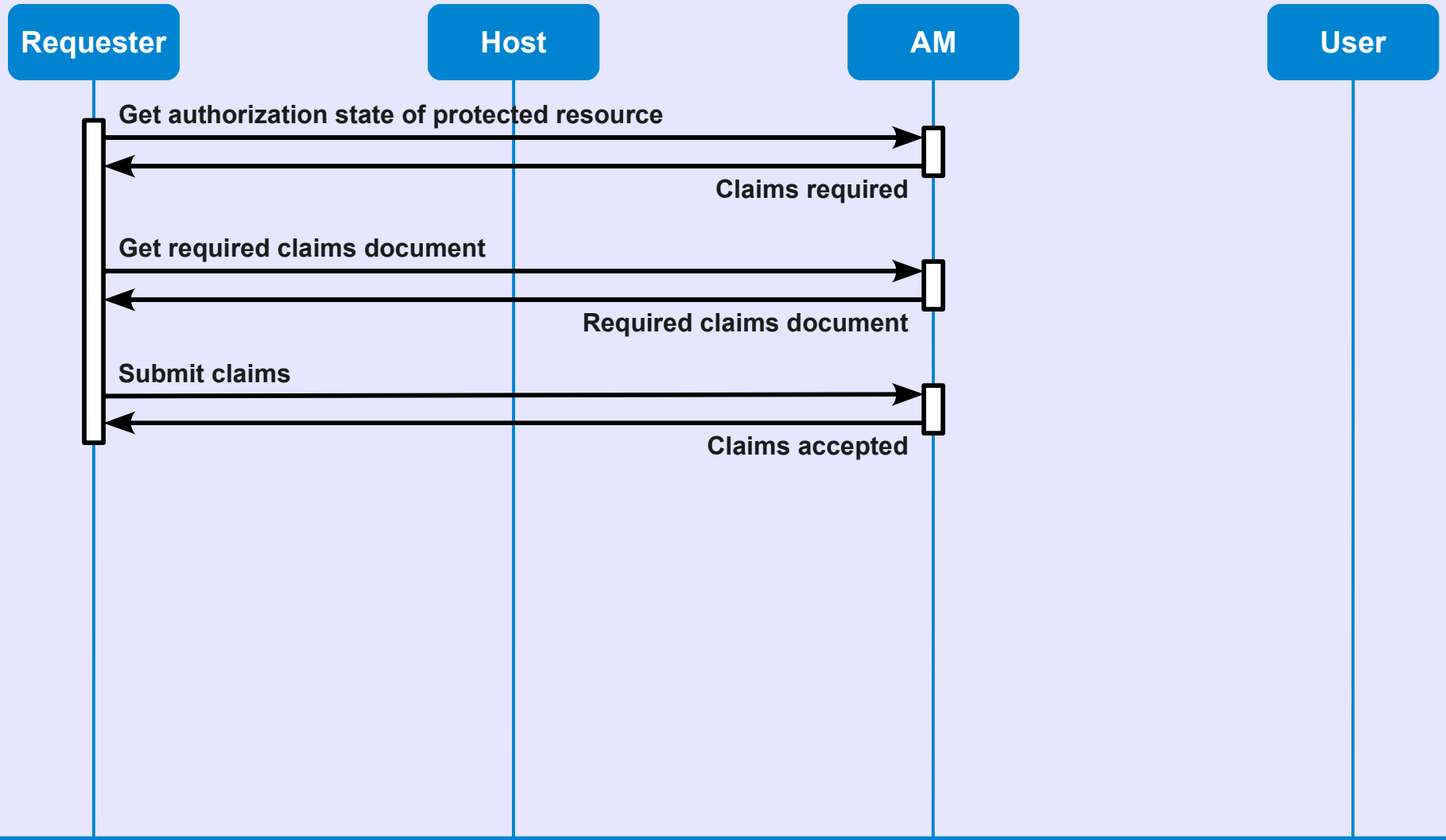


HTTPS request from Requester to Authorization Manager (copmonkey.com:443)

```
POST /requester/authorization/claims?method=GET&resource=http://schedewl.com/…/travel.ics
Authorization: OAuth realm="copmonkey-requester", oauth_consumer_key="3972c639fb72476f",
 oauth_token="4f30db8d0117464e", oauth_signature_method="HMAC-SHA1", …
Content-Type: application/x-claims-format-v2
…
```

Entity contains claims document.

Step 4. Requester negotiates with Authorization Manager

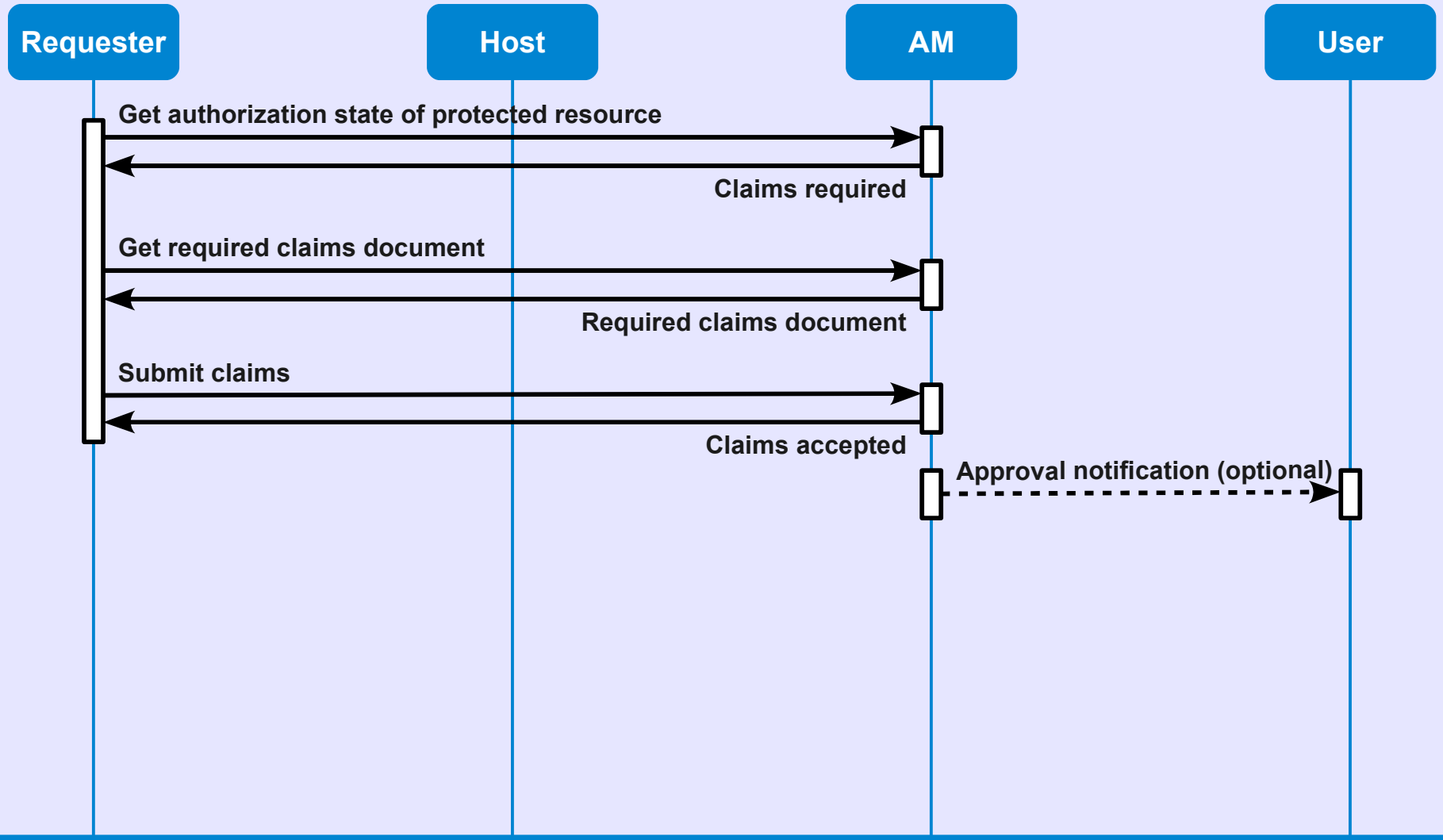


HTTPS response from Authorization Manager to Requester

```
HTTP/1.1 203 Accepted
...
```

Entity contains human-readable explanation.

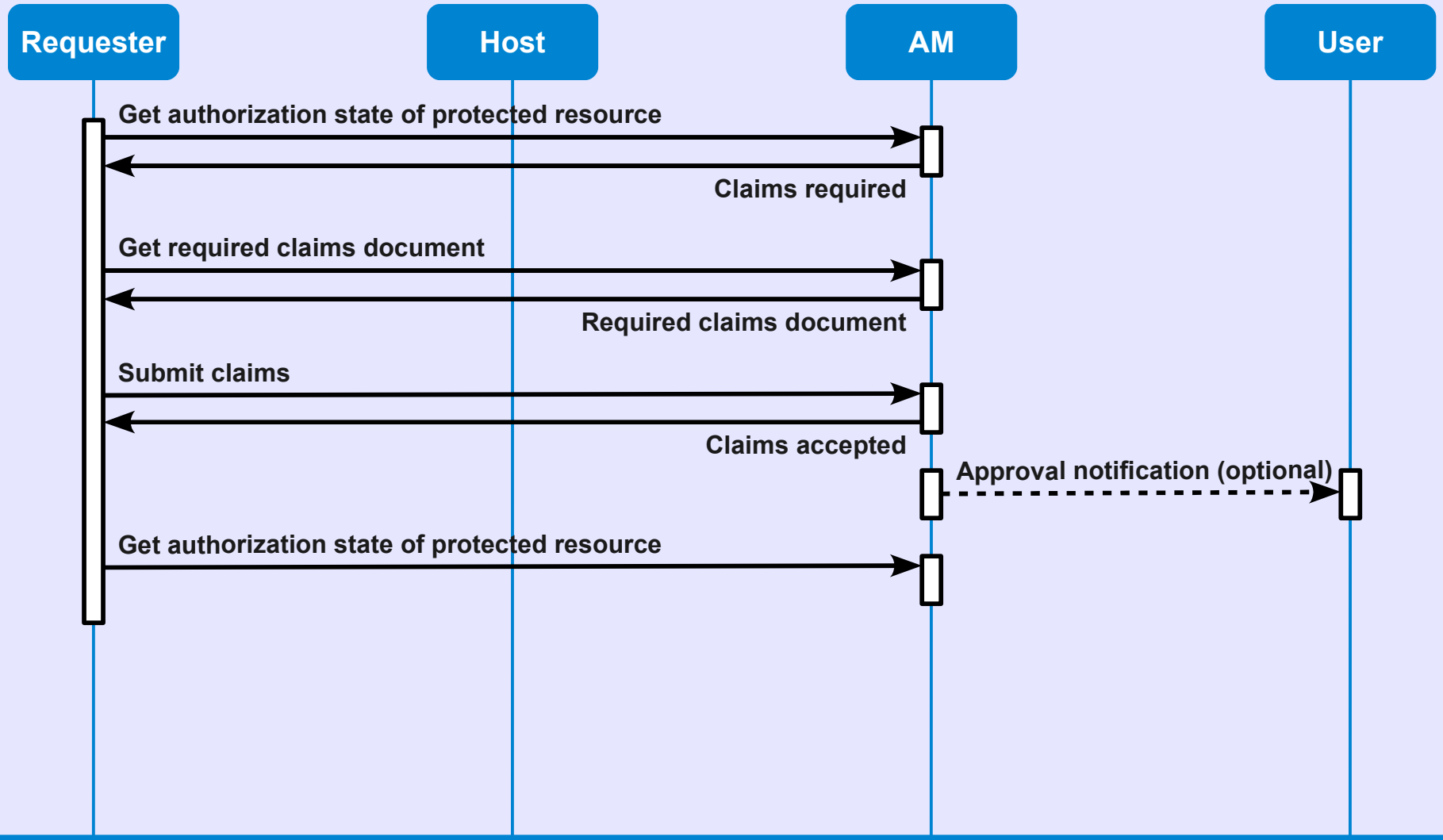
Step 4. Requester negotiates with Authorization Manager



Out-of-band approval notification from Authorization Manager to User (optional)

- User approval is optional, and can be a requirement of a policy defined in Authorization Manager.
- Process and channel are not a part of the core UMA protocol.
- Interesting notification options include: HTTP, XMPP, SMS.

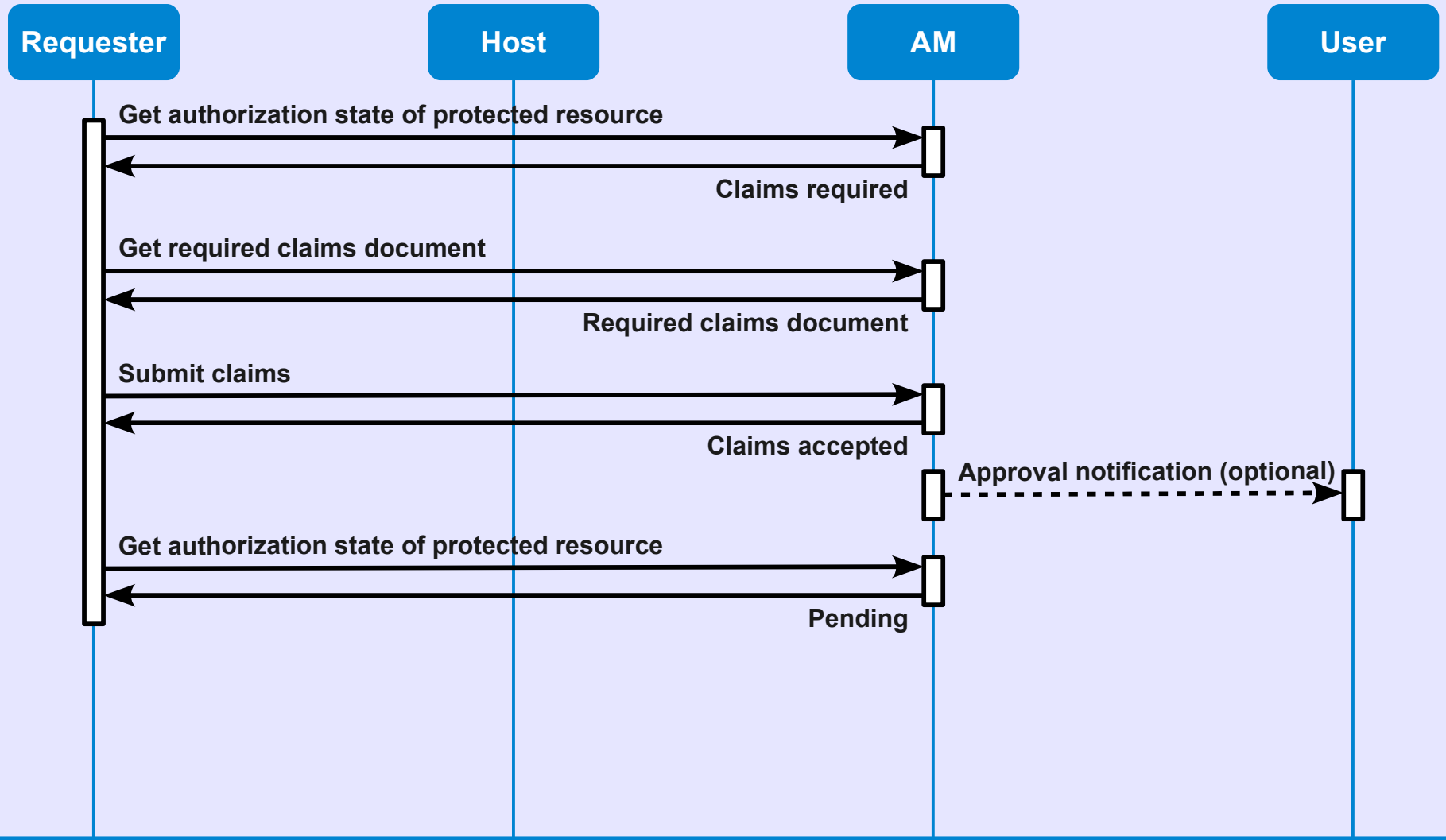
Step 4. Requester negotiates with Authorization Manager



HTTPS request from Requester to Authorization Manager (copmonkey.com:443)

```
GET /requester/authorization/status?method=GET&resource=http://schedewl.com/…/travel.ics
Authorization: OAuth realm="copmonkey-requester", oauth_consumer_key="3972c639fb72476f",
 oauth_token="4f30db8d0117464e", oauth_signature_method="HMAC-SHA1", …
…
```

Step 4. Requester negotiates with Authorization Manager

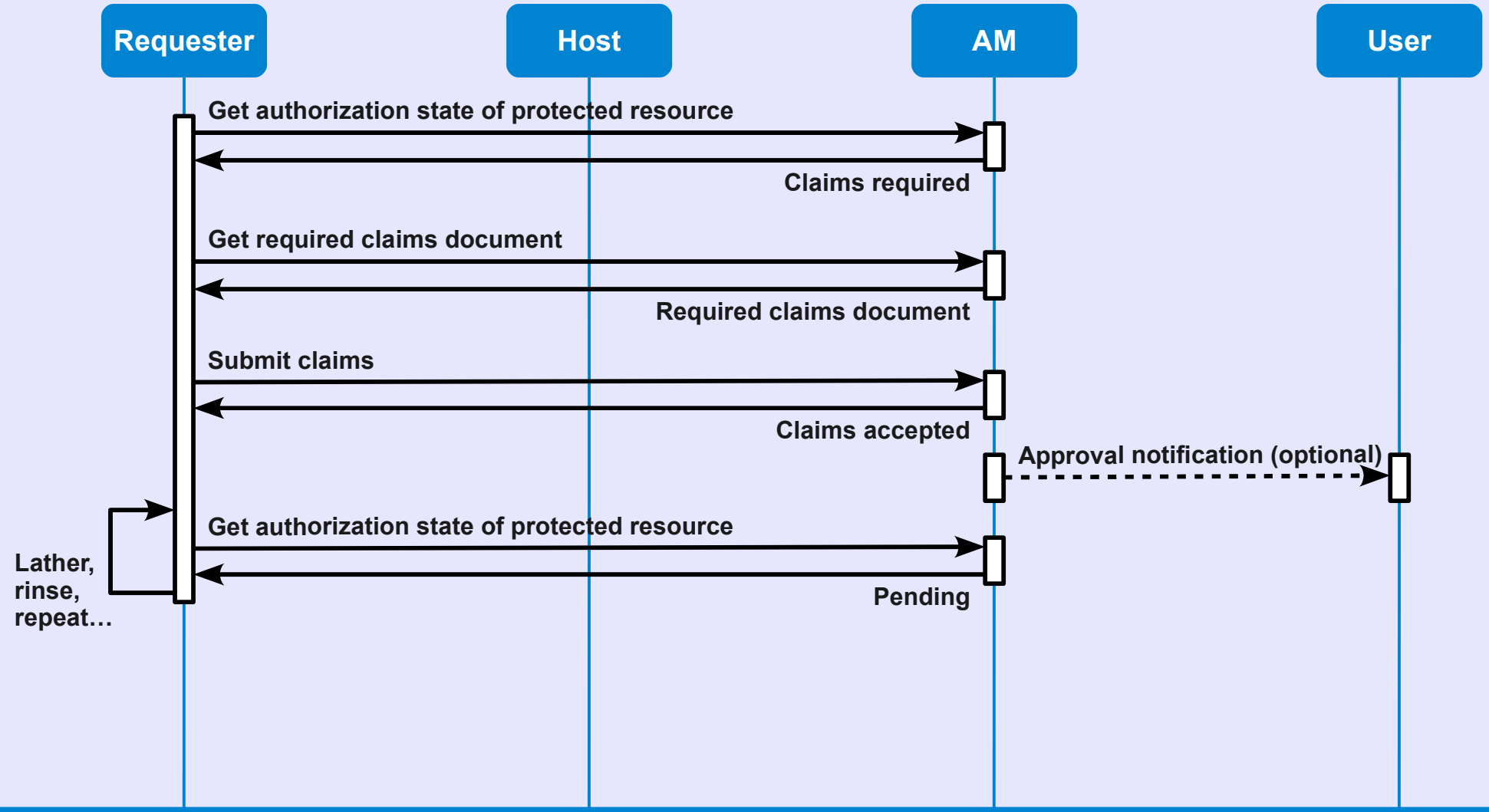


HTTPS response from Authorization Manager to Requester

```
HTTP/1.1 200 OK
Content-Type: application/json
…

{ "authorization": "pending" }
```

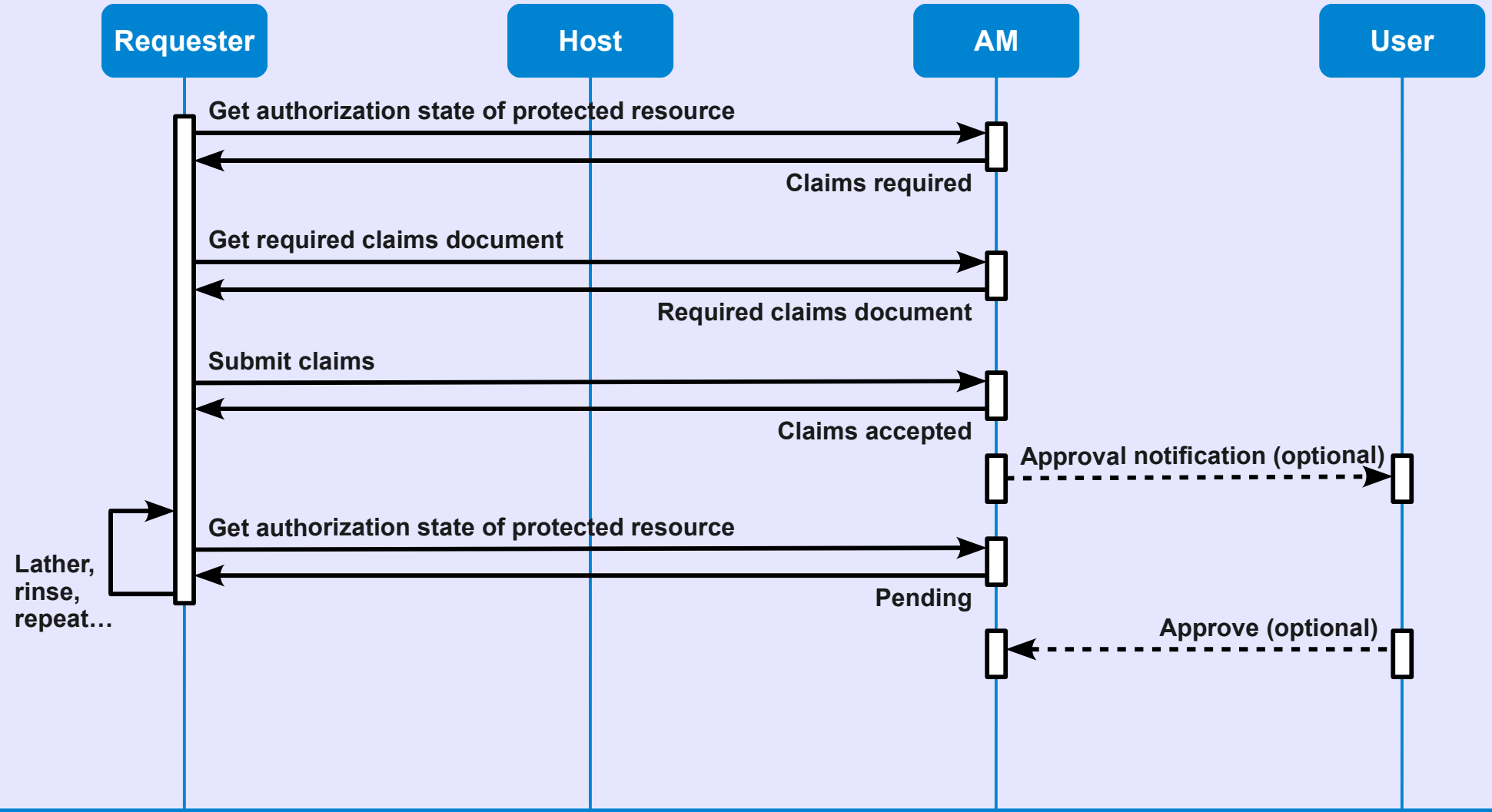
Step 4. Requester negotiates with Authorization Manager



Polling of pending state

- A pending state indicates authorization is awaiting some form of approval process or workflow.
- There is nothing for the Requester to do except wait for a state change; this is intentional.
- Authorization Manager should attempt to resolve pending states in a timely manner.
- Options to reduce polling and latency through asynchronous notification.

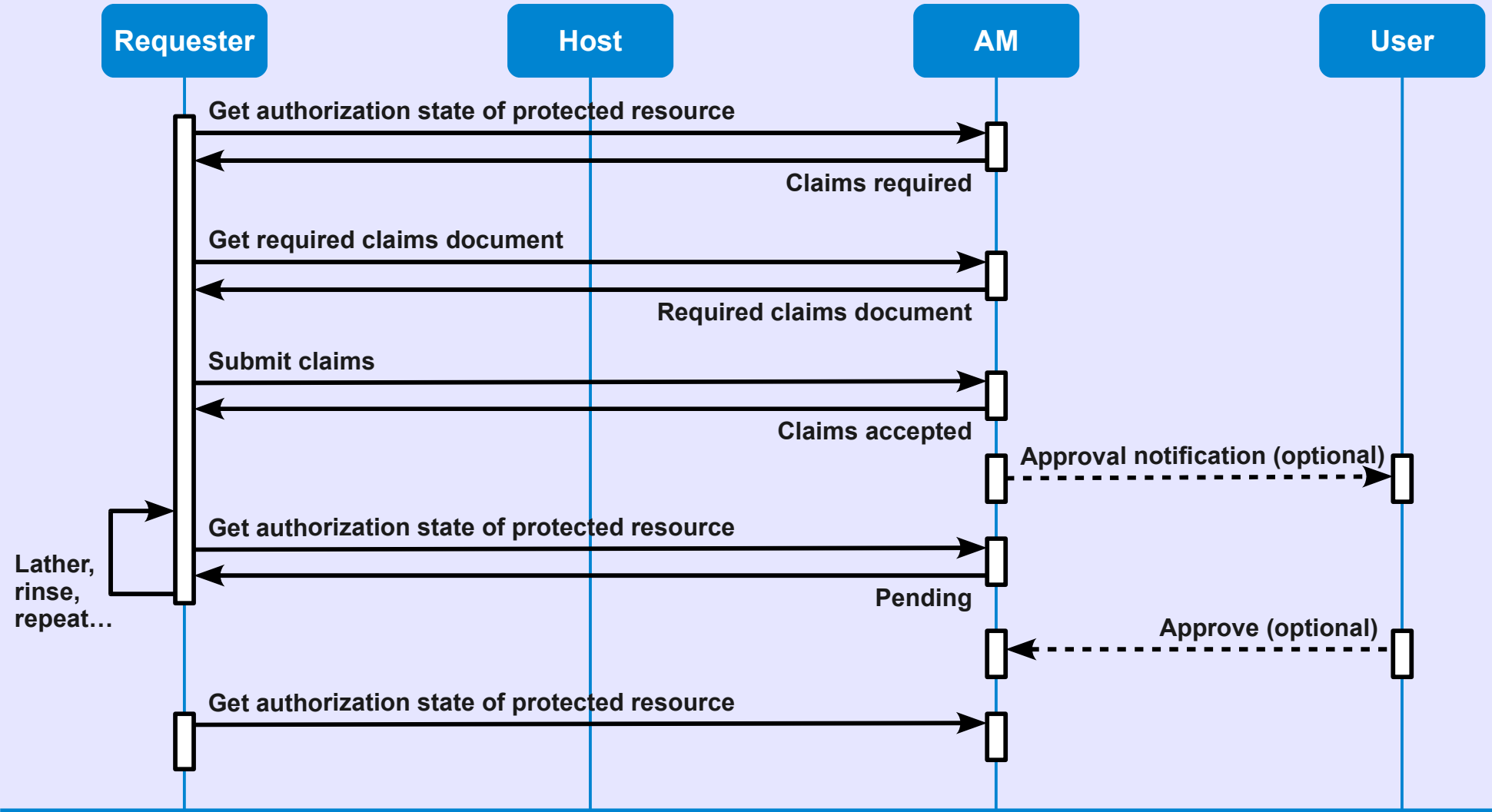
Step 4. Requester negotiates with Authorization Manager



Out-of-band User approval of Requester authorization request (optional)

- User provides approval (or rejection) of the authorization request.
- Not a part of the core UMA protocol.
- Interesting options include reply via: HTTP, XMPP, SMS.

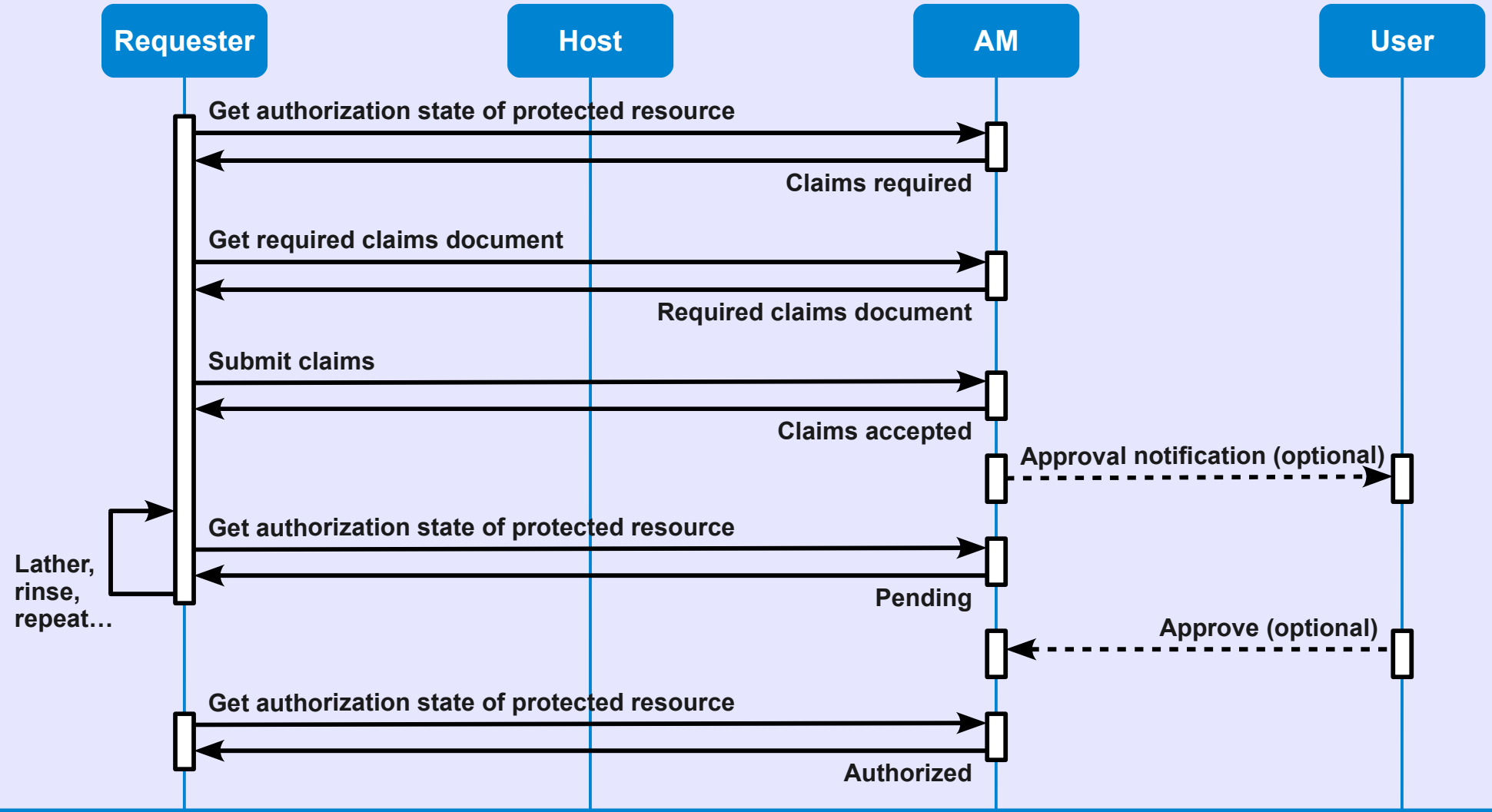
Step 4. Requester negotiates with Authorization Manager



HTTPS request from Requester to Authorization Manager (copmonkey.com:443)

```
GET /requester/authorization/status?method=GET&resource=http://schedewl.com/…/travel.ics
Authorization: OAuth realm="copmonkey-requester", oauth_consumer_key="3972c639fb72476f",
 oauth_token="4f30db8d0117464e", oauth_signature_method="HMAC-SHA1", …
…
```

Step 4. Requester negotiates with Authorization Manager

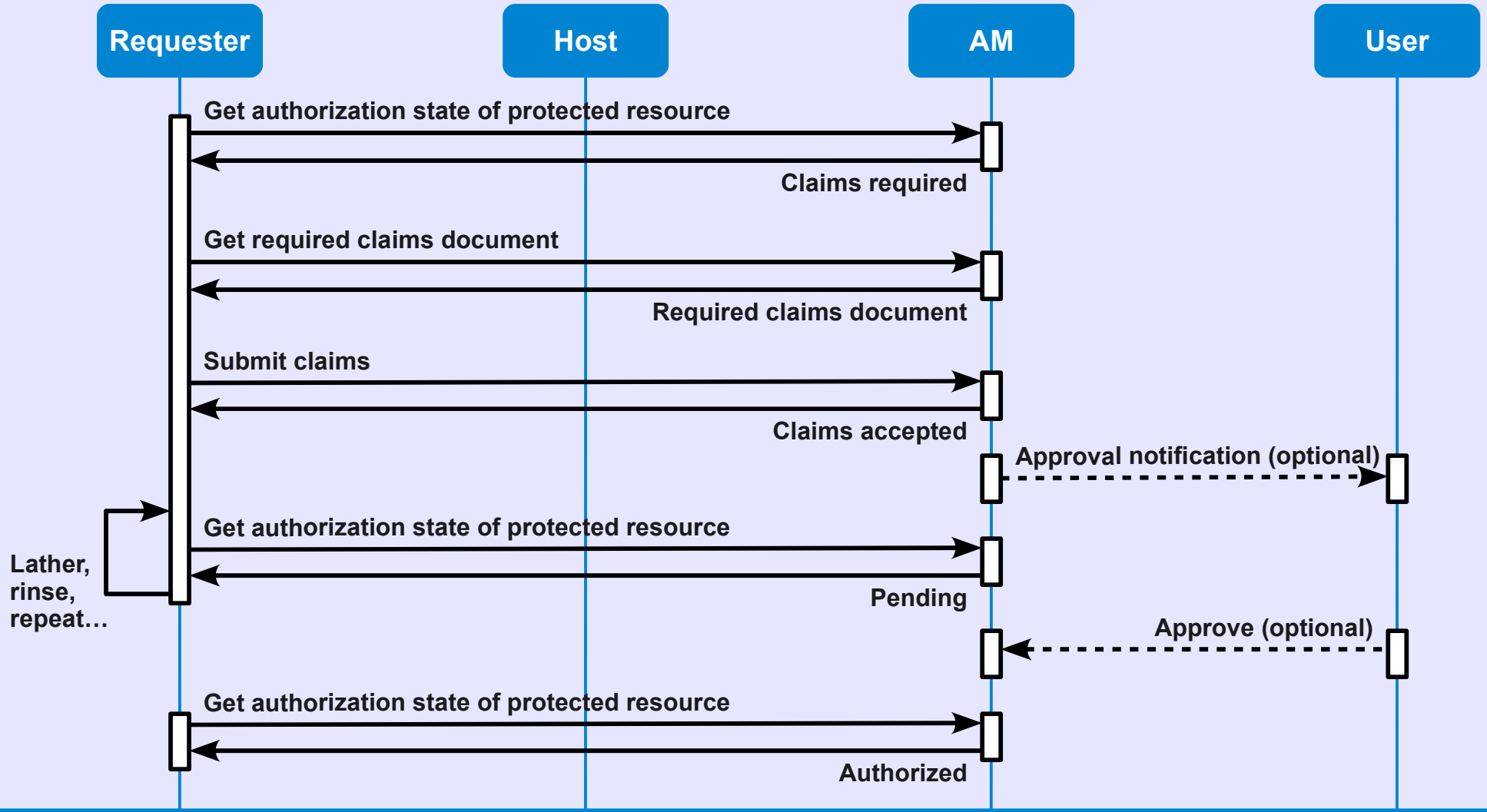


HTTPS response from Authorization Manager to Requester

```
HTTP/1.1 200 OK
Content-Type: application/json
…

{ "authorization": "authorized" }
```

Step 4. Requester negotiates with Authorization Manager

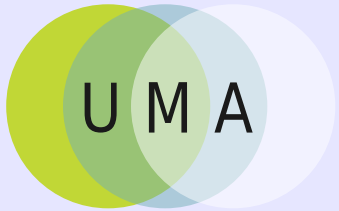


Summary

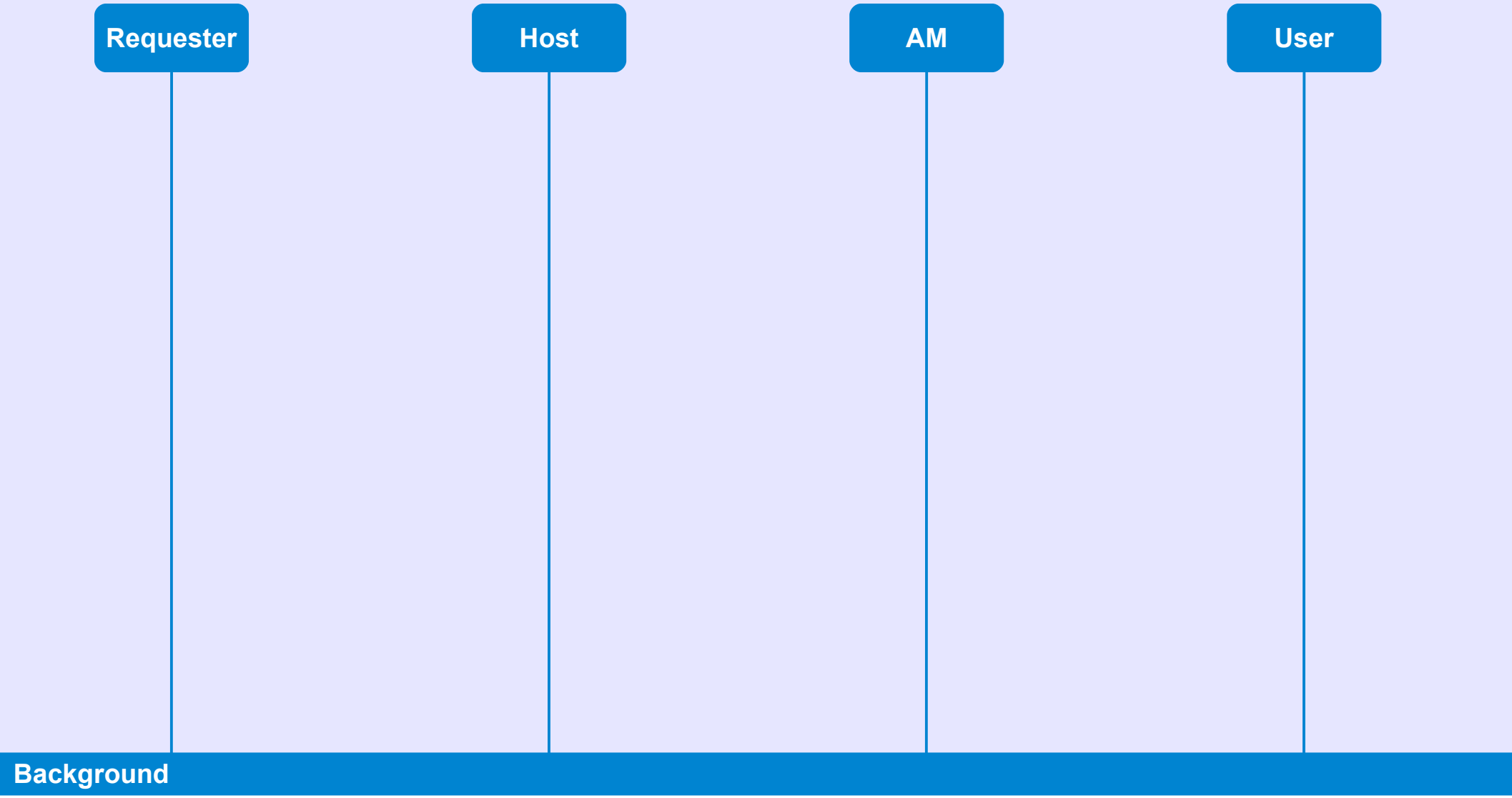
- Authorization is attached to access token; heavy-lifting up-front to allow ongoing efficient access.
- Claims-based access control not just for privacy (example: could require claims of payment).
- Authorization is presumed to persist as long as policy and user discretion allow.
- Requester can always revisit authorization status.
- Not always from pending to authorized—user could approve conditionally, requiring additional claims.

Step 5. Requester accesses resource at Host

The whole reason for the negotiations thus far

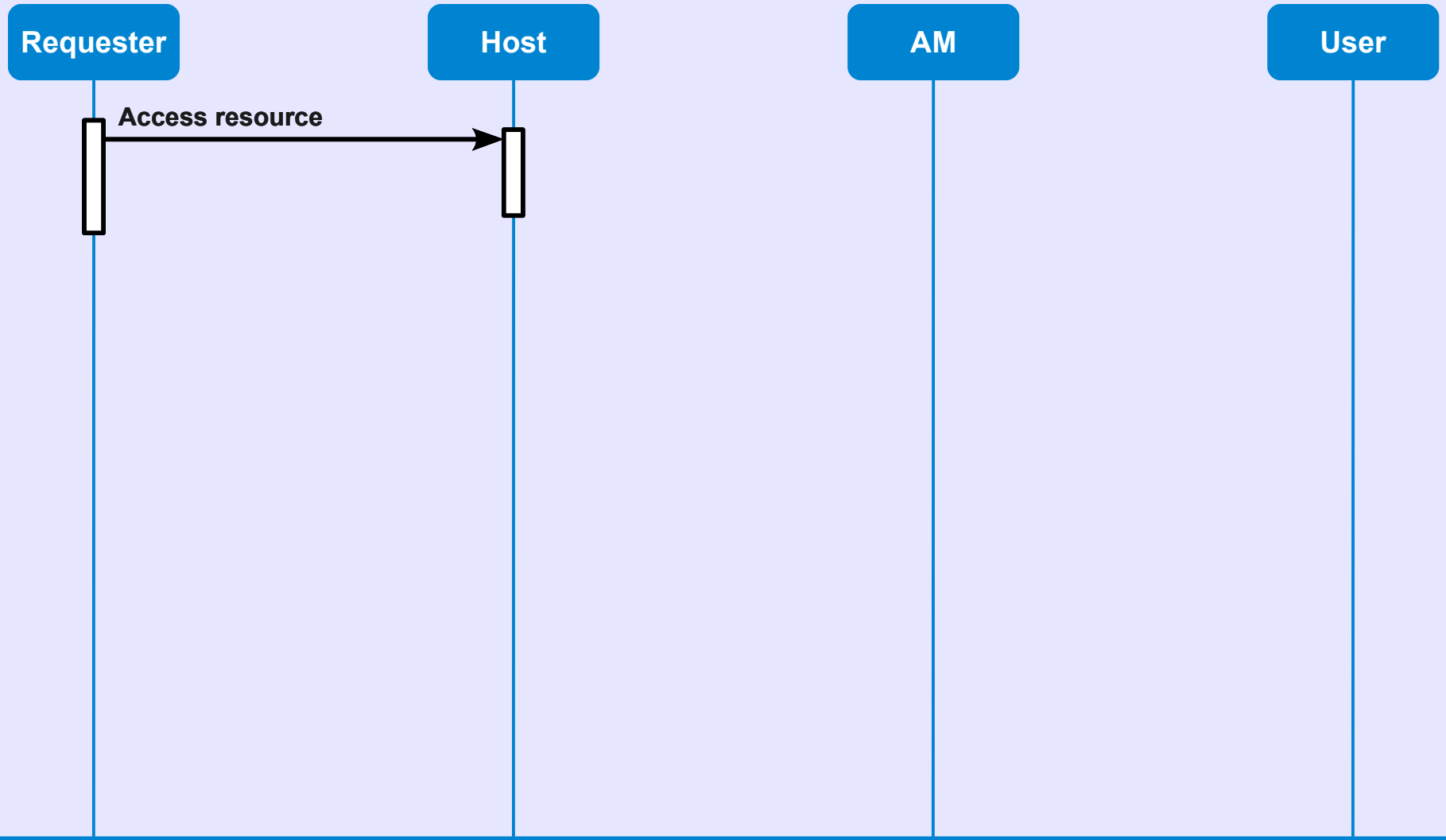


Step 5. Requester accesses resource at Host



Background

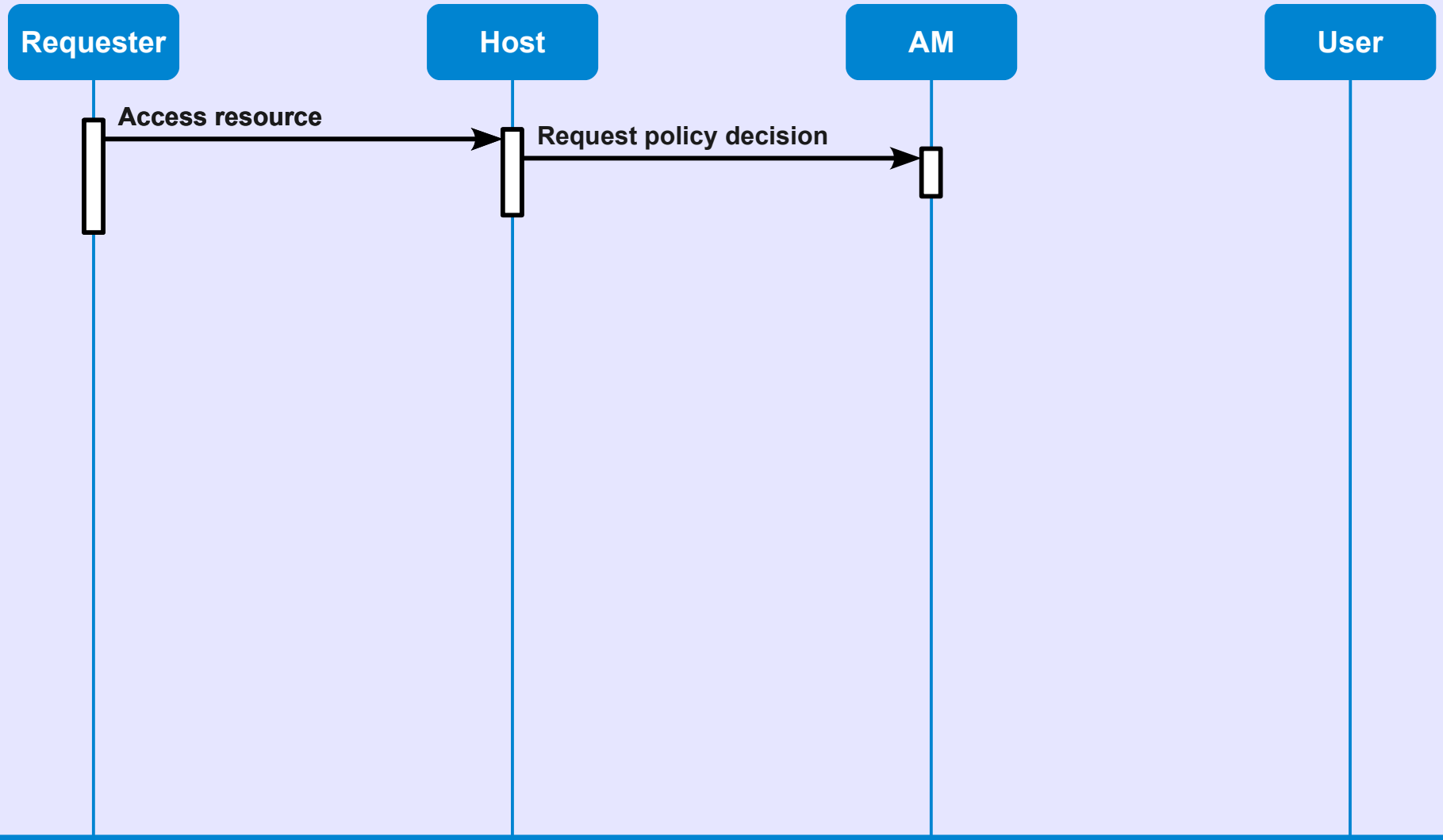
Step 5. Requester accesses resource at Host



HTTP request from Requester to Host (schedewl.com:80)

```
GET /calendar/ical/alice/public/travel.ics
Authorization: OAuth realm="schedewl", oauth_consumer_key="86d2e3ae50f249c0",
 oauth_token="5cdd7b5c68e24908", oauth_signature_method="HMAC-SHA1", …
 oauth_version="1.0"
…
```

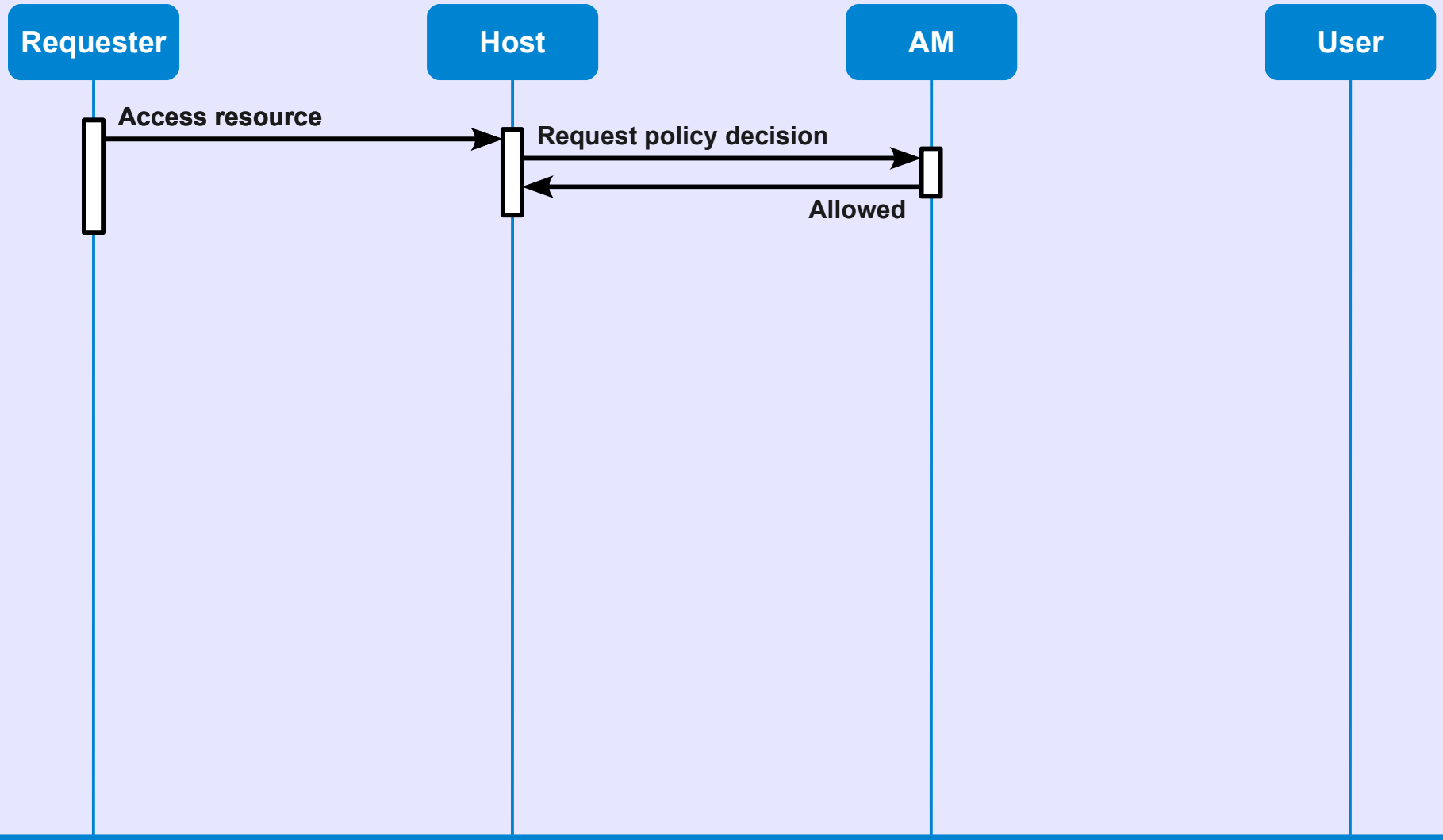
Step 5. Requester accesses resource at Host



HTTPS request from Host to Authorization Manager (copmonkey.com:443)

```
GET /host/75284056/decision?requester_id=5cdd7b5c68e24908&method=GET&resource=http://schedewl.com/⏎
 calendar/ical/alice/public/travel.ics
Authorization: OAuth realm="copmonkey-host", oauth_consumer_key="53032297b44847ed",
 oauth_token="2f5fa6f0613942d9", oauth_signature_method="HMAC-SHA1", …
…
```

Step 5. Requester accesses resource at Host

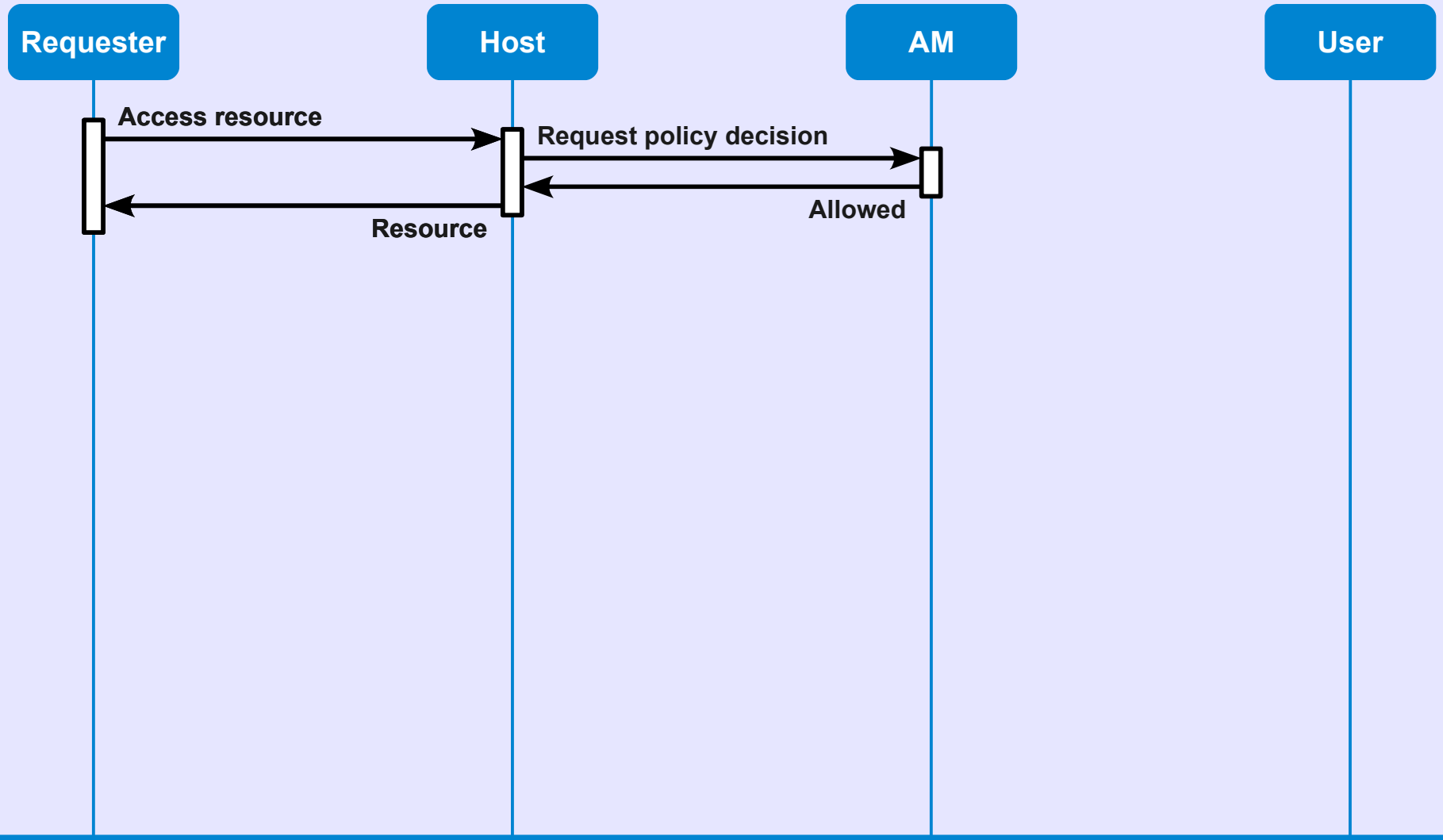


HTTPS response from Authorization Manager to Host

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: private
Expires: Expires: Fri, 29 Jan 2010 16:00:00 GMT
…

{"access": "allowed"}
```

Step 5. Requester accesses resource at Host

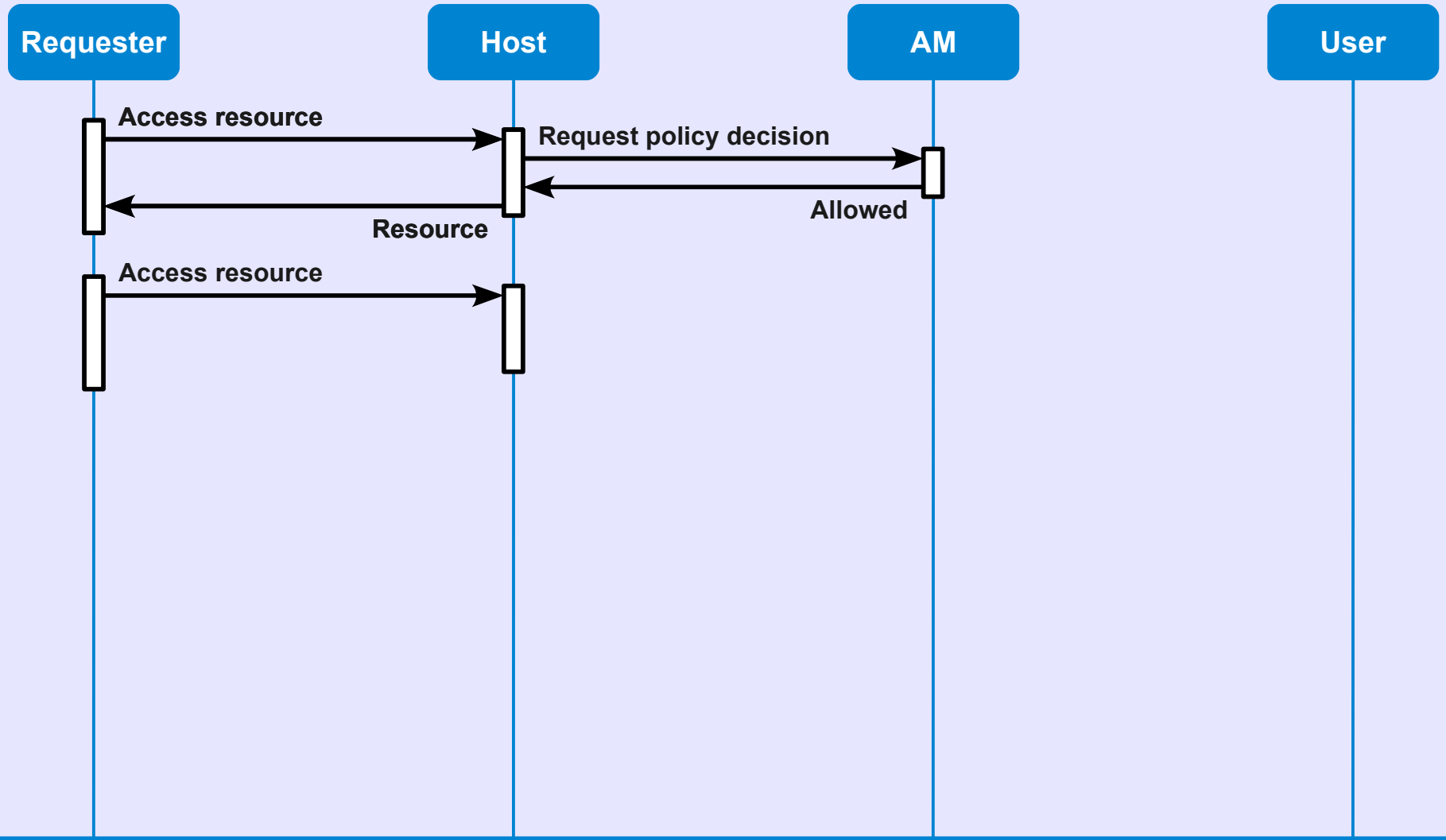


HTTP response from Host to Requester

```
HTTP/1.1 200 OK
Content-Type: text/calendar
…
```

Entity contains resource content.

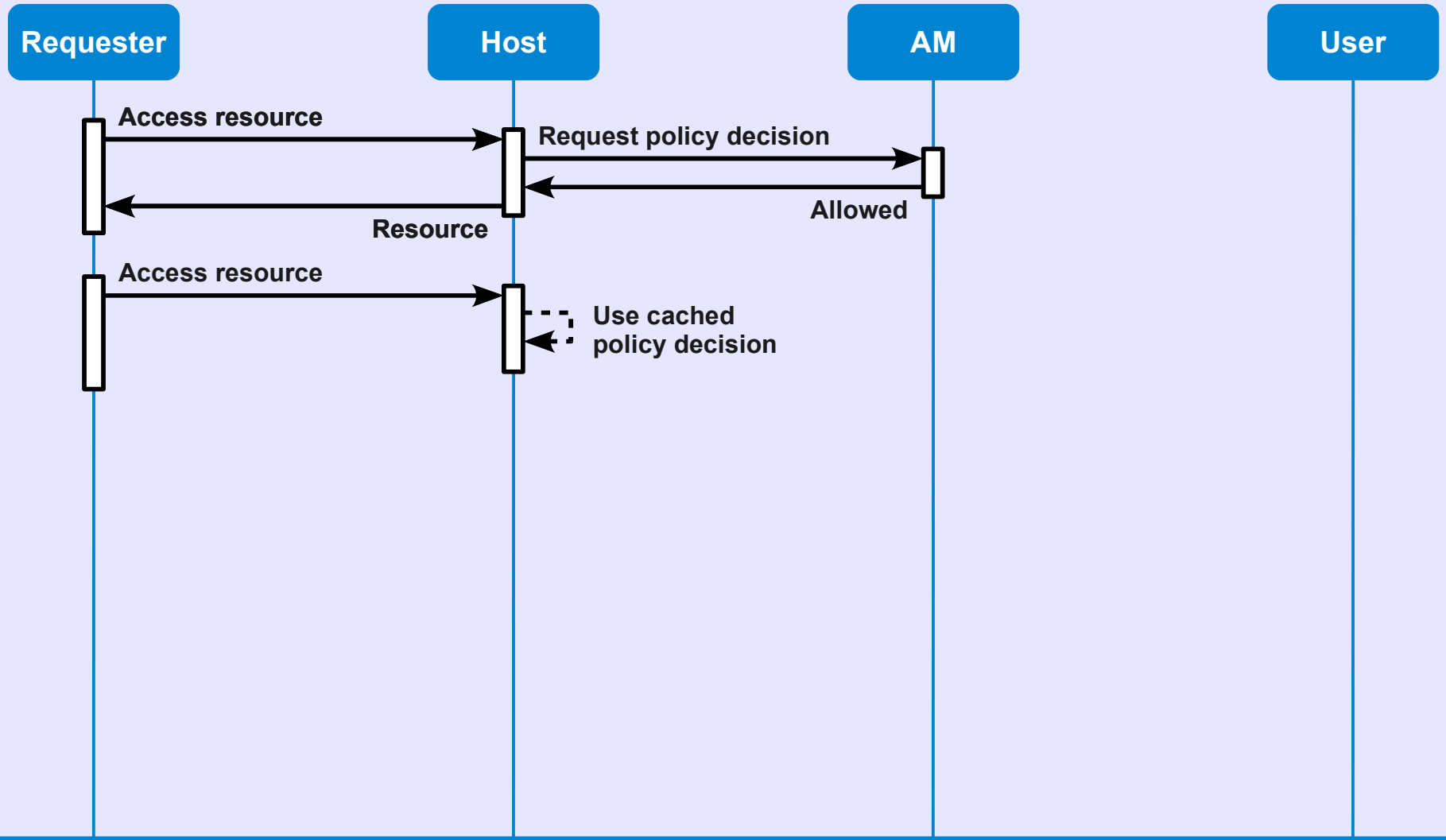
Step 5. Requester accesses resource at Host



HTTP request from Requester to Host (schedewl.com:80)

```
GET /calendar/ical/alice/public/travel.ics
Authorization: OAuth realm="schedewl", oauth_consumer_key="86d2e3ae50f249c0",
 oauth_token="5cdd7b5c68e24908", oauth_signature_method="HMAC-SHA1", …
…
```

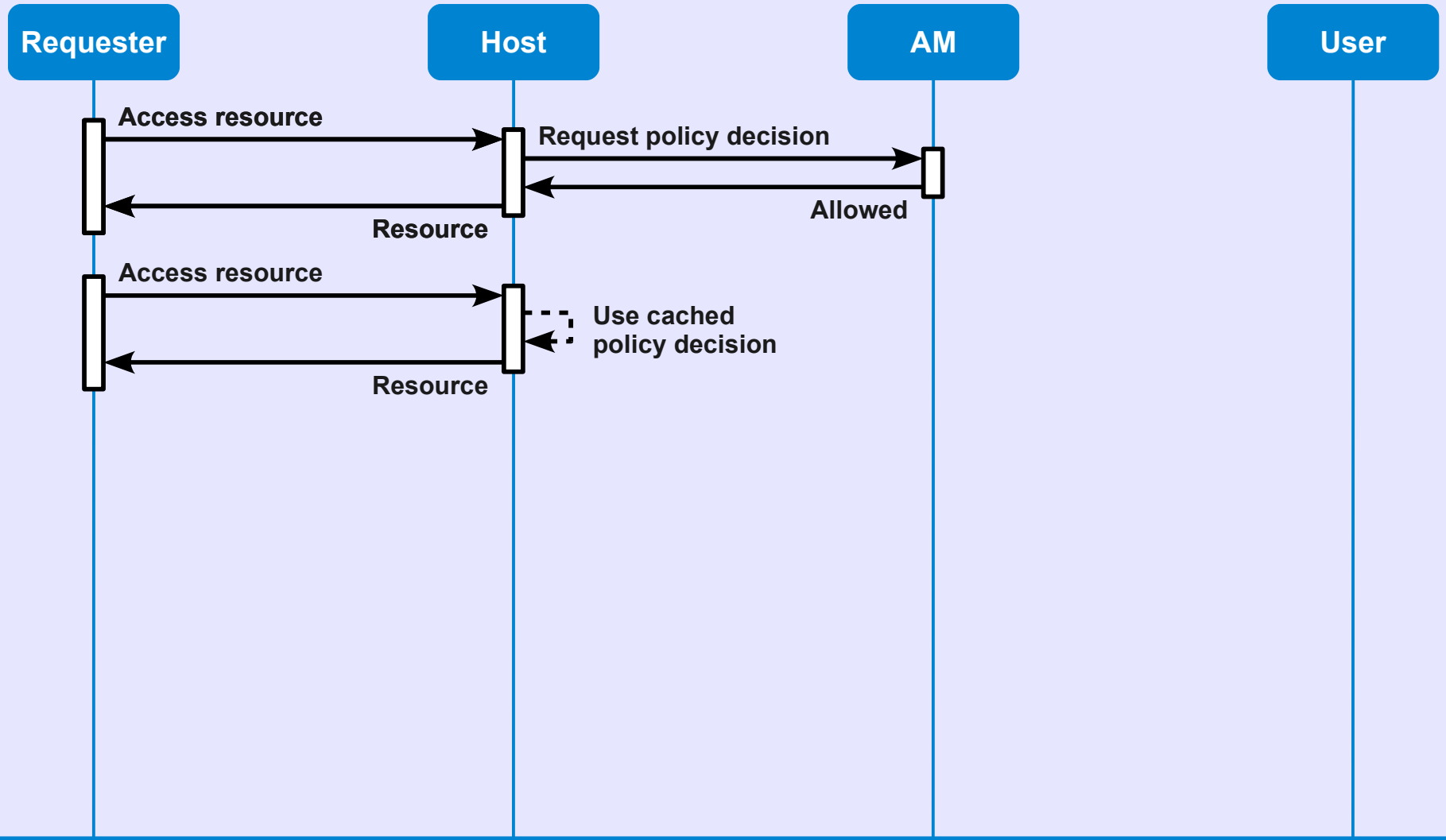
Step 5. Requester accesses resource at Host



Using a cached policy decision

- Host can use HTTP caching mechanism to reuse an existing policy decision.
- Prevents Authorization Manager from being a bottleneck.
- Reduces latency for frequent access to host resources.

Step 5. Requester accesses resource at Host

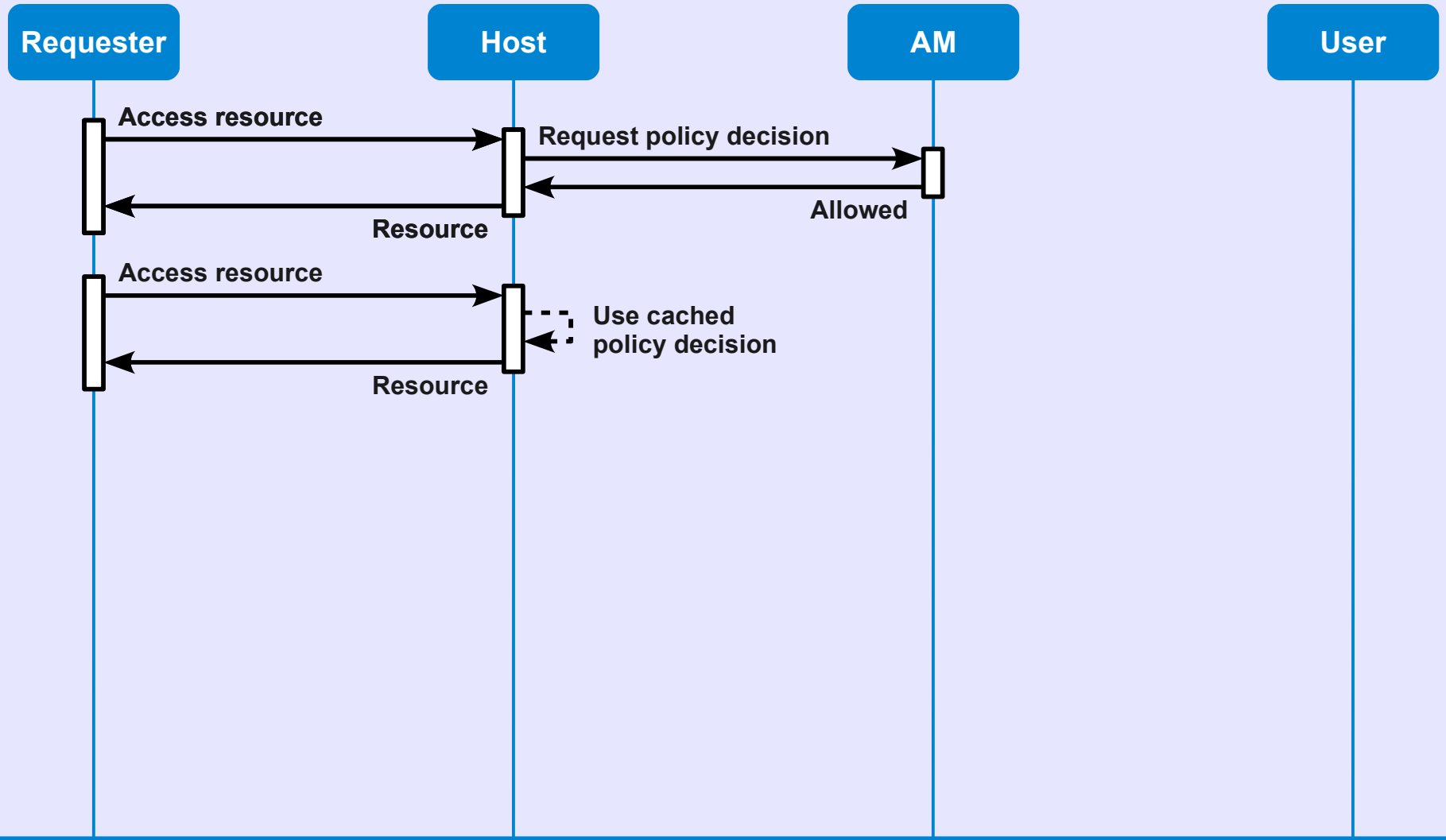


HTTP response from Host to Requester

```
HTTP/1.1 200 OK
Content-Type: text/calendar
…
```

Entity contains resource content.

Step 5. Requester accesses resource at Host



Summary

- **Mission accomplished!**

Questions & Answers