ATP
DocumentationCommunity
1. Home
2. Docs
Guides
Protocol Overview Start here
An introduction to the AT Protocol.
Identity
How the AT Protocol handles user identity.
Data Repositories
A guide to the AT Protocol repo structure.
Lexicon
A schema-driven interoperability framework
Applications
How applications work on the AT Protocol.
FAQ
Frequently Asked Questions about ATP
Specs
ATP
The specification for the Authenticated Transfer Protocol (aka "AT Protocol").
URI Scheme
A URI scheme for addressing ATP repository data.
Lexicon Schemas
A schemas format and distribution network.
XRPC
Cross-system remote procedure calls.
NSID
A specification for global semantic IDs.
DID:Placeholder
A hosted, secure registry of user DIDs.
Lexicons
com.atproto.admin
ATP Lexicon - Admin Schemas
com.atproto.identity
ATP Lexicon - Identity Schemas
com.atproto.label
ATP Lexicon - Label Schemas
com.atproto.moderation
ATP Lexicon - Moderation Schemas
com.atproto.repo
ATP Lexicon - Repo Schemas
com.atproto.server
ATP Lexicon - Server Schemas
com.atproto.sync

ATP Lexicon - Sync Schemas
app.bsky.actor
Bluesky Lexicon - Actor Schemas
app.bsky.embed
Bluesky Lexicon - Embed Schemas
app.bsky.feed
Bluesky Lexicon - Feed Schemas
app.bsky.graph
Bluesky Lexicon - Graph Schemas
app.bsky.notification
Bluesky Lexicon - Notification Schemas
app.bsky.richtext
Bluesky Lexicon - Richtext Schemas

ATP
DocumentationCommunity
1. Home
2. Docs
3. Protocol Overview
GuidesProtocol OverviewIdentityData RepositoriesLexiconApplicationsFAQSpecsATPURI SchemeLexicon SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext
Protocol Overview#
The Authenticated Transfer Protocol, aka atproto, is a federated protocol for large-scale distributed social applications. This document will introduce you to the ideas behind the AT Protocol.
Identity#
Users are identified by domain names on the AT Protocol. These domains map to cryptographic URLs which secure the user's account and its data.
[Image: Identities]
Data repositories#
User data is exchanged in signed data repositories. These repositories are collections of records which include posts, comments, likes, follows, media blobs, etc.
[Image: Data repos]
Federation#

The AT Protocol syncs the repositories in a federated networking model. Federation was chosen to ensure the network is convenient to use and reliably available. Commands are sent between servers using HTTPS + XRPC.
The three main services of our first federation are personal data servers (PDS), big graph services (BGS), and App Views. We're also working on feed generators and labelers.
The lower-level primitives that can get stacked together differently are the repositories, lexicons, and DIDs. We published an overview of our technical decisions around federation architecture on our blog.
[Image: Federation]
Interoperation#
A global schemas network called Lexicon is used to unify the names and behaviors of the calls across the servers. Servers implement "lexicons" to support featuresets, including the core ATP Lexicon for syncing user repositories and the Bsky Lexicon to provide basic social behaviors.
[Image: Interop]
While the Web exchanges documents, the AT Protocol exchanges schematic and semantic information, enabling the software from different orgs to understand each others' data. This gives atproto clients freedom to produce user interfaces independently of the servers, and removes the need to exchange rendering code (HTML/JS/CSS) while browsing content.
Achieving scale#
Personal data servers are your home in the cloud. They host your data, distribute it, manage your identity, and orchestrate requests to other services to give you your views.
Big Graph Services (BGS) handle all of your events, like retrieving large-scale metrics (likes, reposts, followers), content discovery (algorithms), and user search.
[Image: PDS and BGS]
This distinction is intended to achieve scale as well as a high degree of user-choice.
Algorithmic choice#
As with Web search engines, users are free to select their indexers. Each feed, discovery section, or search interface is integrated into the PDS while being served from a third party service.
[Image: Algorithmic choice]
Account portability#
We assume that a Personal Data Server may fail at any time, either by going offline in its entirety, or by ceasing service for specific users. The goal of the AT Protocol is to ensure that a user can migrate their account to a new PDS without the server's involvement.
User data is stored in signed data repositories and verified by DIDs. Signed data repositories are like Git repos but for database records, and DIDs are essentially registries of user certificates, similar in some ways to the TLS certificate system. They are expected to be secure, reliable, and independent of the user's PDS.
[Image: DID Documents]
Each DID document publishes two public keys: a signing key and a recovery key.
* Signing key: Asserts changes to the DID Document and to the user's data repository.
* Recovery key: Asserts changes to the DID Document; may override the signing key within a 72-hour window.
The signing key is entrusted to the PDS so that it can manage the user's data, but the recovery

key is saved by the user, e.g. as a paper key. This makes it possible for the user to update their account to a new PDS without the original host's help.
[Image: Account recovery]
A backup of the user’s data is persistently synced to their client as a backup (contingent on the disk space available). Should a PDS disappear without notice, the user should be able to migrate to a new provider by updating their DID Document and uploading the backup.
Speech, reach, and moderation#
Atproto's model is that speech and reach should be two separate layers, built to work with each other. The “speech” layer should remain neutral, distributing authority and designed to ensure everyone has a voice. The “reach” layer lives on top, built for flexibility and designed to scale.
[Image: Speech vs Reach]
The base layer of atproto (personal data repositories and federated networking) creates a common space for speech where everyone is free to participate, analogous to the Web where anyone can put up a website. The indexing services then enable reach by aggregating content from the network, analogous to a search engine.
Specifications#
Five primary specs comprise the v1 of the AT Protocol. These specs are:
* Authenticated Transfer Protocol
* Cross-system RPC (XRPC)
* Lexicon Schemas
* Namespaced IDs (NSIDs)
* DID:Placeholder (did:plc)
These specs can be organized into three layers of dependency:
[Image: Spec diagram]
From here, you can continue reading the guides and specs.
DocumentationCommunity
1. Home
2. Docs
3. Identity
GuidesProtocol OverviewIdentityData RepositoriesLexiconApplicationsFAQSpecsATPURI SchemeLexicon SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext
TL;DR:
Every user has a domain name like @alice.host.com or just @alice.com
Every user also has a persistent "DID" which enables migration between hosts
The DID maps to users' keys and host addresses
Identity#
The atproto identity system has a number of requirements:
* ID provision. Users should be able to create global IDs which are stable across services. These IDs should rarely change to ensure that links to their content are stable.
* Public key distribution. Distributed systems rely on cryptography to prove the authenticity of data and provide end-to-end privacy. The identity system must publish their public keys with

strong security.
* Key rotation. Users must be able to rotate their key material without disrupting their identity.
* Service discovery. To interact with users, applications must be able to discover the services in use by a given user.
* Usability. Users should have human-readable and memorable names.
* Portability. Identities should be portable across services. Changing a provider should not cause a user to lose their identity, social graph, or content.

Adopting this system should give applications the tools for end-to-end encryption, signed user data, service sign in, and general interoperation.

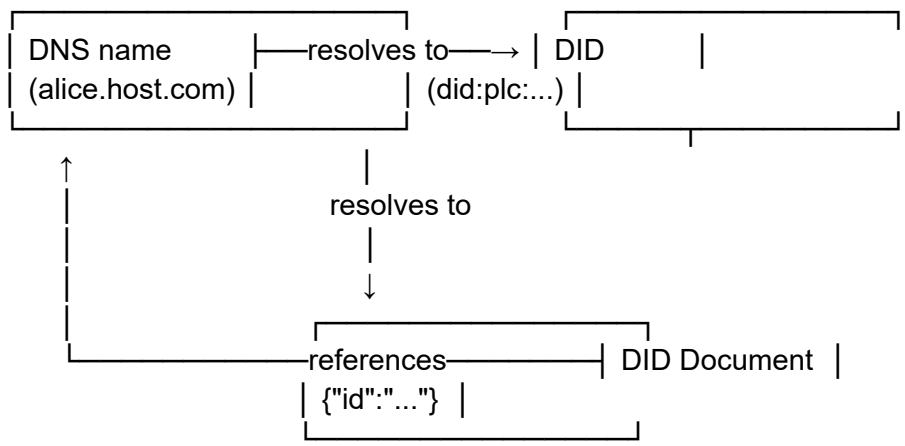
Identifiers#

We use two interrelated forms of identifiers: the handle and the DID. Handles are DNS names while DIDs are an emerging W3C standard which act as secure & stable IDs.

The following are all valid user identifiers:

@alice.host.com
at://alice.host.com
at://did:plc:bv6ggog3tya2z3vxsub7hnal

The relationship between them can be visualized as:



The DNS handle is a user-facing identifier — it should be shown in UIs and promoted as a way to find users. Applications resolve handles to DIDs and then use the DID as the stable canonical identifier. The DID can then be securely resolved to a DID document which includes public keys and user services.

Handles

Handles are DNS names. They are resolved using the com.atproto.identity.resolveHandle() XRPC method and should be confirmed by a matching entry in the DID document.

DIDs

DIDs are an emerging W3C standard for providing stable & secure IDs. They are used as stable, canonical IDs of users.

DID Documents

DID Documents are standardized objects which are hosted by DID registries. They include the following information:

* The handle associated with the DID.
* The signing key.
* The URL of the user's PDS.

DID Methods#

The DID standard supports custom "methods" of publishing and resolving DIDs to the DID Document. A variety of existing methods have been published so we must establish criteria for inclusion in this proposal:

* Strong consistency. For a given DID, a resolution query should produce only one valid document at any time. (In some networks, this may be subject to probabilistic transaction finality.)
* High availability. Resolution queries must succeed reliably.
* Online API. Clients must be able to publish new DID documents through a standard API.
* Secure. The network must protect against attacks from its operators, a MITM, and other users.
* Low cost. Creating and updating DID documents must be affordable to services and users.
* Key rotation. Users must be able to rotate keypairs without losing their identity.
* Decentralized governance. The network should not be governed by a single stakeholder; it must be an open network or a consortium of providers.

At present, none of the DID methods meet our standards fully. Therefore, we have chosen to support did-web and a temporary method we've created called did-placeholder. We expect this situation to evolve as new solutions emerge.

Handle Resolution#

Handles in atproto are domain names which resolve to a DID, which in turn resolves to a DID Document containing the user's signing pubkey and hosting service.

Handle resolution uses the com.atproto.identity.resolveHandle XRPC method. The method call should be sent to the server identified by the handle, and the handle should be passed as a parameter.

Here is the algorithm in pseudo-TypeScript:

```
async function resolveHandle(handle: string) {
  const origin = `https://${handle}`
  const res = await xrpc(origin, 'com.atproto.identity.resolveHandle', {handle})
  assert(typeof res?.did === 'string' && res.did.startsWith('did:'))
  return res.did
}
```

Example: Hosting service#

Consider a scenario where a hosting service is using PLC and is providing the handle for the user as a subdomain:
* The handle: alice.pds.com
* The DID: did:plc:12345
* The hosting service: https://pds.com
At first, all we know is alice.pds.com, so we call com.atproto.identity.resolveHandle() on alice.pds.com. This tells us the DID.

```
await xrpc.service('https://alice.pds.com').com.atproto.identity.resolveHandle() // => {did:
'did:plc:12345'}
```

Next we call the PLC resolution method on the returned DID so that we can learn the hosting service's endpoint and the user's key material.

```
await didPlc.resolve('did:plc:12345') /* => {
  id: 'did:plc:12345',
  alsoKnownAs: `https://alice.pds.com`,
  verificationMethod: [...],
  service: [{serviceEndpoint: 'https://pds.com', ...}]
}*/
```

We can now communicate with https://pds.com to access Alice's data.
Example: Hosting service with separate domain name#
Suppose we have the same scenario as before, except the user has supplied their own domain name:
* The handle: alice.com (this differs from before)
* The DID: did:plc:12345
* The hosting service: https://pds.com
We call com.atproto.identity.resolveHandle() on alice.com to get the DID.

```
await xrpc.service('https://alice.com').com.atproto.identity.resolveHandle() // => {did:
'did:plc:12345'}
```

Then we resolve the DID as before:

```
await didPlc.resolve('did:plc:12345') /* => {
  id: 'did:plc:12345',
  alsoKnownAs: `https://alice.com`,
  verificationMethod: [...],
```

```
  service: [{serviceEndpoint: 'https://pds.com', ...}]
}*/
```

We can now communicate with https://pds.com to access Alice's data. The https://alice.com endpoint only serves to handle the com.atproto.identity.resolveHandle() call. The actual user data lives on pds.com.

Example: Self-hosted#

Let's consider a self-hosting scenario. If it's using did:plc, it would look something like:

* The handle: alice.com
* The DID: did:plc:12345
* The hosting service: https://alice.com

However, if the self-hoster is confident they will retain ownership of the domain name, they can use did:web instead of did:plc:

* The handle: alice.com
* The DID: did:web:alice.com
* The hosting service: https://alice.com

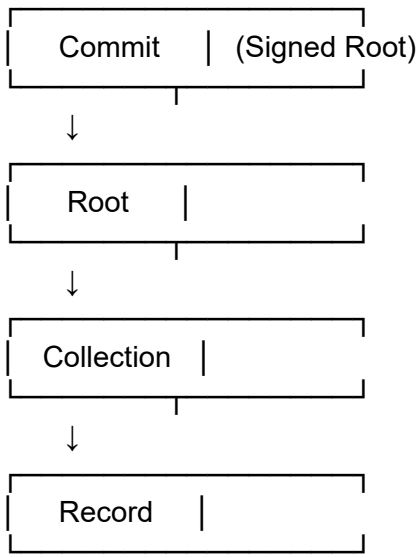
We call com.atproto.identity.resolveHandle() on alice.com to get the DID.

```
await xrpc.service('https://alice.com').com.atproto.identity.resolveHandle() // => {did:
'did:web:alice.com'}
```

We then resolve using did:web:

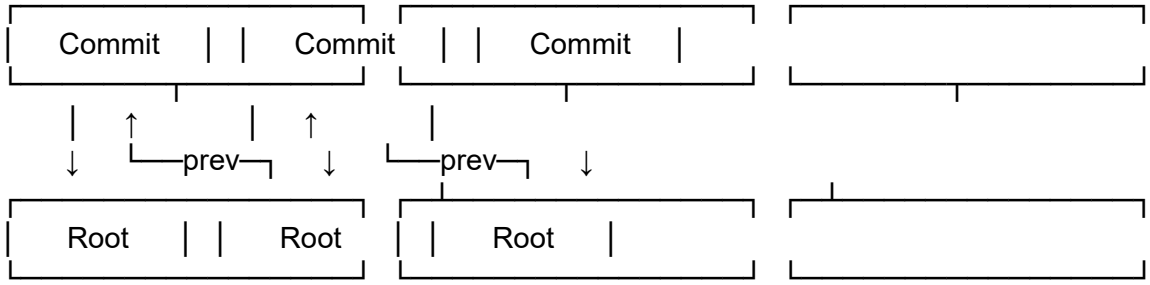
```
await didWeb.resolve('did:web:alice.com') /* => {
  id: 'did:web:alice.com',
  alsoKnownAs: `https://alice.com`,
  verificationMethod: [...],
  service: [{serviceEndpoint: 'https://alice.com', ...}]
}*/
```

2. Docs
3. Personal Data Repositories
GuidesProtocol OverviewIdentityData RepositoriesLexiconApplicationsFAQSpecsATPURI SchemeLexicon SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext
TL;DR:
A data repository is a collection of signed data
They're like Git repos but for database records
Users put their public activity in these repos
Data Repositories#
A data repository is a collection of data published by a single user. Repositories are self-authenticating data structures, meaning each update is signed and can be verified by anyone.
Data Layout#
The content of a repository is laid out in a Merkle Search Tree (MST) which reduces the state to a single root hash. It can be visualized as the following layout:



Every node is an IPLD object (dag-cbor) which is referenced by a CID hash. The arrows in the diagram above represent a CID reference.
This layout is reflected in the URLs:
Root | at://alice.com
Collection | at://alice.com/app.bsky.feed.post
Record | at://alice.com/app.bsky.feed.post/1234

A “commit” to a data repository is simply a keypair signature over a Root node’s CID. Each mutation to the repository produces a new Root node, and every Root node includes the CID of the previous Commit. This produces a linked list which represents the history of changes in a Repository.



Identifier Types#

Multiple types of identifiers are used within a Personal Data Repository.

DIDs

Decentralized IDs (DIDs) identify data repositories. They are broadly used as user IDs, but since every user has one data repository then a DID can be considered a reference to a data repository. The format of a DID varies by the “DID method” used but all DIDs ultimately resolve to a keypair and a list of service providers. This keypair can sign commits to the data repository, or it can authorize UCAN keypairs which then sign commits (see “Permissioning”).

CIDs

Content IDs (CIDs) identify content using a fingerprint hash. They are used throughout the repository to reference the objects (nodes) within it. When a node in the repository changes, its CID also changes. Parents which reference the node must then update their reference, which in turn changes the parent’s CID as well. This chains all the way to the Root node, which is then signed to produce a new commit.

TIDs

Timestamp IDs (TIDs) identify records. They are used in Collections as a key to Records. TIDs are produced using the local device’s monotonic clock e.g. microseconds since Unix epoch. To reduce the potential for collisions, a 10-bit clockID is appended . The resulting number is encoded as a 13 character string in a sort-order-invariant base32 encoding (`3hgb-r7t-ngir-t4`).

ATP

DocumentationCommunity

1. Home
2. Docs
3. Lexicon

GuidesProtocol OverviewIdentityData RepositoriesLexiconApplicationsFAQSpecsATPURI

SchemeLexicon
SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext
TL;DR:
Lexicon is a global schema system
It uses reverse-DNS names like "com.example.ping()"
The definitions are JSON documents, similar to JSON-Schema
It's currently used for RPC methods and repo records
Intro to Lexicon#
Lexicon is a schema system used to define RPC methods and record types. Every Lexicon schema is written in JSON, in a format similar to JSON-Schema for defining constraints.
The schemas are identified using NSIDs which are a reverse-DNS format. Here are some example methods:
com.atproto.repo.getRecord()
com.atproto.identity.resolveHandle()
app.bsky.feed.getPostThread()
app.bsky.notification.listNotifications()

And here are some example record types:
app.bsky.feed.post
app.bsky.feed.like
app.bsky.actor.profile
app.bsky.graph.follow

Why is Lexicon needed?#
Interoperability. An open network like atproto needs a way to agree on behaviors and semantics. Lexicon solves this while making it relatively simple for developers to introduce new schemas.
Lexicon is not RDF. While RDF is effective at describing data, it is not ideal for enforcing schemas. Lexicon is easier to use because it doesn't need the generality that RDF provides. In fact, Lexicon's schemas enable code-generation with types and validation, which makes life much easier!
Schema format#
Schemas are JSON objects which follow this Typescript interface:

```
interface LexiconDoc {
  lexicon: 1
  id: string // an NSID
```

```
  type: 'query' | 'procedure' | 'record' | 'token'
  revision?: number
  description?: string
  defs?: JSONSchema




  // if type == record
  key?: string
  record?: JSONSchema




  // if type == query or procedure
  parameters?: Record<string, XrpcParameter>
  input?: XrpcBody
  output?: XrpcBody
  errors?: XrpcError[]
}
```

Notice the structure differs depending on the type. The meanings of the type are:

Type

Meaning

query

An XRPC "read" method (aka GET).

procedure

An XRPC "modify" method (aka POST).

record

An ATP repository record type.

token

A declared identifier with no behaviors associated.

RPC methods#

The AT Protocol's RPC system, XRPC, is essentially a thin wrapper around HTTPS. Its purpose is to apply the Lexicon to HTTPS.

For example, a call to:

com.example.getProfile()

is actually just an HTTP request:
GET /xrpc/com.example.getProfile

The schemas establish valid query parameters, request bodies, and response bodies.

```
{
  "lexicon": 1,
  "id": "com.example.getProfile",
  "type": "query",
  "parameters": {
    "user": {"type": "string", "required": true}
  },
  "output": {
    "encoding": "application/json",
    "schema": {
      "type": "object",
      "required": ["did", "name"],
      "properties": {
        "did": {"type": "string"},
        "name": {"type": "string"},
        "displayName": {"type": "string", "maxLength": 64},
        "description": {"type": "string", "maxLength": 256}
      }
    }
  }
}
```

With code-generation, these schemas become very easy to use:

```
await client.com.example.getProfile({user: 'bob.com'})
// => {name: 'bob.com', did: 'did:plc:1234', displayName: '...', ...}
```

Record types#
Schemas define the possible values of a record. Every record has a "type" which maps to a schema and also establishes the URL of a record.
For instance, this "follow" record:

```
{
  "$type": "com.example.follow",
  "subject": "at://did:plc:12345",
  "createdAt": "2022-10-09T17:51:55.043Z"
}
```

...would have a URL like:
at://bob.com/com.example.follow/12345

...and a schema like:

```
{
  "lexicon": 1,
  "id": "com.example.follow",
  "type": "record",
  "description": "A social follow",
  "record": {
    "type": "object",
    "required": ["subject", "createdAt"],
    "properties": {
      "subject": { "type": "string" },
      "createdAt": {"type": "string", "format": "date-time"}
    }
  }
}
```

Tokens#
Tokens declare global identifiers which can be used in data.
Let's say a record schema wanted to specify three possible states for a traffic light: 'red', 'yellow', and 'green'.

```
{
  "lexicon": 1,
  "id": "com.example.trafficLight",
  "type": "record",
  "record": {
    "type": "object",
```

```
    "required": ["state"],
    "properties": {
      "state": { "type": "string", "enum": ["red", "yellow", "green"] },
    }
  }
}
```

This is perfectly acceptable, but it's not extensible. You could never add new states, like "flashing yellow" or "purple" (who knows, it could happen).
To add flexibility, you could remove the enum constraint and just document the possible values:

```
{
  "lexicon": 1,
  "id": "com.example.trafficLight",
  "type": "record",
  "record": {
    "type": "object",
    "required": ["state"],
    "properties": {
      "state": {
        "type": "string",
        "description": "Suggested values: red, yellow, green"
      },
    }
  }
}
```

This isn't bad, but it lacks specificity. People inventing new values for state are likely to collide with each other, and there won't be clear documentation on each state.
Instead, you can define Lexicon tokens for the values you use:

```
{
  "lexicon": 1,
  "id": "com.example.green",
  "type": "token",
  "description": "A possible traffic light state.",
}
{
  "lexicon": 1,
  "id": "com.example.yellow",
```

```
  "type": "token",
  "description": "A possible traffic light state.",
}
{
  "lexicon": 1,
  "id": "com.example.red",
  "type": "token",
  "description": "A possible traffic light state.",
}
```

This gives us unambiguous values to use in our trafficLight state. The final schema will still use flexible validation, but other teams will have more clarity on where the values originate from and how to add their own:

```
{
  "lexicon": 1,
  "id": "com.example.trafficLight",
  "type": "record",
  "record": {
    "type": "object",
    "required": ["state"],
    "properties": {
      "state": {
        "type": "string",
        "description": "Suggested values: com.example.red, com.example.yellow,
com.example.green"
      },
    }
  }
}
```

Versioning#

Once a schema is published, it can never change its constraints. Loosening a constraint (adding possible values) will cause old software to fail validation for new data, and tightening a constraint (removing possible values) will cause new software to fail validation for old data. As a consequence, schemas may only add optional constraints to previously unconstrained fields.

If a schema must change a previously-published constraint, it should be published as a new schema under a new NSID.

Schema distribution#

Schemas are designed to be machine-readable and network-accessible. While it is not currently required that a schema is available on the network, it is strongly advised to publish schemas so that a single canonical & authoritative representation is available to consumers of the method.

TL;DR:
Apps sign into the user's PDS to access their account
Apps can directly read and write repo records
Most interactions occur through higher-level lexicons
Applications model#
Applications on the AT Protocol connect to the user's Personal Data Server (PDS) to access their account. Once a session is established, the app can use the lexicons implemented by the PDS to drive behaviors.
In this guide, we'll step through a couple of common patterns (with simple code examples) to help you develop an intuition about this. All APIs shown below are generated using Lexicon's code-generator CLI.
Signing in#
Sign-in and authentication is a simple session-oriented process. The com.atproto.server lexicon includes APIs for creating and managing these sessions.

```
// create an API instance with my PDS
const api = AtpApi.service('my-pds.com')
```

```
// sign in using my identifier and password
const res = await api.com.atproto.server.createSession({
  identifier: 'alice.host.com',
  password: 'hunter2'
})
```

```
// configure future calls to include the token in the Authorization header
api.setHeader('Authorization', `Bearer ${res.data.accessJwt}`)
```

Repo CRUD#

Every user has a public data repository. The application can do basic CRUD on records using the API.

```
await api.com.atproto.repo.listRecords({
  repo: 'alice.com',
  type: 'app.bsky.post'
})
await api.com.atproto.repo.getRecord({
  repo: 'alice.com',
  type: 'app.bsky.post',
  tid: '1'
})
await api.com.atproto.repo.createRecord({
  repo: 'alice.com',
  type: 'app.bsky.post'
}, {
  text: 'Second post!',
  createdAt: (new Date()).toISOString()
})
await api.com.atproto.repo.putRecord({
  repo: 'alice.com',
  type: 'app.bsky.post',
  tid: '1'
}, {
  text: 'Hello universe!',
  createdAt: originalPost.data.createdAt
})
await api.com.atproto.repo.deleteRecord({
  repo: 'alice.com',
  type: 'app.bsky.post',
  tid: '1'
})
```

You may notice that the repo above is identified by a domain name alice.com. Take a look at the Identity guide to learn more about that.

Record types#

If you're noticing the "type" field and wondering how that works, see the Intro to Lexicon guide.

Here is a short list of types that are currently used by the ATP software.
You'll notice "cids" in some of the schemas. A "cid" is a "Content ID," a sha256 hash of some referenced content. These are used to ensure integrity; for instance, a like includes the cid of the post being liked so that a future edit can be detected and noted in the UI.

app.bsky.graph.follow#
A social follow. Example:

```
{
  $type: 'app.bsky.graph.follow',
  subject: {
    did: 'did:plc:bv6ggog3tya2z3vxsub7hnal',
    declarationCid: 'bafyreid27zk7lbis4zw5fz4podbvbs4fc5ivwji3dmrwa6zggnj4bnd57u'
  },
  createdAt: '2022-10-10T00:39:08.609Z'
}
```

app.bsky.feed.like#
A like on a piece of content. Example:

```
{
  $type: 'app.bsky.feed.like',
  subject: {
    uri: 'at://did:plc:bv6ggog3tya2z3vxsub7hnal/app.bsky.post/1',
    cid: 'bafyreif5lqnk3tgbhi5vgqd6wy5dtovfgndhwta6bwla4iqaohuf2yd764'
  }
  createdAt: '2022-10-10T00:39:08.609Z'
}
```

app.bsky.feed.post#
A microblog post. Example:

```
{
  $type: 'app.bsky.feed.post',
  text: 'Hello, world!',
  createdAt: '2022-10-10T00:39:08.609Z'
}
```

app.bsky.actor.profile#

A user profile. Example:

```
{
  $type: 'app.bsky.actor.profile',
  displayName: 'Alice',
  description: 'A cool hacker'
}
```

app.bsky.feed.repost#

A repost of an existing microblog post (similar to retweets). Example:

```
{
  $type: 'app.bsky.feed.repost',
  subject: {
    uri: 'at://did:plc:bv6ggog3tya2z3vxsub7hnal/app.bsky.post/1',
    cid: 'bafyreif5lqnk3tgbhi5vgqd6wy5dtovfgndhwta6bwla4iqaohuf2yd764'
  }
  createdAt: '2022-10-10T00:39:08.609Z'
}
```

Social APIs#

While there's a lot that can be done by repo CRUD and other low-level com.atproto.* APIs, the app.bsky.* lexicon provides more powerful and easy-to-use APIs for social applications.

```
await api.app.bsky.feed.getTimeline()
await api.app.bsky.feed.getAuthorFeed({author: 'alice.com'})
await api.app.bsky.feed.getPostThread({uri: 'at://alice.com/app.bsky.post/1'})
await api.app.bsky.feed.getLikes({uri: 'at://alice.com/app.bsky.post/1'})
await api.app.bsky.feed.getRepostedBy({uri: 'at://alice.com/app.bsky.post/1'})
await api.app.bsky.actor.getProfile({user: 'alice.com'})
await api.app.bsky.graph.getFollowers({user: 'alice.com'})
await api.app.bsky.graph.getFollows({user: 'alice.com'})
await api.app.bsky.notification.listNotifications()
await api.app.bsky.notification.getUnreadCount()
await api.app.bsky.notification.updateSeen()
```

FAQ#

Frequently Asked Questions about the Authenticated Transfer Protocol (ATP). For FAQ about Bluesky, visit here.

Is the AT Protocol a blockchain?#

No. The AT Protocol is a federated protocol. It's not a blockchain nor does it use a blockchain.

Why not use ActivityPub?#

ActivityPub is a federated social networking technology popularized by Mastodon.

Account portability is the major reason why we chose to build a separate protocol. We consider portability to be crucial because it protects users from sudden bans, server shutdowns, and policy disagreements. Our solution for portability requires both signed data repositories and DIDs, neither of which are easy to retrofit into ActivityPub. The migration tools for ActivityPub are comparatively limited; they require the original server to provide a redirect and cannot migrate the user's previous data.

Other smaller differences include: a different viewpoint about how schemas should be handled, a preference for domain usernames over AP’s double-@ email usernames, and the goal of having large scale search and discovery (rather than the hashtag style of discovery that ActivityPub favors).

Why create Lexicon instead of using JSON-LD or RDF?#

Atproto exchanges data and RPC commands across organizations. For the data and RPC to be useful, the software needs to correctly handle schemas created by separate teams. This is the purpose of Lexicon.

We want engineers to feel comfortable using and creating new schemas, and we want developers to enjoy the DX of the system. Lexicon helps us produce strongly typed APIs which are extremely familiar to developers and which provides a variety of runtime correctness checks (which are vital in distributed systems).

RDF is intended for extremely general cases in which the systems share very little infrastructure. It’s conceptually elegant but difficult to use, often adding a lot of syntax which devs don't understand. JSON-LD simplifies the task of consuming RDF vocabularies, but it does so by hiding the underlying concepts, not by making RDF more legible.

We looked very closely at using RDF but just didn't love the developer experience (DX) or the tooling it offered.

What is “XRPC,” and why not use ___?#

XRPC is HTTP with some added conventions.

XRPC uses Lexicon to describe HTTP calls and maps them to /xrpc/{methodId}. For example, this API call:

```
await api.com.atproto.repo.listRecords({
```

```
  user: 'alice.com',
  collection: 'app.bsky.feed.post'
})
```

...maps to:

```
GET /xrpc/com.atproto.repo.listRecords
  ?user=alice.com
  &collection=app.bsky.feed.post
```

Lexicon establishes a shared method id (com.atproto.repo.listRecords) and the expected query params, input body, and output body. By using Lexicon we get runtime checks on the inputs and outputs of the call, and can generate typed code like the API call example above.

Work in progress

This document is not complete. Please check back soon for updates.

Authenticated Transfer Protocol# Glossary#

* Client: The application running on the user's device. Interacts with the network through a PDS.
* Personal Data Server (PDS): A server hosting user data. Acts as the user's personal agent on the network.
* Name server. A server mapping domains to DIDs via the com.atproto.identity.resolveHandle() API. Often a PDS.
* Big Graph Service (BGS). A service that handles all of your events, like retrieving large-scale metrics (likes, reposts, followers), content discovery (algorithms), and user search.

Wire protocol (XRPC)#

Atproto uses a light wrapper over HTTPS called XRPC. XRPC uses Lexicon, a global schema

system, to unify behaviors across hosts. The atproto.com lexicons enumerate all XRPC methods used in ATP.

Identifiers#

The following identifiers are used in atproto:

Identifier	Usage
Domain names	A unique global identifier which weakly identify repositories.
DID	A unique global identifier which strongly identify repositories.
NSID	A unique global identifier which identifies record types and XRPC methods.
TID	A timestamp-based ID which identifies records.

Domain names#

Domain names (aka "handles") weakly identify repositories. They are a convenience which should be used in UIs but rarely used within records to reference data as they may change at any time. The repo DID is preferred to provide a stable identifier.

DIDs#

DIDs are unique global identifiers which strongly identify repositories. They are considered "strong" because they should never change during the lifecycle of a user. They should rarely be used in UIs, but should always be used in records to reference data.

Atproto supports two DID methods:

* Web (did:web). Should be used only when the user is "self-hosting" and therefore directly controls the domain name & server. May also be used during testing.
* Placeholder (did:plc). A method developed in conjunction with atproto to provide global secure IDs which are host-independent.

DIDs resolve to "DID Documents" which provide the address of the repo's host and the public key used to sign the repo's updates.

Timestamp IDs (TID)#

Describe TIDs

URI scheme#

Atproto uses the at:// URI scheme (specified here). Some example at URLs:

Repository	at://alice.host.com
Repository	at://did:plc:bv6ggog3tya2z3vxsub7hnal
Collection	at://alice.host.com/io.example.song
Record	at://alice.host.com/io.example.song/3yl5-c1z-cc2p-1a
Record Field	at://bob.com/io.example.song/3yl5-c1z-cc2p-1a#/title

Schemas#

Atproto uses strict schema definitions for XRPC methods and record types. These schemas are identified using NSIDs and defined using Lexicon.

Repositories#

A data repository is a collection of signed records.

It is an implementation of a Merkle Search Tree (MST). The MST is an ordered, insert-order-independent, deterministic tree. Keys are laid out in alphabetic order. The key insight of an MST is that each key is hashed and starting 0s are counted to determine which layer it falls on (5 zeros for ~32 fanout).

This is a Merkle tree, so each subtree is referred to by its hash (CID). When a leaf is changed, every tree on the path to that leaf is changed as well, thereby updating the root hash.

Repo data layout#

Provide a more detailed description of the data layout and how the MST is organized.

The repository data layout establishes the units of network-transmissible data. It includes the following three major groupings:

Grouping

Description

Repository

Repositories are the dataset of a single "user" in the atproto network. Every user has a single repository which is identified by a DID.

Collection

A collection is an ordered list of records. Every collection is identified by an NSID. Collections only contain records of the type identified by their NSID.

Record

A record is a key/value document. It is the smallest unit of data which can be transmitted over the network. Every record has a type and is identified by a TID.

Every node is an IPLD object (dag-cbor to be specific) which is referenced by a CID hash.

Node Type

Description

Signed Root ("commit")

The Signed Root, or “commit”, is the topmost node in a repo. It contains:

* root The CID of the Root node.
* sig A signature.

Root

The Root node contains:

* did The DID of this repository.
* prev The CID(s) of the previous commit node(s) in this repository’s history.
* data The Merkle Search Tree topmost node.
* auth_token The jwt-encoded UCAN that gives authority to make the write which produced this root.

MST Node

The Merkle Search Tree Nodes contain:

* l (Optional) The CID of the leftmost subtree.
* e An array of MST Entries.

MST Entry

The Merkle Search Tree Entries contain:

* p Prefix count of utf-8 chars that this key shares with the prev key.
* k The rest of the key outside the shared prefix.
* v The CID of the value of the entry.
* t (Optional) The CID of the next subtree (to the right of the leaf).

Repo encodings#

All data in the repository is encoded using CBOR. The following value types are supported:

null

A CBOR simple value (major type 7, subtype 24) with a simple value of 22 (null).

boolean

A CBOR simple value (major type 7, subtype 24) with a simple value of 21 (true) or 20 (false).

integer

A CBOR integer (major type 0 or 1), choosing the shortest byte representation.

float

A CBOR floating-point number (major type 7). All floating point values MUST be encoded as 64-bits (additional type value 27), even for integral values.

string

A CBOR string (major type 3).

list

A CBOR array (major type 4), where each element of the list is added, in order, as a value of the array according to its type.

map

A CBOR map (major type 5), where each entry is represented as a member of the CBOR map. The entry key is expressed as a CBOR string (major type 3) as the key.

Are we missing value types? Binary? CID/Link?

Repo CBOR normalization#

Describe normalization algorithm

Repo records#

Repo records are CBOR-encoded objects (using only JSON-compatible CBOR types). Each record has a "type" which is defined by a lexicon. The type defines which collection will contain the record as well as the expected schema of the record.

The AT Protocol uses dollar ($) prefixed fields as system fields. The following fields are given a system-meaning:

Field

Usage

$type

Declares the type of a record. (Required on records and Union objects)

Client-to-server API#

The client-to-server API drives communication between a client application and the user's PDS. The APIs are dictated by the lexicons implemented by the PDS. It's recommended that every PDS support the full atproto.com lexicon. Application-level lexicons such as bsky.app are also recommended.

Authentication#

Authentication is a simple session-oriented process. View the API call in the applications model section of the docs here.

App passwords#

We also have app passwords, an initial solution for authentication that will let users use third-party clients without having to trust them with their primary password. In the long term, we plan add SSO (Single Sign-On) authentication with scoped permissions.

Users can log into third-party apps with an app password in the same way that they login with their account password. App passwords have most of the same abilities as the user's account password, but they're restricted from destructive actions such as account deletion or account migration. They are also restricted from creating additional app passwords.

No client changes are required to adopt app passwords. However, we strongly encourage you to prompt users to use an app password on login and avoid ever entering their password. For account creation, we encourage redirecting a user to the Bluesky web client.

If you expect users have used their primary password with your client, we also strongly encourage you to delete all existing refresh tokens and re-fetch access/refresh tokens using an app password.

App passwords are of the form xxxx-xxxx-xxxx-xxxx. For your users' safety, you could run a quick check to ensure that they are logging in with an app password and not their account password.

For users to generate an app password, navigate to Settings > Advanced > App passwords.

Atproto core lexicon#

The com.atproto.* lexicons provides the following behaviors:

* com.atproto.identity. Handle resolution and changes.
* com.atproto.server. Account and session management.
* com.atproto.moderation. Moderation reporting.
* com.atproto.repo. Repo CRUD operations.
* com.atproto.sync. Repo content sync and streaming.

Additional lexicons#

For atproto to be practically useful, it needs to support a variety of sophisticated queries and behaviors. While these sophisticated behaviors could be implemented on the user device, doing so would perform more slowly than on the server. Therefore, the PDS is expected to implement lexicons which provide higher-level APIs. The reference PDS created by Bluesky implements the bsky.app lexicon.

Server-to-server API#

The server-to-server APIs enable federation, event delivery, and global indexing. They may also be used to provide application behaviors such as mail delivery and form submission.

Authentication#

Describe how servers may authenticate with each other

AT URI Scheme#

The at URL scheme is used to address records in the atproto network.

```
atp-url   = "at://" authority path [ "#" fragment ]
authority = reg-name / did
path      = [ "/" coll-nsid [ "/" record-id ] ]
coll-nsid = nsid
record-id = 1*pchar
```

did is defined in https://w3c.github.io/did-core/#did-syntax.
reg-name is defined in https://www.rfc-editor.org/rfc/rfc3986#section-3.2.2.
nsid is defined in NameSpaced IDs (NSID).
pchar is defined in https://www.rfc-editor.org/rfc/rfc3986#section-3.3.
fragment is defined in https://www.rfc-editor.org/rfc/rfc3986#section-3.5.
The fragment segment only has meaning if the URL references a record. Its value maps according to "Field pathing" below.
Some example at URLs:
Repository

at://alice.host.com

Repository

at://did:plc:bv6ggog3tya2z3vxsub7hnal

Collection

at://alice.host.com/io.example.song

Record

at://alice.host.com/io.example.song/3yl5-c1z-cc2p-1a

Record Field

at://alice.host.com/io.example.song/3yl5-c1z-cc2p-1a#/title

Field pathing#

All fields in atproto records are addressed using JSON Pointers in the fragment section of the URL.

```
const obj = {
  "foo": 10,
  "arr": [
    { "key": "value1" },
```

```
    { "key": "value2" }
  ]
}
```

```
get(obj, '#/foo') // => 10
get(obj, '#/arr') // => [Object, Object]
get(obj, '#/arr/0/key') // => "value1"
get(obj, '#/arr/1/key') // => "value2"
```


Work in progress

This document is not complete. Please check back soon for updates.

Lexicon#

Lexicon is a schemas document format used to define XRPC methods and repository record types. Every Lexicon schema is written in JSON and follows the interface specified below. The schemas are identified using NSIDs which are then used to identify the methods or record types they describe.

Lexicon URIs#

```
alpha     = "a" / "b" / "c" / "d" / "e" / "f" / "g" / "h" / "i" / "j" / "k" / "l" / "m" / "n" / "o" / "p" / "q" / "r" / "s" / "t" / "u" / "v" / "w" / "x" / "y" / "z" / "A" / "B" / "C" / "D" / "E" / "F" / "G" / "H" / "I" / "J" / "K" / "L" / "M" / "N" / "O" / "P" / "Q" / "R" / "S" / "T" / "U" / "V" / "W" / "X" / "Y" / "Z"
number    = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" / "0"
def-id    = alpha *( alpha / number )
uri       = 'lex:' nsid ['#' def-id]
```

nsid is defined in the NSID spec.

The def-id maps to the keys of the defs subobject within a document. If no def-id is specified, the main definition is referenced.

Interface#

```
// core
// =



type LexRef = string



interface LexiconDoc {
  lexicon: 1
  id: string // an NSID
  revision?: number
  description?: string
  defs: Record<string, LexUserType|LexArray|LexPrimitive|LexRef[]>
}



interface LexUserType {
  type: 'query'
    | 'procedure'
    | 'record'
    | 'token'
    | 'object'
    | 'blob'
    | 'image'
    | 'video'
    | 'audio'
  description?: string
}



interface LexToken extends LexUserType {
  type = 'token'
}
```

```
interface LexObject extends LexUserType {
  type = 'object'
  required?: string[]
  properties: Record<string, LexRef | LexArray | LexPrimitive | LexRef[]>
}



// database
// =



interface LexRecord extends LexUserType {
  type = 'record'
  key?: string
  record: LexObject
}



// XRPC
// =



interface LexXrpcQuery extends LexUserType {
  type = 'query'
  parameters?: Record<string, LexPrimitive>
  output?: LexXrpcBody
  errors?: LexXrpcError[]
}
```

```
interface LexXrpcProcedure extends LexUserType {
  type = 'procedure'
  parameters?: Record<string, LexPrimitive>
  input?: LexXrpcBody
  output?: LexXrpcBody
  errors?: LexXrpcError[]
}



interface LexXrpcBody {
  description?: string
  encoding: string|string[]
  schema: LexObject
}



interface LexXrpcError {
  name: string
  description?: string
}



// blobs
// =



interface LexBlob extends LexUserType {
  type = 'blob'
  accept?: string[]
  maxSize?: number
}



interface LexImage extends LexUserType {
```

```
  type = 'image'
  accept?: string[]
  maxSize?: number
  maxWidth?: number
  maxHeight?: number
}



interface LexVideo extends LexUserType {
  type = 'video'
  accept?: string[]
  maxSize?: number
  maxWidth?: number
  maxHeight?: number
  maxLength?: number
}



interface LexAudio extends LexUserType {
  type = 'audio'
  accept?: string[]
  maxSize?: number
  maxLength?: number
}



// primitives
// =



interface LexArray {
  type = 'array'
  description?: string
  items: LexRef | LexPrimitive | LexRef[]
  minLength?: number
  maxLength?: number
```

```
}
```

```
interface LexPrimitive {
  type: 'boolean' | 'number' | 'integer' | 'string'
  description?: string
}
```

```
interface LexBoolean extends LexPrimitive {
  type = 'boolean'
  default?: boolean
  const?: boolean
}
```

```
interface LexNumber extends LexPrimitive {
  type = 'number'
  default?: number
  minimum?: number
  maximum?: number
  enum?: number[]
  const?: number
}
```

```
interface LexInteger extends LexPrimitive {
  type = 'integer'
  default?: number
  minimum?: number
  maximum?: number
  enum?: number[]
  const?: number
}
```

```
interface LexString extends LexPrimitive {
  type = 'string'
  default?: string
  minLength?: number
  maxLength?: number
  enum?: string[]
  const?: string
  knownValues?: string[]
}
```

Examples# XRPC Method#

```
{
  "lexicon": 1,
  "id": "com.atproto.account.create",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Create an account.",
      "input": {
        "encoding": "application/json",
        "schema": "#inputSchema"
      },
      "output": {
        "encoding": "application/json",
        "schema": "#outputSchema"
      },
      "errors": [
        {"name": "InvalidHandle"},
        {"name": "InvalidPassword"},
        {"name": "InvalidInviteCode"},
        {"name": "HandleNotAvailable"}
      ]
    },
    "inputSchema": {
      "type": "object",
      "required": ["handle", "email", "password"],
      "properties": {
        "email": {"type": "string"},
        "handle": {"type": "string"},
```

```
        "inviteCode": {"type": "string"},
        "password": {"type": "string"},
        "recoveryKey": {"type": "string"}
      }
    },
    "outputSchema": {
      "type": "object",
      "required": ["accessJwt", "refreshJwt", "handle", "did", "declarationCid"],
      "properties": {
        "accessJwt": { "type": "string" },
        "refreshJwt": { "type": "string" },
        "handle": { "type": "string" },
        "did": { "type": "string" },
        "declarationCid": { "type": "string" }
      }
    }
  }
}
```

ATP Record Type#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.post",
  "defs": {
    "main": {
      "type": "record",
      "key": "tid",
      "record": {
        "type": "object",
        "required": ["text", "createdAt"],
        "properties": {
          "text": {"type": "string", "maxLength": 256},
          "entities": {"type": "array", "items": "#entity"},
          "reply": "#reply",
          "createdAt": {"type": "string"}
        }
      }
```

```
    },
    "reply": {
      "type": "object",
      "required": ["root", "parent"],
      "properties": {
        "root": "#postRef",
        "parent": "#postRef"
      }
    },
    "postRef": {
      "type": "object",
      "required": ["uri", "cid"],
      "properties": {
        "uri": {"type": "string"},
        "cid": {"type": "string"}
      }
    },
    "entity": {
      "type": "object",
      "required": ["index", "type", "value"],
      "properties": {
        "index": "#textSlice",
        "type": {
          "type": "string",
          "description": "Expected values are 'mention', 'hashtag', and 'link'."
        },
        "value": {"type": "string"}
      }
    },
    "textSlice": {
      "type": "object",
      "required": ["start", "end"],
      "properties": {
        "start": {"type": "integer", "minimum": 0},
        "end": {"type": "integer", "minimum": 0}
      }
    }
  }
}
```

Token#

```
{
  "lexicon": 1,
  "id": "com.socialapp.actorUser",
  "def": {
    "main": {
      "type": "token",
      "description": "Actor type of 'User'"
    }
  }
}
```

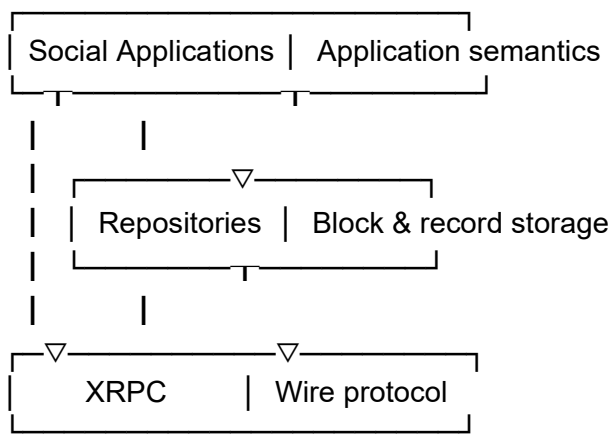
ATP
DocumentationCommunity
1. Home
2. Docs
3. XRPC

Work in progress

This document is not complete. Please check back soon for updates.

XRPC#

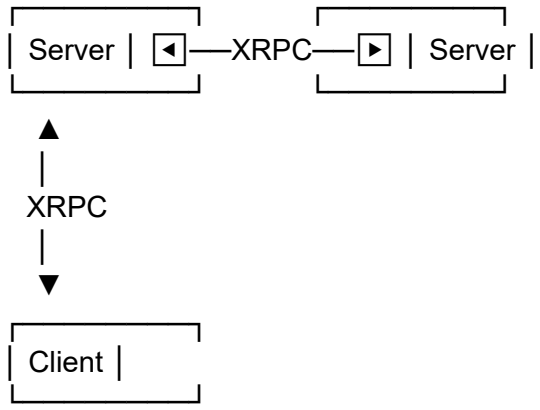
XRPC is a general purpose server-to-server messaging protocol. It was created for the AT Protocol but is a generic communications layer which can be applied to multiple use-cases (and which does not include any atproto-specific semantics). The repository data layer and social applications operate as layers atop XRPC.



Features:
* Contract-oriented. All "methods" in XRPC are declared by schemas which define the accepted inputs and outputs. Schemas are globally identified and published as machine-readable documents. This helps ensure correctness and consistency across an open network of services.
* HTTP-based. XRPC methods are transported using HTTP/S, using GET or POST methods depending on the behavior. This makes XRPC easy to understand and easy to integrate into existing tech stacks.
* Cacheable. XRPC's "query" methods are designed to cache well with common HTTP-based caching techniques.
* Support for multiple encodings. XRPC supports structured data (JSON) and unstructured binary blobs.

TODOs#
* Authentication
* Schema versioning & extensibility
* Define getSchema

Specification#
XRPC supports client-to-server and server-to-server messaging over HTTP/S. Each user has a "Personal Data Server (PDS)" which acts as their agent in the network, meaning most (if not all) of their communication is routed through their PDS.



Methods#
XRPC "Methods" possess the following attributes:
* ID: The ID of the schema for the method's inputs and outputs.
* Type: Query (non-effectful, cacheable) or Procedure (effectful, non-cacheable).
* Parameters: Encoded in the URI query segment. Affects caching.
* Input: The request body.
* Output: The response body.

Calls to a method must specify the ID, Parameters, Input, and certain HTTP Headers for the request. Likewise the return value must provide some information about the HTTP response. Therefore XRPC does not fully abstract away the semantics of HTTP when used in APIs.
Method IDs#
Methods are identified using NSIDs, a form of Reverse Domain-Name Notation.
Some example method IDs:
com.example.status()
io.social.getFeed()
net.users.bob.ping()

Method schemas#
Method schemas are encoded in JSON using Lexicon Schema Documents.
Schema distribution#
Method schemas are designed to be machine-readable and network-accessible. While it is not currently required that a schema is available on the network, it is strongly advised to publish schemas so that a single canonical & authoritative representation is available to consumers of the method.
To fetch a schema, a request must be sent to the builtin getSchema method. This request is sent to the authority of the NSID.
Built-in methods# getSchema#
TODO
Requests# HTTP Method#
The HTTP Method used depends on the type specified by the method schema.
Type
Method
query
GET
procedure
POST
Path#
All requests are sent to the /xrpc/{methodId} path on the target server. For example, a call to the io.social.getFeed method would be sent to /xrpc/io.social.getFeed path.
The parameters (as specified in the Method schema) are encoded as query parameters. The values should be encoded using the following algorithm in pseudo-javascript:

```
function encodeParam (paramType, value) {
  if (paramType === 'boolean') {
    return value ? 'true' : 'false'
  } else {
    return encodeURIComponent(value)
  }
}
```

If a default value is specified in the method schema, that value should be included in requests to ensure consistent caching behaviors.

Headers#

Header

Usage

Content-Type

Must specify the encoding of the request body if present.

Authorization

May specify the authentication data if present. See Authentication for more information.

Responses#

Response types are identified by the HTTP status codes.

200 Request successful#

The request has succeeded. Expectations:

* Content-Type header must be populated.
* Response body will vary by the method interface.

400 Invalid request#

The request is invalid and was not processed.

401 Authentication required#

The request cannot be processed without authentication. Expectations:

* WWW-Authenticate header must be populated with an authentication challenge. See Authentication for more information.

403 Forbidden#

The user lacks the needed permissions to access the method.

404 XRPC not supported#

The interpretation of a 404 response is somewhat unique for XRPC. A 404 indicates that the server does not provide a resource at the given location (/xrpc) meaning the server does not support XRPC.

To indicate that the given procedure is not implemented, use the 501 response.

413 Payload too large#

The payload of the request is larger than the server is willing to process. Payload size-limits are decided by each server.

429 Rate limit exceeded#

The client has sent too many requests. Rate-limits are decided by each server. Expectations:

* Retry-After header may be populated with the amount of time that must pass before the next request.

500 Internal server error#

The server reached an unexpected condition during processing.

501 Method not implemented#

The server does not implement the requested method.

502 A request to upstream failed#

The execution of the procedure depends on a call to another server which has failed.
503 Not enough resources#
The server is under heavy load and can't complete the request.
504 A request to upstream timed out#
The execution of the procedure depends on a call to another server which timed out.
Remaining codes#
Any response code not explicitly enumerated should be handled as follows:
* 1xx treat as a 404
* 2xx treat as a 200
* 3xx treat as a 404 (redirects are not supported)
* 4xx treat as a 400
* 5xx treat as a 500
Authentication#
TODO
Custom error codes and descriptions#
In non-200 (error) responses, services may respond with a JSON body which matches the following schema:

```
interface XrpcErrorDescription {
  error?: string
  message?: string
}
```

The error field of the response body should map to an error name defined in the method's Lexicon schema. This enables more specific error-handling by client software. This is especially advised on 400, 500, and 502 responses where further information will be useful.

NameSpaced IDs (NSIDs)#
NameSpaced IDs (NSIDs) are used throughout ATP to identify methods, records types, and other semantic information.
NSIDs use Reverse Domain-Name Notation with the additional constraint that the segments prior to the final segment must map to a valid domain name. For instance, the owner of example.com could use the ID of com.example.foo but could not use com.example.foo.bar

unless they also control foo.example.com. These rules are to ensure that schemas are globally unique, have a clear authority mapping (to a registered domain), and can potentially be resolved by request.
Some example NSIDs:
com.example.status
io.social.getFeed
net.users.bob.ping

Grammar#
alpha = "a" / "b" / "c" / "d" / "e" / "f" / "g" / "h" / "i" / "j" / "k" / "l" / "m" / "n" / "o" / "p" / "q" / "r" / "s" / "t" / "u" / "v" / "w" / "x" / "y" / "z" / "A" / "B" / "C" / "D" / "E" / "F" / "G" / "H" / "I" / "J" / "K" / "L" / "M" / "N" / "O" / "P" / "Q" / "R" / "S" / "T" / "U" / "V" / "W" / "X" / "Y" / "Z"
number = "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8" / "9" / "0"
delim = "."
segment = alpha *(alpha / number / "-")
authority = segment *(delim segment)
name = segment
nsid = authority delim name
nsid-ns = authority delim "*"

The nsid-ns (a "namespace") can be used in certain situations to designate all names under a namespace, eg com.example.*.
Authority model#
Every NSID asserts a single authority which is identified as the segments prior to the final segment which are then reversed.
com.example.thing
^^^^^^^^^^--------> example.com

The authority controls the namespace of all names within it, however there is no hierarchy or relationship between authorities. That is, the domain example.com does not hold any authority over sub.example.com, and therefore com.example.* is considered completely independent of com.example.sub.*.
Parsing#
NSIDs are comprised of an "authority" and a "name". Some example resolutions of NSIDs to authorities and names:

NSID

Authority

Name

com.example.status

example.com

status

io.social.getFeed

social.io

getFeed

net.users.bob.ping

bob.users.net

ping

The domain can be extracted through the following algorithm:

```
function getNSIDAuthority (nsid) {
  // split the nsid into segments
  const parts = nsid.split('.')
  // remove the last segment
  parts.pop()
  // reverse the order of the segments
  parts.reverse()
  // rejoin the segments
  return parts.join('.')
}
```

The name can be extracted through the following algorithm:

```
function getNSIDName (nsid) {
  // split the nsid into segments
  const parts = nsid.split('.')
  // return the last segment
  return parts.pop()
}
```

ATP

DocumentationCommunity

Work in progress
This document is not complete. Please check back soon for updates.
DID Placeholder (did:plc)#
DID Placeholder is a cryptographic, strongly-consistent, and recoverable DID method.
Motivation#
We introduced DID Placeholder because we weren't totally satisfied with any of the existing DID methods. We wanted a strongly consistent, highly available, recoverable, and cryptographically secure method with cheap and fast propagation of updates.
We cheekily titled the method "Placeholder," because we don't want it to stick around. We're actively hoping to replace it with something less centralized. We expect a method to emerge that fits the bill within the next few years, likely a permissioned DID consortium.
How it works#
This is not a fully-expressive DID format. Though it adheres to the DID spec, it is domain-specific and only allows for representing specific data types in a specific manner. There is the possibility that it could be extended to be more general in the future.
Each DID document is made up of just four pieces of data (for now):
* signingKey
* recoveryKey
* username
* atpPds (Personal Data Server for the related AT Protocol repository)

DID documents are derived from a log of signed operations, ordered by the PLC server. There are 5 operations that can be found in each log: create, rotate_signing_key, rotate_recovery_key, update_username, and update_atp_pds.
Each operation is of the shape:

```
type Operation = {
  type: string // operation type
  prev: CID | null // pointer to the CID of the previous operation in the log
  sig: string // base64url encoded signature of the operation
  ... // other operation-specific data
}
```

Each operation contains a reference to the immediately preceding operation in the log and is signed by either the signingKey or the recoveryKey.

The DID itself is derived from the sha256 hash of the first operation in the log. It is then base32 encoded and truncated to 24 chars.
To illustrate: did:plc:${base32Encode(sha256(createOp)).slice(0,24)}
Operations are verified, ordered, and made available by the PLC server.
The PLC server is constrained in its capabilities. The operation logs are fully self-certifying, with the exception of their ordering.
Therefore, the PLC server's attacks are limited to:
* Denial of service: rejecting valid operations, or refusing to serve some information about the DID
* Misordering: In the event of a fork in DID document history, the server could choose to serve the "wrong" fork
Signing and Recovery Keys#
Both the signingKey and the recoveryKey are permissioned to make changes to the DID document. However, these keys are not equal.
As can be seen in the example document (below), only the signingKey is granted the ability to make assertions and invoke/delegate capabilities.
The recovery key on the other hand is capable of performing a "recovery operation"
Account Recovery#
The PLC server provides a 72hr window during which the recoveryKey can "rewrite" history. This is to be used in adversarial situations in which a user's signingKey leaks or is being held by some custodian who turns out to be a bad actor.
In a situation such as this, the recoveryKey may be used to rotate both the signingKey and recoveryKey.
If a user wishes to recover from this situation, they sign a new operation rotating the signingKey to a key that they hold and set the prev of that operation to point to the most recent pre-attack operation.
Example#
Consider the following operation log:

```
[
  {
    type: 'create',
    signingKey: 'did:key:zDnaejYFhgFiVF89LhJ4UipACLKuqo6PteZf8eKDVKeExXUPk',
    recoveryKey: 'did:key:zDnaeSezF2TgCD71b5DiiFyhHQwKAfsBVqTTHRMvP597Z5Ztn',
    username: 'alice.example.com',
    service: 'https://example.com',
    prev: null,
    sig:
'vi6JAl5W4FfyViD5_BKL9p0rbI3MxTWuh0g_egTFAjtf7gwoSfSe1O3qMOEUPX6QH3H0Q9M4y
7gOLGblWkEwfQ'
  },
  {
    type: 'update_username',
    username: 'ali.example2.com',
    prev: 'bafyreih2gihqzgq5qd6uqktyfpyxqxvpdnrpu2qunnkaxugbyquxumisuq',
```

```
    sig: 'KL98ORpGmAJTqDsC9mWAYbhoDIv_-
eZ3Nv0YqiPkbgx0ra96gYa3fQhIpZVxXFyNbu_4Y3JhPCvyJb8yDMe9Sg'
  },
  {
    type: 'update_atp_pds',
    service: 'https://example2.com',
    prev: 'bafyreickw7v7mwncrganw645agsmwjciolknt4f6f5an5wt3nrjepqaoiu',
    sig: 'AS-APea3xxR5-
sq2i5v9IOsgbM5G5qAnB92tExZ8Z4vEy_GQbV8jmfY7zTx76P88AVXInZsO6yWX4UO7_xAIfg'
  },
  {
    type: 'rotate_signing_key',
    key: 'did:key:zDnaeh9v2RmcMo13Du2d6pjUf5bZwtauYxj3n9dYjw4EZUAR7',
    prev: 'bafyreictfsrkdt5azni355vapqka5a7erqjsa3vv7iaf52yjlqqbzkwgga',
    sig: 'VvcCoYVDluLZghv3i6ARyk1r7m1M32BPryJlTma1HTOx2CdbmIOUkVUbFa2LWi571fe-
2yjTWY0IEAKfRiPAZg'
  },
  {
    type: 'rotate_recovery_key',
    key: 'did:key:zDnaedvvAsDE6H3BDdBejpx9ve2Tz95cymyCAKF66JbyMh1Lt',
    prev: 'bafyreiazzldal6642usrcowrpztb5gjb73qla343ifnt5dfbxz4swmf5vi',
    sig: 'Um1GVZZT9JgB2SKEbwoF4_Sip05QjH7r_g-Hcx7lIY-
OhIg88ZKcN_N4TgzljgBGwe6qZb0u_0Vaq0c-S2WSDg'
  }
]
```

The log produces the following document data:

```
{
  did: 'did:plc:7iza6de2dwap2sbkpav7c6c6',
  signingKey: 'did:key:zDnaeh9v2RmcMo13Du2d6pjUf5bZwtauYxj3n9dYjw4EZUAR7',
  recoveryKey: 'did:key:zDnaedvvAsDE6H3BDdBejpx9ve2Tz95cymyCAKF66JbyMh1Lt',
  username: 'ali.example2.com',
  atpPds: 'https://example2.com'
}
```

And the following DID document:

```
{
  '@context': [
```

```
    'https://www.w3.org/ns/did/v1',
    'https://w3id.org/security/suites/ecdsa-2019/v1'
  ],
  id: 'did:plc:7iza6de2dwap2sbkpav7c6c6',
  alsoKnownAs: [ 'https://ali.example2.com' ],
  verificationMethod: [
    {
      id: 'did:plc:7iza6de2dwap2sbkpav7c6c6#signingKey',
      type: 'EcdsaSecp256r1VerificationKey2019',
      controller: 'did:plc:7iza6de2dwap2sbkpav7c6c6',
      publicKeyMultibase:
'zSSa7w8s5aApu6td45gWTAAFkqCnaWY6ZsJ8DpyzDdYmVy4fARKqbn5F1UYBUMeVvYTBso
SoLvZnPdjd3pVHbmAHP'
    },
    {
      id: 'did:plc:7iza6de2dwap2sbkpav7c6c6#recoveryKey',
      type: 'EcdsaSecp256r1VerificationKey2019',
      controller: 'did:plc:7iza6de2dwap2sbkpav7c6c6',
      publicKeyMultibase:
'zRV2EDDvop2r2aKWTcCtei3NvuNEnR5ucTVd9U4CSCnJEiha2QFyTjdxoFZ6629iHxhmTMod
ThGQzX1495ZS6iD4V'
    }
  ],
  assertionMethod: [ 'did:plc:7iza6de2dwap2sbkpav7c6c6#signingKey' ],
  capabilityInvocation: [ 'did:plc:7iza6de2dwap2sbkpav7c6c6#signingKey' ],
  capabilityDelegation: [ 'did:plc:7iza6de2dwap2sbkpav7c6c6#signingKey' ],
  service: [
    {
      id: 'did:plc:7iza6de2dwap2sbkpav7c6c6#atpPds',
      type: 'AtpPersonalDataServer',
      serviceEndpoint: 'https://example2.com'
    }
  ]
}
```

SchemeLexicon
SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext

---

com.atproto.admin.defs#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.defs",
  "defs": {
    "actionView": {
      "type": "object",
      "required": [
        "id",
        "action",
        "subject",
        "subjectBlobCids",
        "reason",
        "createdBy",
        "createdAt",
        "resolvedReportIds"
      ],
      "properties": {
        "id": {
          "type": "integer"
        },
        "action": {
          "type": "ref",
          "ref": "#actionType"
        },
        "subject": {
          "type": "union",
          "refs": [
            "#repoRef",
            "com.atproto.repo.strongRef"
          ]
        },
        "subjectBlobCids": {
          "type": "array",
          "items": {
            "type": "string"
          }
        },
        "createLabelVals": {
```

```
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "negateLabelVals": {
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "reason": {
        "type": "string"
      },
      "createdBy": {
        "type": "string",
        "format": "did"
      },
      "createdAt": {
        "type": "string",
        "format": "datetime"
      },
      "reversal": {
        "type": "ref",
        "ref": "#actionReversal"
      },
      "resolvedReportIds": {
        "type": "array",
        "items": {
          "type": "integer"
        }
      }
    }
  },
  "actionViewDetail": {
    "type": "object",
    "required": [
      "id",
      "action",
      "subject",
      "subjectBlobs",
      "reason",
      "createdBy",
      "createdAt",
```

```
    "resolvedReports"
  ],
  "properties": {
    "id": {
      "type": "integer"
    },
    "action": {
      "type": "ref",
      "ref": "#actionType"
    },
    "subject": {
      "type": "union",
      "refs": [
        "#repoView",
        "#recordView"
      ]
    },
    "subjectBlobs": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "#blobView"
      }
    },
    "createLabelVals": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "negateLabelVals": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "reason": {
      "type": "string"
    },
    "createdBy": {
      "type": "string",
      "format": "did"
    },
    "createdAt": {
```

```
      "type": "string",
      "format": "datetime"
    },
    "reversal": {
      "type": "ref",
      "ref": "#actionReversal"
    },
    "resolvedReports": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "#reportView"
      }
    }
  }
},
"actionViewCurrent": {
  "type": "object",
  "required": [
    "id",
    "action"
  ],
  "properties": {
    "id": {
      "type": "integer"
    },
    "action": {
      "type": "ref",
      "ref": "#actionType"
    }
  }
},
"actionReversal": {
  "type": "object",
  "required": [
    "reason",
    "createdBy",
    "createdAt"
  ],
  "properties": {
    "reason": {
      "type": "string"
    },
    "createdBy": {
```

```
          "type": "string",
          "format": "did"
        },
        "createdAt": {
          "type": "string",
          "format": "datetime"
        }
      }
    },
    "actionType": {
      "type": "string",
      "knownValues": [
        "#takedown",
        "#flag",
        "#acknowledge"
      ]
    },
    "takedown": {
      "type": "token",
      "description": "Moderation action type: Takedown. Indicates that content should not be served by the PDS."
    },
    "flag": {
      "type": "token",
      "description": "Moderation action type: Flag. Indicates that the content was reviewed and considered to violate PDS rules, but may still be served."
    },
    "acknowledge": {
      "type": "token",
      "description": "Moderation action type: Acknowledge. Indicates that the content was reviewed and not considered to violate PDS rules."
    },
    "reportView": {
      "type": "object",
      "required": [
        "id",
        "reasonType",
        "subject",
        "reportedBy",
        "createdAt",
        "resolvedByActionIds"
      ],
      "properties": {
        "id": {
```

```
      "type": "integer"
    },
    "reasonType": {
      "type": "ref",
      "ref": "com.atproto.moderation.defs#reasonType"
    },
    "reason": {
      "type": "string"
    },
    "subject": {
      "type": "union",
      "refs": [
        "#repoRef",
        "com.atproto.repo.strongRef"
      ]
    },
    "reportedBy": {
      "type": "string",
      "format": "did"
    },
    "createdAt": {
      "type": "string",
      "format": "datetime"
    },
    "resolvedByActionIds": {
      "type": "array",
      "items": {
        "type": "integer"
      }
    }
  }
},
"reportViewDetail": {
  "type": "object",
  "required": [
    "id",
    "reasonType",
    "subject",
    "reportedBy",
    "createdAt",
    "resolvedByActions"
  ],
  "properties": {
    "id": {
```

```
      "type": "integer"
    },
    "reasonType": {
      "type": "ref",
      "ref": "com.atproto.moderation.defs#reasonType"
    },
    "reason": {
      "type": "string"
    },
    "subject": {
      "type": "union",
      "refs": [
        "#repoView",
        "#recordView"
      ]
    },
    "reportedBy": {
      "type": "string",
      "format": "did"
    },
    "createdAt": {
      "type": "string",
      "format": "datetime"
    },
    "resolvedByActions": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.admin.defs#actionView"
      }
    }
  }
},
"repoView": {
  "type": "object",
  "required": [
    "did",
    "handle",
    "relatedRecords",
    "indexedAt",
    "moderation"
  ],
  "properties": {
    "did": {
```

```
      "type": "string",
      "format": "did"
    },
    "handle": {
      "type": "string",
      "format": "handle"
    },
    "email": {
      "type": "string"
    },
    "relatedRecords": {
      "type": "array",
      "items": {
        "type": "unknown"
      }
    },
    "indexedAt": {
      "type": "string",
      "format": "datetime"
    },
    "moderation": {
      "type": "ref",
      "ref": "#moderation"
    },
    "invitedBy": {
      "type": "ref",
      "ref": "com.atproto.server.defs#inviteCode"
    }
  }
},
"repoViewDetail": {
  "type": "object",
  "required": [
    "did",
    "handle",
    "relatedRecords",
    "indexedAt",
    "moderation"
  ],
  "properties": {
    "did": {
      "type": "string",
      "format": "did"
    },
```

```
    "handle": {
      "type": "string",
      "format": "handle"
    },
    "email": {
      "type": "string"
    },
    "relatedRecords": {
      "type": "array",
      "items": {
        "type": "unknown"
      }
    },
    "indexedAt": {
      "type": "string",
      "format": "datetime"
    },
    "moderation": {
      "type": "ref",
      "ref": "#moderationDetail"
    },
    "labels": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.label.defs#label"
      }
    },
    "invitedBy": {
      "type": "ref",
      "ref": "com.atproto.server.defs#inviteCode"
    },
    "invites": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.server.defs#inviteCode"
      }
    }
  }
},
"repoRef": {
  "type": "object",
  "required": [
```

```
      "did"
    ],
    "properties": {
      "did": {
        "type": "string",
        "format": "did"
      }
    }
  },
  "recordView": {
    "type": "object",
    "required": [
      "uri",
      "cid",
      "value",
      "blobCids",
      "indexedAt",
      "moderation",
      "repo"
    ],
    "properties": {
      "uri": {
        "type": "string",
        "format": "at-uri"
      },
      "cid": {
        "type": "string",
        "format": "cid"
      },
      "value": {
        "type": "unknown"
      },
      "blobCids": {
        "type": "array",
        "items": {
          "type": "string",
          "format": "cid"
        }
      },
      "indexedAt": {
        "type": "string",
        "format": "datetime"
      },
      "moderation": {
```

```
      "type": "ref",
      "ref": "#moderation"
    },
    "repo": {
      "type": "ref",
      "ref": "#repoView"
    }
  }
},
"recordViewDetail": {
  "type": "object",
  "required": [
    "uri",
    "cid",
    "value",
    "blobs",
    "indexedAt",
    "moderation",
    "repo"
  ],
  "properties": {
    "uri": {
      "type": "string",
      "format": "at-uri"
    },
    "cid": {
      "type": "string",
      "format": "cid"
    },
    "value": {
      "type": "unknown"
    },
    "blobs": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "#blobView"
      }
    },
    "labels": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.label.defs#label"
```

```
      }
    },
    "indexedAt": {
      "type": "string",
      "format": "datetime"
    },
    "moderation": {
      "type": "ref",
      "ref": "#moderationDetail"
    },
    "repo": {
      "type": "ref",
      "ref": "#repoView"
    }
  }
},
"moderation": {
  "type": "object",
  "required": [],
  "properties": {
    "currentAction": {
      "type": "ref",
      "ref": "#actionViewCurrent"
    }
  }
},
"moderationDetail": {
  "type": "object",
  "required": [
    "actions",
    "reports"
  ],
  "properties": {
    "currentAction": {
      "type": "ref",
      "ref": "#actionViewCurrent"
    },
    "actions": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "#actionView"
      }
    },
```

```
    "reports": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "#reportView"
      }
    }
  }
},
"blobView": {
  "type": "object",
  "required": [
    "cid",
    "mimeType",
    "size",
    "createdAt"
  ],
  "properties": {
    "cid": {
      "type": "string",
      "format": "cid"
    },
    "mimeType": {
      "type": "string"
    },
    "size": {
      "type": "integer"
    },
    "createdAt": {
      "type": "string",
      "format": "datetime"
    },
    "details": {
      "type": "union",
      "refs": [
        "#imageDetails",
        "#videoDetails"
      ]
    },
    "moderation": {
      "type": "ref",
      "ref": "#moderation"
    }
  }
```

```
    },
    "imageDetails": {
      "type": "object",
      "required": [
        "width",
        "height"
      ],
      "properties": {
        "width": {
          "type": "integer"
        },
        "height": {
          "type": "integer"
        }
      }
    },
    "videoDetails": {
      "type": "object",
      "required": [
        "width",
        "height",
        "length"
      ],
      "properties": {
        "width": {
          "type": "integer"
        },
        "height": {
          "type": "integer"
        },
        "length": {
          "type": "integer"
        }
      }
    }
  }
}
```

---

com.atproto.admin.disableInviteCodes#

```
{
```

```
  "lexicon": 1,
  "id": "com.atproto.admin.disableInviteCodes",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Disable some set of codes and/or all codes associated with a set of users",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "properties": {
            "codes": {
              "type": "array",
              "items": {
                "type": "string"
              }
            },
            "accounts": {
              "type": "array",
              "items": {
                "type": "string"
              }
            }
          }
        }
      }
    }
  }
}
```

---

com.atproto.admin.getInviteCodes#

```
{
  "lexicon": 1,
  "id": "com.atproto.admin.getInviteCodes",
  "defs": {
    "main": {
      "type": "query",
      "description": "Admin view of invite codes",
      "parameters": {
        "type": "params",
```

```
        "properties": {
          "sort": {
            "type": "string",
            "knownValues": [
              "recent",
              "usage"
            ],
            "default": "recent"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 500,
            "default": 100
          },
          "cursor": {
            "type": "string"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "codes"
          ],
          "properties": {
            "cursor": {
              "type": "string"
            },
            "codes": {
              "type": "array",
              "items": {
                "type": "ref",
                "ref": "com.atproto.server.defs#inviteCode"
              }
            }
          }
        }
      }
    }
  }
}
```

---

com.atproto.admin.getModerationAction#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.getModerationAction",
  "defs": {
    "main": {
      "type": "query",
      "description": "View details about a moderation action.",
      "parameters": {
        "type": "params",
        "required": [
          "id"
        ],
        "properties": {
          "id": {
            "type": "integer"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "ref",
          "ref": "com.atproto.admin.defs#actionViewDetail"
        }
      }
    }
  }
}
```


---

com.atproto.admin.getModerationActions#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.getModerationActions",
  "defs": {
```

```
    "main": {
      "type": "query",
      "description": "List moderation actions related to a subject.",
      "parameters": {
        "type": "params",
        "properties": {
          "subject": {
            "type": "string"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
            "type": "string"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "actions"
          ],
          "properties": {
            "cursor": {
              "type": "string"
            },
            "actions": {
              "type": "array",
              "items": {
                "type": "ref",
                "ref": "com.atproto.admin.defs#actionView"
              }
            }
          }
        }
      }
    }
  }
}
```

---

com.atproto.admin.getModerationReport#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.getModerationReport",
  "defs": {
    "main": {
      "type": "query",
      "description": "View details about a moderation report.",
      "parameters": {
        "type": "params",
        "required": [
          "id"
        ],
        "properties": {
          "id": {
            "type": "integer"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "ref",
          "ref": "com.atproto.admin.defs#reportViewDetail"
        }
      }
    }
  }
}
```


---

com.atproto.admin.getModerationReports#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.getModerationReports",
  "defs": {
```

```
"main": {
  "type": "query",
  "description": "List moderation reports related to a subject.",
  "parameters": {
    "type": "params",
    "properties": {
      "subject": {
        "type": "string"
      },
      "resolved": {
        "type": "boolean"
      },
      "limit": {
        "type": "integer",
        "minimum": 1,
        "maximum": 100,
        "default": 50
      },
      "cursor": {
        "type": "string"
      }
    }
  },
  "output": {
    "encoding": "application/json",
    "schema": {
      "type": "object",
      "required": [
        "reports"
      ],
      "properties": {
        "cursor": {
          "type": "string"
        },
        "reports": {
          "type": "array",
          "items": {
            "type": "ref",
            "ref": "com.atproto.admin.defs#reportView"
          }
        }
      }
    }
  }
```

```
    }
  }
}
```

---

com.atproto.admin.getRecord#

```
{
  "lexicon": 1,
  "id": "com.atproto.admin.getRecord",
  "defs": {
    "main": {
      "type": "query",
      "description": "View details about a record.",
      "parameters": {
        "type": "params",
        "required": [
          "uri"
        ],
        "properties": {
          "uri": {
            "type": "string",
            "format": "at-uri"
          },
          "cid": {
            "type": "string",
            "format": "cid"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "ref",
          "ref": "com.atproto.admin.defs#recordViewDetail"
        }
      }
    }
  }
}
```

---

com.atproto.admin.getRepo#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.getRepo",
  "defs": {
    "main": {
      "type": "query",
      "description": "View details about a repository.",
      "parameters": {
        "type": "params",
        "required": [
          "did"
        ],
        "properties": {
          "did": {
            "type": "string",
            "format": "did"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "ref",
          "ref": "com.atproto.admin.defs#repoViewDetail"
        }
      }
    }
  }
}
```


---

com.atproto.admin.resolveModerationReports#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.resolveModerationReports",
  "defs": {
    "main": {
```

```
    "type": "procedure",
    "description": "Resolve moderation reports by an action.",
    "input": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "actionId",
          "reportIds",
          "createdBy"
        ],
        "properties": {
          "actionId": {
            "type": "integer"
          },
          "reportIds": {
            "type": "array",
            "items": {
              "type": "integer"
            }
          },
          "createdBy": {
            "type": "string",
            "format": "did"
          }
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "ref",
        "ref": "com.atproto.admin.defs#actionView"
      }
    }
  }
 }
}
```

---

com.atproto.admin.reverseModerationAction#

```
{
  "lexicon": 1,
  "id": "com.atproto.admin.reverseModerationAction",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Reverse a moderation action.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "id",
            "reason",
            "createdBy"
          ],
          "properties": {
            "id": {
              "type": "integer"
            },
            "reason": {
              "type": "string"
            },
            "createdBy": {
              "type": "string",
              "format": "did"
            }
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "ref",
          "ref": "com.atproto.admin.defs#actionView"
        }
      }
    }
  }
}
```

com.atproto.admin.searchRepos#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.searchRepos",
  "defs": {
    "main": {
      "type": "query",
      "description": "Find repositories based on a search term.",
      "parameters": {
        "type": "params",
        "properties": {
          "term": {
            "type": "string"
          },
          "invitedBy": {
            "type": "string"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
            "type": "string"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "repos"
          ],
          "properties": {
            "cursor": {
              "type": "string"
            },
            "repos": {
              "type": "array",
              "items": {
                "type": "ref",
```

```
            "ref": "com.atproto.admin.defs#repoView"
          }
        }
      }
    }
  }
 }
}
```

---

com.atproto.admin.takeModerationAction#

```
{
  "lexicon": 1,
  "id": "com.atproto.admin.takeModerationAction",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Take a moderation action on a repo.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "action",
            "subject",
            "reason",
            "createdBy"
          ],
          "properties": {
            "action": {
              "type": "string",
              "knownValues": [
                "com.atproto.admin.defs#takedown",
                "com.atproto.admin.defs#flag",
                "com.atproto.admin.defs#acknowledge"
              ]
            },
            "subject": {
              "type": "union",
              "refs": [
```

```
          "com.atproto.admin.defs#repoRef",
          "com.atproto.repo.strongRef"
        ]
      },
      "subjectBlobCids": {
        "type": "array",
        "items": {
          "type": "string",
          "format": "cid"
        }
      },
      "createLabelVals": {
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "negateLabelVals": {
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "reason": {
        "type": "string"
      },
      "createdBy": {
        "type": "string",
        "format": "did"
      }
    }
  }
},
"output": {
  "encoding": "application/json",
  "schema": {
    "type": "ref",
    "ref": "com.atproto.admin.defs#actionView"
  }
},
"errors": [
  {
    "name": "SubjectHasAction"
  }
```

```
      ]
    }
  }
}
```

---

com.atproto.admin.updateAccountEmail#

```
{
  "lexicon": 1,
  "id": "com.atproto.admin.updateAccountEmail",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Administrative action to update an account's email",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "account",
            "email"
          ],
          "properties": {
            "account": {
              "type": "string",
              "format": "at-identifier",
              "description": "The handle or DID of the repo."
            },
            "email": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

com.atproto.admin.updateAccountHandle#


```
{
  "lexicon": 1,
  "id": "com.atproto.admin.updateAccountHandle",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Administrative action to update an account's handle",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "did",
            "handle"
          ],
          "properties": {
            "did": {
              "type": "string",
              "format": "did"
            },
            "handle": {
              "type": "string",
              "format": "handle"
            }
          }
        }
      }
    }
  }
}
```

3. com.atproto.identity
GuidesProtocol OverviewIdentityData RepositoriesLexiconApplicationsFAQSpecsATPURI SchemeLexicon SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext
com.atproto.identity.resolveHandle#


```
{
  "lexicon": 1,
  "id": "com.atproto.identity.resolveHandle",
  "defs": {
    "main": {
      "type": "query",
      "description": "Provides the DID of a repo.",
      "parameters": {
        "type": "params",
        "properties": {
          "handle": {
            "type": "string",
            "format": "handle",
            "description": "The handle to resolve. If not supplied, will resolve the host's own handle."
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "did"
          ],
          "properties": {
            "did": {
              "type": "string",
              "format": "did"
            }
          }
        }
      }
    }
  }
}
```

com.atproto.identity.updateHandle#

```
{
  "lexicon": 1,
  "id": "com.atproto.identity.updateHandle",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Updates the handle of the account",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "handle"
          ],
          "properties": {
            "handle": {
              "type": "string",
              "format": "handle"
            }
          }
        }
      }
    }
  }
}
```

SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext

---

com.atproto.label.defs#


```
{
  "lexicon": 1,
  "id": "com.atproto.label.defs",
  "defs": {
    "label": {
      "type": "object",
      "description": "Metadata tag on an atproto resource (eg, repo or record)",
      "required": [
        "src",
        "uri",
        "val",
        "cts"
      ],
      "properties": {
        "src": {
          "type": "string",
          "format": "did",
          "description": "DID of the actor who created this label"
        },
        "uri": {
          "type": "string",
          "format": "uri",
          "description": "AT URI of the record, repository (account), or other resource which this label applies to"
        },
        "cid": {
          "type": "string",
          "format": "cid",
          "description": "optionally, CID specifying the specific version of 'uri' resource this label applies to"
        },
        "val": {
          "type": "string",
          "maxLength": 128,
          "description": "the short string name of the value or type of this label"
        },
        "neg": {
          "type": "boolean",
          "description": "if true, this is a negation label, overwriting a previous label"
```

```
        },
        "cts": {
          "type": "string",
          "format": "datetime",
          "description": "timestamp when this label was created"
        }
      }
    }
  }
}
```

---

com.atproto.label.queryLabels#

```
{
  "lexicon": 1,
  "id": "com.atproto.label.queryLabels",
  "defs": {
    "main": {
      "type": "query",
      "description": "Find labels relevant to the provided URI patterns.",
      "parameters": {
        "type": "params",
        "required": [
          "uriPatterns"
        ],
        "properties": {
          "uriPatterns": {
            "type": "array",
            "items": {
              "type": "string"
            },
            "description": "List of AT URI patterns to match (boolean 'OR'). Each may be a prefix (ending with '*'; will match inclusive of the string leading to '*'), or a full URI"
          },
          "sources": {
            "type": "array",
            "items": {
              "type": "string",
              "format": "did"
            },
            "description": "Optional list of label sources (DIDs) to filter on"
```

```
        },
        "limit": {
          "type": "integer",
          "minimum": 1,
          "maximum": 250,
          "default": 50
        },
        "cursor": {
          "type": "string"
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "labels"
        ],
        "properties": {
          "cursor": {
            "type": "string"
          },
          "labels": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "com.atproto.label.defs#label"
            }
          }
        }
      }
    }
  }
}
```

---

com.atproto.label.subscribeLabels#

```
{
  "lexicon": 1,
```

```
"id": "com.atproto.label.subscribeLabels",
"defs": {
  "main": {
    "type": "subscription",
    "description": "Subscribe to label updates",
    "parameters": {
      "type": "params",
      "properties": {
        "cursor": {
          "type": "integer",
          "description": "The last known event to backfill from."
        }
      }
    },
    "message": {
      "schema": {
        "type": "union",
        "refs": [
          "#labels",
          "#info"
        ]
      }
    },
    "errors": [
      {
        "name": "FutureCursor"
      }
    ]
  },
  "labels": {
    "type": "object",
    "required": [
      "seq",
      "labels"
    ],
    "properties": {
      "seq": {
        "type": "integer"
      },
      "labels": {
        "type": "array",
        "items": {
          "type": "ref",
          "ref": "com.atproto.label.defs#label"
```

```
        }
       }
      }
    },
    "info": {
      "type": "object",
      "required": [
        "name"
      ],
      "properties": {
        "name": {
          "type": "string",
          "knownValues": [
            "OutdatedCursor"
          ]
        },
        "message": {
          "type": "string"
        }
      }
    }
  }
}
```


---

com.atproto.moderation.createReport#


```
{
```

```
"lexicon": 1,
"id": "com.atproto.moderation.createReport",
"defs": {
  "main": {
    "type": "procedure",
    "description": "Report a repo or a record.",
    "input": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "reasonType",
          "subject"
        ],
        "properties": {
          "reasonType": {
            "type": "ref",
            "ref": "com.atproto.moderation.defs#reasonType"
          },
          "reason": {
            "type": "string"
          },
          "subject": {
            "type": "union",
            "refs": [
              "com.atproto.admin.defs#repoRef",
              "com.atproto.repo.strongRef"
            ]
          }
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "id",
          "reasonType",
          "subject",
          "reportedBy",
          "createdAt"
        ],
        "properties": {
```

```
          "id": {
            "type": "integer"
          },
          "reasonType": {
            "type": "ref",
            "ref": "com.atproto.moderation.defs#reasonType"
          },
          "reason": {
            "type": "string"
          },
          "subject": {
            "type": "union",
            "refs": [
              "com.atproto.admin.defs#repoRef",
              "com.atproto.repo.strongRef"
            ]
          },
          "reportedBy": {
            "type": "string",
            "format": "did"
          },
          "createdAt": {
            "type": "string",
            "format": "datetime"
          }
        }
      }
    }
  }
}
```

---

com.atproto.moderation.defs#

```
{
  "lexicon": 1,
  "id": "com.atproto.moderation.defs",
  "defs": {
    "reasonType": {
      "type": "string",
      "knownValues": [
```

```
        "com.atproto.moderation.defs#reasonSpam",
        "com.atproto.moderation.defs#reasonViolation",
        "com.atproto.moderation.defs#reasonMisleading",
        "com.atproto.moderation.defs#reasonSexual",
        "com.atproto.moderation.defs#reasonRude",
        "com.atproto.moderation.defs#reasonOther"
      ]
    },
    "reasonSpam": {
      "type": "token",
      "description": "Spam: frequent unwanted promotion, replies, mentions"
    },
    "reasonViolation": {
      "type": "token",
      "description": "Direct violation of server rules, laws, terms of service"
    },
    "reasonMisleading": {
      "type": "token",
      "description": "Misleading identity, affiliation, or content"
    },
    "reasonSexual": {
      "type": "token",
      "description": "Unwanted or mis-labeled sexual content"
    },
    "reasonRude": {
      "type": "token",
      "description": "Rude, harassing, explicit, or otherwise unwelcoming behavior"
    },
    "reasonOther": {
      "type": "token",
      "description": "Other: reports not falling under another report category"
    }
  }
}
```

---

com.atproto.moderation.createReport#


```
{
  "lexicon": 1,
  "id": "com.atproto.moderation.createReport",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Report a repo or a record.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "reasonType",
            "subject"
          ],
          "properties": {
            "reasonType": {
              "type": "ref",
              "ref": "com.atproto.moderation.defs#reasonType"
            },
            "reason": {
              "type": "string"
            },
            "subject": {
              "type": "union",
              "refs": [
                "com.atproto.admin.defs#repoRef",
                "com.atproto.repo.strongRef"
              ]
            }
          }
        }
      },
      "output": {
```

```
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "id",
            "reasonType",
            "subject",
            "reportedBy",
            "createdAt"
          ],
          "properties": {
            "id": {
              "type": "integer"
            },
            "reasonType": {
              "type": "ref",
              "ref": "com.atproto.moderation.defs#reasonType"
            },
            "reason": {
              "type": "string"
            },
            "subject": {
              "type": "union",
              "refs": [
                "com.atproto.admin.defs#repoRef",
                "com.atproto.repo.strongRef"
              ]
            },
            "reportedBy": {
              "type": "string",
              "format": "did"
            },
            "createdAt": {
              "type": "string",
              "format": "datetime"
            }
          }
        }
      }
    }
  }
}
```

com.atproto.moderation.defs#

```
{
  "lexicon": 1,
  "id": "com.atproto.moderation.defs",
  "defs": {
    "reasonType": {
      "type": "string",
      "knownValues": [
        "com.atproto.moderation.defs#reasonSpam",
        "com.atproto.moderation.defs#reasonViolation",
        "com.atproto.moderation.defs#reasonMisleading",
        "com.atproto.moderation.defs#reasonSexual",
        "com.atproto.moderation.defs#reasonRude",
        "com.atproto.moderation.defs#reasonOther"
      ]
    },
    "reasonSpam": {
      "type": "token",
      "description": "Spam: frequent unwanted promotion, replies, mentions"
    },
    "reasonViolation": {
      "type": "token",
      "description": "Direct violation of server rules, laws, terms of service"
    },
    "reasonMisleading": {
      "type": "token",
      "description": "Misleading identity, affiliation, or content"
    },
    "reasonSexual": {
      "type": "token",
      "description": "Unwanted or mis-labeled sexual content"
    },
    "reasonRude": {
      "type": "token",
      "description": "Rude, harassing, explicit, or otherwise unwelcoming behavior"
    },
    "reasonOther": {
      "type": "token",
      "description": "Other: reports not falling under another report category"
    }
  }
```

}

com.atproto.repo Lexicon#

Definitions related to repositories in ATP.

---

com.atproto.repo.applyWrites#


```
{
  "lexicon": 1,
  "id": "com.atproto.repo.applyWrites",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Apply a batch transaction of creates, updates, and deletes.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "repo",
            "writes"
          ],
          "properties": {
            "repo": {
              "type": "string",
              "format": "at-identifier",
              "description": "The handle or DID of the repo."
            },
            "validate": {
              "type": "boolean",
              "default": true,
              "description": "Validate the records?"
            },
```

```
        "writes": {
          "type": "array",
          "items": {
            "type": "union",
            "refs": [
              "#create",
              "#update",
              "#delete"
            ],
            "closed": true
          }
        },
        "swapCommit": {
          "type": "string",
          "format": "cid"
        }
      }
    }
  },
  "errors": [
    {
      "name": "InvalidSwap"
    }
  ]
},
"create": {
  "type": "object",
  "description": "Create a new record.",
  "required": [
    "action",
    "collection",
    "value"
  ],
  "properties": {
    "collection": {
      "type": "string",
      "format": "nsid"
    },
    "rkey": {
      "type": "string"
    },
    "value": {
      "type": "unknown"
    }
```

```
      }
    },
    "update": {
      "type": "object",
      "description": "Update an existing record.",
      "required": [
        "action",
        "collection",
        "rkey",
        "value"
      ],
      "properties": {
        "collection": {
          "type": "string",
          "format": "nsid"
        },
        "rkey": {
          "type": "string"
        },
        "value": {
          "type": "unknown"
        }
      }
    },
    "delete": {
      "type": "object",
      "description": "Delete an existing record.",
      "required": [
        "action",
        "collection",
        "rkey"
      ],
      "properties": {
        "collection": {
          "type": "string",
          "format": "nsid"
        },
        "rkey": {
          "type": "string"
        }
      }
    }
  }
}
```

com.atproto.repo.createRecord#


```
{
  "lexicon": 1,
  "id": "com.atproto.repo.createRecord",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Create a new record.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "repo",
            "collection",
            "record"
          ],
          "properties": {
            "repo": {
              "type": "string",
              "format": "at-identifier",
              "description": "The handle or DID of the repo."
            },
            "collection": {
              "type": "string",
              "format": "nsid",
              "description": "The NSID of the record collection."
            },
            "rkey": {
              "type": "string",
              "description": "The key of the record."
            },
            "validate": {
              "type": "boolean",
              "default": true,
              "description": "Validate the record?"
            },
            "record": {
              "type": "unknown",
```

```
            "description": "The record to create."
          },
          "swapCommit": {
            "type": "string",
            "format": "cid",
            "description": "Compare and swap with the previous commit by cid."
          }
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "uri",
          "cid"
        ],
        "properties": {
          "uri": {
            "type": "string",
            "format": "at-uri"
          },
          "cid": {
            "type": "string",
            "format": "cid"
          }
        }
      }
    },
    "errors": [
      {
        "name": "InvalidSwap"
      }
    ]
  }
 }
}
```


---

com.atproto.repo.deleteRecord#

```
{
  "lexicon": 1,
  "id": "com.atproto.repo.deleteRecord",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Delete a record, or ensure it doesn't exist.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "repo",
            "collection",
            "rkey"
          ],
          "properties": {
            "repo": {
              "type": "string",
              "format": "at-identifier",
              "description": "The handle or DID of the repo."
            },
            "collection": {
              "type": "string",
              "format": "nsid",
              "description": "The NSID of the record collection."
            },
            "rkey": {
              "type": "string",
              "description": "The key of the record."
            },
            "swapRecord": {
              "type": "string",
              "format": "cid",
              "description": "Compare and swap with the previous record by cid."
            },
            "swapCommit": {
              "type": "string",
              "format": "cid",
              "description": "Compare and swap with the previous commit by cid."
            }
          }
        }
      },
```

```
      "errors": [
        {
          "name": "InvalidSwap"
        }
      ]
    }
  }
}
```

---

com.atproto.repo.describeRepo#

```
{
  "lexicon": 1,
  "id": "com.atproto.repo.describeRepo",
  "defs": {
    "main": {
      "type": "query",
      "description": "Get information about the repo, including the list of collections.",
      "parameters": {
        "type": "params",
        "required": [
          "repo"
        ],
        "properties": {
          "repo": {
            "type": "string",
            "format": "at-identifier",
            "description": "The handle or DID of the repo."
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "handle",
            "did",
            "didDoc",
            "collections",
            "handleIsCorrect"
```

```
        ],
        "properties": {
          "handle": {
            "type": "string",
            "format": "handle"
          },
          "did": {
            "type": "string",
            "format": "did"
          },
          "didDoc": {
            "type": "unknown"
          },
          "collections": {
            "type": "array",
            "items": {
              "type": "string",
              "format": "nsid"
            }
          },
          "handleIsCorrect": {
            "type": "boolean"
          }
        }
      }
    }
  }
}
```

---

com.atproto.repo.getRecord#

```
{
  "lexicon": 1,
  "id": "com.atproto.repo.getRecord",
  "defs": {
    "main": {
      "type": "query",
      "description": "Get a record.",
      "parameters": {
        "type": "params",
```

```
        "required": [
          "repo",
          "collection",
          "rkey"
        ],
        "properties": {
          "repo": {
            "type": "string",
            "format": "at-identifier",
            "description": "The handle or DID of the repo."
          },
          "collection": {
            "type": "string",
            "format": "nsid",
            "description": "The NSID of the record collection."
          },
          "rkey": {
            "type": "string",
            "description": "The key of the record."
          },
          "cid": {
            "type": "string",
            "format": "cid",
            "description": "The CID of the version of the record. If not specified, then return the most recent version."
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "uri",
            "value"
          ],
          "properties": {
            "uri": {
              "type": "string",
              "format": "at-uri"
            },
            "cid": {
              "type": "string",
              "format": "cid"
```

```
          },
          "value": {
            "type": "unknown"
          }
        }
      }
    }
  }
}
```

---

com.atproto.repo.listRecords#

```
{
  "lexicon": 1,
  "id": "com.atproto.repo.listRecords",
  "defs": {
    "main": {
      "type": "query",
      "description": "List a range of records in a collection.",
      "parameters": {
        "type": "params",
        "required": [
          "repo",
          "collection"
        ],
        "properties": {
          "repo": {
            "type": "string",
            "format": "at-identifier",
            "description": "The handle or DID of the repo."
          },
          "collection": {
            "type": "string",
            "format": "nsid",
            "description": "The NSID of the record type."
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
```

```
        "default": 50,
        "description": "The number of records to return."
      },
      "cursor": {
        "type": "string"
      },
      "rkeyStart": {
        "type": "string",
        "description": "DEPRECATED: The lowest sort-ordered rkey to start from (exclusive)"
      },
      "rkeyEnd": {
        "type": "string",
        "description": "DEPRECATED: The highest sort-ordered rkey to stop at (exclusive)"
      },
      "reverse": {
        "type": "boolean",
        "description": "Reverse the order of the returned records?"
      }
    }
  },
  "output": {
    "encoding": "application/json",
    "schema": {
      "type": "object",
      "required": [
        "records"
      ],
      "properties": {
        "cursor": {
          "type": "string"
        },
        "records": {
          "type": "array",
          "items": {
            "type": "ref",
            "ref": "#record"
          }
        }
      }
    }
  }
},
"record": {
  "type": "object",
```

```
      "required": [
        "uri",
        "cid",
        "value"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "at-uri"
        },
        "cid": {
          "type": "string",
          "format": "cid"
        },
        "value": {
          "type": "unknown"
        }
      }
    }
  }
}
```

---

com.atproto.repo.putRecord#

```
{
  "lexicon": 1,
  "id": "com.atproto.repo.putRecord",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Write a record, creating or updating it as needed.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "repo",
            "collection",
            "rkey",
            "record"
          ],
```

```
    "nullable": [
      "swapRecord"
    ],
    "properties": {
      "repo": {
        "type": "string",
        "format": "at-identifier",
        "description": "The handle or DID of the repo."
      },
      "collection": {
        "type": "string",
        "format": "nsid",
        "description": "The NSID of the record collection."
      },
      "rkey": {
        "type": "string",
        "description": "The key of the record."
      },
      "validate": {
        "type": "boolean",
        "default": true,
        "description": "Validate the record?"
      },
      "record": {
        "type": "unknown",
        "description": "The record to write."
      },
      "swapRecord": {
        "type": "string",
        "format": "cid",
        "description": "Compare and swap with the previous record by cid."
      },
      "swapCommit": {
        "type": "string",
        "format": "cid",
        "description": "Compare and swap with the previous commit by cid."
      }
    }
  }
},
"output": {
  "encoding": "application/json",
  "schema": {
    "type": "object",
```

```
          "required": [
            "uri",
            "cid"
          ],
          "properties": {
            "uri": {
              "type": "string",
              "format": "at-uri"
            },
            "cid": {
              "type": "string",
              "format": "cid"
            }
          }
        }
      },
      "errors": [
        {
          "name": "InvalidSwap"
        }
      ]
    }
  }
}
```

---

com.atproto.repo.strongRef#
A URI with a content-hash fingerprint.

```
{
  "lexicon": 1,
  "id": "com.atproto.repo.strongRef",
  "description": "A URI with a content-hash fingerprint.",
  "defs": {
    "main": {
      "type": "object",
      "required": [
        "uri",
        "cid"
      ],
      "properties": {
        "uri": {
```

```
        "type": "string",
        "format": "at-uri"
      },
      "cid": {
        "type": "string",
        "format": "cid"
      }
    }
  }
 }
}
```

---

com.atproto.repo.uploadBlob#

```
{
 "lexicon": 1,
 "id": "com.atproto.repo.uploadBlob",
 "defs": {
  "main": {
   "type": "procedure",
   "description": "Upload a new blob to be added to repo in a later request.",
   "input": {
    "encoding": "*/*"
   },
   "output": {
    "encoding": "application/json",
    "schema": {
     "type": "object",
     "required": [
      "blob"
     ],
     "properties": {
      "blob": {
       "type": "blob"
      }
     }
    }
   }
  }
 }
}
```

com.atproto.server Lexicon#

Definitions related to server behaviors in ATP.

---

com.atproto.server.createAccount#


```
{
  "lexicon": 1,
  "id": "com.atproto.server.createAccount",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Create an account.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "handle",
            "email",
            "password"
          ],
          "properties": {
            "email": {
              "type": "string"
            },
            "handle": {
              "type": "string",
              "format": "handle"
            },
            "inviteCode": {
              "type": "string"
```

```
      },
      "password": {
        "type": "string"
      },
      "recoveryKey": {
        "type": "string"
      }
    }
  }
},
"output": {
  "encoding": "application/json",
  "schema": {
    "type": "object",
    "required": [
      "accessJwt",
      "refreshJwt",
      "handle",
      "did"
    ],
    "properties": {
      "accessJwt": {
        "type": "string"
      },
      "refreshJwt": {
        "type": "string"
      },
      "handle": {
        "type": "string",
        "format": "handle"
      },
      "did": {
        "type": "string",
        "format": "did"
      }
    }
  }
},
"errors": [
  {
    "name": "InvalidHandle"
  },
  {
    "name": "InvalidPassword"
```

```
      },
      {
        "name": "InvalidInviteCode"
      },
      {
        "name": "HandleNotAvailable"
      },
      {
        "name": "UnsupportedDomain"
      }
    ]
  }
 }
}
```

---

com.atproto.server.createAppPassword#

```
{
 "lexicon": 1,
 "id": "com.atproto.server.createAppPassword",
 "defs": {
  "main": {
   "type": "procedure",
   "description": "Create an app-specific password.",
   "input": {
    "encoding": "application/json",
    "schema": {
     "type": "object",
     "required": [
      "name"
     ],
     "properties": {
      "name": {
       "type": "string"
      }
     }
    }
   },
   "output": {
    "encoding": "application/json",
    "schema": {
```

```
          "type": "ref",
          "ref": "#appPassword"
        }
      },
      "errors": [
        {
          "name": "AccountTakedown"
        }
      ]
    },
    "appPassword": {
      "type": "object",
      "required": [
        "name",
        "password",
        "createdAt"
      ],
      "properties": {
        "name": {
          "type": "string"
        },
        "password": {
          "type": "string"
        },
        "createdAt": {
          "type": "string",
          "format": "datetime"
        }
      }
    }
  }
}
```

---

com.atproto.server.createInviteCode#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.createInviteCode",
  "defs": {
    "main": {
      "type": "procedure",
```

```
    "description": "Create an invite code.",
    "input": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "useCount"
        ],
        "properties": {
          "useCount": {
            "type": "integer"
          },
          "forAccount": {
            "type": "string",
            "format": "did"
          }
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "code"
        ],
        "properties": {
          "code": {
            "type": "string"
          }
        }
      }
    }
  }
}
```

---

com.atproto.server.createInviteCodes#

```
{
  "lexicon": 1,
```

```
"id": "com.atproto.server.createInviteCodes",
"defs": {
  "main": {
    "type": "procedure",
    "description": "Create an invite code.",
    "input": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "codeCount",
          "useCount"
        ],
        "properties": {
          "codeCount": {
            "type": "integer",
            "default": 1
          },
          "useCount": {
            "type": "integer"
          },
          "forAccounts": {
            "type": "array",
            "items": {
              "type": "string",
              "format": "did"
            }
          }
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "codes"
        ],
        "properties": {
          "codes": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "#accountCodes"
```

```
            }
          }
        }
      }
    }
  },
  "accountCodes": {
    "type": "object",
    "required": [
      "account",
      "codes"
    ],
    "properties": {
      "account": {
        "type": "string"
      },
      "codes": {
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    }
  }
}
```

---

com.atproto.server.createSession#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.createSession",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Create an authentication session.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
```

```
          "identifier",
          "password"
        ],
        "properties": {
          "identifier": {
            "type": "string",
            "description": "Handle or other identifier supported by the server for the authenticating
user."
          },
          "password": {
            "type": "string"
          }
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "accessJwt",
          "refreshJwt",
          "handle",
          "did"
        ],
        "properties": {
          "accessJwt": {
            "type": "string"
          },
          "refreshJwt": {
            "type": "string"
          },
          "handle": {
            "type": "string",
            "format": "handle"
          },
          "did": {
            "type": "string",
            "format": "did"
          },
          "email": {
            "type": "string"
          }
        }
```

```
      }
     },
     "errors": [
      {
       "name": "AccountTakedown"
      }
     ]
    }
   }
}
```

---

com.atproto.server.defs#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.defs",
  "defs": {
    "inviteCode": {
      "type": "object",
      "required": [
        "code",
        "available",
        "disabled",
        "forAccount",
        "createdBy",
        "createdAt",
        "uses"
      ],
      "properties": {
        "code": {
          "type": "string"
        },
        "available": {
          "type": "integer"
        },
        "disabled": {
          "type": "boolean"
        },
        "forAccount": {
          "type": "string"
        },
```

```
        "createdBy": {
          "type": "string"
        },
        "createdAt": {
          "type": "string",
          "format": "datetime"
        },
        "uses": {
          "type": "array",
          "items": {
            "type": "ref",
            "ref": "#inviteCodeUse"
          }
        }
      }
    },
    "inviteCodeUse": {
      "type": "object",
      "required": [
        "usedBy",
        "usedAt"
      ],
      "properties": {
        "usedBy": {
          "type": "string",
          "format": "did"
        },
        "usedAt": {
          "type": "string",
          "format": "datetime"
        }
      }
    }
  }
}
```

---

com.atproto.server.deleteAccount#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.deleteAccount",
```

```
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Delete a user account with a token and password.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "did",
            "password",
            "token"
          ],
          "properties": {
            "did": {
              "type": "string",
              "format": "did"
            },
            "password": {
              "type": "string"
            },
            "token": {
              "type": "string"
            }
          }
        }
      },
      "errors": [
        {
          "name": "ExpiredToken"
        },
        {
          "name": "InvalidToken"
        }
      ]
    }
  }
}
```

---

com.atproto.server.deleteSession#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.deleteSession",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Delete the current session."
    }
  }
}
```

---

com.atproto.server.describeServer#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.describeServer",
  "defs": {
    "main": {
      "type": "query",
      "description": "Get a document describing the service's accounts configuration.",
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "availableUserDomains"
          ],
          "properties": {
            "inviteCodeRequired": {
              "type": "boolean"
            },
            "availableUserDomains": {
              "type": "array",
              "items": {
                "type": "string"
              }
            },
            "links": {
              "type": "ref",
              "ref": "#links"
            }
```

```
        }
       }
      }
    },
    "links": {
      "type": "object",
      "properties": {
        "privacyPolicy": {
          "type": "string"
        },
        "termsOfService": {
          "type": "string"
        }
      }
    }
  }
}
```

---

com.atproto.server.getAccountInviteCodes#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.getAccountInviteCodes",
  "defs": {
    "main": {
      "type": "query",
      "description": "Get all invite codes for a given account",
      "parameters": {
        "type": "params",
        "properties": {
          "includeUsed": {
            "type": "boolean",
            "default": true
          },
          "createAvailable": {
            "type": "boolean",
            "default": true
          }
        }
      },
      "output": {
```

```
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "codes"
        ],
        "properties": {
          "codes": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "com.atproto.server.defs#inviteCode"
            }
          }
        }
      }
    },
    "errors": [
      {
        "name": "DuplicateCreate"
      }
    ]
  }
 }
}
```

---

com.atproto.server.getSession#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.getSession",
  "defs": {
    "main": {
      "type": "query",
      "description": "Get information about the current session.",
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "handle",
```

```
          "did"
        ],
        "properties": {
          "handle": {
            "type": "string",
            "format": "handle"
          },
          "did": {
            "type": "string",
            "format": "did"
          },
          "email": {
            "type": "string"
          }
        }
      }
    }
  }
}
```

---

com.atproto.server.listAppPasswords#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.listAppPasswords",
  "defs": {
    "main": {
      "type": "query",
      "description": "List all app-specific passwords.",
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "passwords"
          ],
          "properties": {
            "passwords": {
              "type": "array",
              "items": {
```

```
              "type": "ref",
              "ref": "#appPassword"
            }
          }
        }
      }
    },
    "errors": [
      {
        "name": "AccountTakedown"
      }
    ]
  },
  "appPassword": {
    "type": "object",
    "required": [
      "name",
      "createdAt"
    ],
    "properties": {
      "name": {
        "type": "string"
      },
      "createdAt": {
        "type": "string",
        "format": "datetime"
      }
    }
  }
 }
}
```

---

com.atproto.server.refreshSession#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.refreshSession",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Refresh an authentication session.",
```

```
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "accessJwt",
          "refreshJwt",
          "handle",
          "did"
        ],
        "properties": {
          "accessJwt": {
            "type": "string"
          },
          "refreshJwt": {
            "type": "string"
          },
          "handle": {
            "type": "string",
            "format": "handle"
          },
          "did": {
            "type": "string",
            "format": "did"
          }
        }
      }
    },
    "errors": [
      {
        "name": "AccountTakedown"
      }
    ]
  }
 }
}
```

---

com.atproto.server.requestAccountDelete#

```
{
  "lexicon": 1,
```

```
  "id": "com.atproto.server.requestAccountDelete",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Initiate a user account deletion via email."
    }
  }
}
```

---

com.atproto.server.requestPasswordReset#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.requestPasswordReset",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Initiate a user account password reset via email.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "email"
          ],
          "properties": {
            "email": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

---

com.atproto.server.resetPassword#

```
{
  "lexicon": 1,
  "id": "com.atproto.server.resetPassword",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Reset a user account password using a token.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "token",
            "password"
          ],
          "properties": {
            "token": {
              "type": "string"
            },
            "password": {
              "type": "string"
            }
          }
        }
      },
      "errors": [
        {
          "name": "ExpiredToken"
        },
        {
          "name": "InvalidToken"
        }
      ]
    }
  }
}
```


---

com.atproto.server.revokeAppPassword#


```
{
  "lexicon": 1,
```

```
  "id": "com.atproto.server.revokeAppPassword",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Revoke an app-specific password by name.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "name"
          ],
          "properties": {
            "name": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```


com.atproto.sync Lexicon#

Definitions related to cross-server sync in ATP.

---

com.atproto.sync.getBlob#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.getBlob",
  "defs": {
    "main": {
      "type": "query",
      "description": "Get a blob associated with a given repo.",
      "parameters": {
        "type": "params",
        "required": [
          "did",
          "cid"
        ],
        "properties": {
          "did": {
            "type": "string",
            "format": "did",
            "description": "The DID of the repo."
          },
          "cid": {
            "type": "string",
            "format": "cid",
            "description": "The CID of the blob to fetch"
          }
        }
      },
      "output": {
        "encoding": "*/*"
      }
    }
  }
}
```


---

com.atproto.sync.getBlocks#


```
{
  "lexicon": 1,
  "id": "com.atproto.sync.getBlocks",
  "defs": {
    "main": {
      "type": "query",
```

```
    "description": "Gets blocks from a given repo.",
    "parameters": {
      "type": "params",
      "required": [
        "did",
        "cids"
      ],
      "properties": {
        "did": {
          "type": "string",
          "format": "did",
          "description": "The DID of the repo."
        },
        "cids": {
          "type": "array",
          "items": {
            "type": "string",
            "format": "cid"
          }
        }
      }
    },
    "output": {
      "encoding": "application/vnd.ipld.car"
    }
  }
 }
}
```

---

com.atproto.sync.getCheckout#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.getCheckout",
  "defs": {
    "main": {
      "type": "query",
      "description": "Gets the repo state.",
      "parameters": {
        "type": "params",
        "required": [
```

```
        "did"
      ],
      "properties": {
        "did": {
          "type": "string",
          "format": "did",
          "description": "The DID of the repo."
        },
        "commit": {
          "type": "string",
          "format": "cid",
          "description": "The commit to get the checkout from. Defaults to current HEAD."
        }
      }
    },
    "output": {
      "encoding": "application/vnd.ipld.car"
    }
  }
 }
}
```

---

com.atproto.sync.getCommitPath#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.getCommitPath",
  "defs": {
    "main": {
      "type": "query",
      "description": "Gets the path of repo commits",
      "parameters": {
        "type": "params",
        "required": [
          "did"
        ],
        "properties": {
          "did": {
            "type": "string",
            "format": "did",
            "description": "The DID of the repo."
```

```
        },
        "latest": {
          "type": "string",
          "format": "cid",
          "description": "The most recent commit"
        },
        "earliest": {
          "type": "string",
          "format": "cid",
          "description": "The earliest commit to start from"
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "commits"
        ],
        "properties": {
          "commits": {
            "type": "array",
            "items": {
              "type": "string",
              "format": "cid"
            }
          }
        }
      }
    }
  }
}
```

---

com.atproto.sync.getHead#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.getHead",
  "defs": {
```

```
    "main": {
      "type": "query",
      "description": "Gets the current HEAD CID of a repo.",
      "parameters": {
        "type": "params",
        "required": [
          "did"
        ],
        "properties": {
          "did": {
            "type": "string",
            "format": "did",
            "description": "The DID of the repo."
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "root"
          ],
          "properties": {
            "root": {
              "type": "string",
              "format": "cid"
            }
          }
        }
      }
    }
  }
}
```

---

com.atproto.sync.getRecord#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.getRecord",
  "defs": {
```

```
    "main": {
      "type": "query",
      "description": "Gets blocks needed for existence or non-existence of record.",
      "parameters": {
        "type": "params",
        "required": [
          "did",
          "collection",
          "rkey"
        ],
        "properties": {
          "did": {
            "type": "string",
            "format": "did",
            "description": "The DID of the repo."
          },
          "collection": {
            "type": "string",
            "format": "nsid"
          },
          "rkey": {
            "type": "string"
          },
          "commit": {
            "type": "string",
            "format": "cid",
            "description": "An optional past commit CID."
          }
        }
      },
      "output": {
        "encoding": "application/vnd.ipld.car"
      }
    }
  }
}
```

---

com.atproto.sync.getRepo#

```
{
  "lexicon": 1,
```

```
  "id": "com.atproto.sync.getRepo",
  "defs": {
    "main": {
      "type": "query",
      "description": "Gets the repo state.",
      "parameters": {
        "type": "params",
        "required": [
          "did"
        ],
        "properties": {
          "did": {
            "type": "string",
            "format": "did",
            "description": "The DID of the repo."
          },
          "earliest": {
            "type": "string",
            "format": "cid",
            "description": "The earliest commit in the commit range (not inclusive)"
          },
          "latest": {
            "type": "string",
            "format": "cid",
            "description": "The latest commit in the commit range (inclusive)"
          }
        }
      },
      "output": {
        "encoding": "application/vnd.ipld.car"
      }
    }
  }
}
```

---

com.atproto.sync.listBlobs#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.listBlobs",
  "defs": {
```

```
"main": {
  "type": "query",
  "description": "List blob cids for some range of commits",
  "parameters": {
    "type": "params",
    "required": [
      "did"
    ],
    "properties": {
      "did": {
        "type": "string",
        "format": "did",
        "description": "The DID of the repo."
      },
      "latest": {
        "type": "string",
        "format": "cid",
        "description": "The most recent commit"
      },
      "earliest": {
        "type": "string",
        "format": "cid",
        "description": "The earliest commit to start from"
      }
    }
  },
  "output": {
    "encoding": "application/json",
    "schema": {
      "type": "object",
      "required": [
        "cids"
      ],
      "properties": {
        "cids": {
          "type": "array",
          "items": {
            "type": "string",
            "format": "cid"
          }
        }
      }
    }
  }
```

```
    }
  }
}
```

---

com.atproto.sync.listRepos#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.listRepos",
  "defs": {
    "main": {
      "type": "query",
      "description": "List dids and root cids of hosted repos",
      "parameters": {
        "type": "params",
        "properties": {
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 1000,
            "default": 500
          },
          "cursor": {
            "type": "string"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "repos"
          ],
          "properties": {
            "cursor": {
              "type": "string"
            },
            "repos": {
              "type": "array",
              "items": {
```

```
              "type": "ref",
              "ref": "#repo"
            }
          }
        }
      }
    }
  },
  "repo": {
    "type": "object",
    "required": [
      "did",
      "head"
    ],
    "properties": {
      "did": {
        "type": "string",
        "format": "did"
      },
      "head": {
        "type": "string",
        "format": "cid"
      }
    }
  }
 }
}
```

---

com.atproto.sync.notifyOfUpdate#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.notifyOfUpdate",
  "defs": {
    "main": {
      "type": "query",
      "description": "Notify a crawling service of a recent update. Often when a long break between updates causes the connection with the crawling service to break.",
      "parameters": {
        "type": "params",
        "required": [
```

```
        "hostname"
      ],
      "properties": {
        "hostname": {
          "type": "string",
          "description": "Hostname of the service that is notifying of update."
        }
      }
    }
  }
 }
}
```

---

com.atproto.sync.requestCrawl#

```
{
  "lexicon": 1,
  "id": "com.atproto.sync.requestCrawl",
  "defs": {
    "main": {
      "type": "query",
      "description": "Request a service to persistently crawl hosted repos.",
      "parameters": {
        "type": "params",
        "required": [
          "hostname"
        ],
        "properties": {
          "hostname": {
            "type": "string",
            "description": "Hostname of the service that is requesting to be crawled."
          }
        }
      }
    }
  }
}
```

com.atproto.sync.subscribeRepos#


```
{
  "lexicon": 1,
  "id": "com.atproto.sync.subscribeRepos",
  "defs": {
    "main": {
      "type": "subscription",
      "description": "Subscribe to repo updates",
      "parameters": {
        "type": "params",
        "properties": {
          "cursor": {
            "type": "integer",
            "description": "The last known event to backfill from."
          }
        }
      },
      "message": {
        "schema": {
          "type": "union",
          "refs": [
            "#commit",
            "#handle",
            "#migrate",
            "#tombstone",
            "#info"
          ]
        }
      },
      "errors": [
        {
          "name": "FutureCursor"
        }
      ]
    },
    "commit": {
      "type": "object",
      "required": [
        "seq",
        "rebase",
        "tooBig",
        "repo",
        "commit",
```

```
      "prev",
      "blocks",
      "ops",
      "blobs",
      "time"
    ],
    "nullable": [
      "prev"
    ],
    "properties": {
      "seq": {
        "type": "integer"
      },
      "rebase": {
        "type": "boolean"
      },
      "tooBig": {
        "type": "boolean"
      },
      "repo": {
        "type": "string",
        "format": "did"
      },
      "commit": {
        "type": "cid-link"
      },
      "prev": {
        "type": "cid-link"
      },
      "blocks": {
        "type": "bytes",
        "description": "CAR file containing relevant blocks",
        "maxLength": 1000000
      },
      "ops": {
        "type": "array",
        "items": {
          "type": "ref",
          "ref": "#repoOp"
        },
        "maxLength": 200
      },
      "blobs": {
        "type": "array",
```

```
      "items": {
        "type": "cid-link"
      }
    },
    "time": {
      "type": "string",
      "format": "datetime"
    }
  }
},
"handle": {
  "type": "object",
  "required": [
    "seq",
    "did",
    "handle",
    "time"
  ],
  "properties": {
    "seq": {
      "type": "integer"
    },
    "did": {
      "type": "string",
      "format": "did"
    },
    "handle": {
      "type": "string",
      "format": "handle"
    },
    "time": {
      "type": "string",
      "format": "datetime"
    }
  }
},
"migrate": {
  "type": "object",
  "required": [
    "seq",
    "did",
    "migrateTo",
    "time"
  ],
```

```
    "nullable": [
      "migrateTo"
    ],
    "properties": {
      "seq": {
        "type": "integer"
      },
      "did": {
        "type": "string",
        "format": "did"
      },
      "migrateTo": {
        "type": "string"
      },
      "time": {
        "type": "string",
        "format": "datetime"
      }
    }
  },
  "tombstone": {
    "type": "object",
    "required": [
      "seq",
      "did",
      "time"
    ],
    "properties": {
      "seq": {
        "type": "integer"
      },
      "did": {
        "type": "string",
        "format": "did"
      },
      "time": {
        "type": "string",
        "format": "datetime"
      }
    }
  },
  "info": {
    "type": "object",
    "required": [
```

```
        "name"
      ],
      "properties": {
        "name": {
          "type": "string",
          "knownValues": [
            "OutdatedCursor"
          ]
        },
        "message": {
          "type": "string"
        }
      }
    },
    "repoOp": {
      "type": "object",
      "required": [
        "action",
        "path",
        "cid"
      ],
      "nullable": [
        "cid"
      ],
      "properties": {
        "action": {
          "type": "string",
          "knownValues": [
            "create",
            "update",
            "delete"
          ]
        },
        "path": {
          "type": "string"
        },
        "cid": {
          "type": "cid-link"
        }
      }
    }
  }
}
```

app.bsky.actor Lexicon#

Definitions related to "actors," a general term for users in Bluesky.

---

app.bsky.actor.defs#

A reference to an actor in the network.


```
{
  "lexicon": 1,
  "id": "app.bsky.actor.defs",
  "description": "A reference to an actor in the network.",
  "defs": {
    "profileViewBasic": {
      "type": "object",
      "required": [
        "did",
        "handle"
      ],
      "properties": {
        "did": {
          "type": "string",
          "format": "did"
        },
        "handle": {
          "type": "string",
          "format": "handle"
        },
        "displayName": {
          "type": "string",
          "maxGraphemes": 64,
          "maxLength": 640
        },
        "avatar": {
```

```
      "type": "string"
    },
    "viewer": {
      "type": "ref",
      "ref": "#viewerState"
    },
    "labels": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.label.defs#label"
      }
    }
  }
},
"profileView": {
  "type": "object",
  "required": [
    "did",
    "handle"
  ],
  "properties": {
    "did": {
      "type": "string",
      "format": "did"
    },
    "handle": {
      "type": "string",
      "format": "handle"
    },
    "displayName": {
      "type": "string",
      "maxGraphemes": 64,
      "maxLength": 640
    },
    "description": {
      "type": "string",
      "maxGraphemes": 256,
      "maxLength": 2560
    },
    "avatar": {
      "type": "string"
    },
    "indexedAt": {
```

```
      "type": "string",
      "format": "datetime"
    },
    "viewer": {
      "type": "ref",
      "ref": "#viewerState"
    },
    "labels": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.label.defs#label"
      }
    }
  }
},
"profileViewDetailed": {
  "type": "object",
  "required": [
    "did",
    "handle"
  ],
  "properties": {
    "did": {
      "type": "string",
      "format": "did"
    },
    "handle": {
      "type": "string",
      "format": "handle"
    },
    "displayName": {
      "type": "string",
      "maxGraphemes": 64,
      "maxLength": 640
    },
    "description": {
      "type": "string",
      "maxGraphemes": 256,
      "maxLength": 2560
    },
    "avatar": {
      "type": "string"
    },
```

```
    "banner": {
      "type": "string"
    },
    "followersCount": {
      "type": "integer"
    },
    "followsCount": {
      "type": "integer"
    },
    "postsCount": {
      "type": "integer"
    },
    "indexedAt": {
      "type": "string",
      "format": "datetime"
    },
    "viewer": {
      "type": "ref",
      "ref": "#viewerState"
    },
    "labels": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.label.defs#label"
      }
    }
  }
},
"viewerState": {
  "type": "object",
  "properties": {
    "muted": {
      "type": "boolean"
    },
    "blockedBy": {
      "type": "boolean"
    },
    "blocking": {
      "type": "string",
      "format": "at-uri"
    },
    "following": {
      "type": "string",
```

```
          "format": "at-uri"
        },
        "followedBy": {
          "type": "string",
          "format": "at-uri"
        }
      }
    }
  }
}
```

---

app.bsky.actor.getProfile#

```
{
  "lexicon": 1,
  "id": "app.bsky.actor.getProfile",
  "defs": {
    "main": {
      "type": "query",
      "parameters": {
        "type": "params",
        "required": [
          "actor"
        ],
        "properties": {
          "actor": {
            "type": "string",
            "format": "at-identifier"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "ref",
          "ref": "app.bsky.actor.defs#profileViewDetailed"
        }
      }
    }
  }
}
```

app.bsky.actor.getProfiles#


```
{
  "lexicon": 1,
  "id": "app.bsky.actor.getProfiles",
  "defs": {
    "main": {
      "type": "query",
      "parameters": {
        "type": "params",
        "required": [
          "actors"
        ],
        "properties": {
          "actors": {
            "type": "array",
            "items": {
              "type": "string",
              "format": "at-identifier"
            },
            "maxLength": 25
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "profiles"
          ],
          "properties": {
            "profiles": {
              "type": "array",
              "items": {
                "type": "ref",
                "ref": "app.bsky.actor.defs#profileViewDetailed"
              }
            }
          }
```

```
      }
     }
    }
  }
}
```

---

app.bsky.actor.getSuggestions#

```
{
  "lexicon": 1,
  "id": "app.bsky.actor.getSuggestions",
  "defs": {
    "main": {
      "type": "query",
      "description": "Get a list of actors suggested for following. Used in discovery UIs.",
      "parameters": {
        "type": "params",
        "properties": {
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
            "type": "string"
          }
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "actors"
          ],
          "properties": {
            "cursor": {
              "type": "string"
            },
            "actors": {
```

```
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "app.bsky.actor.defs#profileView"
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.actor.profile#

```
{
  "lexicon": 1,
  "id": "app.bsky.actor.profile",
  "defs": {
    "main": {
      "type": "record",
      "key": "literal:self",
      "record": {
        "type": "object",
        "properties": {
          "displayName": {
            "type": "string",
            "maxGraphemes": 64,
            "maxLength": 640
          },
          "description": {
            "type": "string",
            "maxGraphemes": 256,
            "maxLength": 2560
          },
          "avatar": {
            "type": "blob",
            "accept": [
              "image/png",
              "image/jpeg"
            ],
```

```
          "maxSize": 1000000
        },
        "banner": {
          "type": "blob",
          "accept": [
            "image/png",
            "image/jpeg"
          ],
          "maxSize": 1000000
        }
      }
    }
  }
}
```

---

app.bsky.actor.searchActors#

```
{
  "lexicon": 1,
  "id": "app.bsky.actor.searchActors",
  "defs": {
    "main": {
      "type": "query",
      "description": "Find actors matching search criteria.",
      "parameters": {
        "type": "params",
        "properties": {
          "term": {
            "type": "string"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
            "type": "string"
          }
        }
```

```
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "actors"
          ],
          "properties": {
            "cursor": {
              "type": "string"
            },
            "actors": {
              "type": "array",
              "items": {
                "type": "ref",
                "ref": "app.bsky.actor.defs#profileView"
              }
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.actor.searchActorsTypeahead#

```
{
  "lexicon": 1,
  "id": "app.bsky.actor.searchActorsTypeahead",
  "defs": {
    "main": {
      "type": "query",
      "description": "Find actor suggestions for a search term.",
      "parameters": {
        "type": "params",
        "properties": {
          "term": {
            "type": "string"
          },
```

```
        "limit": {
          "type": "integer",
          "minimum": 1,
          "maximum": 100,
          "default": 50
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "actors"
        ],
        "properties": {
          "actors": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "app.bsky.actor.defs#profileViewBasic"
            }
          }
        }
      }
    }
  }
}
```

Definitions related to "embeds," content which is embedded within other records (e.g. links or images in posts).

---

app.bsky.embed.external#
A representation of some externally linked content, embedded in another form of content


```
{
  "lexicon": 1,
  "id": "app.bsky.embed.external",
  "description": "A representation of some externally linked content, embedded in another form of content",
  "defs": {
    "main": {
      "type": "object",
      "required": [
        "external"
      ],
      "properties": {
        "external": {
          "type": "ref",
          "ref": "#external"
        }
      }
    },
    "external": {
      "type": "object",
      "required": [
        "uri",
        "title",
        "description"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "uri"
        },
        "title": {
          "type": "string"
        },
        "description": {
          "type": "string"
        },
        "thumb": {
          "type": "blob",
          "accept": [
```

```
            "image/*"
          ],
          "maxSize": 1000000
        }
      }
    },
    "view": {
      "type": "object",
      "required": [
        "external"
      ],
      "properties": {
        "external": {
          "type": "ref",
          "ref": "#viewExternal"
        }
      }
    },
    "viewExternal": {
      "type": "object",
      "required": [
        "uri",
        "title",
        "description"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "uri"
        },
        "title": {
          "type": "string"
        },
        "description": {
          "type": "string"
        },
        "thumb": {
          "type": "string"
        }
      }
    }
  }
}
```

app.bsky.embed.images#

A set of images embedded in some other form of content


```
{
  "lexicon": 1,
  "id": "app.bsky.embed.images",
  "description": "A set of images embedded in some other form of content",
  "defs": {
    "main": {
      "type": "object",
      "required": [
        "images"
      ],
      "properties": {
        "images": {
          "type": "array",
          "items": {
            "type": "ref",
            "ref": "#image"
          },
          "maxLength": 4
        }
      }
    },
    "image": {
      "type": "object",
      "required": [
        "image",
        "alt"
      ],
      "properties": {
        "image": {
          "type": "blob",
          "accept": [
            "image/*"
          ],
          "maxSize": 1000000
        },
        "alt": {
          "type": "string"
        }
```

```
    }
   },
   "view": {
    "type": "object",
    "required": [
     "images"
    ],
    "properties": {
     "images": {
      "type": "array",
      "items": {
       "type": "ref",
       "ref": "#viewImage"
      },
      "maxLength": 4
     }
    }
   },
   "viewImage": {
    "type": "object",
    "required": [
     "thumb",
     "fullsize",
     "alt"
    ],
    "properties": {
     "thumb": {
      "type": "string"
     },
     "fullsize": {
      "type": "string"
     },
     "alt": {
      "type": "string"
     }
    }
   }
  }
}
```

app.bsky.embed.record#
A representation of a record embedded in another form of content


```
{
  "lexicon": 1,
  "id": "app.bsky.embed.record",
  "description": "A representation of a record embedded in another form of content",
  "defs": {
    "main": {
      "type": "object",
      "required": [
        "record"
      ],
      "properties": {
        "record": {
          "type": "ref",
          "ref": "com.atproto.repo.strongRef"
        }
      }
    },
    "view": {
      "type": "object",
      "required": [
        "record"
      ],
      "properties": {
        "record": {
          "type": "union",
          "refs": [
            "#viewRecord",
            "#viewNotFound",
            "#viewBlocked"
          ]
        }
      }
    },
    "viewRecord": {
      "type": "object",
      "required": [
        "uri",
        "cid",
        "author",
        "value",
        "indexedAt"
      ],
```

```
  "properties": {
    "uri": {
      "type": "string",
      "format": "at-uri"
    },
    "cid": {
      "type": "string",
      "format": "cid"
    },
    "author": {
      "type": "ref",
      "ref": "app.bsky.actor.defs#profileViewBasic"
    },
    "value": {
      "type": "unknown"
    },
    "labels": {
      "type": "array",
      "items": {
        "type": "ref",
        "ref": "com.atproto.label.defs#label"
      }
    },
    "embeds": {
      "type": "array",
      "items": {
        "type": "union",
        "refs": [
          "app.bsky.embed.images#view",
          "app.bsky.embed.external#view",
          "app.bsky.embed.record#view",
          "app.bsky.embed.recordWithMedia#view"
        ]
      }
    },
    "indexedAt": {
      "type": "string",
      "format": "datetime"
    }
  }
},
"viewNotFound": {
  "type": "object",
  "required": [
```

```
        "uri"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "at-uri"
        }
      }
    },
    "viewBlocked": {
      "type": "object",
      "required": [
        "uri"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "at-uri"
        }
      }
    }
  }
}
```

---

app.bsky.embed.recordWithMedia#

A representation of a record embedded in another form of content, alongside other compatible embeds

```
{
  "lexicon": 1,
  "id": "app.bsky.embed.recordWithMedia",
  "description": "A representation of a record embedded in another form of content, alongside other compatible embeds",
  "defs": {
    "main": {
      "type": "object",
      "required": [
        "record",
        "media"
      ],
      "properties": {
```

```
        "record": {
          "type": "ref",
          "ref": "app.bsky.embed.record"
        },
        "media": {
          "type": "union",
          "refs": [
            "app.bsky.embed.images",
            "app.bsky.embed.external"
          ]
        }
      }
    },
    "view": {
      "type": "object",
      "required": [
        "record",
        "media"
      ],
      "properties": {
        "record": {
          "type": "ref",
          "ref": "app.bsky.embed.record#view"
        },
        "media": {
          "type": "union",
          "refs": [
            "app.bsky.embed.images#view",
            "app.bsky.embed.external#view"
          ]
        }
      }
    }
  }
}
```

app.bsky.feed Lexicon#

Definitions related to content & activity published in Bluesky.

---

app.bsky.feed.defs#


```
{
  "lexicon": 1,
  "id": "app.bsky.feed.defs",
  "defs": {
    "postView": {
      "type": "object",
      "required": [
        "uri",
        "cid",
        "author",
        "record",
        "indexedAt"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "at-uri"
        },
        "cid": {
          "type": "string",
          "format": "cid"
        },
        "author": {
          "type": "ref",
          "ref": "app.bsky.actor.defs#profileViewBasic"
        },
        "record": {
          "type": "unknown"
        },
        "embed": {
          "type": "union",
          "refs": [
            "app.bsky.embed.images#view",
            "app.bsky.embed.external#view",
            "app.bsky.embed.record#view",
```

```
          "app.bsky.embed.recordWithMedia#view"
        ]
      },
      "replyCount": {
        "type": "integer"
      },
      "repostCount": {
        "type": "integer"
      },
      "likeCount": {
        "type": "integer"
      },
      "indexedAt": {
        "type": "string",
        "format": "datetime"
      },
      "viewer": {
        "type": "ref",
        "ref": "#viewerState"
      },
      "labels": {
        "type": "array",
        "items": {
          "type": "ref",
          "ref": "com.atproto.label.defs#label"
        }
      }
    }
  },
  "viewerState": {
    "type": "object",
    "properties": {
      "repost": {
        "type": "string",
        "format": "at-uri"
      },
      "like": {
        "type": "string",
        "format": "at-uri"
      }
    }
  },
  "feedViewPost": {
    "type": "object",
```

```
  "required": [
    "post"
  ],
  "properties": {
    "post": {
      "type": "ref",
      "ref": "app.bsky.feed.defs#postView"
    },
    "reply": {
      "type": "ref",
      "ref": "#replyRef"
    },
    "reason": {
      "type": "union",
      "refs": [
        "#reasonRepost"
      ]
    }
  }
},
"replyRef": {
  "type": "object",
  "required": [
    "root",
    "parent"
  ],
  "properties": {
    "root": {
      "type": "ref",
      "ref": "app.bsky.feed.defs#postView"
    },
    "parent": {
      "type": "ref",
      "ref": "app.bsky.feed.defs#postView"
    }
  }
},
"reasonRepost": {
  "type": "object",
  "required": [
    "by",
    "indexedAt"
  ],
  "properties": {
```

```
    "by": {
      "type": "ref",
      "ref": "app.bsky.actor.defs#profileViewBasic"
    },
    "indexedAt": {
      "type": "string",
      "format": "datetime"
    }
  }
},
"threadViewPost": {
  "type": "object",
  "required": [
    "post"
  ],
  "properties": {
    "post": {
      "type": "ref",
      "ref": "#postView"
    },
    "parent": {
      "type": "union",
      "refs": [
        "#threadViewPost",
        "#notFoundPost",
        "#blockedPost"
      ]
    },
    "replies": {
      "type": "array",
      "items": {
        "type": "union",
        "refs": [
          "#threadViewPost",
          "#notFoundPost",
          "#blockedPost"
        ]
      }
    }
  }
},
"notFoundPost": {
  "type": "object",
  "required": [
```

```
        "uri",
        "notFound"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "at-uri"
        },
        "notFound": {
          "type": "boolean",
          "const": true
        }
      }
    },
    "blockedPost": {
      "type": "object",
      "required": [
        "uri",
        "blocked"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "at-uri"
        },
        "blocked": {
          "type": "boolean",
          "const": true
        }
      }
    }
  }
}
```

---

app.bsky.feed.getAuthorFeed#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.getAuthorFeed",
  "defs": {
    "main": {
```

```
"type": "query",
"description": "A view of an actor's feed.",
"parameters": {
  "type": "params",
  "required": [
    "actor"
  ],
  "properties": {
    "actor": {
      "type": "string",
      "format": "at-identifier"
    },
    "limit": {
      "type": "integer",
      "minimum": 1,
      "maximum": 100,
      "default": 50
    },
    "cursor": {
      "type": "string"
    }
  }
},
"output": {
  "encoding": "application/json",
  "schema": {
    "type": "object",
    "required": [
      "feed"
    ],
    "properties": {
      "cursor": {
        "type": "string"
      },
      "feed": {
        "type": "array",
        "items": {
          "type": "ref",
          "ref": "app.bsky.feed.defs#feedViewPost"
        }
      }
    }
  }
},
```

```
      "errors": [
        {
          "name": "BlockedActor"
        },
        {
          "name": "BlockedByActor"
        }
      ]
    }
  }
}
```

---

app.bsky.feed.getLikes#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.getLikes",
  "defs": {
    "main": {
      "type": "query",
      "parameters": {
        "type": "params",
        "required": [
          "uri"
        ],
        "properties": {
          "uri": {
            "type": "string",
            "format": "at-uri"
          },
          "cid": {
            "type": "string",
            "format": "cid"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
```

```
        "type": "string"
      }
    }
  },
  "output": {
    "encoding": "application/json",
    "schema": {
      "type": "object",
      "required": [
        "uri",
        "likes"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "at-uri"
        },
        "cid": {
          "type": "string",
          "format": "cid"
        },
        "cursor": {
          "type": "string"
        },
        "likes": {
          "type": "array",
          "items": {
            "type": "ref",
            "ref": "#like"
          }
        }
      }
    }
  }
},
"like": {
  "type": "object",
  "required": [
    "indexedAt",
    "createdAt",
    "actor"
  ],
  "properties": {
    "indexedAt": {
```

```
          "type": "string",
          "format": "datetime"
        },
        "createdAt": {
          "type": "string",
          "format": "datetime"
        },
        "actor": {
          "type": "ref",
          "ref": "app.bsky.actor.defs#profileView"
        }
      }
    }
  }
}
```

---

app.bsky.feed.getPostThread#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.getPostThread",
  "defs": {
    "main": {
      "type": "query",
      "parameters": {
        "type": "params",
        "required": [
          "uri"
        ],
        "properties": {
          "uri": {
            "type": "string",
            "format": "at-uri"
          },
          "depth": {
            "type": "integer"
          }
        }
      },
      "output": {
        "encoding": "application/json",
```

```
      "schema": {
        "type": "object",
        "required": [
          "thread"
        ],
        "properties": {
          "thread": {
            "type": "union",
            "refs": [
              "app.bsky.feed.defs#threadViewPost",
              "app.bsky.feed.defs#notFoundPost",
              "app.bsky.feed.defs#blockedPost"
            ]
          }
        }
      }
    },
    "errors": [
      {
        "name": "NotFound"
      }
    ]
  }
 }
}
```

---

app.bsky.feed.getPosts#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.getPosts",
  "defs": {
    "main": {
      "type": "query",
      "description": "A view of an actor's feed.",
      "parameters": {
        "type": "params",
        "required": [
          "uris"
        ],
        "properties": {
```

```
        "uris": {
          "type": "array",
          "items": {
            "type": "string",
            "format": "at-uri"
          },
          "maxLength": 25
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "posts"
        ],
        "properties": {
          "posts": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "app.bsky.feed.defs#postView"
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.feed.getRepostedBy#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.getRepostedBy",
  "defs": {
    "main": {
      "type": "query",
      "parameters": {
```

```
  "type": "params",
  "required": [
    "uri"
  ],
  "properties": {
    "uri": {
      "type": "string",
      "format": "at-uri"
    },
    "cid": {
      "type": "string",
      "format": "cid"
    },
    "limit": {
      "type": "integer",
      "minimum": 1,
      "maximum": 100,
      "default": 50
    },
    "cursor": {
      "type": "string"
    }
  }
},
"output": {
  "encoding": "application/json",
  "schema": {
    "type": "object",
    "required": [
      "uri",
      "repostedBy"
    ],
    "properties": {
      "uri": {
        "type": "string",
        "format": "at-uri"
      },
      "cid": {
        "type": "string",
        "format": "cid"
      },
      "cursor": {
        "type": "string"
      },
```

```
          "repostedBy": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "app.bsky.actor.defs#profileView"
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.feed.getTimeline#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.getTimeline",
  "defs": {
    "main": {
      "type": "query",
      "description": "A view of the user's home timeline.",
      "parameters": {
        "type": "params",
        "properties": {
          "algorithm": {
            "type": "string"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
            "type": "string"
          }
        }
      },
      "output": {
```

```
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "feed"
          ],
          "properties": {
            "cursor": {
              "type": "string"
            },
            "feed": {
              "type": "array",
              "items": {
                "type": "ref",
                "ref": "app.bsky.feed.defs#feedViewPost"
              }
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.feed.like#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.like",
  "defs": {
    "main": {
      "type": "record",
      "key": "tid",
      "record": {
        "type": "object",
        "required": [
          "subject",
          "createdAt"
        ],
        "properties": {
          "subject": {
```

```
          "type": "ref",
          "ref": "com.atproto.repo.strongRef"
        },
        "createdAt": {
          "type": "string",
          "format": "datetime"
        }
      }
    }
  }
 }
}
```

---

app.bsky.feed.post#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.post",
  "defs": {
    "main": {
      "type": "record",
      "key": "tid",
      "record": {
        "type": "object",
        "required": [
          "text",
          "createdAt"
        ],
        "properties": {
          "text": {
            "type": "string",
            "maxLength": 3000,
            "maxGraphemes": 300
          },
          "entities": {
            "type": "array",
            "description": "Deprecated: replaced by app.bsky.richtext.facet.",
            "items": {
              "type": "ref",
              "ref": "#entity"
            }
```

```
      },
      "facets": {
        "type": "array",
        "items": {
          "type": "ref",
          "ref": "app.bsky.richtext.facet"
        }
      },
      "reply": {
        "type": "ref",
        "ref": "#replyRef"
      },
      "embed": {
        "type": "union",
        "refs": [
          "app.bsky.embed.images",
          "app.bsky.embed.external",
          "app.bsky.embed.record",
          "app.bsky.embed.recordWithMedia"
        ]
      },
      "createdAt": {
        "type": "string",
        "format": "datetime"
      }
    }
  }
},
"replyRef": {
  "type": "object",
  "required": [
    "root",
    "parent"
  ],
  "properties": {
    "root": {
      "type": "ref",
      "ref": "com.atproto.repo.strongRef"
    },
    "parent": {
      "type": "ref",
      "ref": "com.atproto.repo.strongRef"
    }
  }
```

```
    },
    "entity": {
      "type": "object",
      "description": "Deprecated: use facets instead.",
      "required": [
        "index",
        "type",
        "value"
      ],
      "properties": {
        "index": {
          "type": "ref",
          "ref": "#textSlice"
        },
        "type": {
          "type": "string",
          "description": "Expected values are 'mention' and 'link'."
        },
        "value": {
          "type": "string"
        }
      }
    },
    "textSlice": {
      "type": "object",
      "description": "Deprecated. Use app.bsky.richtext instead -- A text segment. Start is
inclusive, end is exclusive. Indices are for utf16-encoded strings.",
      "required": [
        "start",
        "end"
      ],
      "properties": {
        "start": {
          "type": "integer",
          "minimum": 0
        },
        "end": {
          "type": "integer",
          "minimum": 0
        }
      }
    }
  }
}
```

---

app.bsky.feed.repost#

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.repost",
  "defs": {
    "main": {
      "type": "record",
      "key": "tid",
      "record": {
        "type": "object",
        "required": [
          "subject",
          "createdAt"
        ],
        "properties": {
          "subject": {
            "type": "ref",
            "ref": "com.atproto.repo.strongRef"
          },
          "createdAt": {
            "type": "string",
            "format": "datetime"
          }
        }
      }
    }
  }
}
```

---

app.bsky.feed.getFeedSkeleton#

We are actively developing Feed Generator integration into the Bluesky PDS. Though we are reasonably confident about the general shape laid out here, this lexicon is subject to change.

```
{
  "lexicon": 1,
  "id": "app.bsky.feed.getFeedSkeleton",
```

```
  "defs": {
    "main": {
      "type": "query",
      "description": "A skeleton of a feed provided by a feed generator",
      "parameters": {
        "type": "params",
        "required": ["feed"],
        "properties": {
          "feed": {"type": "string", "format": "at-uri"},
          "limit": {"type": "integer", "minimum": 1, "maximum": 100, "default": 50},
          "cursor": {"type": "string"}
        }
      },
      "output": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": ["feed"],
          "properties": {
            "cursor": {"type": "string"},
            "feed": {
              "type": "array",
              "items": {"type": "ref", "ref": "app.bsky.feed.defs#skeletonFeedPost"}
            }
          }
        }
      }
    }
  }
}
```

SchemeLexicon
SchemasXRPCNSIDDID:PlaceholderLexiconscom.atproto.admincom.atproto.identitycom.atproto.labelcom.atproto.moderationcom.atproto.repocom.atproto.servercom.atproto.syncapp.bsky.actorapp.bsky.embedapp.bsky.feedapp.bsky.graphapp.bsky.notificationapp.bsky.richtext
app.bsky.graph Lexicon#
Definitions related to the social graph in Bluesky.

---

app.bsky.graph.block#


```
{
  "lexicon": 1,
  "id": "app.bsky.graph.block",
  "defs": {
    "main": {
      "type": "record",
      "description": "A block.",
      "key": "tid",
      "record": {
        "type": "object",
        "required": [
          "subject",
          "createdAt"
        ],
        "properties": {
          "subject": {
            "type": "string",
            "format": "did"
          },
          "createdAt": {
            "type": "string",
            "format": "datetime"
          }
        }
      }
    }
  }
}
```


---

app.bsky.graph.follow#


```
{
  "lexicon": 1,
```

```
  "id": "app.bsky.graph.follow",
  "defs": {
    "main": {
      "type": "record",
      "description": "A social follow.",
      "key": "tid",
      "record": {
        "type": "object",
        "required": [
          "subject",
          "createdAt"
        ],
        "properties": {
          "subject": {
            "type": "string",
            "format": "did"
          },
          "createdAt": {
            "type": "string",
            "format": "datetime"
          }
        }
      }
    }
  }
}
```

---

app.bsky.graph.getBlocks#

```
{
  "lexicon": 1,
  "id": "app.bsky.graph.getBlocks",
  "defs": {
    "main": {
      "type": "query",
      "description": "Who is the requester's account blocking?",
      "parameters": {
        "type": "params",
        "properties": {
          "limit": {
            "type": "integer",
```

```
          "minimum": 1,
          "maximum": 100,
          "default": 50
        },
        "cursor": {
          "type": "string"
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "blocks"
        ],
        "properties": {
          "cursor": {
            "type": "string"
          },
          "blocks": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "app.bsky.actor.defs#profileView"
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.graph.getFollowers#

```
{
  "lexicon": 1,
  "id": "app.bsky.graph.getFollowers",
  "defs": {
    "main": {
```

```
"type": "query",
"description": "Who is following an actor?",
"parameters": {
  "type": "params",
  "required": [
    "actor"
  ],
  "properties": {
    "actor": {
      "type": "string",
      "format": "at-identifier"
    },
    "limit": {
      "type": "integer",
      "minimum": 1,
      "maximum": 100,
      "default": 50
    },
    "cursor": {
      "type": "string"
    }
  }
},
"output": {
  "encoding": "application/json",
  "schema": {
    "type": "object",
    "required": [
      "subject",
      "followers"
    ],
    "properties": {
      "subject": {
        "type": "ref",
        "ref": "app.bsky.actor.defs#profileView"
      },
      "cursor": {
        "type": "string"
      },
      "followers": {
        "type": "array",
        "items": {
          "type": "ref",
          "ref": "app.bsky.actor.defs#profileView"
```

```
          }
         }
        }
       }
      }
     }
    }
```

---

app.bsky.graph.getFollows#

```
{
  "lexicon": 1,
  "id": "app.bsky.graph.getFollows",
  "defs": {
    "main": {
      "type": "query",
      "description": "Who is an actor following?",
      "parameters": {
        "type": "params",
        "required": [
          "actor"
        ],
        "properties": {
          "actor": {
            "type": "string",
            "format": "at-identifier"
          },
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
            "type": "string"
          }
        }
      },
      "output": {
        "encoding": "application/json",
```

```
        "schema": {
          "type": "object",
          "required": [
            "subject",
            "follows"
          ],
          "properties": {
            "subject": {
              "type": "ref",
              "ref": "app.bsky.actor.defs#profileView"
            },
            "cursor": {
              "type": "string"
            },
            "follows": {
              "type": "array",
              "items": {
                "type": "ref",
                "ref": "app.bsky.actor.defs#profileView"
              }
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.graph.getMutes#

```
{
  "lexicon": 1,
  "id": "app.bsky.graph.getMutes",
  "defs": {
    "main": {
      "type": "query",
      "description": "Who does the viewer mute?",
      "parameters": {
        "type": "params",
        "properties": {
          "limit": {
```

```
          "type": "integer",
          "minimum": 1,
          "maximum": 100,
          "default": 50
        },
        "cursor": {
          "type": "string"
        }
      }
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "mutes"
        ],
        "properties": {
          "cursor": {
            "type": "string"
          },
          "mutes": {
            "type": "array",
            "items": {
              "type": "ref",
              "ref": "app.bsky.actor.defs#profileView"
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.graph.muteActor#

```
{
  "lexicon": 1,
  "id": "app.bsky.graph.muteActor",
  "defs": {
```

```
    "main": {
      "type": "procedure",
      "description": "Mute an actor by did or handle.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "actor"
          ],
          "properties": {
            "actor": {
              "type": "string",
              "format": "at-identifier"
            }
          }
        }
      }
    }
  }
}
```

---

app.bsky.graph.unmuteActor#

```
{
  "lexicon": 1,
  "id": "app.bsky.graph.unmuteActor",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Unmute an actor by did or handle.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "actor"
          ],
          "properties": {
            "actor": {
              "type": "string",
```

```
            "format": "at-identifier"
          }
         }
        }
       }
     }
   }
}
```


app.bsky.notification Lexicon#

Definitions related to notifications.

---

app.bsky.notification.getUnreadCount#


```
{
  "lexicon": 1,
  "id": "app.bsky.notification.getUnreadCount",
  "defs": {
    "main": {
      "type": "query",
      "parameters": {
        "type": "params",
        "properties": {
          "seenAt": {
            "type": "string",
            "format": "datetime"
          }
        }
```

```
    },
    "output": {
      "encoding": "application/json",
      "schema": {
        "type": "object",
        "required": [
          "count"
        ],
        "properties": {
          "count": {
            "type": "integer"
          }
        }
      }
    }
  }
}
```

---

app.bsky.notification.listNotifications#

```
{
  "lexicon": 1,
  "id": "app.bsky.notification.listNotifications",
  "defs": {
    "main": {
      "type": "query",
      "parameters": {
        "type": "params",
        "properties": {
          "limit": {
            "type": "integer",
            "minimum": 1,
            "maximum": 100,
            "default": 50
          },
          "cursor": {
            "type": "string"
          },
          "seenAt": {
            "type": "string",
```

```
        "format": "datetime"
      }
    }
  },
  "output": {
    "encoding": "application/json",
    "schema": {
      "type": "object",
      "required": [
        "notifications"
      ],
      "properties": {
        "cursor": {
          "type": "string"
        },
        "notifications": {
          "type": "array",
          "items": {
            "type": "ref",
            "ref": "#notification"
          }
        }
      }
    }
  }
},
"notification": {
  "type": "object",
  "required": [
    "uri",
    "cid",
    "author",
    "reason",
    "record",
    "isRead",
    "indexedAt"
  ],
  "properties": {
    "uri": {
      "type": "string",
      "format": "at-uri"
    },
    "cid": {
      "type": "string",
```

```
          "format": "cid"
        },
        "author": {
          "type": "ref",
          "ref": "app.bsky.actor.defs#profileView"
        },
        "reason": {
          "type": "string",
          "description": "Expected values are 'like', 'repost', 'follow', 'mention', 'reply', and 'quote'.",
          "knownValues": [
            "like",
            "repost",
            "follow",
            "mention",
            "reply",
            "quote"
          ]
        },
        "reasonSubject": {
          "type": "string",
          "format": "at-uri"
        },
        "record": {
          "type": "unknown"
        },
        "isRead": {
          "type": "boolean"
        },
        "indexedAt": {
          "type": "string",
          "format": "datetime"
        },
        "labels": {
          "type": "array",
          "items": {
            "type": "ref",
            "ref": "com.atproto.label.defs#label"
          }
        }
      }
    }
  }
}
```

app.bsky.notification.updateSeen#


```
{
  "lexicon": 1,
  "id": "app.bsky.notification.updateSeen",
  "defs": {
    "main": {
      "type": "procedure",
      "description": "Notify server that the user has seen notifications.",
      "input": {
        "encoding": "application/json",
        "schema": {
          "type": "object",
          "required": [
            "seenAt"
          ],
          "properties": {
            "seenAt": {
              "type": "string",
              "format": "datetime"
            }
          }
        }
      }
    }
  }
}
```

app.bsky.richtext.facet#


```
{
  "lexicon": 1,
  "id": "app.bsky.richtext.facet",
  "defs": {
    "main": {
      "type": "object",
      "required": [
        "index",
        "features"
      ],
      "properties": {
        "index": {
          "type": "ref",
          "ref": "#byteSlice"
        },
        "features": {
          "type": "array",
          "items": {
            "type": "union",
            "refs": [
              "#mention",
              "#link"
            ]
          }
        }
      }
    },
    "mention": {
      "type": "object",
      "description": "A facet feature for actor mentions.",
      "required": [
        "did"
      ],
      "properties": {
        "did": {
          "type": "string",
          "format": "did"
        }
      }
    },
    "link": {
      "type": "object",
      "description": "A facet feature for links.",
```

```
      "required": [
        "uri"
      ],
      "properties": {
        "uri": {
          "type": "string",
          "format": "uri"
        }
      }
    },
    "byteSlice": {
      "type": "object",
      "description": "A text segment. Start is inclusive, end is exclusive. Indices are for utf8-encoded strings.",
      "required": [
        "byteStart",
        "byteEnd"
      ],
      "properties": {
        "byteStart": {
          "type": "integer",
          "minimum": 0
        },
        "byteEnd": {
          "type": "integer",
          "minimum": 0
        }
      }
    }
  }
}
```