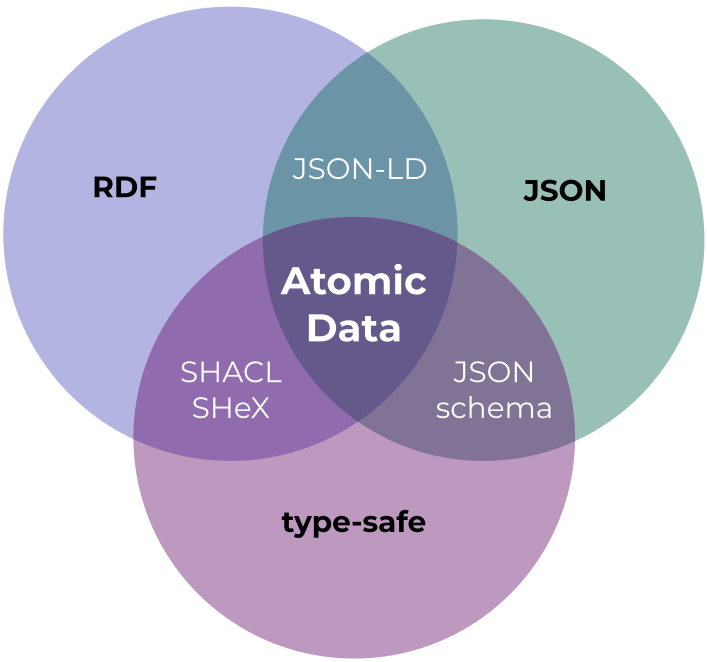
Atomic Data

Atomic Data is a modular specification for sharing, modifying and modeling graph data. It combines the ease of use of JSON, the connectivity of RDF (linked data) and the reliability of type-safety.



Atomic Data uses links to connect pieces of data, and therefore makes it easier to connect datasets to each other - even when these datasets exist on separate machines.

Atomic Data has been designed with the following goals in mind:

- Give people more control over their data
- Make linked data easier to use
- Make it easier for developers to build highly interoperable apps
- Make standardization easier and cheaper

Atomic Data is Linked Data, as it is a strict subset of RDF. It is type-safe (you know if something is a `string`, `number`, `date`, `URL`, etc.) and extensible through Atomic Schema, which means that you can re-use or define your own Classes, Properties and Datatypes.

The default serialization format for Atomic Data is JSON-AD, which is simply JSON where each key is a URL of an Atomic Property. These Properties are responsible for setting the `datatype` (to ensure type-safety) and setting `shortnames` (which help to keep names short, for example in JSON serialization) and `descriptions` (which provide semantic explanation of what a property should be used for).

Read more about Atomic Data Core

Atomic Data Extended

Atomic Data Extended is a set of extra modules (on top of Atomic Data Core) that deal with data that changes over time, authentication, and authorization.

- Commits communicate state changes. These Commits are signed using cryptographic keys, which ensures that every change can be audited. Commits are also used to construct a history of versions.
- Agents are Users that enable authentication. They are Resources with their own Public and Private keys, which they use to identify themselves.
- Collections: querying, filtering, sorting and pagination.
- Paths: traverse graphs.
- Hierarchies used for authorization and keeping data organized. Similar to folder structures on file-systems.
- Invites: create new users and provide them with rights.
- WebSockets: real-time updates.
- Endpoints: provide machine-readable descriptions of web services.
- Files: upload, download and metadata for files.

Atomizing: how to create, convert and host Atomic Data

Atomic Data has been designed to be very easy to create and host. In the Atomizing section, we'll show you how you can create Atomic Data in three ways:

- Using Atomic Server, from your browser
- By creating JSON-AD (and optionally importing it)
- By upgrading your existing application

Tools & libraries

- Browser app atomic-data-browser (demo on atomicdata.dev)
- Build a react app using typescript & react libraries. Start with the react template on codesandbox
- Host your own atomic-server (powers atomicdata.dev, run with `docker run -p 80:80 -p 443:443 -v atomic-storage:/atomic-storage joepmeneer/atomic-server`)
- Discover the command line tool: atomic-cli (`cargo install atomic-cli`)
- Use the Rust library: atomic-lib

Get involved

Make sure to join our Discord if you'd like to discuss Atomic Data with others.

Status

Keep in mind that none of the Atomic Data projects has reached a v1, which means that breaking changes can happen.

Reading these docs

This is written mostly as a book, so reading it in the order of the Table of Contents will probably give you the best experience. That being said, feel free to jump around - links are often used to refer to earlier discussed concepts. If you encounter any issues while reading, please leave an issue on Github. Use the arrows on the side / bottom to go to the next page.

Table of contents

Specification (extended)

Create Atomic Data

Use Atomic Data

Acknowledgements | Newsletter | Get involved

Found a bug? Edit this page on GitHub.