Package: libipldr (via r-universe)

June 11, 2026

Title R Bindings to the Rust IPLD Library

Version 0.0.1

Description Provides R bindings to decode DAG-CBOR encoded data, CIDs (Content Identifiers), and CAR (Content Addressable aRchive) files using Rust's IPLD library. This is especially useful for working with data from IPFS and AtProto (Bluesky) applications.

License MIT + file LICENSE

URL https://github.com/JBGruber/libipldr

BugReports https://github.com/JBGruber/libipldr/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/rextendr/version 0.4.2

SystemRequirements Cargo (Rust's package manager), rustc >= 1.65

Suggests testthat (>= 3.0.0), rextendr (>= 0.2.0), httr2, jsonlite

Config/testthat/edition 3

Depends R (>= 4.2)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libclang-dev

Repository https://jbgruber.r-universe.dev

Date/Publication 2026-05-12 16:27:57 UTC

RemoteUrl https://github.com/jbgruber/libipldr

RemoteRef HEAD

RemoteSha ed3dc42154db12974380ee2a80014576b526c9d5

Contents

`libipldr-package` *libipldr: R Bindings to the Rust IPLD Library*

Description

Provides R bindings to decode DAG-CBOR encoded data, CIDs (Content Identifiers), and CAR (Content Addressable aRchive) files using Rust's IPLD library. This is especially useful for working with data from IPFS and AtProto (Bluesky) applications.

Author(s)

Maintainer: Johannes B. Gruber <JohannesB.Gruber@gmail.com> (ORCID)

Authors:

- Johannes B. Gruber <JohannesB.Gruber@gmail.com> (ORCID)

Other contributors:

- Ilya Siamionau (Author of the original python-libipld Rust implementation) [contributor]

See Also

Useful links:

- https://github.com/JBGruber/libipldr
- Report bugs at https://github.com/JBGruber/libipldr/issues

`decode_car` *Decode a Content Addressable aRchive (CAR) file*

Description

This function decodes a CAR file from a raw vector, extracting the header and blocks.

Usage

```
decode_car(data)
```

Arguments

`data`	A raw vector containing a CAR file

Value

A list with header information and decoded blocks

Examples

```
## Not run:
# When you have a CAR file as raw data:
car_data <- decode_car(raw_car_data)

## End(Not run)
```

---

decode_cid *Decode a Content IDentifier (CID) string*

Description

This function decodes a CID string into its components (version, codec, and hash).

Usage

```
decode_cid(cid_str)
```

Arguments

cid_str	A string containing a valid CID

Value

A list with CID components

Examples

```
# Decode a CID:
cid_info <- decode_cid("bafyreib775pirw4o3rz4iwdjwi3rz7q4z5t4xjyfrwnk2yukhzo2wyr4ye")
```

---

decode_dag_cbor *Decode DAG-CBOR encoded data to an R object*

Description

This function decodes a raw vector containing DAG-CBOR encoded data into an R object. DAG-CBOR is a deterministic subset of the CBOR format, used by IPFS and AtProto (Bluesky) for data representation.

Usage

```
decode_dag_cbor(data)
```

Arguments

data	A raw vector containing DAG-CBOR encoded data

Value

An R object representing the decoded data

Examples

```
# Decode a simple DAG-CBOR map {"a": "Hello", "b": "World!"}
cbor_data <- as.raw(c(
  0xa2, 0x61, 0x61, 0x65, 0x48, 0x65, 0x6c, 0x6c, 0x6f,
  0x61, 0x62, 0x66, 0x57, 0x6f, 0x72, 0x6c, 0x64, 0x21
))
decode_dag_cbor(cbor_data)
```

---

decode_dag_cbor_multi *Decode multiple DAG-CBOR objects from a byte stream*

Description

This function decodes multiple consecutive DAG-CBOR objects from a single raw vector. The returned list includes a 'bytes_consumed' attribute indicating how many bytes from the beginning of the input were successfully processed. This is useful for streaming applications where you need to know where to continue reading.

Usage

```
decode_dag_cbor_multi(data)
```

Arguments

data	A raw vector containing multiple DAG-CBOR encoded objects

Value

A list of R objects, each representing a decoded DAG-CBOR object

Examples

```
# Decode two consecutive DAG-CBOR objects from a single byte stream
cbor_data <- as.raw(c(
  0xa2, 0x61, 0x61, 0x65, 0x48, 0x65, 0x6c, 0x6c, 0x6f,
  0x61, 0x62, 0x66, 0x57, 0x6f, 0x72, 0x6c, 0x64, 0x21,
  0xa1, 0x61, 0x63, 0x01
))
results <- decode_dag_cbor_multi(cbor_data)
attr(results, "bytes_consumed")
```

Index