kaiv for ASN.1 Users

A Cookbook

The kaiv Project

for kaiv 0.4.0

1 Introduction

You work with ASN.1 — the oldest discipline in structured data, the one that already knew, in 1984, that data needs a schema, constraints need syntax, and encodings need canonical forms. Of every format kaiv converts, ASN.1 is the *peer*: the closest to kaiv in constraint expressiveness, and the only one kaiv does not cover completely — about 95%, with the remainder examined honestly below.

What kaiv offers an ASN.1 practice is a second surface for the same rigor: a self-describing *text* canonical form — DER's determinism without DER's opacity — one grammar for data and schema alike, and a toolchain that is a single lexer and a forward scan rather than a compiler ecosystem and six codecs.

Every example is a verified transcript of the reference `kaiv` tool (`cargo install kaiv-cli`), run against the fixture files shipped beside this document.

2 Hello, kaiv

A DER `SEQUENCE` of a `UTF8String`, an `INTEGER`, and a `BOOLEAN`, built byte by byte so everything is on the page:

```
$ printf '\x30\x0b\x0c\x03Ada\x02\x01\x07\x01\x01\xff' >
   ↪  rec.der
$ kaiv import --asn1 rec.der
.!kaiv 1

/@seq+=Ada
```

```
!int
/@seq+=7
!bool
/@seq+=true
```

ASN.1 fields are positional — the schema, not the wire, carries the names — so a bare `SEQUENCE` imports as the ordered array `@seq`, each member annotated with its decoded universal type. And because DER and kaiv are both canonical-by-construction, the trip back is byte-identical:

```
$ kaiv import --asn1 rec.der | kaiv build > rec.daiv
$ kaiv export --asn1 rec.daiv | xxd
00000000: 300b 0c03 4164 6102 0107 0101 ff
   ↪  0...Ada......
```

(A word on that export: kaiv namespaces are named, ASN.1 fields are not, so only sequence-shaped documents like this one have a DER form — the exporter says so loudly for anything else rather than inventing an encoding.)

3 A Real Certificate, Greppable

The everyday encounter with DER is X.509. kaiv auto-imports `.crt`/`.pem` as ASN.1 — and the certificate becomes plain, typed, addressable text:

```
$ kaiv import demo.crt | head -14
.!kaiv 1
.!types std/enc
.!types std/time

!int
/@seq/0/@seq/0/@c0;=2
!int
/@seq/0/@seq::1=701316311132569874803253130111925826802265752411
/@seq/0/@seq/2/@seq+:=oid=1.3.101.112
/@seq/0/@seq/3/@seq/0/@set/0/@seq/0:=oid=2.5.4.3
/@seq/0/@seq/3/@seq/0/@set/0/@seq::1=kaiv.io
/@seq/0/@seq/3/@seq/1/@set/0/@seq/0:=oid=2.5.4.10
```

```
/@seq/0/@seq/3/@seq/1/@set/0/@seq::1=AIV Data
&datetime
```

OIDs decode to dotted form, context tags keep their position (`@c0`), the serial number is an unbounded `!int`, and the Ed25519 public key rides the `&bin` channel. Which means the questions you usually route through `openssl x509 -text` become one-liners:

```
$ kaiv import demo.crt | grep 'utc='
/@seq/0/@seq/4/@seq+:=utc=2026-07-17T16:35:09Z
/@seq/0/@seq/4/@seq+:=utc=2036-07-14T16:35:09Z
```

Issued and expires, as `std/time` datetimes — `UTCTime`'s two-digit years normalized to full ISO form on the way in, per DER's own interpretation rules.

4 The Mapping — and the Honest 5%

ASN.1	kaiv	
`SEQUENCE`	`@seq` array	positional; schema names
`SEQUENCE OF`	`@`-array	
`CHOICE`	tagged union	
`ENUMERATED`	enum constraint	`{a,b,c}`
`INTEGER (1..100)`	`!int[1,100]`	unbounded
`UTF8String SIZE`	`#[m,n]`	
`OCTET/BIT STRING`	`&bin`	base64url
`OBJECT IDENTIFIER`	dotted `oid=` fields	decoded
`UTCTime` etc.	`&datetime`	DER-normalized
`OPTIONAL` / `DEFAULT`	`?=` / defaults	direct alignment
`SET (OF)`	ordered `SEQUENCE`	as DER requires
`WITH SYNTAX`, generics	—	not covered
BER / PER / XER / OER	—	kaiv is one text encoding

The last three rows are the ∼5%. Unordered `SET` would cost the linear-time validation guarantee — and DER already forces `SET OF` into sorted order, so kaiv's ordered-only stance is what your canonical encoder does today. The `WITH SYNTAX` macro layer and parameterized types are schema-authoring conveniences with no kaiv equivalent; and of ASN.1's family of encoding rules, kaiv is a single text encoding — if you need PER's bit-packing, you need ASN.1.

5 When to Stay with ASN.1

An honest map marks the roads not taken. ASN.1 remains the right choice when:

- the protocol is specified in it — 3GPP, ITU-T, ICAO, PKI: the schema *is* the standard, and interoperability means X.680 to the letter;
- PER's bit-level compactness is the budget — radio interfaces where kaiv's text form is not a candidate;
- your assurance case cites forty years of formal methodology and tooling.

kaiv's pull is for the systems *around* the protocol: inspection, archival, diffing, test fixtures, and new contracts that want ASN.1's constraint discipline without the compiler ecosystem — one grammar, one scan, and DER kept byte-faithful at the boundary.

6 Where Next

From here:

- the kaiv specification — the formal grammar and semantics;
- `kaiv help` — the full toolchain surface;
- the sibling cookbooks — kaiv for JSON, YAML, TOML, XML, Protocol Buffers, Avro, CBOR, and GraphQL users.