ARIES FRAMEWORK JAVASCRIPT RFC0454 PRESENT PROOF V2 TECHNICAL & PLANNING NOTES

03rd December 2021

Prasad Katkar | Amit Padmani
prasad@northernblock.io I amit@northernblock.io

http://www.northernblock.io

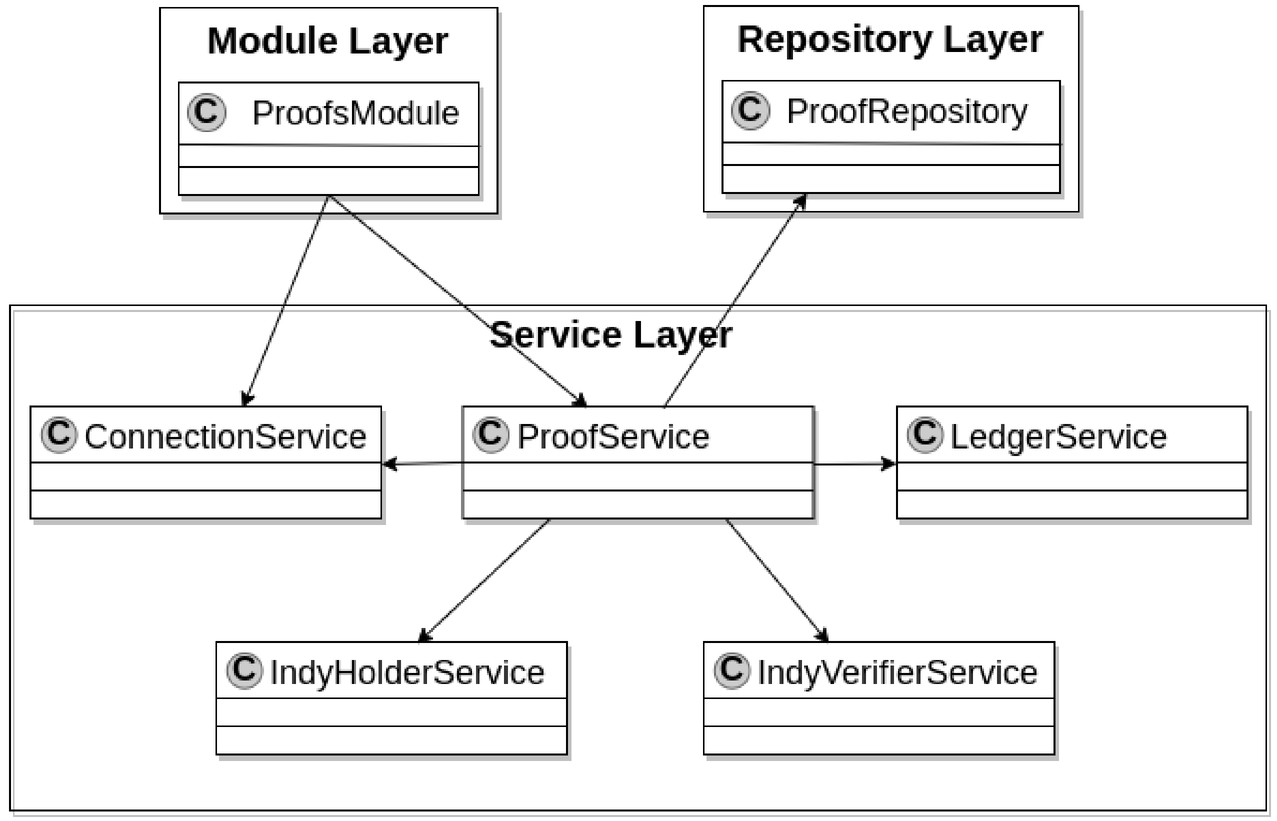
TECHNICAL OVERVIEW

Northern Block

- CURRENT AND PROPOSED ARCHITECTURE
- FOLDER STRUCTURE
- USE CASES / SUBTASKS
- PLANNING/SEQUENCING OF TASKS
- UML CLASS DIAGRAMS (INITIAL INTERFACES)
- PROOF ATTACHMENT FORMATS

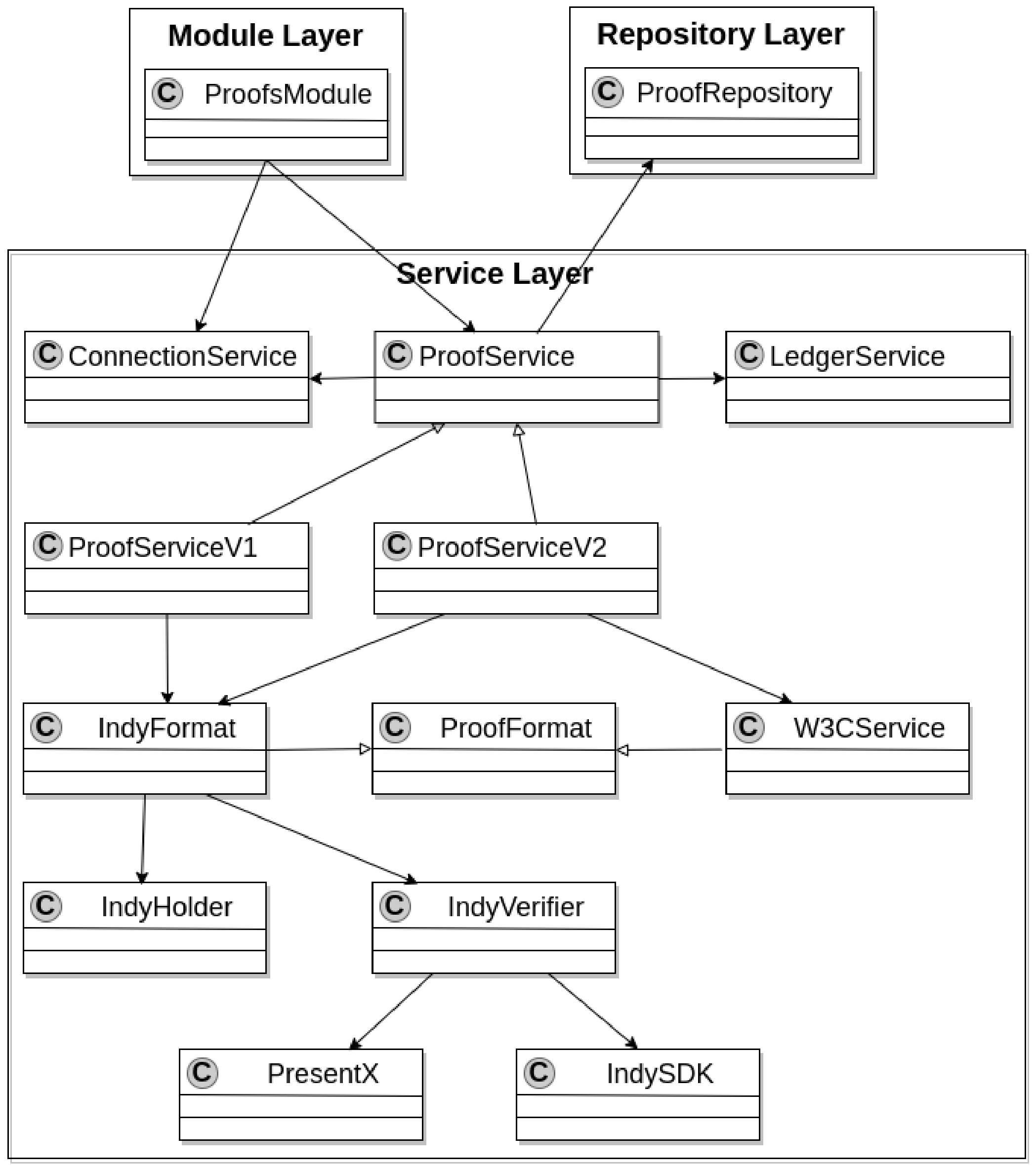
CURRENT ARCHITECTURE

Northern Block



PROPOSED ARCHITECTURE

Northern Block



One module as entry
Point to different services

V2 reuses V1 Indy code and format(s) but within pluggable architecture

PROPOSED ARCHITECTURE

Continued…

ProofModule
Contains various classes, functions for proofs.

ProofRepository
Helps to store the proof records into the wallet.

ConnectionService
Used to fetch / check the connection record from the wallet.

ProofService
Provides various service methods to do proof verification like Propose, Request, Present Presentation/ Proof.

LedgerService
Provides service methods to fetch various data from the Ledger. for e.g get the credential definition details for proof from ledger.

ProofServiceV1
Has service methods specifically for present-proof version 1 transactions.

ProofServiceV2
Has service methods specifically for present-proof version 2 transactions.

IndyFormat
Validate indy attachment data for a specific message type.

ProofFormat
Validate the present-proof message attachment format for Indy or W3C(dif)

W3CServices
Validate w3c(dif) attachment data for a specific message type.

IndyHolder
Services for holder to fetch data from the wallet required for presenting proof using Indy SDK

IndyVerifier
Services for the verifier to verify the proof presented by the holder using Indy SDK.

PresentX
Presentation Exchange Record

IndySDK
Various Indy methods to perform proof transactions for Issuer, Holder and Verifier

FOLDER STRUCTURE

Service -> Credential Format
V1 -> Indy (only)
V2 -> JSON-LD initially (others to follow)

Protocol
Indy or
W3C

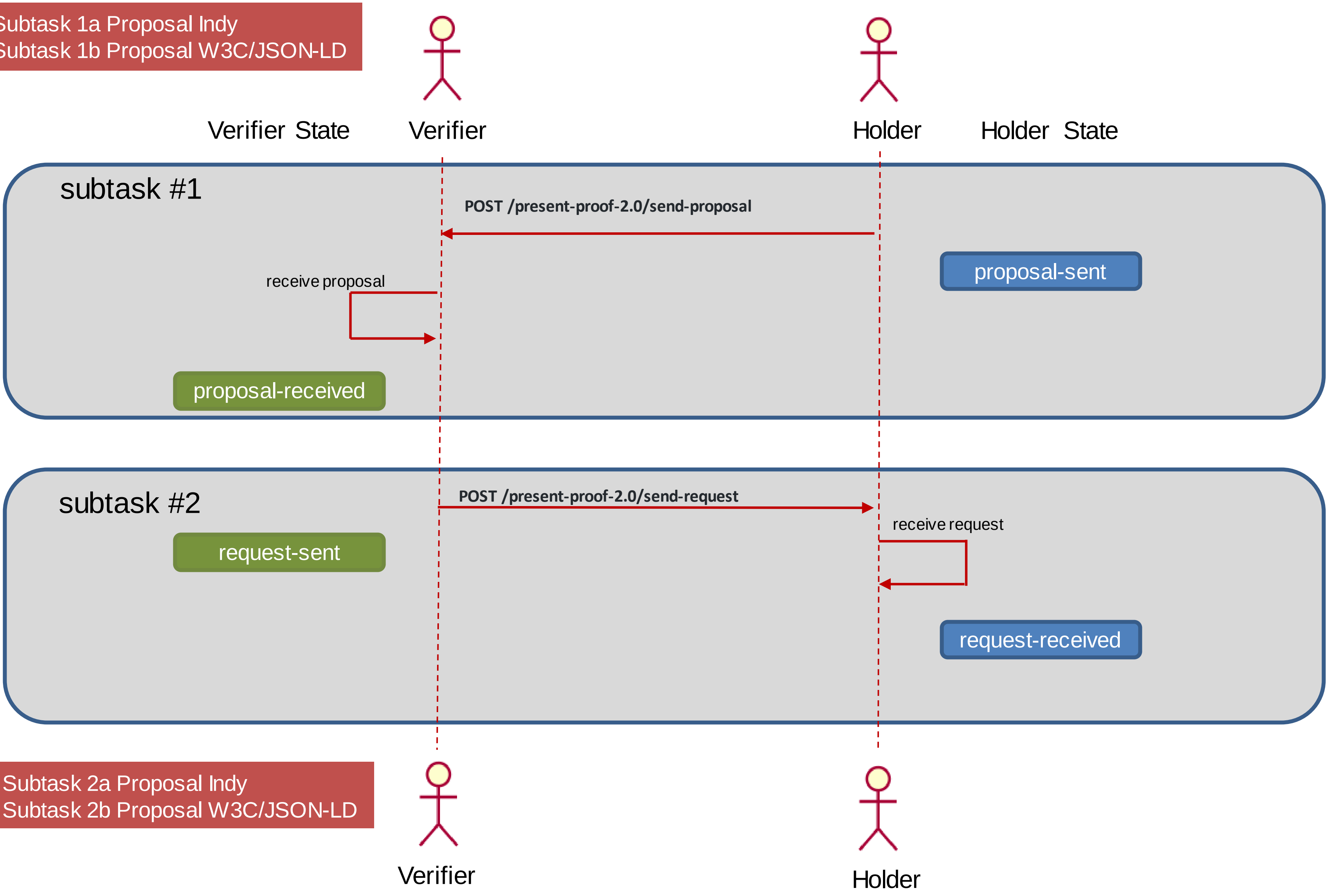
```
.
└── src
    └── modules
        └── proofs
            ├── v1
            │   ├── messages
            │   │   ├── V1ProposePresentationMessage.ts
            │   │   └── ... other messages ...
            │   ├── handlers
            │   │   ├── V1ProposePresentationHandler.ts
            │   │   └── ... other messages ...
            │   └── V1PresentationService.ts                       # only supports indy format, so can use that class directly
            ├── v2
            │   ├── messages
            │   │   ├── V2ProposePresentationMessage.ts
            │   │   └── ... other messages ...
            │   ├── handlers
            │   │   ├── V2ProposePresentationHandler.ts
            │   │   └── ... other messages ...
            │   ├── V2PresentationService.ts                      # declares which formats it supports through mapping
            │   ├── formats
            │   │   ├── PresentationFormatService.ts
            │   │   ├── indy
            │   │   │   └── IndyPresentationFormatService.ts
            │   │   └── jsonld
            │   │       └── jsonldPresentationFormatService.ts
            │   ├── PresentationRecord.ts
            │   ├── PresentationRepository.ts
            │   ├── PresentationService.ts                       # can return the implementation specific handlers (v1, v2, etc...) for
            │   └── PresentationModule.ts                          registration in the dispatcher to the module
            ├── indy                                             # all logic in this folder is agnostic of DIDComm and its protocols
            │   ├── IndyHolderService.ts                         # get the credential for proof request
            │   └── IndyVerifierService.ts                       # handles verification of indy credentials
            └── w3c                                              # all logic in this folder is agnostic of DIDComm and its protocols
                ├── W3cVcService.ts                              # handles issuing/ holding/ proving/ verifying w3c credentials
                ├── W3cVcRecord.ts                               # w3c vc record to fetch stored credentials
                └── W3cVcRepository.ts
```

RFC0454
PROPOSAL USE CASE

PRESENT PROOF V2 PROPOSAL/ REQUEST USE CASE

Northern Block

Subtask 1a Proposal Indy
Subtask 1b Proposal W3C/JSON-LD



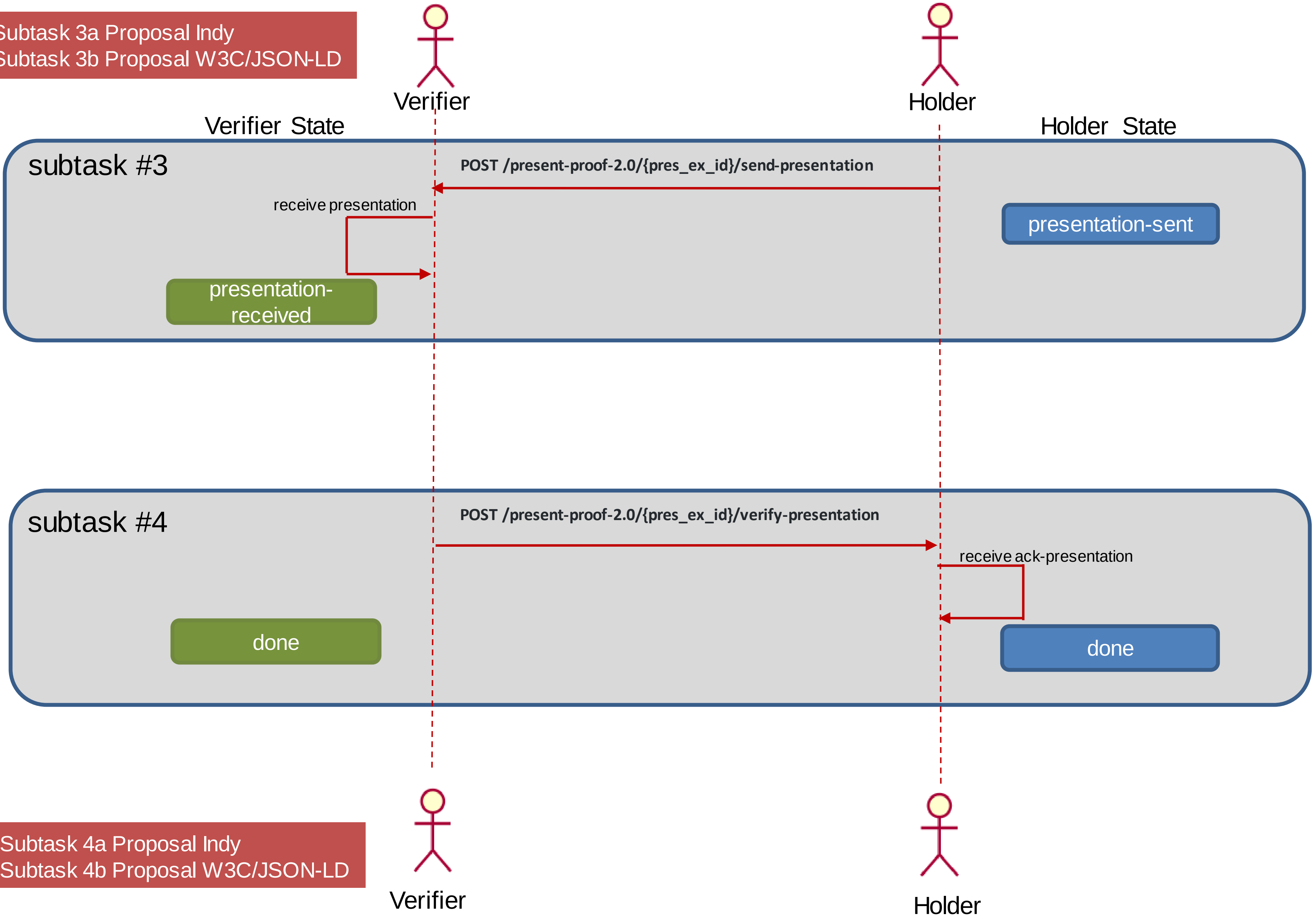
Subtask 2a Proposal Indy
Subtask 2b Proposal W3C/JSON-LD

RFC0454 PRESENTATION USE CASE

PRESENT PROOF V2 PRESENTATION / VERIFY USE CASE

Northern Block

Subtask 3a Proposal Indy
Subtask 3b Proposal W3C/JSON-LD

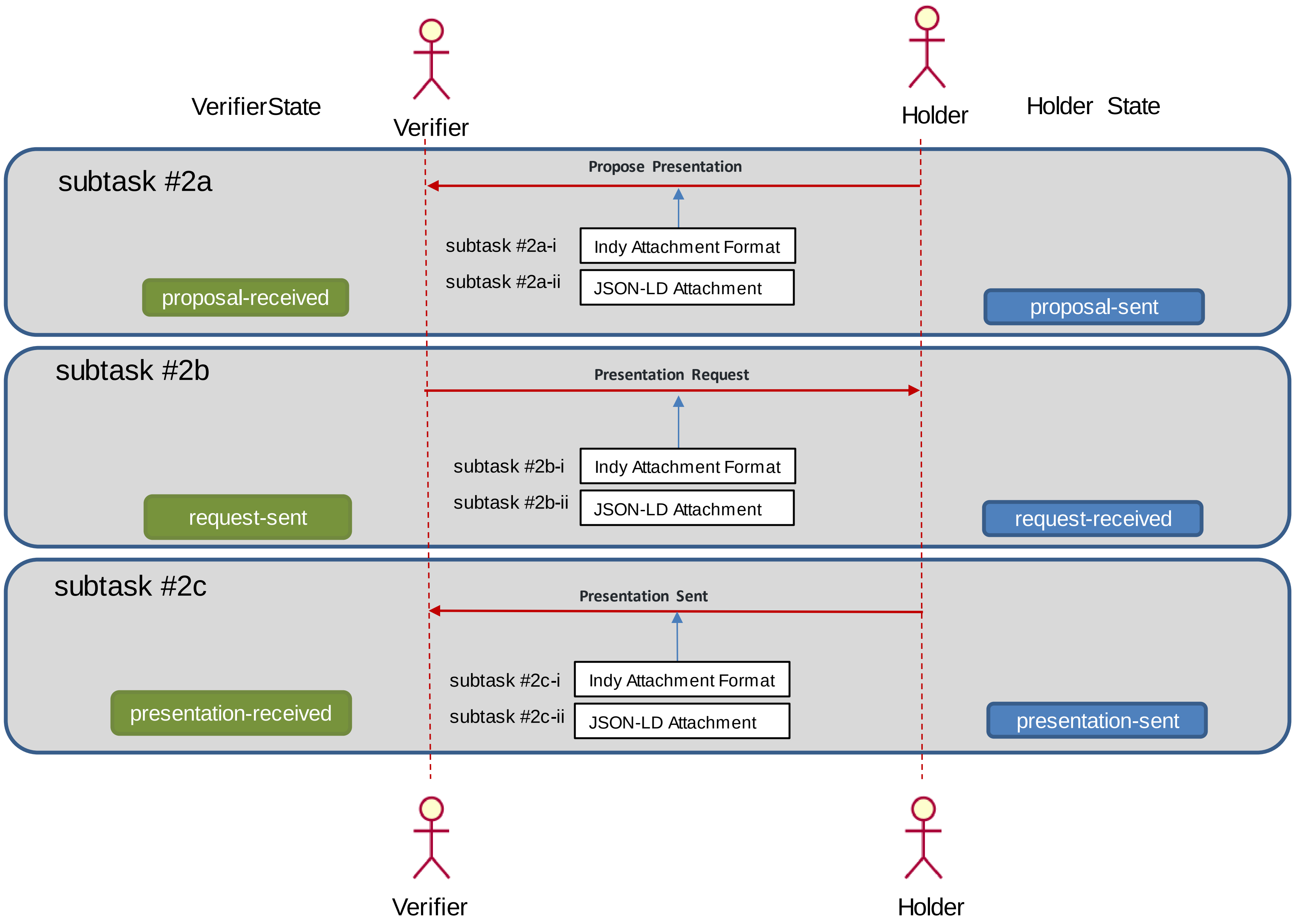


Subtask 4a Proposal Indy
Subtask 4b Proposal W3C/JSON-LD

RFC0592/0510 FORMATS

PROOF REQUEST V2 ATTACHMENT FORMATS

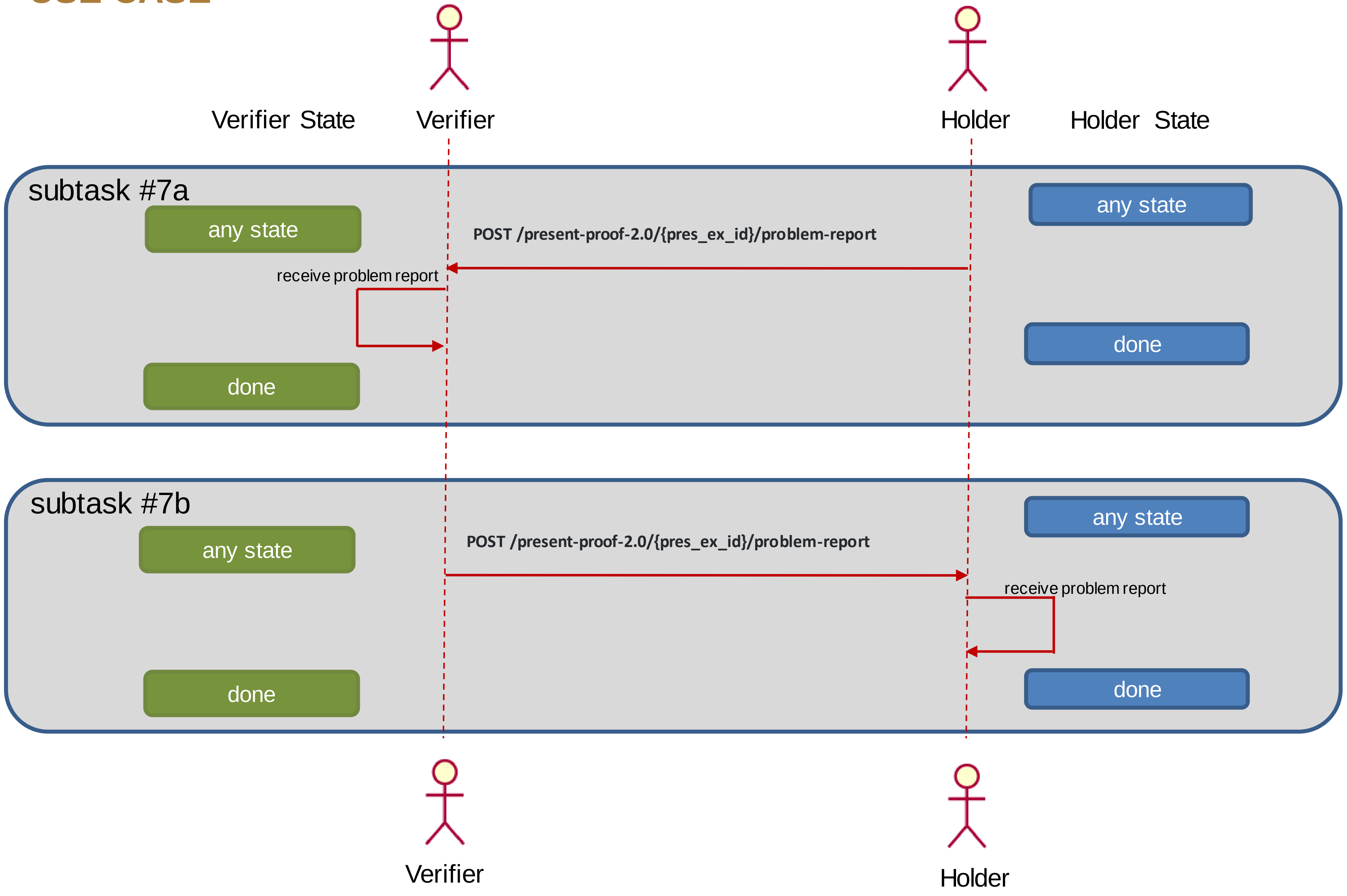
Northern Block



RFC0454
PROBLEM REPORT
USE CASE

PRESENT PROOF V2 PROBLEM REPORT

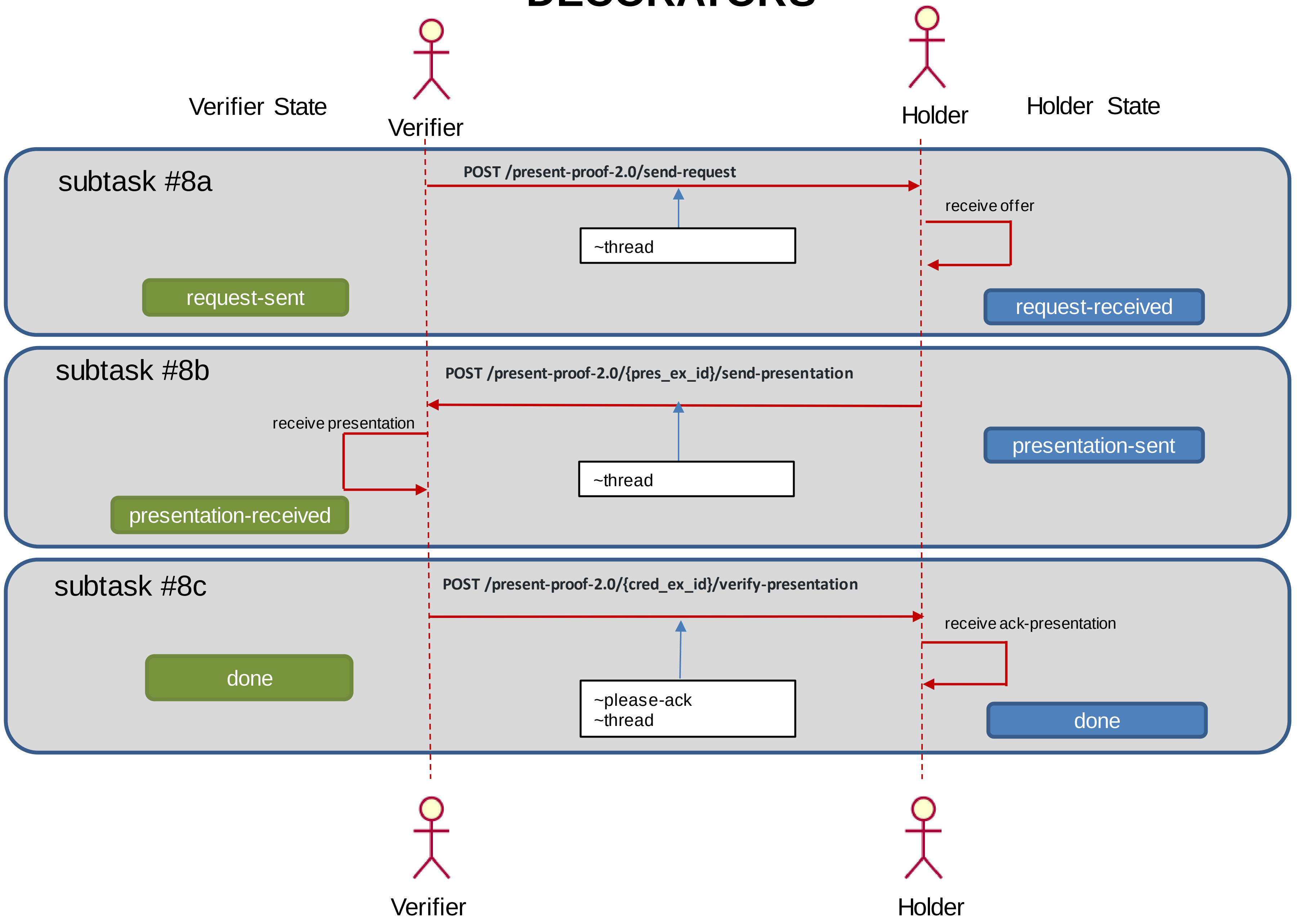
Northern Block



https://github.com/hyperledger/aries-rfcs/blob/main/features/0035-report-problem/README.md

RFC0454
DECORATOR USE CASE

PRESENT PROOF V2 DECORATORS



PLANNING

PRESENT PROOF V2 PROPOSED SEQUENCING

Northern Block

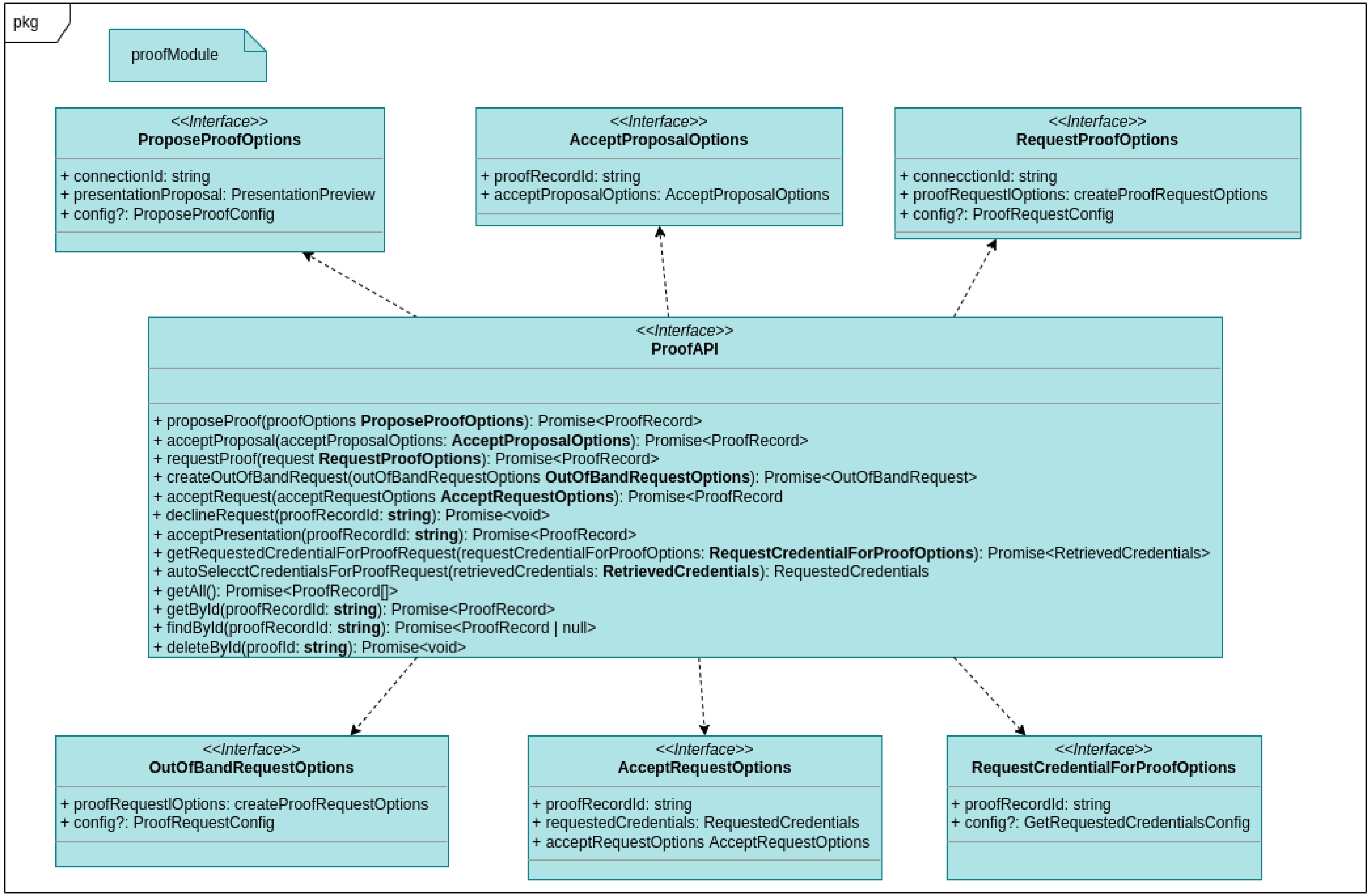
V2 Proofs (Indy)

Sprint 1				
Subtask 1a Send Proposal (Indy)	Subtask 2a Send Request (Indy)	Subtask 3a Send Presentation (Indy)	Subtask 4a Verify Presentation	Subtask 7 Problem Report
Subtask 2a-i Indy Attachment Formats (Proposal)	Subtask 2b-i Indy Attachment Formats (Request)	Subtask 2c-i Indy Attachment Formats (Presentation)	Subtask 8a Proof Decorators	
Subtask 9 Repackage Indy Proofs V1			Subtask 8b Send Presentation Ack	

Repeat above timeline for V2 Proofs (W3C/JSON-LD)

CLASS DIAGRAM

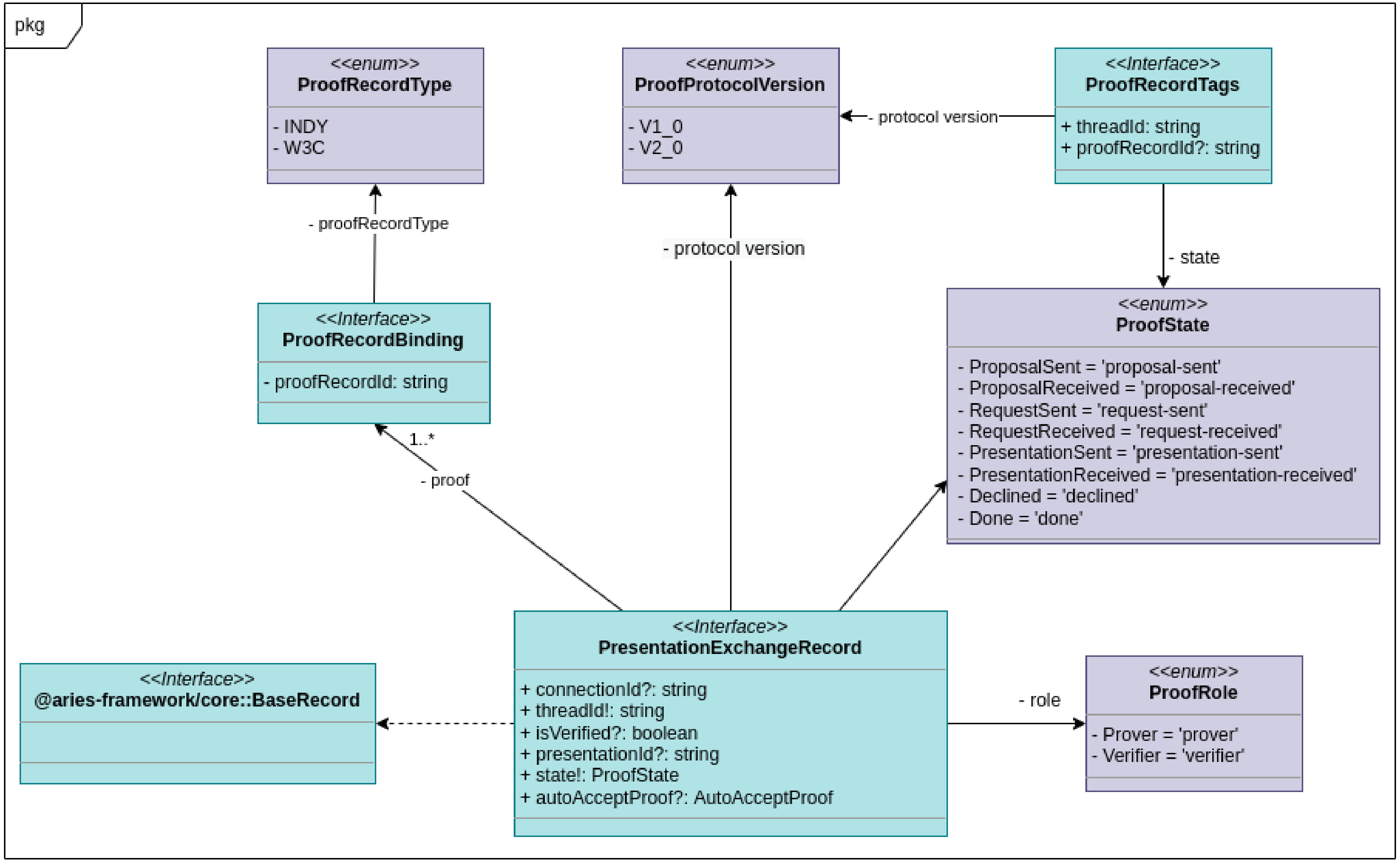
ProofModule


pkg
proofModule
<<Interface>>
ProposeProofOptions
+ connectionId: string
+ presentationProposal: PresentationPreview
+ config?: ProposeProofConfig
<<Interface>>
AcceptProposalOptions
+ proofRecordId: string
+ acceptProposalOptions: AcceptProposalOptions
<<Interface>>
RequestProofOptions
+ connecctionId: string
+ proofRequestIOptions: createProofRequestOptions
+ config?: ProofRequestConfig
<<Interface>>
ProofAPI
+ proposeProof(proofOptions ProposeProofOptions): Promise<ProofRecord>
+ acceptProposal(acceptProposalOptions: AcceptProposalOptions): Promise<ProofRecord>
+ requestProof(request RequestProofOptions): Promise<ProofRecord>
+ createOutOfBandRequest(outOfBandRequestOptions OutOfBandRequestOptions): Promise<OutOfBandRequest>
+ acceptRequest(acceptRequestOptions AcceptRequestOptions): Promise<ProofRecord
+ declineRequest(proofRecordId: string): Promise<void>
+ acceptPresentation(proofRecordId: string): Promise<ProofRecord>
+ getRequestedCredentialForProofRequest(requestCredentialForProofOptions: RequestCredentialForProofOptions): Promise<RetrievedCredentials>
+ autoSelecctCredentialsForProofRequest(retrievedCredentials: RetrievedCredentials): RequestedCredentials
+ getAll(): Promise<ProofRecord[]>
+ getById(proofRecordId: string): Promise<ProofRecord>
+ findById(proofRecordId: string): Promise<ProofRecord | null>
+ deleteById(proofId: string): Promise<void>
<<Interface>>
OutOfBandRequestOptions
+ proofRequestIOptions: createProofRequestOptions
+ config?: ProofRequestConfig
<<Interface>>
AcceptRequestOptions
+ proofRecordId: string
+ requestedCredentials: RequestedCredentials
+ acceptRequestOptions AcceptRequestOptions
<<Interface>>
RequestCredentialForProofOptions
+ proofRecordId: string
+ config?: GetRequestedCredentialsConfig

CLASS DIAGRAM

PresentationExchangeRecord MODULE

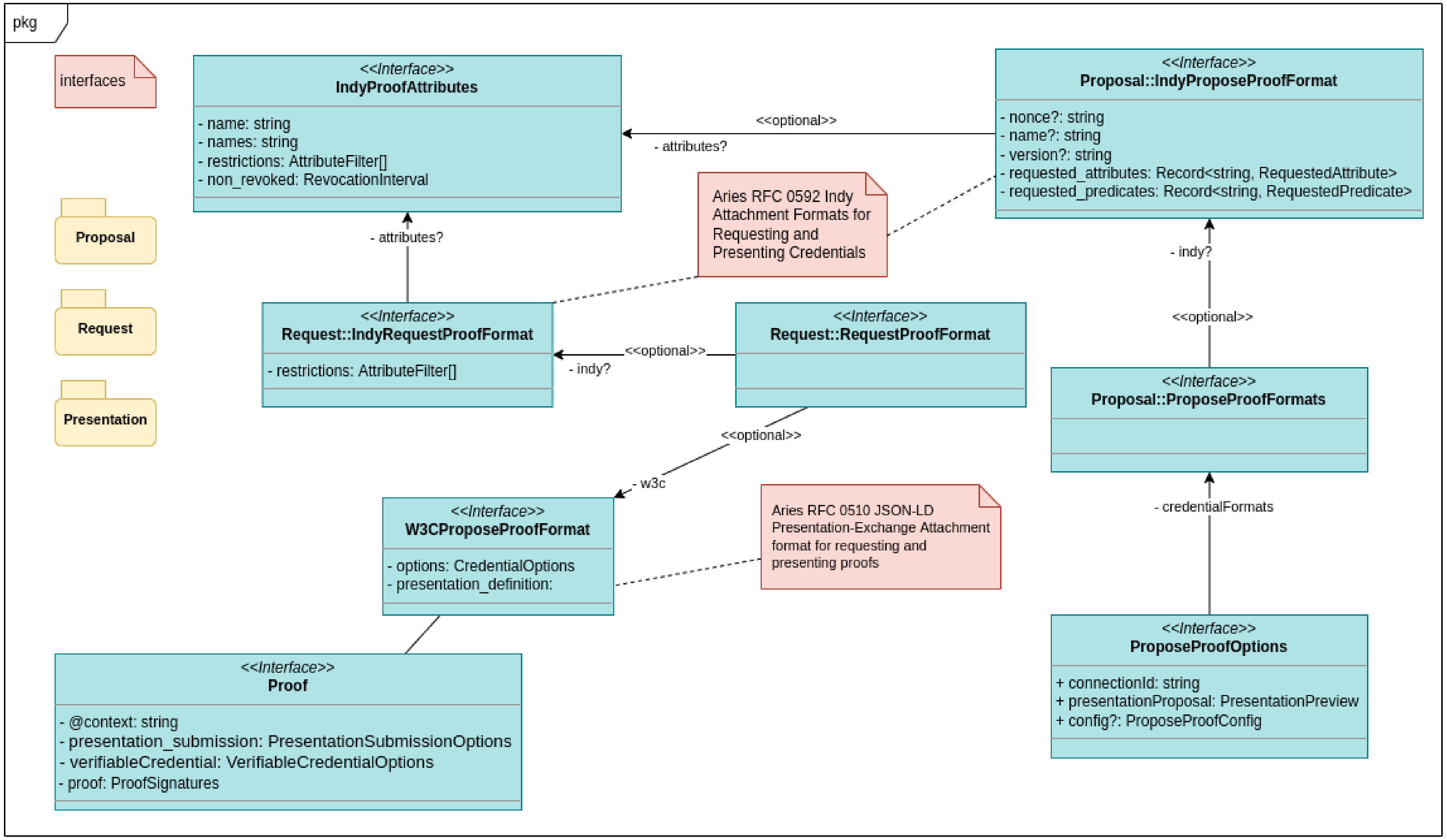
Northern Block



CLASS DIAGRAM

Interfaces MODULE

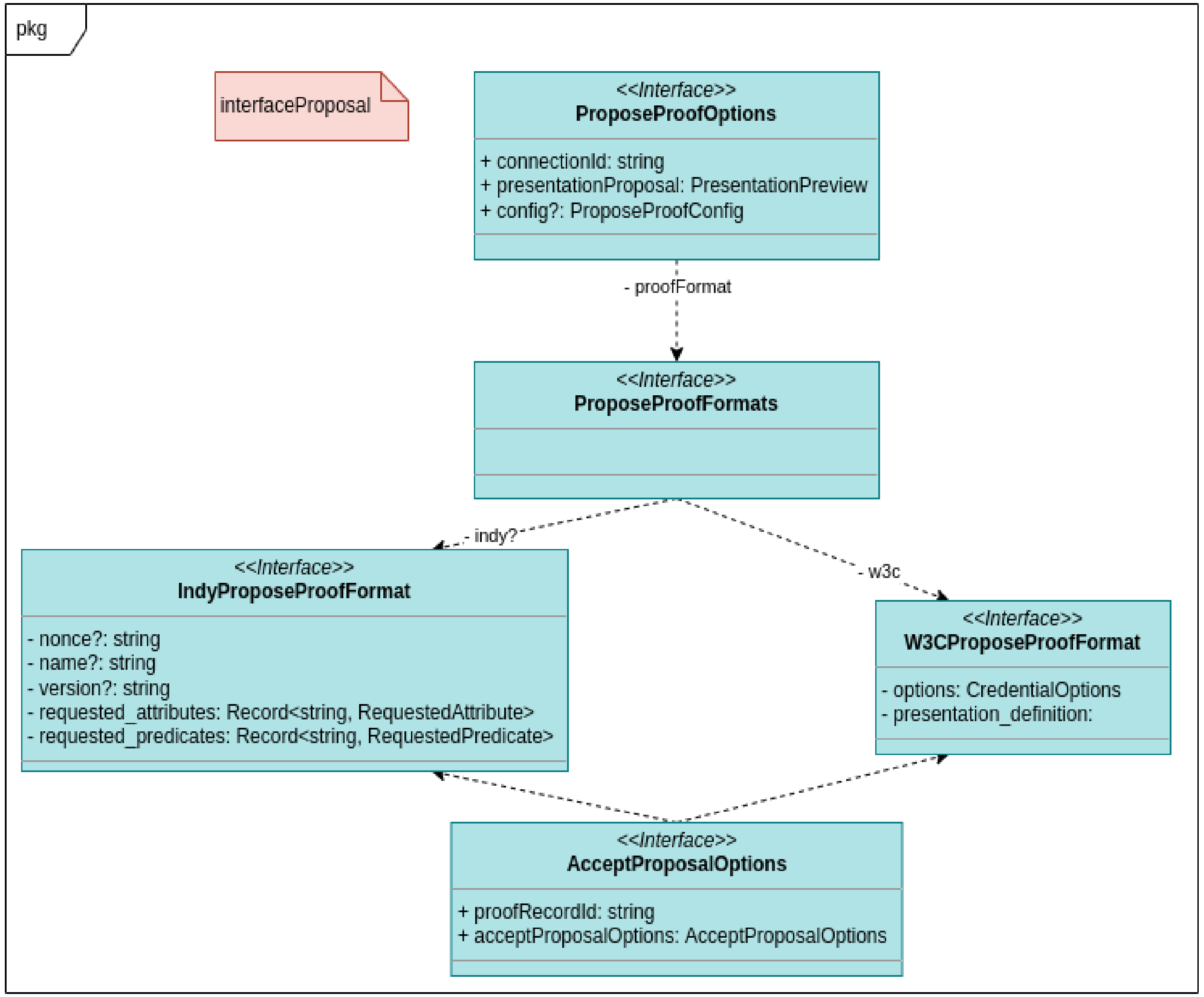
Northern Block


pkg
interfaces
Proposal
Request
Presentation
<<Interface>>
IndyProofAttributes
- name: string
- names: string
- restrictions: AttributeFilter[]
- non_revoked: RevocationInterval
<<Interface>>
Proposal::IndyProposeProofFormat
- nonce?: string
- name?: string
- version?: string
- requested_attributes: Record<string, RequestedAttribute>
- requested_predicates: Record<string, RequestedPredicate>
<<optional>>
- attributes?
Aries RFC 0592 Indy Attachment Formats for Requesting and Presenting Credentials
- attributes?
- indy?
<<optional>>
<<Interface>>
Request::IndyRequestProofFormat
- restrictions: AttributeFilter[]
<<optional>>
- indy?
<<Interface>>
Request::RequestProofFormat
<<Interface>>
Proposal::ProposeProofFormats
<<optional>>
- w3c
<<Interface>>
W3CProposeProofFormat
- options: CredentialOptions
- presentation_definition:
Aries RFC 0510 JSON-LD Presentation-Exchange Attachment format for requesting and presenting proofs
- credentialFormats
<<Interface>>
Proof
- @context: string
- presentation_submission: PresentationSubmissionOptions
- verifiableCredential: VerifiableCredentialOptions
- proof: ProofSignatures
<<Interface>>
ProposeProofOptions
+ connectionId: string
+ presentationProposal: PresentationPreview
+ config?: ProposeProofConfig

CLASS DIAGRAM

Interfaces/Proposal MODULE

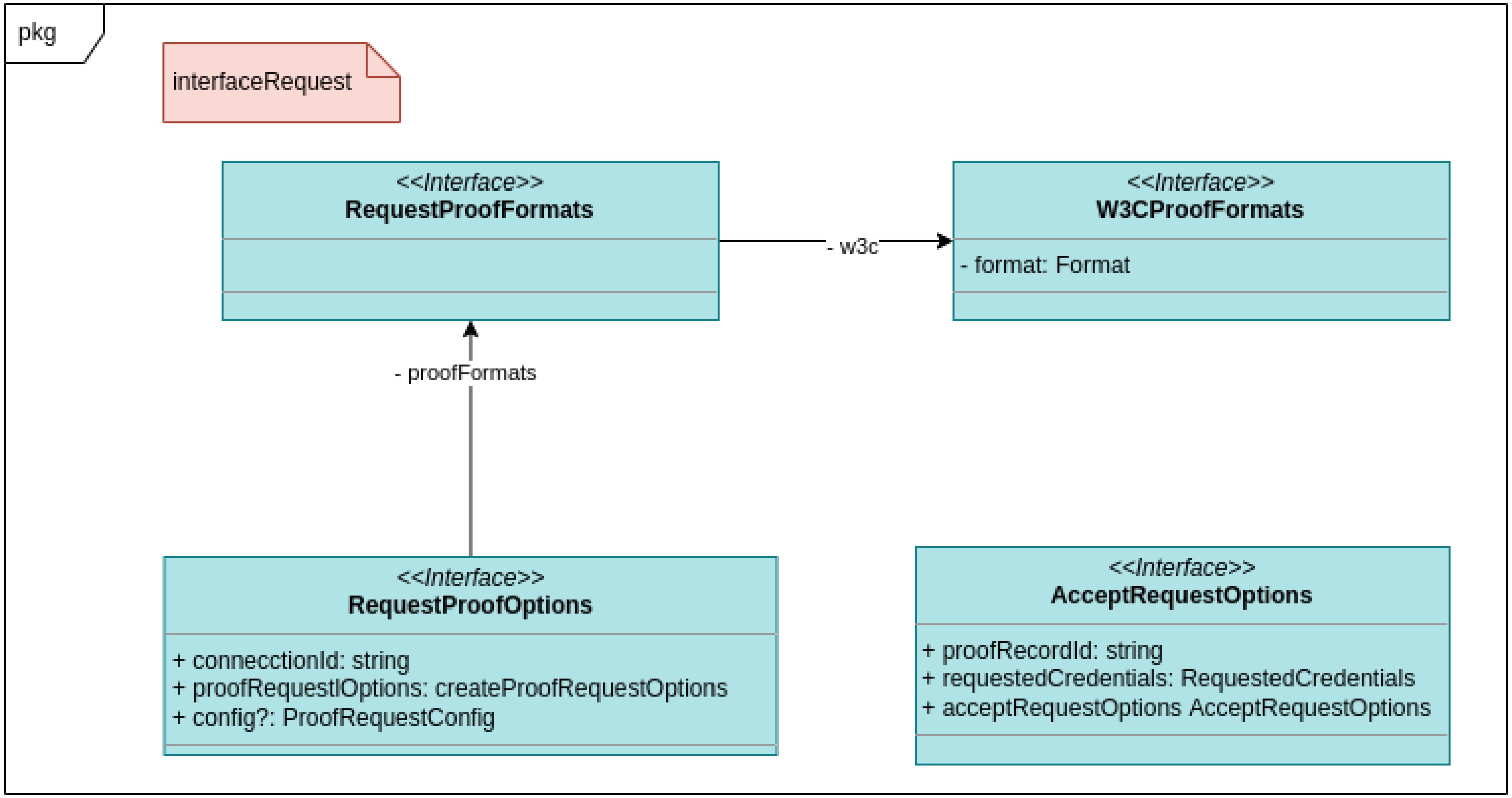
Northern Block



CLASS DIAGRAM

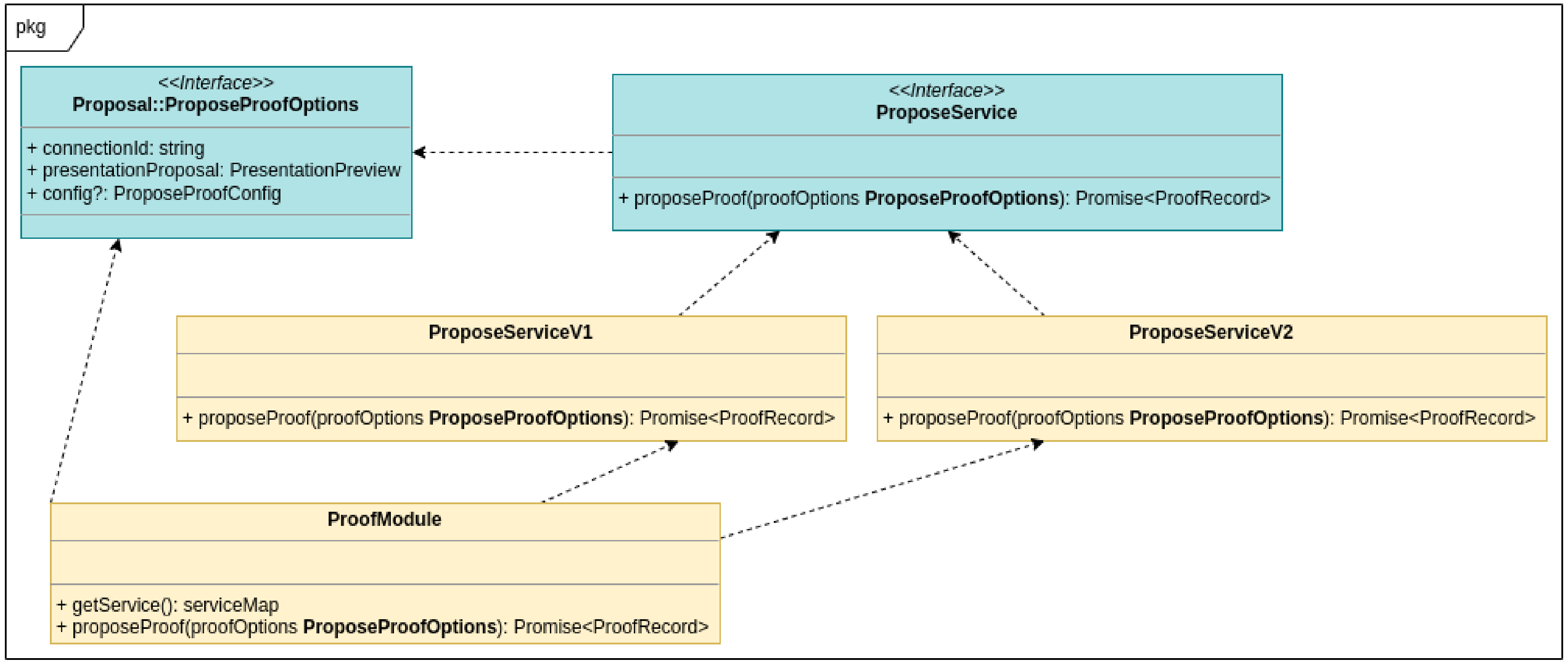
Interfaces/Request MODULE

Northern Block



CLASS DIAGRAM

service-example MODULE

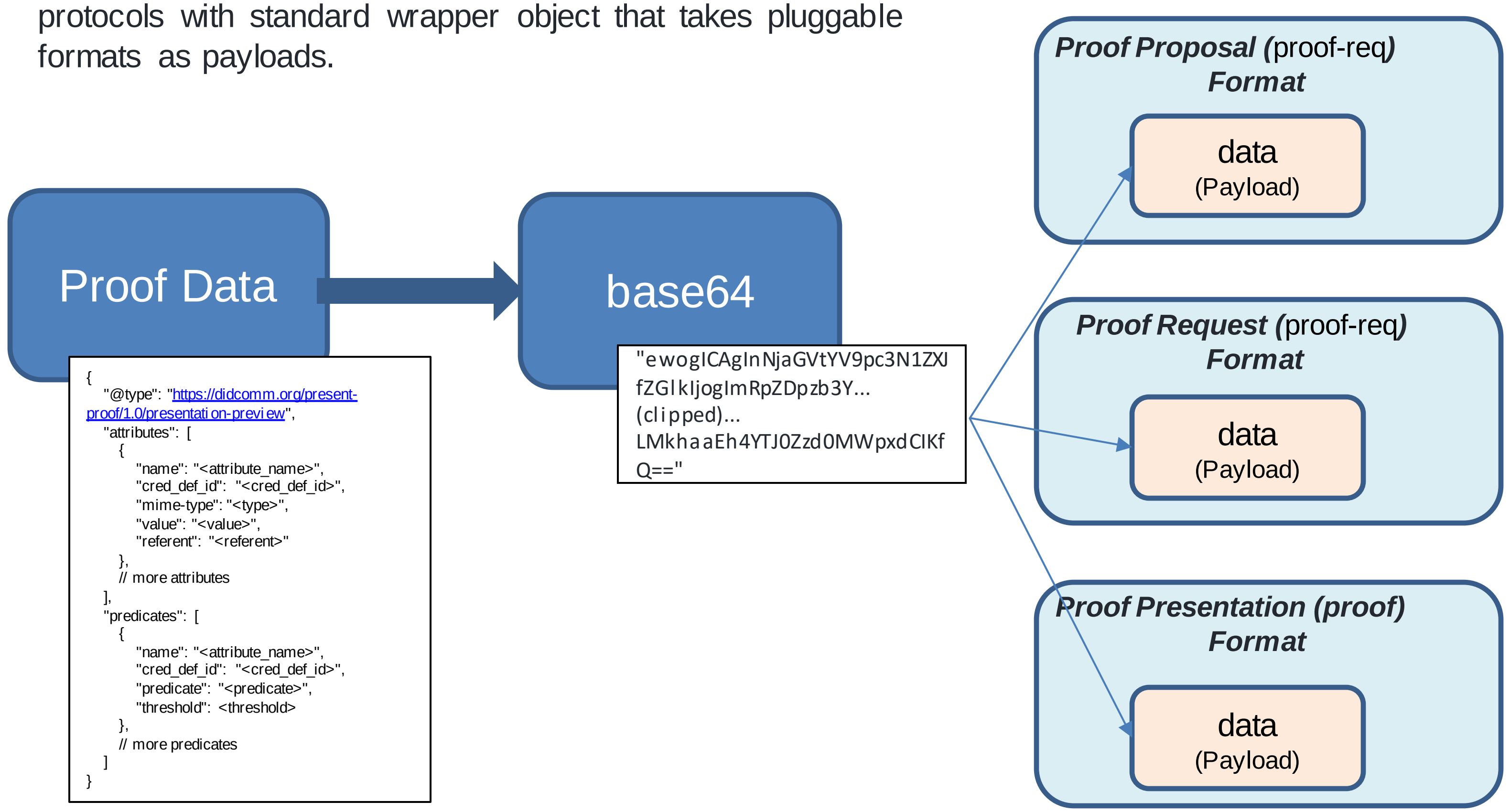


pkg
<<Interface>>
Proposal::ProposeProofOptions
+ connectionId: string
+ presentationProposal: PresentationPreview
+ config?: ProposeProofConfig
<<Interface>>
ProposeService
+ proposeProof(proofOptions ProposeProofOptions): Promise<ProofRecord>
ProposeServiceV1
+ proposeProof(proofOptions ProposeProofOptions): Promise<ProofRecord>
ProposeServiceV2
+ proposeProof(proofOptions ProposeProofOptions): Promise<ProofRecord>
ProofModule
+ getService(): serviceMap
+ proposeProof(proofOptions ProposeProofOptions): Promise<ProofRecord>

INDY PROOF ATTACHMENT FORMATS

RFC 0592 (Indy Proof Format)

Allows Indy-style Proofs to be used with credential-related protocols with standard wrapper object that takes pluggable formats as payloads.


Proof Data
base64
Proof Proposal (proof-req) *Format*
data (Payload)
Proof Request (proof-req) *Format*
data (Payload)
Proof Presentation (proof) Format
data (Payload)

```
{
   "@type": "https://didcomm.org/present-proof/1.0/presentation-preview",
   "attributes": [
      {
         "name": "<attribute_name>",
         "cred_def_id": "<cred_def_id>",
         "mime-type": "<type>",
         "value": "<value>",
         "referent": "<referent>"
      },
      // more attributes
   ],
   "predicates": [
      {
         "name": "<attribute_name>",
         "cred_def_id": "<cred_def_id>",
         "predicate": "<predicate>",
         "threshold": <threshold>
      },
      // more predicates
   ]
}
```

"ewogICAgInNjaGVtYV9pc3N1ZXJfZGlkIjogImRpZDpzb3Y...(clipped)...LMkhaaEh4YTJ0ZzdOMWpxdCIKfQ=="

For details of all formats see:
https://github.com/hyperledger/aries-rfcs/blob/main/features/0592-indy-attachments/README.md

INDY ATTACHMENT FORMATS - EXAMPLE

Northern Block

Example: Proof Request Format
(used for Proof Request sent by holder to verifier)

Payload:

```
{
   "nonce": “2934823091873049823740198370q23984710239847”,
   "name":"proof_req_1",
   "version":"0.1",
   "requested_attributes":{
      "attr1_referent": {"name":"sex"},
      "attr2_referent": {"name":"phone"},
      "attr3_referent": {"names": ["name", "height"], "restrictions":
<restrictions specifying government-issued ID>}
   },
   "requested_predicates":{
      "predicate1_referent":{"name":"age","p_type":">=","p_value":18}
   }
}
```

```
{
    "@id": "<uuid of propose message>",
    "@type": "https://didcomm.org/present-
proof/%VER/request-presentation",
    "comment": "<some comment>",
    "formats" : [{
        "attach_id": "<attach@id value>",
        "format": "hlindy/proof-req@v2.0"
    }],
    "filters~attach": [{
        "@id": "<attach@id value>",
        "mime-type": "application/json",
        "data": {
            "base64":
          "ewogICAgInNjaGVtYV9pc3N1ZXJfZGlkIjogImRpZDpzb3Y.
..        (clipped)... LMkhaaEh4YTJ0Zzd0MWpxdCIKfQ=="
        }
    }]
}
```

For details of all formats see:

https://github.com/hyperledger/aries-rfcs/blob/main/features/0592-indy-attachments/README.md

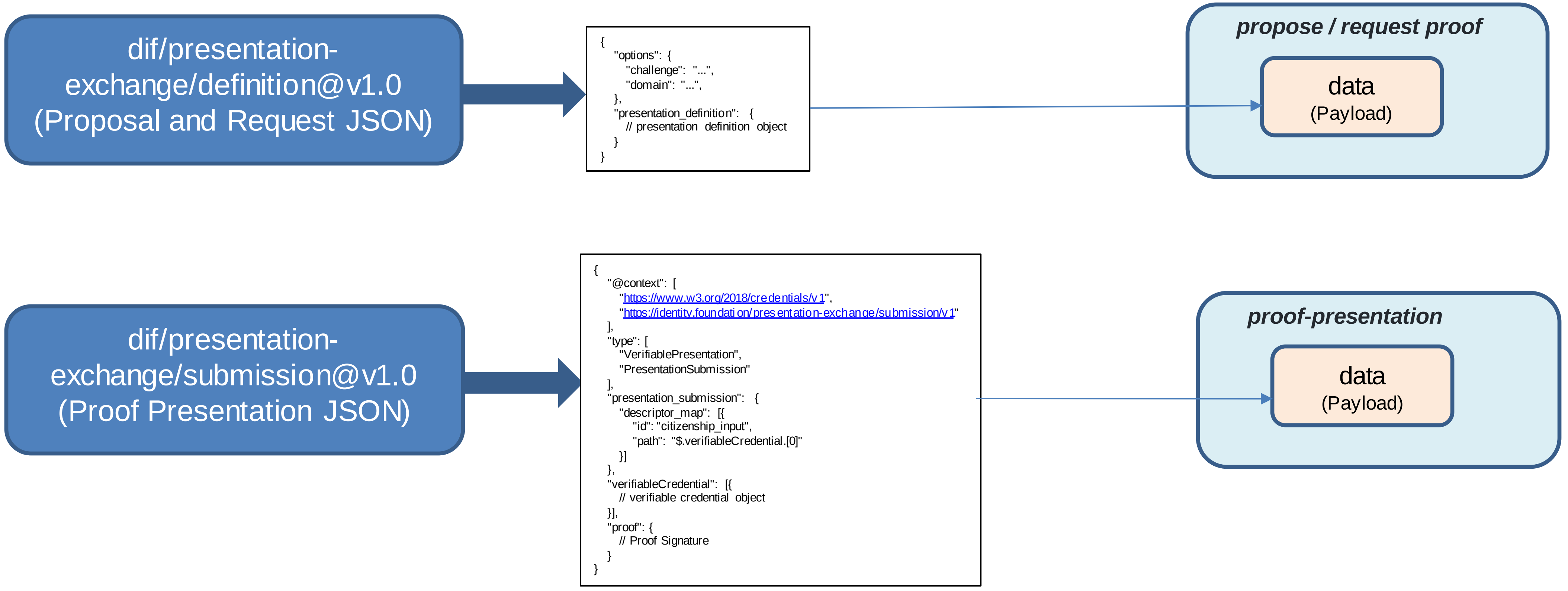
FORMATS

JSON-LD ATTACHMENT FORMATS

Northern Block

RFC 0510(Presentation-Exchange Attachment format for requesting and presenting proofs)

Presentation Exchange defines a data format capable of articulating a rich set of proof requirements from Verifiers, and also provides a means of describing the formats in which Provers must submit those proofs.


dif/presentation-exchange/definition@v1.0 (Proposal and Request JSON)
propose / request proof
data (Payload)
dif/presentation-exchange/submission@v1.0 (Proof Presentation JSON)
proof-presentation
data (Payload)

```
{
  "options": {
    "challenge": "...",
    "domain": "...",
  },
  "presentation_definition": {
    // presentation definition object
  }
}
```

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://identity.foundation/presentation-exchange/submission/v1"
  ],
  "type": [
    "VerifiablePresentation",
    "PresentationSubmission"
  ],
  "presentation_submission": {
    "descriptor_map": [{
      "id": "citizenship_input",
      "path": "$.verifiableCredential.[0]"
    }]
  },
  "verifiableCredential": [{
    // verifiable credential object
  }],
  "proof": {
    // Proof Signature
  }
}
```

https://github.com/hyperledger/aries-rfcs/blob/b3a3942ef0/features/0510-dif-pres-exch-attach/README.md

JSON-LD ATTACHMENT FORMATS - EXAMPLE

Northern Block

Example: **dif/presentation-exchange/definition@v1.0**
Attachment Format
(used to formally propose, request proof)

Payload:

```
{
   "json": {
      "options": {
         "challenge": "23516943-1d79-4ebd-8981-623f036365ef",
         "domain": "us.gov/DriversLicense"
      },
      "presentation_definition": {
         "input_descriptors": [{
            "id": "citizenship_input",
            "name": "US Passport",
            "group": ["A"],
            "schema": [{
               "uri": "hub://did:foo:123/Collections/schema.us.gov/passport.json"
            }],
            "constraints": {
               "fields": [{
                  "path": ["$.credentialSubject.birth_date", "$.birth_date"],
                  "filter": {
                     "type": "date",
                     "minimum": "1999-5-16"
                  }
               }]
            }
         }],
         "format": {
            "ldp_vp": {
               "proof_type": [["BbsBlsSignatureProof2020"] / "Ed25519Signature2018"]
            }
         }
      }
   }
}
```

Proof Proposal message

```
{
   "@type": "https://didcomm.org/present-proof/%VER/propose-presentation",
   "@id": "fce30ed1-96f8-44c9-95cf-b274288009dc",
   "comment": "some comment",
   "formats" : [{
      "attach_id" : "143c458d-1b1c-40c7-ab85-4d16808ddf0a",
      "format" : "dif/presentation-exchange/definition@v1.0"
   }],
   "proposal~attach": [{
      "@id": "143c458d-1b1c-40c7-ab85-4d16808ddf0a",
      "mime-type": "application/json",
      "data": {
         ...
      }
   }]
}
```

For details of all formats see:
https://github.com/hyperledger/aries-rfcs/blob/b3a3942ef0/features/0510-dif-pres-exch-attach/README.md