A YANG Data Model for CMIS Access and Control

draft-hi-ccamp-cmis-control-yang-01

6th November, 2025

Shunsuke Homma (NTT)

Hitoshi Irino (NTT)

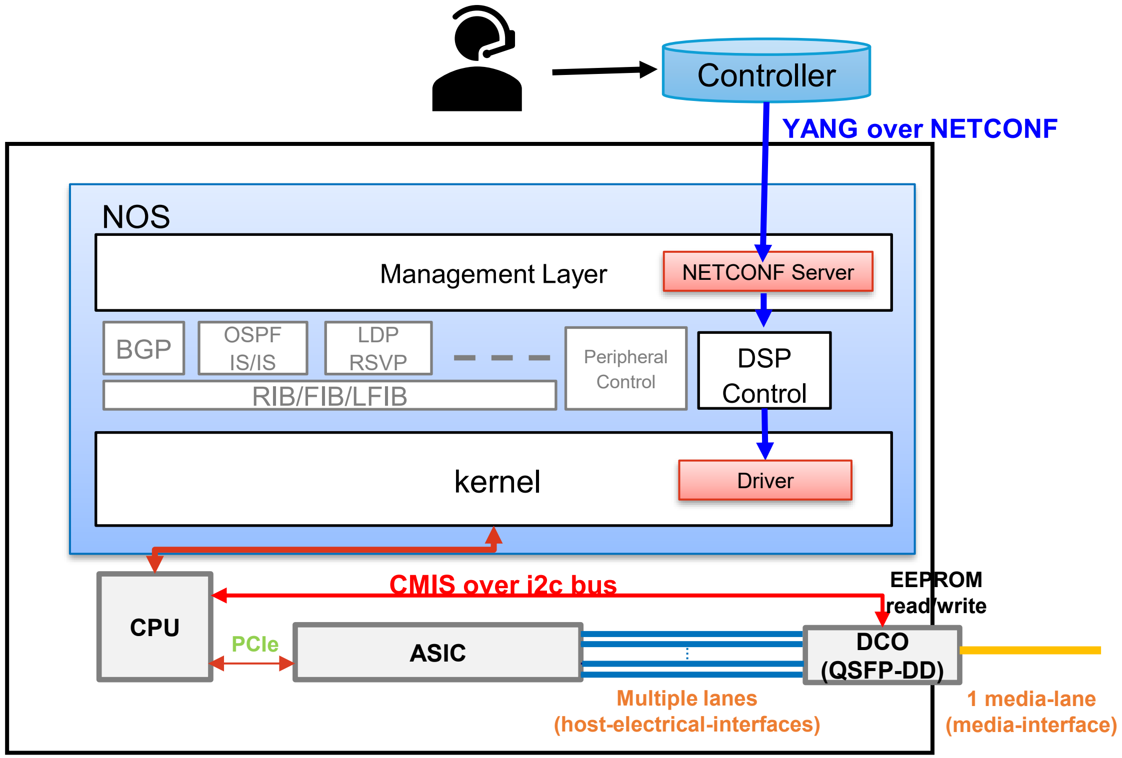
Toru Mano (NTT)

Kazuya Anazawa (NTT)

Yuji Tochio (1Finity)

Recap: Concept of Proposal

- Proposed YANG model enables external systems to directly manage DSP registers with CMIS schema.
  - “page-num” and “bank” are used for indicating target memory
  - Read/Written data is defined with “offset”, “size”, and “data-value”.
- This YANG is augmented to “ietf-interfaces”.




```
module: ietf-cmis-control

  augment /if:interfaces/if:interface:
    +--rw cmis-control
       +--ro cmis-enabled?    boolean
       +--ro cmis-version?    string
       +--rw cmis-page* [page-num]
          +--rw page-num              uint8
          +--rw bank                  uint8
          +--ro page-access-type?     access-type
          +--rw description?          string
          +--rw value* [offset]
             +--rw offset                   uint8
             +--rw size                     uint8
             +--ro value-access-type?       access-type
             +--rw value-data               binary
             +--rw description?             string
```

Page Info.

Value Info.

Tree diagram of proposed YANG

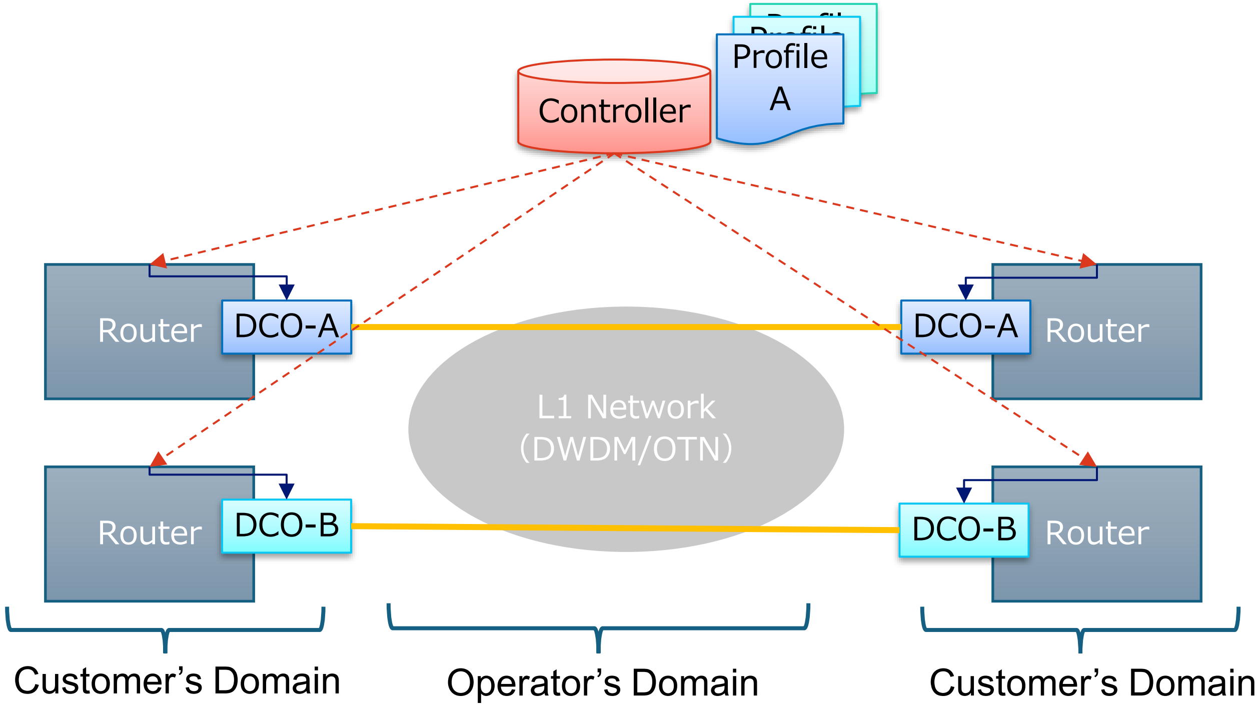
Recap: Use Cases

- **Use Case1: Centralized Control of Pluggable DCO Modules (L0/L1 Private Network Service)**

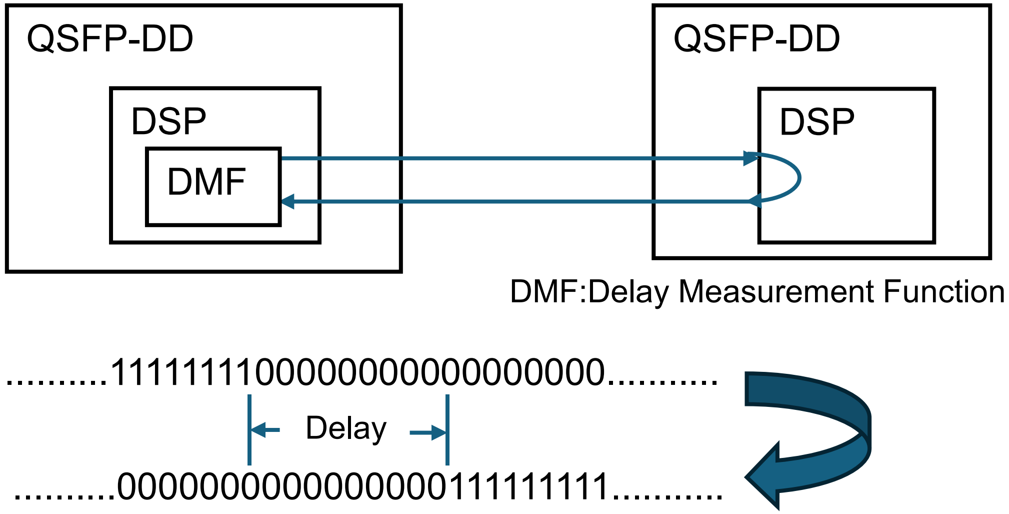
  An L0/L1 private network service provider can control only pluggable DCO transceivers equipped with customer's routers independently of routers' implementations.

- **Use Case2: Using Custom Pages**

  High accuracy metrics of the optical layer can be measured by using custom pages (e.g., measurement of optical characteristics, nano-sec ordered delay)



<Centralized Control of Pluggable DCO Modules>



<Using Custom Pages(High Accuracy Delay Measurement)>

Update Points

- Yuji Tochio (1Finity) joined as a co-author (welcome!)

- Enhanced Use Case Description
  - Supplemented with rationale for remote monitoring requirements.

- Expanded YANG module Coverage
  - Added modules for Action, RPC, and Monitor functionalities.

- Fixed Editorial Issues
  - Aligned terminology (e.g., “pluggable module” -> “pluggable DCO transceiver”).

Action Module

Define read action

```
augment "/if:interfaces/if:interface" {
  description
   "Add CMIS read/write actions under interface.";

  action cmis-read {
    description
     "Read CMIS register via action context.";
    input {
     leaf page {
       type uint8;
       mandatory true;
       description "The number of the CMIS page.";
     }
     leaf bank {
       type uint8;
       mandatory true;
       description "The banks corresponding to the CMIS page.";
     }
     leaf offset {
       type uint8;
       mandatory true;
       description "The memory address of the value.";
     }
     leaf size {
       type uint8{
         range "1 .. 128";
       }
       mandatory true;
       description "The memory size of the value.";
     }
    }
    output {
     leaf data {
       type binary;
       description "Raw register data.";
     }
    }
  }
}
```

Indicate target interface &register

Define output information

Define write action

```
action cmis-write {
   description
    "Write CMIS register data via action context.";
   input {
    leaf page {
      type uint8;
      mandatory true;
      description "The number of the CMIS page.";
    }
    leaf bank {
      type uint8;
      mandatory true;
      description "The banks corresponding to the CMIS page.";
    }
    leaf offset {
      type uint8;
      mandatory true;
      description "The memory address of the value.";
    }
    leaf data {
      type binary;
      mandatory true;
      description "Data to write.";
    }
   }
```

Indicate target register & written data

```
output {
    leaf status {
     type enumeration {
      enum success;
      enum not-permitted {
       description "Write request was rejected due to access-type or policy";
      }
      enum io-error {
       description "I/O error during write";
      }
      enum invalid-params {
       description "Bad parameters";
      }
     }
     description "Result of the write operation.";
    }
    leaf post-write-value {
     type binary;
     description
      "Optional read-back of the target value after write.
       Present only if the implementation performed a read-back
       (e.g., for 'rw' registers). Not present for 'wo' registers
       or when no-readback was requested/possible.";
    }
   }
  }
 }
}
```

Define output information

Define result status

Definition response of written data

RPC Module

Define read action

Indicate target interface &register

Define output information

```
rpc cmis-read {
  description
    "Read a CMIS register from a pluggable DCO transceiver.";
  input {
    leaf interface-name {
      type leafref{
                      path "/if:interfaces/if:interface/if:name";
                    }
      mandatory true;
      description "Target interface name.";
    }
    leaf page {
      type uint8;
      mandatory true;
    }
    leaf bank {
      type uint8;
      mandatory true;
    }
    leaf offset {
      type uint8;
      mandatory true;
    }
    leaf size {
      type uint8;
      default 1;
      description "Number of bytes to read.";
    }
  }
  output {
    leaf data {
      type binary;
      description "Raw register data.";
    }
  }
}
```

Define write action

Indicate target register & written data

```
rpc cmis-write {
  description
    "Write CMIS register data to a pluggable module.";
  input {
    leaf interface-name {
      type leafref{
        path "/if:interfaces/if:interface/if:name";
      }
      mandatory true;
      description "Target interface name.";
    }
    leaf page {
      type uint8;
      mandatory true;
      description "The number of the CMIS page.";
    }
    leaf bank {
      type uint8;
      mandatory true;
      description "The banks corresponding to the CMIS page.";
    }
    leaf offset {
      type uint8;
      mandatory true;
      description "The memory address of the value.";
    }
    leaf data {
      type binary;
      mandatory true;
      description "Data to write.";
    }
  }
```

Define output information

Define result status

Definition response of written data

```
output {
  leaf status {
    type enumeration {
      enum success;
      enum not-permitted {
        description "Write request was rejected due to access-type or policy";
      }
      enum io-error {
        description "I/O error during write";
      }
      enum invalid-params {
        description "Bad parameters";
      }
    }
    description "Result of the write operation.";
  }

  leaf post-write-value {
    type binary;
    description
     "Optional read-back of the target value after write.
      Present only if the implementation performed a read-back
      (e.g., for 'rw' registers). Not present for 'wo' registers
      or when no-readback was requested/possible.";
  }
 }
}
}
```

Monitor Module

Define target register map

```
grouping monitor-target {
  description
    "Target CMIS register to monitor.";
  leaf page {
    type uint8;
    mandatory true;
    description "The number of the CMIS page.";
  }
  leaf bank {
    type uint8;
    mandatory true;
    description "The bank of the CMIS page.";
  }
  leaf offset {
    type uint8;
    mandatory true;
    description "The memory address of the value.";
  }
  leaf size {
    type uint8 {
      range "1 .. 128";
    }
    description "The memory size of the monitored value.";
  }
}
```

Define monitor rules

Set rule name

Set target interface

Set notification rule

Set monitoring interval

Configure enable/disable

```
container monitors {
  description "Container for all monitor rules.";
  list monitor-rule {
    key "id";
    description "Monitoring rule.";
    leaf id {
      type string;
      description "Unique identifier of the rule.";
    }
    leaf interface-name {
      type leafref {
        path "/if:interfaces/if:interface/if:name";
      }
      mandatory true;
      description "Target interface of the monitored CMIS module.";
    }
    container monitor-target {
      uses monitor-target;
    }
    container condition {
      description "Condition to trigger notification.";
      leaf condition-type {
        type enumeration {
          enum threshold;
          enum delta-rate;
        }
        mandatory true;
        description "Type of condition.";
      }
      leaf threshold {
        type decimal64 {
          fraction-digits 2;
        }
        description "Threshold value for triggering notification (only used for threshold type).";
      }
      leaf delta-rate {
        type decimal64 {
          fraction-digits 2;
        }
        description "Maximum allowed change per interval (only used for delta-rate type).";
      }
    }
    leaf interval-ms {
      type uint32;
      default 1000;
      description "Monitoring interval in milliseconds.";
    }
    leaf enabled {
      type boolean;
      default true;
      description "Enable or disable this monitor rule.";
    }
  }
}
```

Define notification event

Define notified information

Define notification timestamp

```
notification cmis-monitor-event {
  description "Notification raised when monitor rule condition is met.";
  leaf interface-name {
    type string;
    description "Interface name of the monitored module.";
  }
  leaf rule-id {
    type string;
    description "ID of the rule that triggered this notification.";
  }
  container monitor-target {
    uses monitor-target;
  }
  leaf condition-type {
    type enumeration {
      enum threshold;
      enum delta-rate;
    }
    description "The type of condition that was met.";
  }

  leaf current-value {
    type binary;
    description "Current value of the monitored register.";
  }

  leaf threshold {
    type decimal64 {
      fraction-digits 2;
    }
    description "Threshold value (present if threshold type).";
  }

  leaf delta-rate {
    type decimal64 {
      fraction-digits 2;
    }
    description "Delta-rate value (present if delta-rate type).";
  }
  leaf timestamp {
    type yang:date-and-time;
    description "Time when the notification was generated.";
  }
 }
}
```

Discussion

- Should this YANG be allowed to handle standard pages? (Raised from Roberto)
  - This proposal is motivated from the following aspects:
    1. Lack of sufficient DCO abstraction in current implementations
    2. Challenges in supporting new pluggable DCO transceivers (especially for using whitebox and OSS)
    3. Potential for unified control between standard and custom pages
  - Note that CMIS-based control of DCO transceivers is performed by NOS
-> This YANG does not violate the definition of CMIS as a protocol for host-module interaction
  - Limiting the scope (e.g., vender-specific configurations) helps avoid cases where YANG-based management cannot meet timing requirements (e.g., counter collection)

- Are there better solution for handling CMIS custom page?

Next Actions

- Need feedback on whether this draft should focus on the control of the CMIS custom page.
- Clarify the problem statements by investigating gaps between use cases and CMIS specification
- Proceed in coordination with the related documents.
  - draft-rokui-ccamp-actn-wdm-pluggable-modelling
  - draft-ietf-ccamp-actn-poi-pluggable-usecases-gaps

Thank you