

NETCONFを転用して、 設定ファイルエディターをつくる話

株式会社コーダンス

Shintaro Kojima / @codeout

事前公開版です。最新版はこちら→

<https://speakerdeck.com/codeout/config-editor-implementation-with-netconf-or-yang>

コンフィグ、手で書いていますか？

「エディターがあったら…」 って思ったことありませんか？

実機にログインしなくても

- **補完** が効く
- **文法チェック** できる
- **“Go to Definition”** できる

それ、NETCONF / YANG で できます！！

補完・ 文法チェック

実機ログインせずに
やりたい

```
10
11
12 | set interfaces xe-0/0/0 descripton "white space"
13 | set interfaces xe-0/0/0 unit 0 family inet address 10.0.0.1/30
14 | set interfaces xe-0/0/1 unit 0 family inet filter input foo-filter
15 | set interfaces xe-0/0/1 unit 0 family inet filter input foo-filter
16 | set interfaces xe-0/0/1 unit 0 family inet
17 |
18 | set protocols bgp group bgp-group import fo
19 | set protocols bgp group bgp-group import fo
20 | set protocols mpls interface xe-0/0/0.0
21 | set protocols mpls interface xe-0/0/0.1
22 | set protocols mpls interface xe-0/0/1
23 | set protocols
24 |
25 | set policy-options prefix-list foo-prefix 1
26 | set policy-options community foo-community
27 | set policy-options as-path foo-as-path "650
28 | set policy-options as-path-group foo-as-pat
29 | set policy-options policy-statement foo-statement from prefix-list foo-prefix
30 | set policy-options policy-statement foo-statement from prefix-list foo-prefix
31 | set policy-options policy-statement foo-statement from community foo-community
32 | set policy-options policy-statement foo-statement from community foo-community
33 | set policy-options policy-statement foo-statement from as-path foo-as-path
34 | set policy-options policy-statement foo-statement from as-path foo-as-path
35 | set policy-options policy-statement foo-statement from as-path-group foo-as-path-group
36 | set policy-options policy-statement foo-statement from as-path-group foo-as-path-group
37 | set firewall filter foo-filter term foo then accept
38
```

abc	accounting
abc	address
abc	allow-filter-on-re
abc	apply-groups
abc	arp-max-cache
abc	arp-new-hold-limit
abc	demux-destination
abc	demux-source
abc	destination-class-usage
abc	dhcp
abc	filter
abc	ingress-queuing-filter

Configure in
s

Go to Definition

ショートカットキー
一発で、定義を見たい

```
5 set protocols bgp group bgp-group import
6 set policy-options policy-statement foo-statement from community
7 set policy-options policy-statement foo-statement from as-path
8 set policy-options policy-statement foo-statement from as-path-group

junos.conf ~/darwin/wip/vscode/vscode-junos/client/testFixture - 12 definitions
23 set protocols
24
25 set policy-options prefix-list foo-prefix 10.0.0.0/8
26 set policy-options community foo-community members 65000:100
27 set policy-options as-path foo-as-path "65000+"
28 set policy-options as-path-group foo-as-path-group as-path foo-as-path "65000+"
29 set policy-options policy-statement foo-statement from prefix-list foo-prefix
30 set policy-options policy-statement foo-statement from prefix-list foo-prefix
31 set policy-options policy-statement foo-statement from community foo-community
32 set policy-options policy-statement foo-statement from community foo-community
33 set policy-options policy-statement foo-statement from as-path foo-as-path
34 set policy-options policy-statement foo-statement from as-path foo-as-path
35 set policy-options policy-statement foo-statement from as-path-group foo-as-path-group
36 set policy-options policy-statement foo-statement from as-path-group foo-as-path-group
37 set firewall filter foo-filter term foo then accept
38

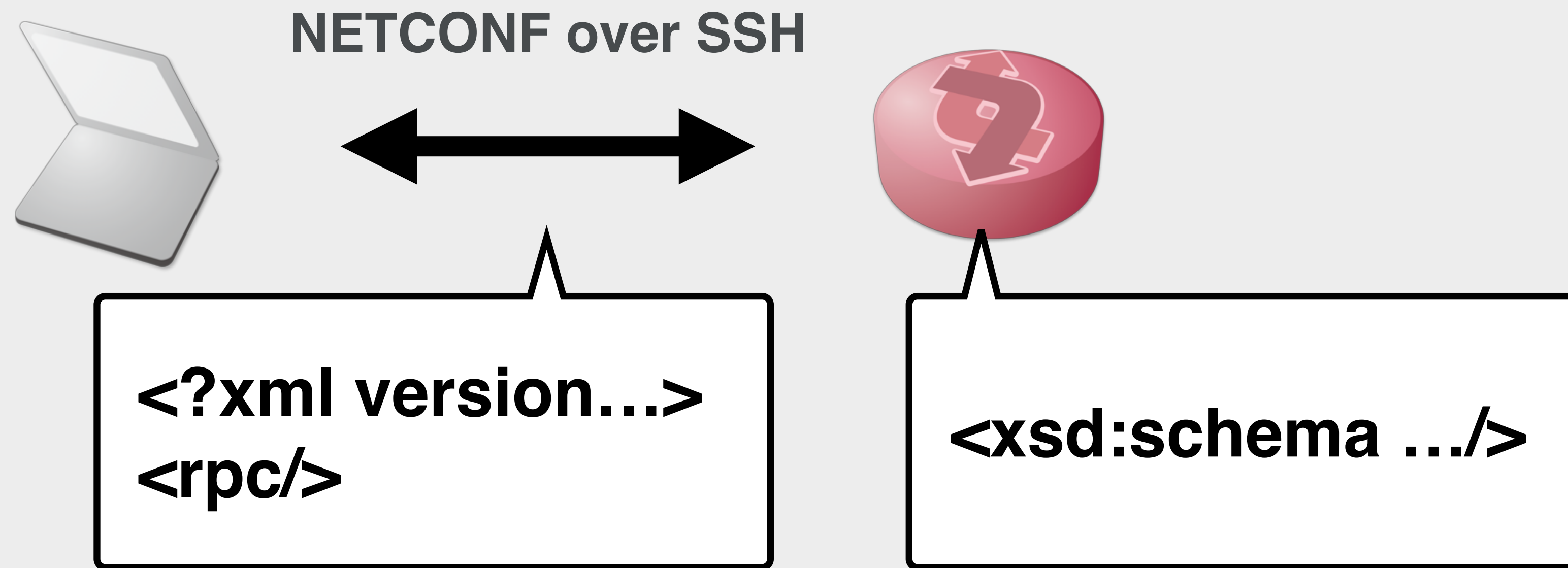
9 set interfaces xe-0/0/1 unit 0 family inet filter input
10
11
12 set interfaces xe-0/0/0 description "white space"
13 set interfaces xe-0/0/0 unit 0 family inet address 10.0.0.1/30
14 set interfaces xe-0/0/1 unit 0 family inet filter input foo-filter
15 set interfaces xe-0/0/1 unit 0 family inet filter input foo-filter
16 set interfaces xe-0/0/1 unit 0 family inet
17
18 set protocols bgp group bgp-group import foo-statement
19 set protocols bgp group bgp-group import foo-statement
20 set protocols mpls interface xe-0/0/0.0
21 set protocols mpls interface xe-0/0/0.1
22 set protocols mpls interface xe-0/0/1
23 set protocols
```

なにそれ、便利そう！

でも… 文法はどこから？ 🤔

→ NETCONF / YANG から抽出します

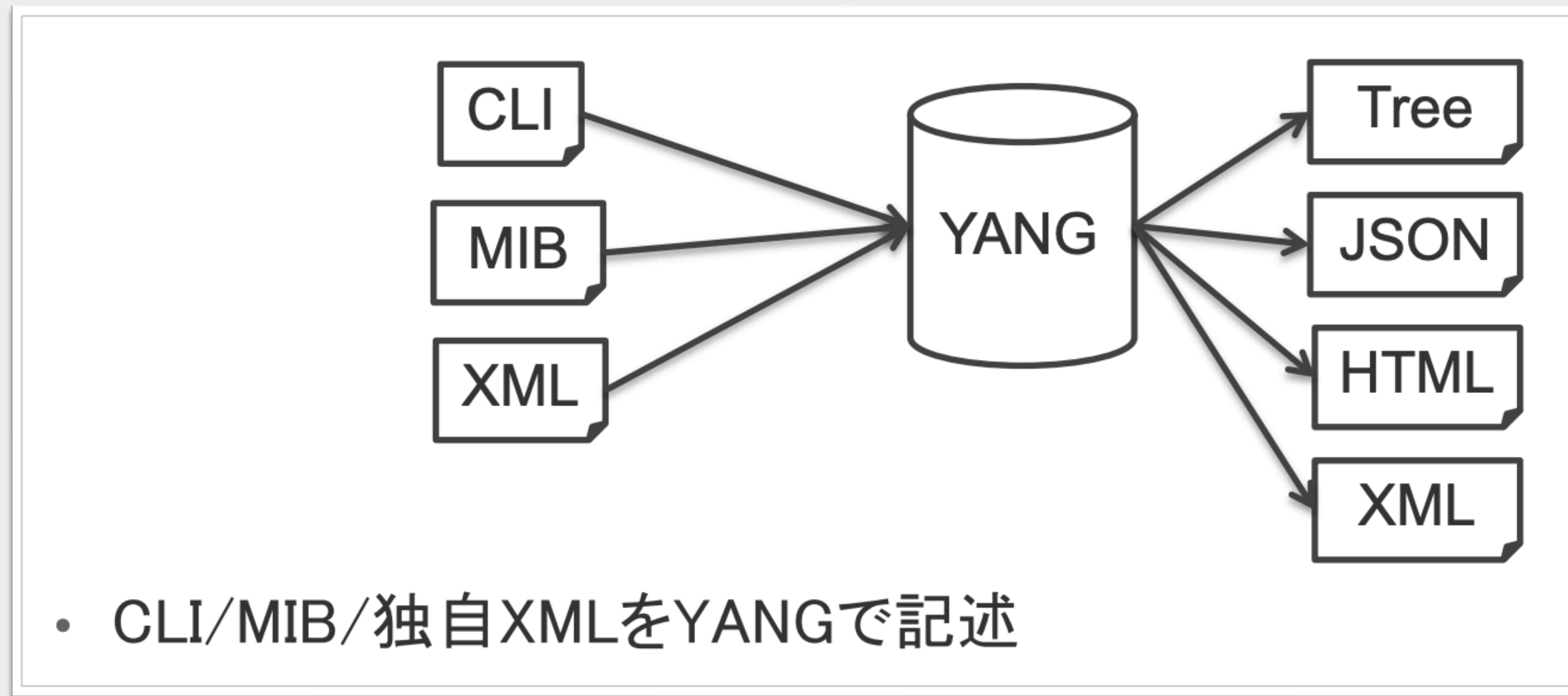
NETCONF / YANG



NETCONF自体は XML-RPC。

コマンド= XML は、ベンダー独自のXMLスキーマ(XSD) で文法チェックされる。

NETCONF / YANG

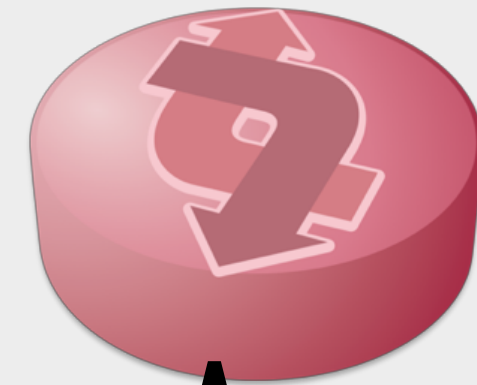


<https://www.janog.gr.jp/meeting/janog36/application/files/7714/3662/8399/janog36-netconf-shishio-03.pdf>

**YANG はNETCONFで使われるデータモデルであり、
「XMLスキーマを共通言語で記述したもの」と言える。**

XSD(NETCONF) / YANG vs. CLI

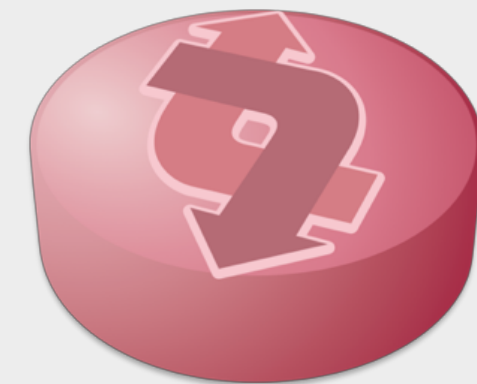
NETCONF



`<?xml version...>`
`<rpc/>`

`<xsd:schema .../>`
or yang

CLI



text

XSD(NETCONF) / YANG vs. CLI

NETCONF



`<?xml version...>`
`<rpc/>`

`<xsd:schema .../>`
or yang

この構造が似ているため

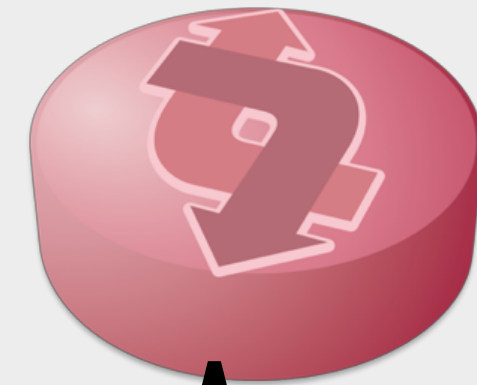
CLI



text

NETCONF / YANG vs. CLI

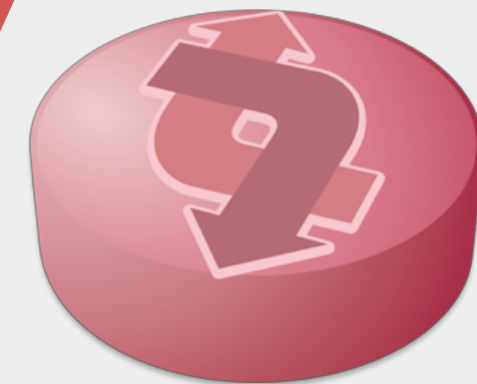
NETCONF



`<?xml version...>`
`<rpc/>`

`<xsd:schema .../>`
or yang

CLI



text

文法の注入が可能

XSD (NETCONF)

```
<xsd:complexType name="interfaces-type">
  <xsd:sequence>
    <xsd:element name="name">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:restriction base="key-attribute-string-type">
            <xsd:enumeration value="$junos-interface-ifd-name">
            </xsd:enumeration>
            <xsd:enumeration value="interface-name">
            </xsd:enumeration>
          </xsd:restriction>
        </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
    <!-- </name> -->
    <xsd:element name="unit" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="name">
            <xsd:complexType>
              <xsd:simpleContent>
                <xsd:restriction base="xsd:anyType">
                  <xsd:simpleType>
                    <xsd:union memberTypes="xsd:unsignedLong">
                      <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                          <xsd:enumeration value="$junos-underlying-
interface-unit">
```

CLI

```
set interfaces
<name> unit <number>
```

(JUNOS の例)

YANG

```
grouping interfaces_type {
  description "Physical interface";
  leaf name {
    type string;
  }
  list unit {
    key name;
    description "Logical interface";
    leaf name {
      type string;
    }
  }
  list unit {
    key name;
    description "Logical interface";
    leaf name {
      type string;
    }
  }
}
```

CLI

```
set interfaces  
<name> unit <number>
```

(JUNOS の例)

エディターの作りかた (概要)

1. XSD の取得
 - `<get-schema/>` コマンド (NETCONF)
2. もしくは、YANG モデル取得
 - ベンダーが公開している
3. 1 or 2 をパースして抽象構文木をつくる
 - 任意のCLI 階層で、一段下のノード(枝)リストが取れる
 - = 文法チェックができる
4. エディターを実装する 💪

Junos の参考実装 (vscode 拡張)



codeout/vscode-junos

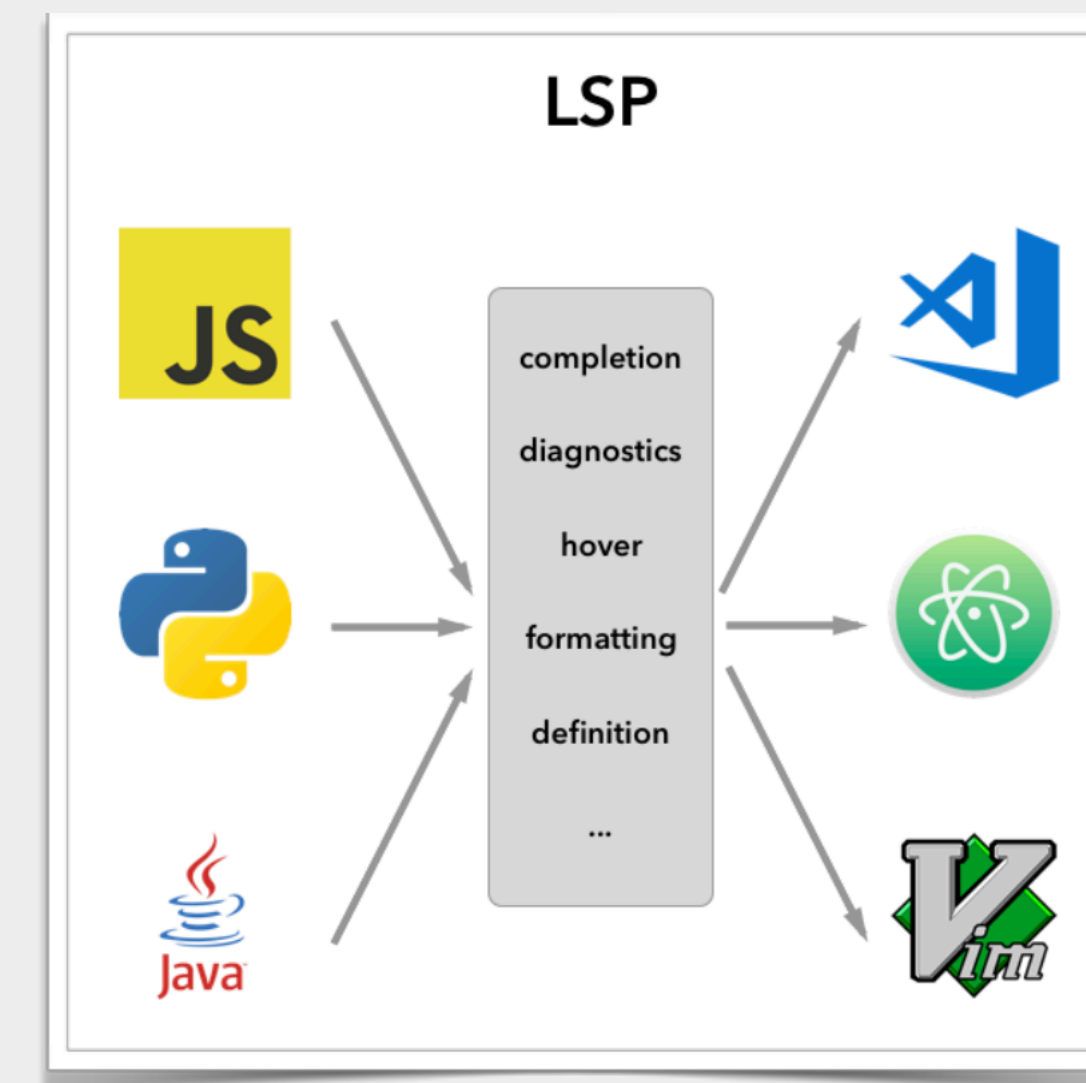
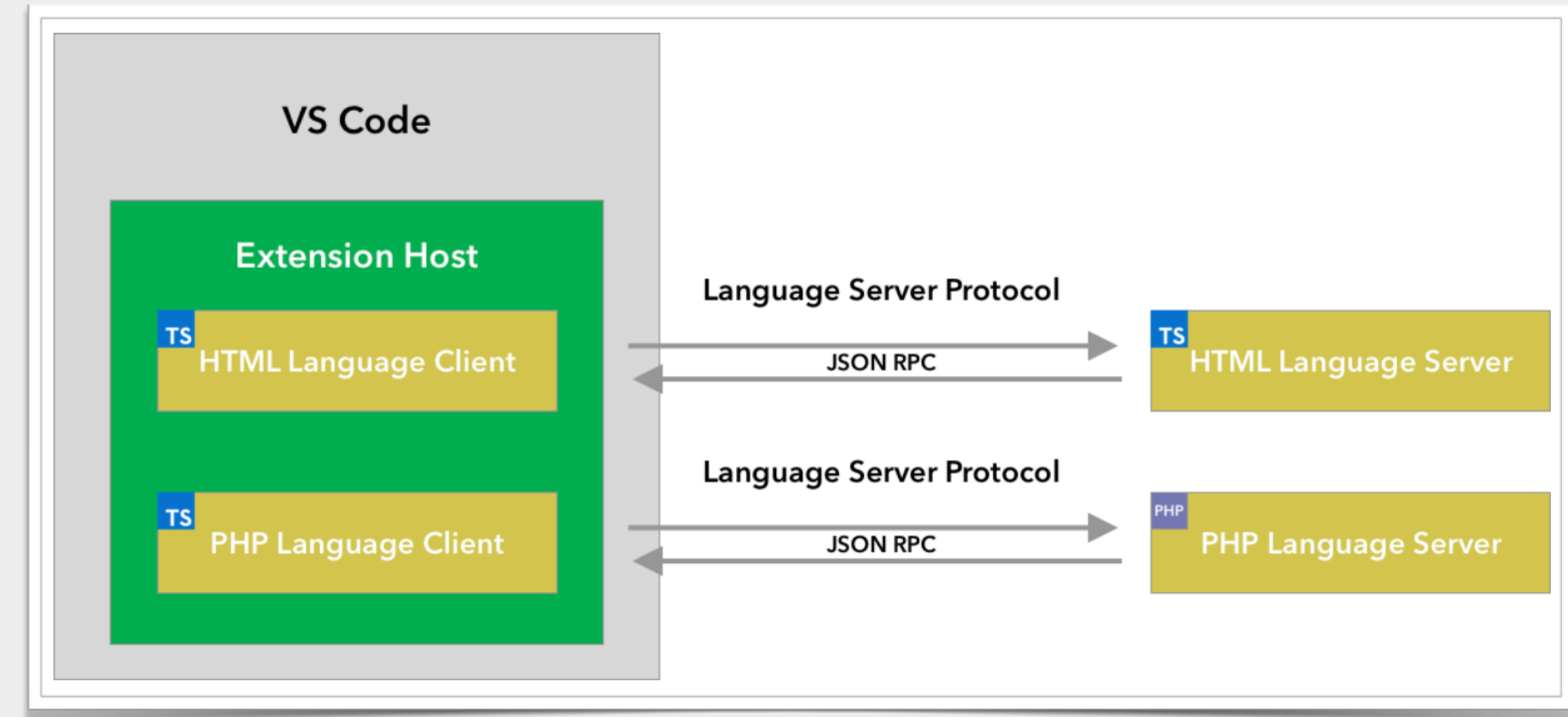
- <https://github.com/codeout/vscode-junos>
- <https://marketplace.visualstudio.com/items?itemName=codeout.vscode-junos>
- “junos” で検索 @ Visual Studio Code

エディターの作りかた (ヒント)

- XSD よりは、YANG のほうがとっつきやすい (オススメ)
 - ただし、メタ情報が落ちているため精度に限界がある
 - 「ここからここまで、一行で書きますよ」とか
- NETCONF をサポートしない機器もある
- 抽象構文木の時点で、オレオレ言語で記述しておくといよい
 - NETCONF とCLI で微妙に違う場合があって、オレオレ記述に文法追加できるように。隠しコマンドなど
 - (わたしはここだけ ruby でやりました)
- ライセンスは要確認

脱線: Language Server Protocol (LSP)

- vscode-junos はLSP で実装
- vscode上のクライアント + vscode以外とも話せる言語サーバー
- Vim やEmacs など、他エディターも対応可能 (たぶん)
- Vim 力が低すぎてなかなか進まない...



まとめ

- NETCONF / YANG を転用すると、ユーザー側で
コンフィグエディターを作れる
- コンフィグ作成コストがかなり下がった 🎉
- みなさまがお使いの機器でも、できる気がしませんか？
 - 「面白そう」「やってみよう」と思われる方がいらっ
しゃれば嬉しいです (動いたら是非公開を…！)