ネットワーク自動化の対応チャレンジ

JANOG 50 Day3
2022年7月15日(金)
10:45〜11:30

セイコーソリューションズ株式会社
中山 真一

自己紹介

名前　　：中山　真一

所属　　：セイコーソリューションズ株式会社

仕事内容　　：担当製品のソフトウェア開発業務全般

（入社以来16年間担当）

興味ある技術

- ネットワークやサーバの運用に関する技術
- 運用自動化、ネットワーク自動化

※JANOG参加：40～50

今回初発表です！

フクロモモンガ飼ってます

元気過ぎる子供二人に毎日振り回されてる
パパ エンジニアです
・息子 6歳：慎重派　かわいい
・娘　 3歳：わがまま全盛期　かわいい

今回の発表について①

- 2017年から約5年間、担当製品を「ネットワーク自動化」に
  対応させる為に様々なアプローチをしてきました。
  その経験談と道のりを共有したいと考えています。

- 「ネットワーク自動化」が今後広がる為には、
  対応する機器が増えていく事が必要と考えていますが、
  対応方法についての情報は非常に少なく　苦労しました。

- 本発表が直接刺さる方は少ないかもしれません、が
  対応する側の知見共有と、ネットワーク自動化が広がっていく為の
  一助になればと考えています。

今回の発表について ②

- 担当製品について
  - コンソールサーバー SmartCS の開発をしています

    - 役割
      ・コンソールポートを集約して、遠隔からコンソールアクセスを行う為の装置。
      ・ルーター スイッチ といった主信号を扱うネットワーク機器ではありません。
    - 使われる環境
      ・ネットワーク運用の現場。
      ・常にマルチベンダー環境。

補足

・本発表では、「担当製品の機能」については取り扱いません。
・本資料に記載の各API対応は、「ネットワーク機器本体の情報取得/設定」となります。
※SmartCSに接続されている機器について ではありません

今回の発表について ③

- ネットワーク運用の現場で扱われる製品の開発担当として

  - JANOGを中心とした多くのコミュニティや勉強会で、
    ネットワーク運用をしている方々の
    「ネットワーク自動化を広げていきたい」
    という気持ちから刺激を受けて製品開発を進めてきました。

  - その応えとして本発表で少しでも知見をコミュニティに還元できればと考えています。

Agenda

- 発表内容
  - 対応背景
  - Ansible
  - RESTAPI
  - 振り返り
  - 今後について
- 議論
- 参考

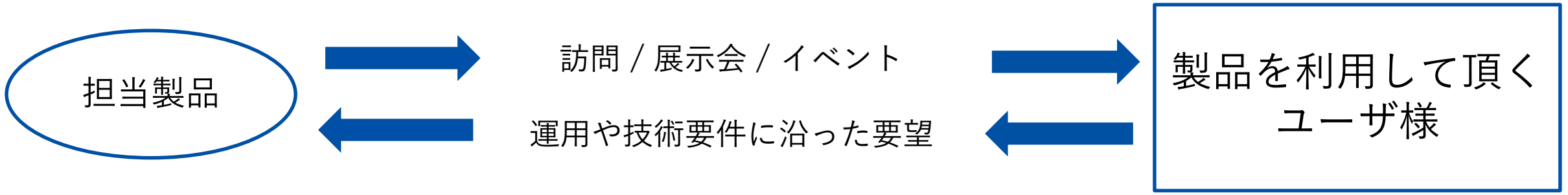
文言定義

API	広義の Application Program Interface の意味として記載しています。
RESTAPI	広義（多くの人がイメージしやすい）の意味として 記載しています。 ・http通信を使ってxmlやjsonフォーマットのデータをやり取り ・URI + Method でリソースにアクセスするという概念

対応背景

対応までのアプローチ

- 従来の機能開発



- 担当製品をご利用頂いているユーザ様からの要望が開発のきっかけ
  - 通信プロトコル(IPv6, IPsec) 、運用連携(Radius, Tacacs+)。
  - 製品に特化した機能の追加や改善など。

- 要望を頂く → 調査 → 仕様検討 → 実装 → 評価・相互接続 → リリース

対応背景

対応までのアプローチ

- 検討初期：担当製品に機能追加　という目線

担当製品 ⇄ 自分のバックボーン（担当製品）目線で見る ⇄ イベント/セミナー 展示会 勉強会

技術や刺激を機能やサービスに

- 担当製品に、外から得た知見を機能として入れ込めないか　という考え。
  - 「運用自動化」「NAPALM」とは 2017年位に出会い、要望も頂きました。

当時の検討

マルチベンダー環境
異なるCLI体系
運用が大変

「show version」

RESTAPI

CLIを抽象化できたら
嬉しいのかもしれない

telnet/ssh

ベンダーA用
「show version」

ベンダーB用
「show version」

console

対応背景

対応までのアプローチ

- 検討後期：担当製品がネットワーク運用の中でどう役立つか　という目線

ネットワーク運用分野の技術トレンド

⇒ 製品の分野を考え、トレンドを知る ⇒ 担当製品

⇐ そこで活かせる機能を考える ⇐

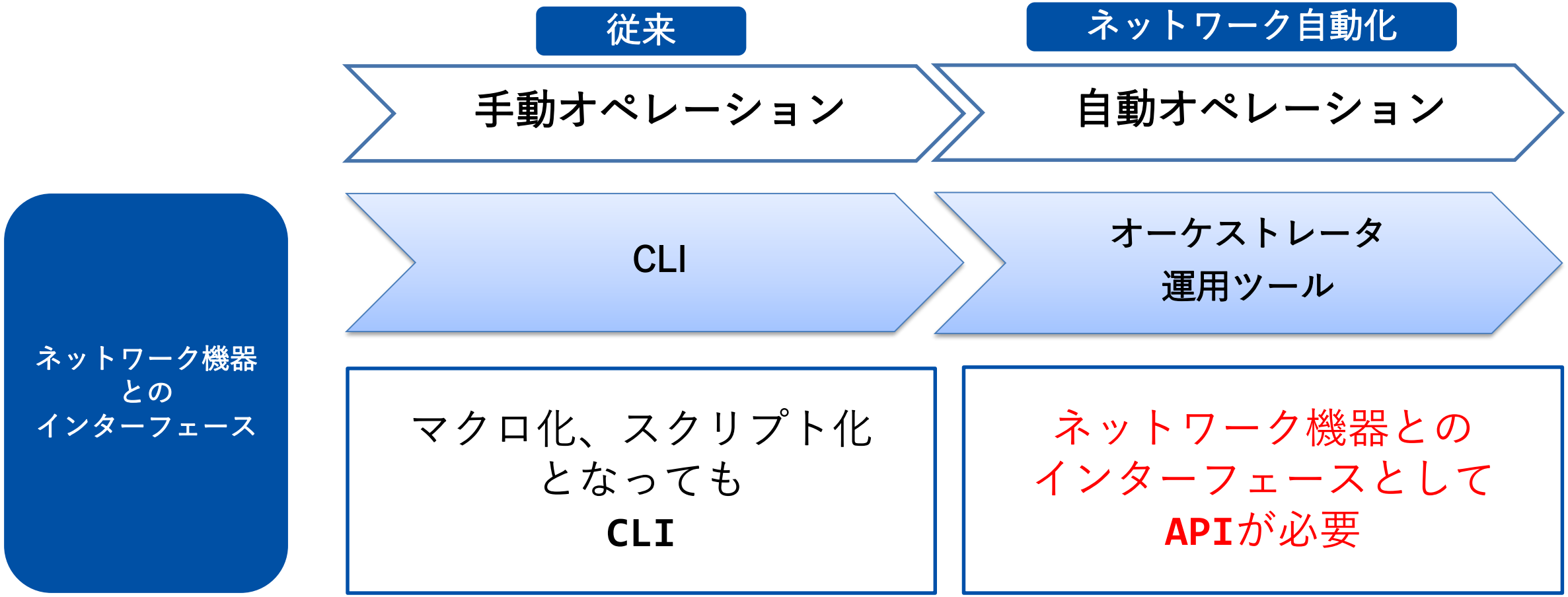
- ネットワーク自動化　という技術トレンドにどう役立てるか

自分の担当製品は、ネットワーク運用の中では
１つの機能/部品/パーツだと考える
↓
使いやすい部品を提供する

対応背景

運用ツールとの連携

- ネットワーク自動化 という環境変化の中でどう使われるのか



対応背景

SEIKO

運用ツールとの連携

- まずはどの運用ツールと連携するか（＝どのAPIに対応するか）
  ※2017-18当時
  ※多くの勉強会参加、ユーザ様ヒアリング を経て絞った実際の選択肢

運用ツール	API	開発内容	個人的な所感
ANSIBLE	Connection Plugin ・CLI ・RESTAPI(WEBAPI) ・NETCONF	Ansible モジュール開発	・運用で使い始めているユーザは少なかった。 ・多くネットワーク機器ベンダーがサポートを開始し始めていた時期。 ・Agentlessという事でサーバ側実装がない。
内製ツール	RESTAPI	RESTAPI サーバ側実装	・内製ツールで運用しているユーザは多い。 ・APIとして、RESTAPIは呼び出しやすい（連携しやすい）という事であった。
内製ツール 商用管理ツール NSO Network Services Orchestrator (cisco社)	NETCONF (CLI)	NETCONF サーバ側実装	・製品の特性上使っているユーザは多くはなかったが、非常に大規模であった。 ・NETCONFのサーバ側実装の難易度が高い。

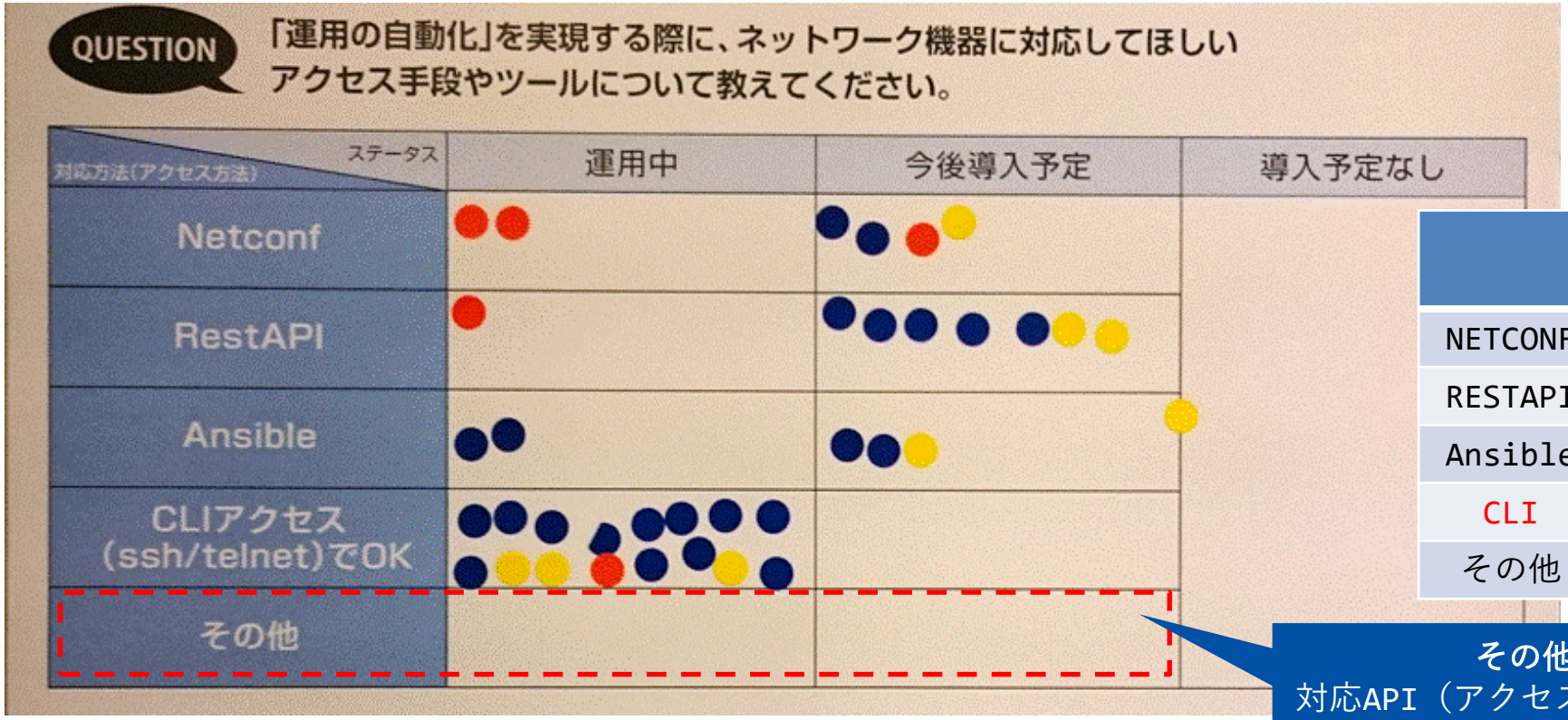
対応背景

SEIKO

運用ツールとの連携

- まずはどの運用ツールと連携するか（＝どのAPIに対応するか）

参考：JANOG41 in 広島（2018.01） 弊社ブースアンケート



	実運用中	今後導入予定
NETCONF	2	4
RESTAPI	1	7
Ansible	2	4
CLI	16	0
その他	0	0

その他が 0
対応API（アクセス方法）として、
当時これで問題なさそうと判断

Ansible対応

SEIKO

最初に選んだ理由

- 運用自動化の直球プロダクト
  - ネットワークも含んだ「運用自動化」を実現するツール
  - デファクトになりそうと感じた
    - ・2018.7月　JANOG42 Ansibleネットワーク自動化チュートリアル
    - ・2018.11月　InternetWeek2018～（Ansibleによるネットワーク運用自動化ハンズオン）

- モック開発が早く行えた
  - 展示会等で実際に動くものを見せる事は重要

- 開発モチベーション
  - 従来のネットワーク通信やプロトコル対応とは異なる開発をしてみたい
  - OSSのツールに対応するというチャレンジ　(社内ボトムアップ)

Ansible対応

SEIKO

開発アプローチ

- 担当製品とどう繋ぐか
  - Ansibleの Network Connection Plugin を決める。

Value of ansible_connection	Protocol	自社製品の対応
ansible.netcommon.network_cli	CLI over SSH	○
ansible.netcommon.netconf	XML over SSH	×
ansible.netcommon.httpapi	API over HTTP/HTTPS	×

  - 今回はnetwork_cli の開発内容となります。

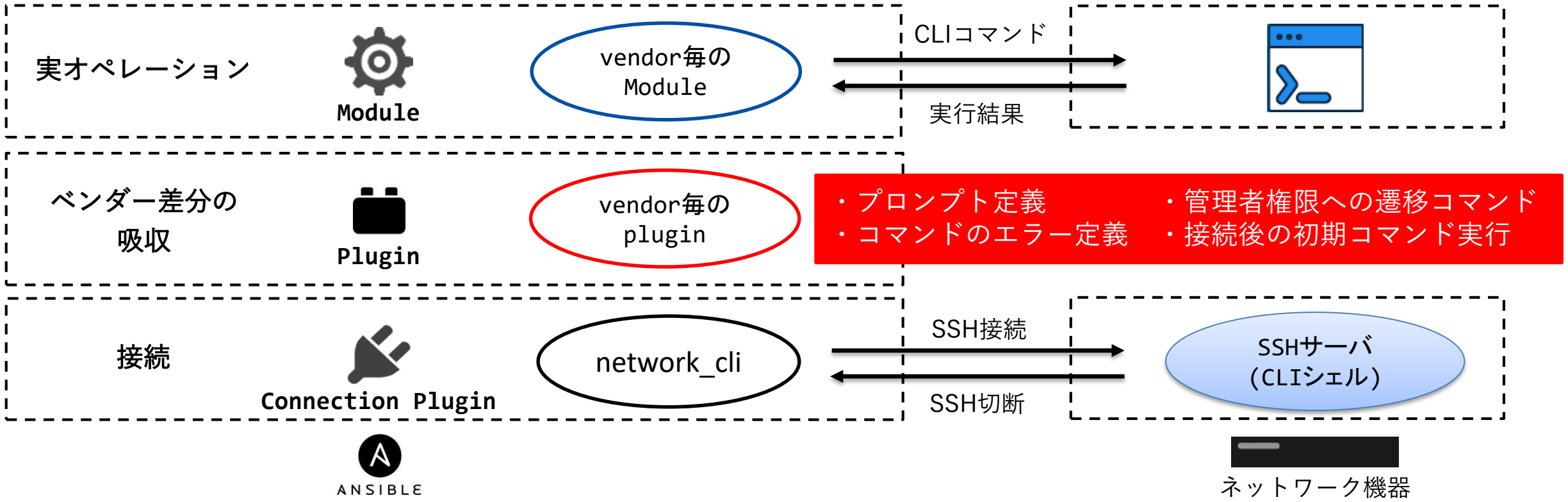
https://docs.ansible.com/ansible/latest/network/getting_started/network_differences.html#multiple-communication-protocols

Ansible対応

SEIKO

開発の苦労話

- network_cli は Ansibleモジュールで CLIオペレーションを実行する
  - モジュールのプラグイン部分でベンダー毎のCLI差分を吸収する



Ansible対応

SEIKO

開発の苦労話

- CLI との闘い
  - プラグイン開発

<プロンプト定義>

```
(0)NS-2250>
```

```
(^|¥r|¥n)¥([0-9]{1,3}¥)(¥[[0-9:]{8}¥])?[a-zA-Z0-9][a-zA-Z0-9-_.]{0,63}(?:[>#])[ ]$
```

<コマンドのエラー定義>：これは大変でした

エラーを全て洗い出す	~~どこかに綺麗に全て定義されていると思っていた時がありました~~ 洗い出した内容を正規表現で定義！
どこまで定義するか	文法エラー、権限エラー、コマンド実行後のエラー　を正規表現で定義 ※linuxコマンドをほぼそのまま使っているものはエラーの定義しない （他ベンダーの実装と比べて同じようにしました）

Ansible対応

SEIKO

開発の苦労話

- CLI との闘い
  - モジュール開発

表示系モジュール (xxx_command)	・プラグイン定義があれば比較的容易 ・全CLIきちんと動くか（エラー誤検出しないか）の確認が必要
設定系モジュール (xxx_config)	・冪等性の担保をするのはモジュール内 →指定されたコマンドが設定済みか、そうでないか →それが現在のコンフィグ（running-config）から判断できるか
その他	・特定の条件によって出力内容が変わるものも理解しておく必要がある。 （HWのモジュール装着時等）

モジュール開発は製品仕様を一番理解している機器ベンダーでないとやはり難しい。

Ansible対応の為に、既存のCLI仕様を変える事だけは（したくなりましたが）しませんでした。

Ansible対応

SEIKO

開発振り返り

- どんなモジュールを用意すればいいか
  - おそらく基本となる3点セット

Module	用途
vendor.product.product_**facts**	設定情報やコンフィグの収集
vendor.product.product_**command**	CLIコマンド(表示系)の実行
vendor.product.product_**config**	CLIコマンド(設定系)の実行

  - 自社製品はどう使われているか　という想像

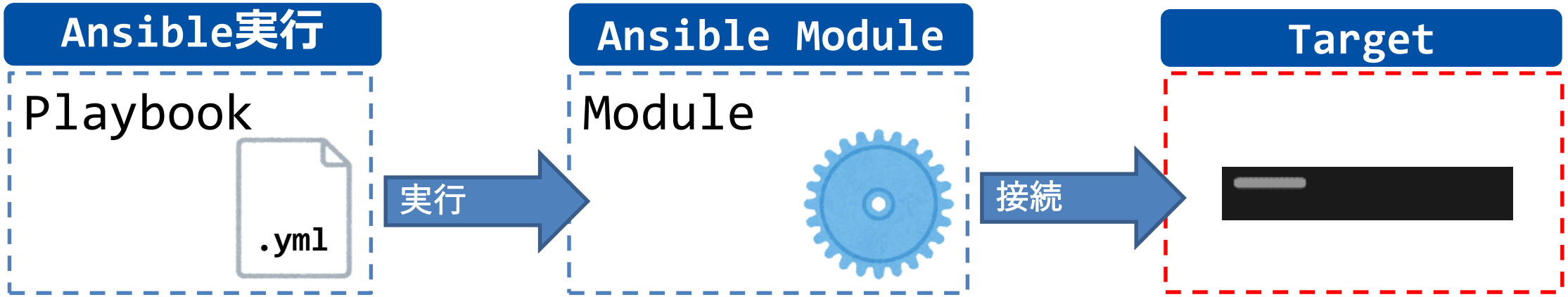
使用例	必要と思われるモジュール
複数台に導入され、微妙に異なる設定を投入する必要がある。	xxx_config, xxx_command
運用時に特定の（複雑な）設定や特徴的なオペレーションが多い。	xyz_operation
その環境で１台のみで使われることが多く　設定変更がほぼない。	xxx_facts

Ansible対応

SEIKO

開発振り返り

- 全ての機能は製品側に



- Moduleでしか実現できない機能は用意しない。
- 提供する機能は全て製品側に用意して、Module側から呼び出す。

見られてもいい情報（公開情報）のみを使って機能を実現
ModuleはOSSとして公開される事を意識

Ansible対応

SEIKO

OSS対応の苦労

- 会社としての考え方
  - 弊社としてはOSS公開が初のケースだったので、大変な事は多かったです
    部門内外含めて後押しする声が多くて非常に助かりました。
    - 特に知財部門とは公開前にしっかりと相談しました。
  - 基本的な考え方は『公開情報』（会社のホームページと同じ）
  - あくまで、対応製品 の価値を上げる（付加価値）為のツールとして扱う。
    ※単独で販売するソフトウェアではない

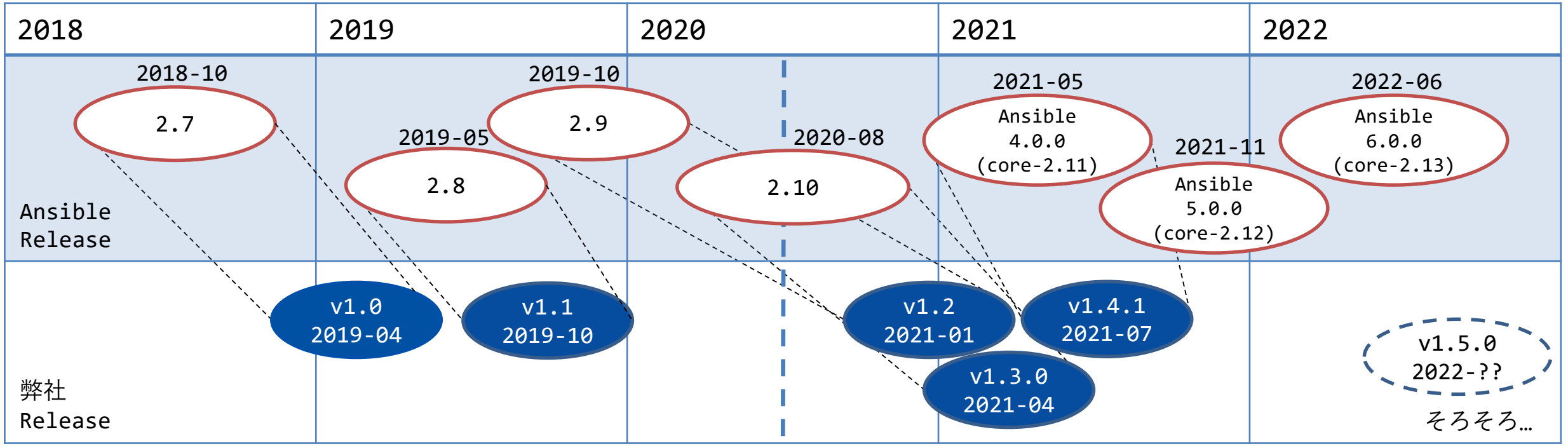
販促ツール

Ansible対応

SEIKO

OSS対応の苦労

- Ansibleのバージョンアップに追従していく



着手

初期リリース

メンテナンス（機能追加・不具合対応・AnsibleのVerup対応）

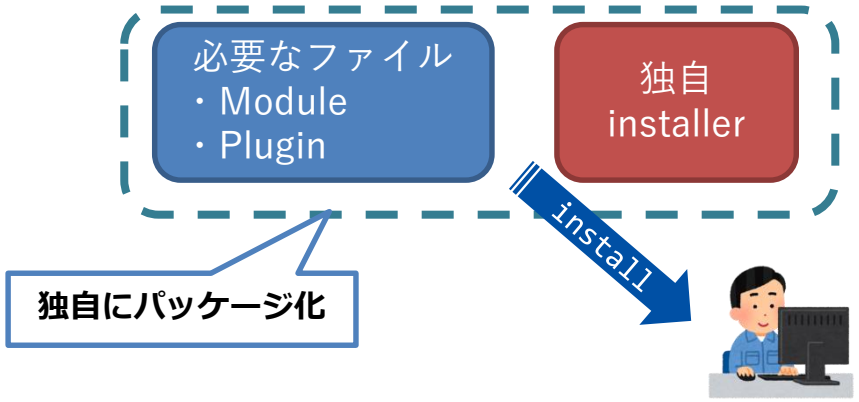
Ansible対応

OSS対応の苦労

- Ansibleのバージョンアップに追従していく
  - 提供方法は一度大きく変わりました。

2018	2019	2020	2021	2022

独自の提供方法から → Ansibleの仕組みに対応していく



OSSのAnsibleパッケージに含まれていない為、
ユーザのAnsible環境にインストールできるよう準備
弊社からHP等で直接お客様に提供

Ansible対応

SEIKO

OSS対応の苦労

- バージョンアップ方針
  - 可能な限り、メジャーバージョンアップや変化に追従していく
    （＝Ansibleのエコシステムに沿う事がユーザにとって使いやすい）

- 開発面
  - パッケージ作成の自動化、評価の自動化
  - 社内に配布サイト（Ansible Galaxy）を検証の為にローカルに構築
  - 専用のリリースフローの用意（QA部門とも調整）

- 課題
  - ソース管理の共通化

Ansible対応

SEIKO

Ansible対応の種類

- Ansibleモジュールやプラグインの開発
  - Ansible Collections の仕組みに沿って開発。
    https://docs.ansible.com/ansible/latest/dev_guide/developing_collections.html
- 提供方法
  - Ansible Galaxy / GitHub で公開
  - モジュール開発ベンダーから直接提供
    - Ansible Collections 形式であれば、ローカルインストールも可能。
  - Red Hat社の認定パートナー制度
    - Red Hat社と共同で製品のモジュールをサポートをする（専用サイトからDL）。

Ansible対応

SEIKO

よかったこと

- 新しい文化が社内に！
  - 開発部門以外でも、Ansibleを利用して業務の自動化が広がる。
  - GitHub利用
    ・会社のHP以外にも製品コンテンツを公開できる仕組みができた☺
    - ハンズオンコンテンツ
    - 技術情報

  - Ansibleを取り扱ったサービス等を提供する企業様との連携。
  - コミュニティ活動。

RESTAPI対応

SEIKO

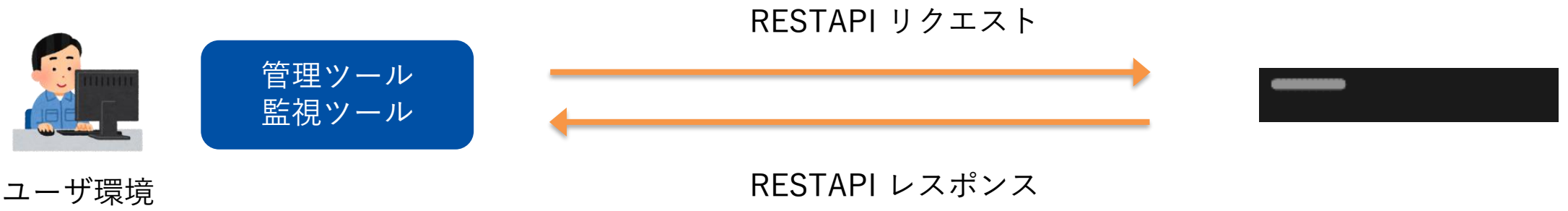
自動化対応　第２弾

- 次は NETCONF？　RESTAPI？
  - Ansible対応を通して、『自動化対応』の反響は非常に大きかった。
  - より幅広い連携が可能となる　という予想と、
    実際に要望の声が多かったRESTAPI を次のターゲットに。
    - 2021年夏位より着手
  - NETCONF を効率よく開発する方法をまだ確立できていない。

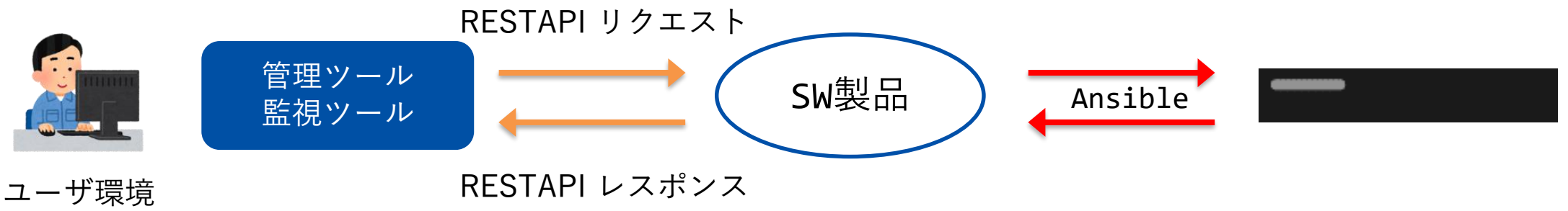
RESTAPI対応

開発アプローチ

- 対応形態
  - 製品に組み込む（比較的従来の開発に近い）



  - SW製品で対応する



RESTAPI対応

SEIKO

開発アプローチ

- 対応形態
  - 製品に組み込む ○

利用ユーザー面	開発面
・追加コスト無し(低)で利用可能 (ネットワーク機器のVerup) ・既存運用システムの構成変更少ない	・ネットワーク機器のVerupで対応可能 ・言語やフレームワークの制約多い (組み込み開発) ・Webサーバから用意する必要有

  - SW製品で対応する

利用ユーザー面	開発面
・SW製品分の追加コスト発生 ・既存の運用システムに組み込みにくい	・新しい製品を作るコスト大きい ・検討項目多い ・言語やフレームワークの制約少ない ・RESTAPI未対応の他の製品群への転用

RESTAPI対応

開発アプローチ

- 従来の機能開発と同じアプローチ
  - Webサーバ / RESTAPIフレームワーク選定

特に重要

軽量

メモリ使用量少ない
余計な機能不要
(クロスコンパイル可能)

情報

公式情報
メンテナンス状況
Web上の知見

拡張性

機能の豊富さ
有効/無効のしやすさ

- 新たに調査・検討が必要な部分
  - RESTAPI の リクエスト / レスポンス仕様

他社実装を調査して参考にはできるが、
通信プロトコルに対応する場合のRFCのような厳密な規定や規格がなく
各ベンダー毎の独自仕様となっている。

RESTAPI対応

SEIKO

開発の苦労話(仕様)

- リクエスト / レスポンス 仕様
  - ① CLIベース

リクエスト	・リクエストデータ（Body）にCLIを指定 ・実行順番も意識（上から実行） ・状態/権限遷移のコマンドも含める ・URIの用意は１つでよい （例 http://<IP>/xxx/xxx/cliapi/）
レスポンス	・CLIの実行結果をほぼそのまま返す ・CLIの出力内容を文字列+配列で扱う （JSONフォーマット）

http://<IP>/xxx/xxx/cliapi
POST

```
"cli" : [
    "show version",
    "show ip"
]
```

```
"response" : [
  [
    "System                 : Software Ver 3.0",
    "Boot Status            : Reboot (05:80:00)",
    "System Up Time         : 2022/05/31 14:47:13",
    "Local MAC Address      : 00:80:15:42:00:08",
    "Number of MAC Address  : 2",
    "Model                  : NS-2250-16 (16 port)",
    "Serial No.             : 56000050",
    "BootROM                : Ver 1.0",
    "Boot System            : main (Ver 3.0)",
    "Boot Config            : external startup1",
    "Main System            : Ver 3.0",
    "Backup System          : Ver 2.2"
  ],
  [
    "Hostname          : NS-2250",
    "IPaddress(eth1)   : 172.31.8.136/16",
    "IPaddress(eth2)   : 10.208.39.136/8"
  ]
]
```

RESTAPI対応

SEIKO

開発の苦労話(仕様)

- リクエスト / レスポンス 仕様
  - ② URI + Method でCLIを表現する RESTAPI

リクエスト	・URIの設計 ・各Methodの定義 GET : 情報の取得 POST : 新規に登録 create系コマンド PUT : 設定の変更 set/unset系コマンド DELETE : 設定の削除 delete系コマンド
レスポンス	・CLIをJSONに整形しなおして返す （Key:Value の形式）

http://<IP>/xxx/xxx/system/version
<GET>

```
{
"response": {
    "Boot": {
      "System": {
        "Version": "3.0",
        "Unit": "main"
      },
      "Status": "Reboot",
      "Config": {
        "Unit": "external",
        "Startup": "startup1"
      },
      "ROM": {
        "Version": "1.0"
      }
    },
    "SystemUpTime": "2022/05/31 14:47:13",
    "HW": {
      "Model": "NS-2250-16",
      "SerialNo": "56000050",
      "MAC": {
        "Local_Address": "00:80:15:42:00:08",
        "Number": "2"
      },
    },
    "System": {
      "Main": "3.0",
      "Backup": "2.2"
    }
  }
}
```

RESTAPI対応

SEIKO

開発の苦労話(仕様)

- リクエスト / レスポンス 仕様
  - 比較

	① CLIベース	② URI + MethodでCLIを表現
ユーザーの利便性視点	<リクエスト> ・CLIベースとなるので、 新たに覚えてもらう事少ない <レスポンス> ・CLIベースだと整形が必要な場面も	<リクエスト> ・新たに覚えてもらう必要がある <レスポンス> ・Key:Value の方がプログラムとの連携容易
開発視点	・CLIを意図的に弾かない限りは全て投入可能となる（＝評価工数が増える） ・CLIベースなので開発工数 抑えられる	・CLIを URI + Method に再設計 ※全てのCLIを対応するのは難しいので 必要なものから用意（スモールスタート） ・CLI → JSON にフォーマット変換が必要 (Ansible対応時は不要だった)

RESTAPI対応

SEIKO

開発の苦労話(仕様)

- URI設計
  - 設計基準
    ・自由に決める事が出来るので、チームでリファレンスを決めて仕様検討。
    (参考) Web API: The Good Parts / O'REILLY
  - API仕様は人にわかりやすく、扱うデータはプログラムしやすく。

URI	Method	概要	CLI例
/users	GET	ユーザ情報一覧の取得	show user
	POST	ユーザ作成	create user nakayama
/users/{name}	GET	ユーザ情報の取得	show user nakayama
	PUT	ユーザ情報の編集	set user usergroup set user password set user sshkey
	DELETE	ユーザ削除	delete user

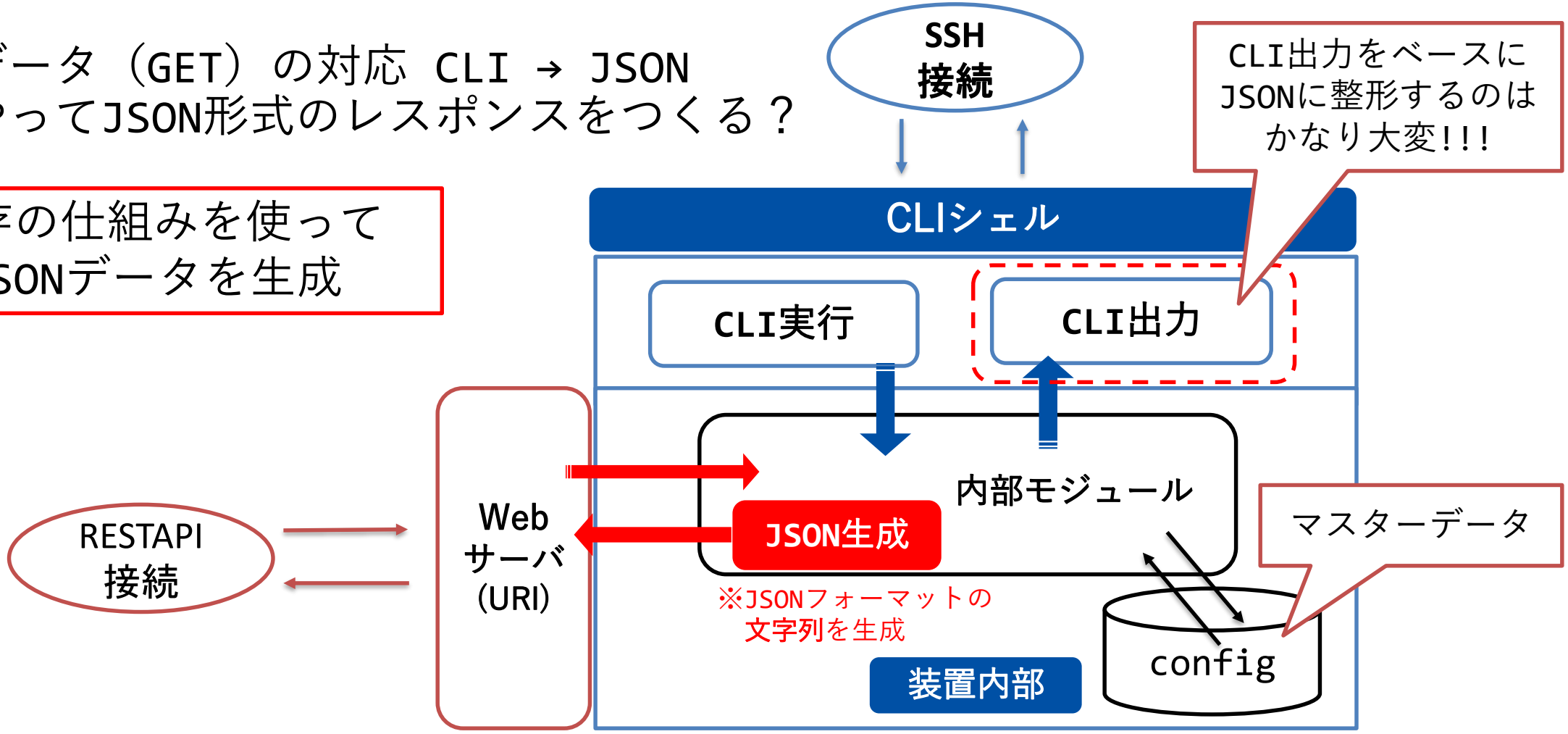
RESTAPI対応

開発の苦労話(実装)

苦労 ①

- 表示データ（GET）の対応 CLI → JSON
  どうやってJSON形式のレスポンスをつくる？

既存の仕組みを使って
JSONデータを生成



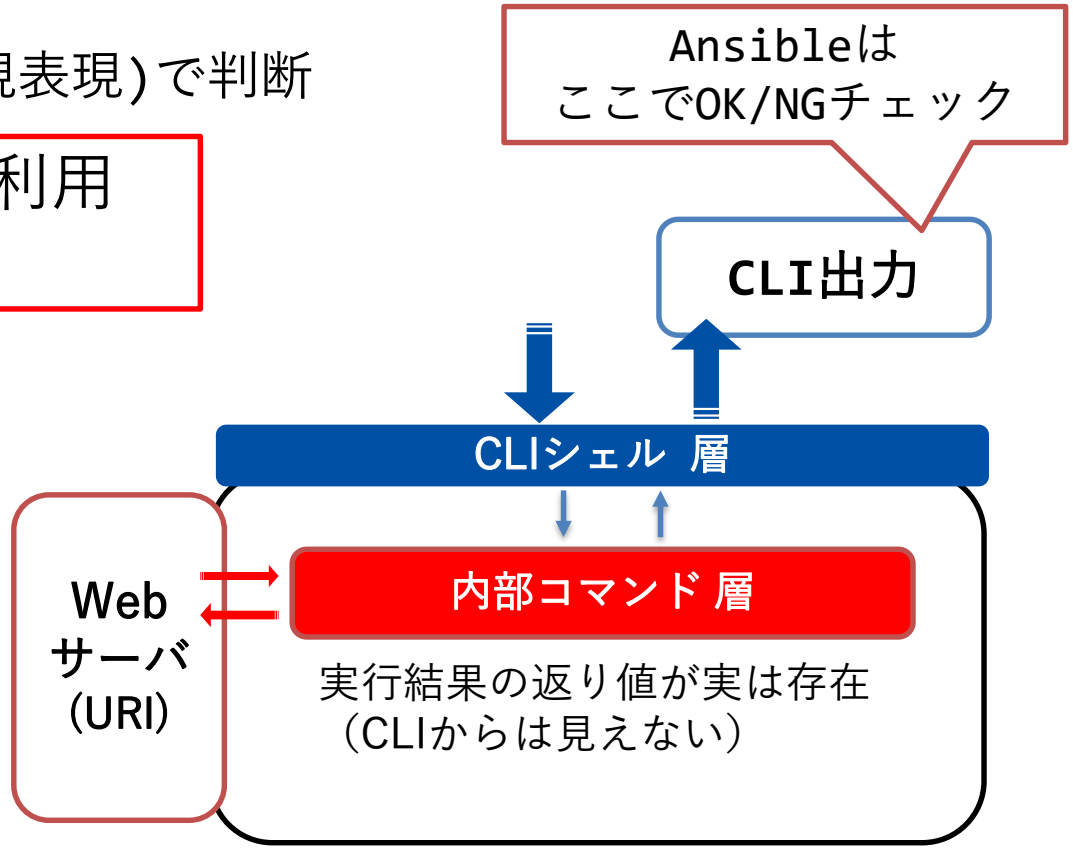
RESTAPI対応

開発の苦労話(実装)

苦労 ②

- 設定オペレーション（POST,PUT,DELETE）の対応
  ・OK/NG の判定どうする？
  Ansibleの場合：CLI出力結果のフォーマット(正規表現)で判断

CLI内部で実行しているコマンドの返り値を利用
→ CLIとRESTAPIで設定処理を共通化



この仕組み(API/CLI共通の処理レイヤー)が無かったら？

Ansibleのように実行結果で判断するアプローチ （プラグインの役割を装置内に用意）
RESTAPI用に新しく設定の仕組みを用意 （ただし今後対応するAPIが増える分だけ･･･）
「CLIベース」の仕様に変更する

RESTAPI対応

SEIKO

開発の苦労話(仕様・実装)

- リクエストデータ、レスポンスデータ
  - ここも自由に作れてしまう、「使いやすさ」をキーワードに開発。
  - Key:Value 形式にはするが、使いやすいAPIを考える。

実行結果の判断 をしやすく	・レスポンスデータについて、全てのURI + Method で共通の返り値を用意。 ・常に同じ部分を参照して判断できるように。 0: 正常終了、エラー文無し 1: エラー　、エラー文有り
設定を簡易に	・GETで取得したレスポンスデータの一部を使って（コピペして） そのまま設定（POST/PUT）オペレーションのリクエストで使えるように。 ・人にとって使いやすい ＝ プログラムで取り扱いやすいか？

```
"info": {
  "result": 0,
  "message": ""
},
```

```
"users": [
 {
  "name": "nakayama",
  "group": "normal",
  "uid": 100,
  "sshkey": ""
 },
```

RESTAPI対応

SEIKO

開発振り返りとこれから

- 開発を終えて
  - 規定や規格のない機能開発は難しい。
    - そんな中でも基準や方針を決める事で対応。
  - 独自仕様となる機能はドキュメントでのフォローも大事。

- 新しい連携の可能性
  - Ansible対応はAnsibleとの連携
  - RESTAPIは様々なクライアントがサポートしている為　可能性を感じる。

- APIを育てる
  - リリースして終わり、ではなく利用して頂いたユーザ様からの
    フィードバックを受けて成長させていく（Ansible対応と同じく）

振り返り

SEIKO

AnsibleとRESTAPIの用途の違い

※担当製品（ネットワーク機器）の対応において感じた個人的な見解です。

- Ansible
  - 「運用内容」や「作業手順」を実現したい場合に使う　事が多い印象

    Playbookの柔軟性によって細かい部分まで再現できる
    (例)show系コマンド：設定作業後の内容確認（確認手順の１つ）

- RESTAPI
  - 「コマンド実行結果＝情報」を取得したい場合に使う　事が多い印象

    取得した値は、運用/監視ツール側の　判断材料
    (例)show系コマンド：特定の情報（設定値/統計値/状態）取得の為

ネットワーク自動化対応　＝　API対応　だけど、APIにも用途や特性がある

振り返り

SEIKO

ネットワーク自動化 対応前後の「CHANGE」

- 変わった事
  - 担当製品の使われ方
    - 「ネットワーク自動化」の中で使える１つのパーツとして
      少しずつ認識されてきたのではないかなと思っています。

- 変えてはいけない部分と思った事
  - 主機能は何か を忘れてはいけない

各APIは、運用で使われやすい機能を連携しやすくするのが目的
API対応によって、主機能（本業）が影響を受ける事がないように、
というのは常に意識して開発。

今後について

- 構想から5年かけて2つのAPIに対応しました。
  - Ansible
  - RESTAPI

- ネットワーク運用の特性（安定した技術を長く使う）からもしばらくこの２つで大丈夫だといいなー と思っていますが、5年、10年先はどうなるか。

- 引き続き ネットワーク運用のトレンドや必要となる技術は注視していきます。

議論

ネットワーク自動化に**対応する側** 目線	ネットワーク自動化を**構築/運用する側** 目線
●弊社の今回のアプローチについてご意見 ●APIやツールに対応する場合においての各社のアプローチ ・選定基準/対応方法 ・規格のない機能の考え方 ●デファクトとなる技術の見極め	●マルチベンダー環境下で求められるAPIやツールの選定基準/重要視するポイント ●「とりあえずAPI対応してほしい」という場合の具体的な内容は？（=RESTAPI？） ●使いやすさ　について

●その他

- この先の「ネットワーク自動化」に必要となりそうな機能
- ネットワーク機器のAPIに求めたい事
- Ansible, RESTAPI 以外で対応してほしいと思うAPI
- NETCONF って対応した方がよいですかね？（サーバ側実装のアドバイス下さい）

参考①

- 参加した イベント・展示会
  - JANOG
  - IntenetWeek
  - INTEROP
- 勉強会
  - ネットワークプログラマビリティ勉強会
  - NetOpsCoding

- 検討初期に 特に 参考にさせて頂いた資料

カテゴリ	資料名
ネットワーク自動化	NAPALMでつくる ネットワークオペレーション自動化への道のり https://www.slideshare.net/ssuser6a8d29/napalm
	なぜネットワーク運用自動化が進まないのか ～とあるNW屋の泣き言～ https://www.slideshare.net/taijitsuchiya5/ss-47398248
	運用自動化の為にネットワーク機器に求めるもの http://yuyarin.hatenablog.com/entry/2015/12/01/000001
	明日からはじめるネットワーク運用自動化（JANOG41） https://www.janog.gr.jp/meeting/janog41/program/auto
Ansible	Ansible ネットワーク自動化チュートリアル（JANOG42） https://www.janog.gr.jp/meeting/janog42/program/ASBL
ツール・API	ネットワーク自動化、なに使う？ ～自動化ツール紹介～ https://tekunabe.hatenablog.jp/entry/2017/08/18/network_automation_tools
	ネットワーク機器のAPIあれこれ入門 https://www.slideshare.net/kentaroebisawa/api-59059827

イベントを主催して頂いている方々
資料や考えをアウトプットしている方々
本当にいつもありがとうございます m(_ _)m

参考②

SEIKO

Ansible対応

- Ansible ユーザー会 ではもう少し濃い情報を公開しています。
  興味ある方、これからモジュール開発の検討をしている方の
  参考になれば幸いです。

  - 国内NW機器ベンダーがAnsible対応した話
    https://speakerdeck.com/nakashin1/developing-ansible-network-module
  - Ansible Collections 対応してみたベンダー 体験談
    https://speakerdeck.com/nakashin1/supporting-ansiblecollections
  - モジュール開発と運用
    https://speakerdeck.com/nakashin1/module-developing-and-operation

SEIKO

セイコーソリューションズ株式会社