SEIKO

SmartCS × { REST : API }

SmartCS　REST API機能説明資料

2023.9 Rev.6

- 本資料ではSmartCS NS-2250のREST API機能について説明しています。
- NS-2250の設定など、本資料に記載が無い内容はREST API運用ガイド等を参照ください。

目次

SmartCSの概要

・コンソールポートとは

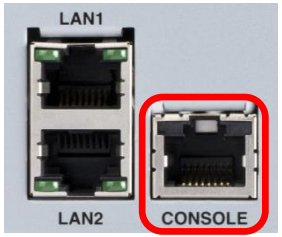
・SmartCSとは

コンソールポートとは

- コンソールポートの役割
  IP通信ではなく、シリアル通信でオペレーションをするためのインターフェース

  ・**初期設定**
  IP設定、ユーザ作成、SSHの有効化 などの初期設定

  ・**緊急時のオペレーション**
  LANポート障害、ネットワーク障害 などの影響で、
  装置へIPアクセスできなくなった際の最後のアクセス手段



RJ45

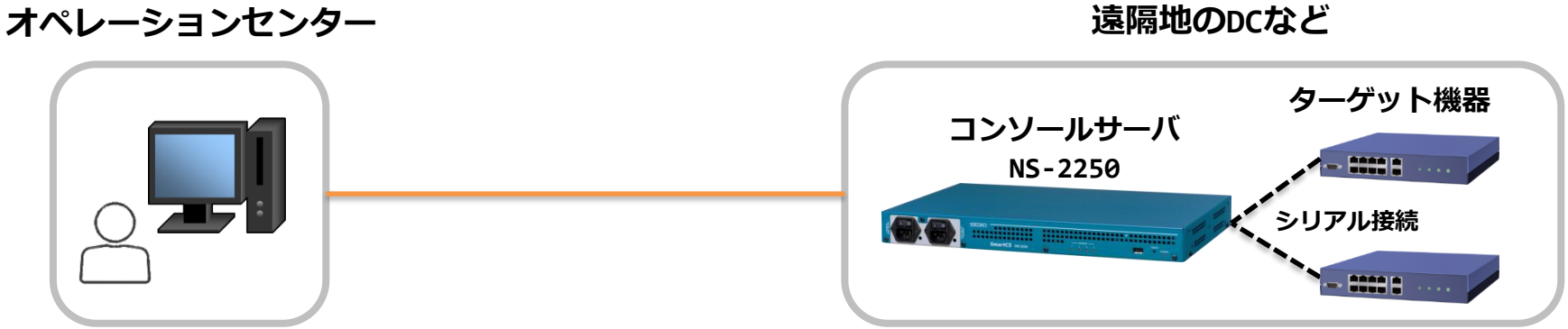
DB9

SmartCSとは

- SmartCSの役割
  コンソールポートを集約し、リモートでアクセスできるようにする装置

・**リモートアクセス**
IPアクセスできない状態の装置へ、リモートでアクセス可能

・**オペレーション範囲の拡大**
リモートからは実施しづらい作業を、安全に実行可能
ACL/ルーティングなどの設定変更、ファーウェアアップデート など



SmartCSと運用自動化

- REST API機能概要
- URL、APIリソース
- curlコマンドの実行例
- ユースケース例
- Zabbix経由での実行例

{ REST : API }

REST API機能概要

- REST API機能に対応したFWをリリースしました！

コンソールサーバーSmartCSがAPIによるITインフラとの連携機能を強化

－REST API対応でシステム全体の運用自動化の実現を促進－

2022年6月7日　ニュースリリース

セイコーソリューションズ株式会社（代表取締役社長：関根 淳、本社：千葉県千葉市、以下 セイコーソリューションズ）は、コンソールサーバー「SmartCS」でREST APIに対応した新ファームウエア v3.0を開発し、2022年6月中旬以降に提供を開始します。今回のバージョンアップにより、さまざまな運用管理ツールとの連携機能を強化し、システム全体の運用自動化を実現します。

SmartCSは、複数のネットワーク機器へのコンソール接続を集約するアプライアンス機器です。今回、アプリケーションやシステム間連携で使われている主要なAPIの一つである REST APIに対応することで、ネットワーク運用の現場で使われる多種多様な運用管理ツールとの連携が可能となりました。SmartCSの設定変更や情報取得といった作業がAPI経由で行えるようになるほか、SmartCSに接続しているターゲット機器が出力するコンソールログやオペレーション情報をAPI経由で取得することが可能となります。さらに、これらの作業を運用システムと連携して行うことで、オペレーション作業の省力化につながります。

また、APIに対応していないネットワーク機器についても、SmartCSに接続することでREST API経由でのオペレーションや情報取得が可能となるため、従来は難しかったシリアル通信情報を運用システムと連携できるようになり、システム全体のネットワーク機器の包括的な管理実現が期待できます。

https://www.seiko-sol.co.jp/archives/69074/

REST API機能概要

アプリケーション、システム間の
連携で使われる主要なAPIとは？

→ { REST : API }

- SmartCSをv3.0にバージョンアップするだけで利用可能
- REST(Representational State Transfer)の原則に沿って設計されたWeb API
- 運用・監視ツールと連携させることで、より有効にSmartCSを活用することができます
- SmartCS自身、およびシリアルポートに接続されている監視対象装置をREST APIで操作可能

できること

SmartCSへのコマンド実行	ターゲット機器へのコマンド実行
・ユーザ、シリアルポートの**設定変更、各種情報取得** ・保存されている**コンソールログの取得、検索**	・障害により**IPリーチできない状態の機器**にコマンド実行 ・**APIに対応していない機器**へのオペレーション ・**APIやSNMPで取得できない情報**を取得

URLとHTTPメソッド

SEIKO

- ベースURL
  SmartCSのREST API機能にアクセスする ベースURL は以下のアドレスとなります。
  “xxx” の部分には、各APIリソースのURLを指定します。

プロトコル	ベースURL	デフォルトポート番号
HTTP	http://<IPアドレス>:<httpポート番号>/api/v1/xxx	10080
HTTPS	https://<IPアドレス>:<httpsポート番号>/api/v1/xxx	10443

- HTTPメソッド
  SmartCSのREST API機能は、以下のHTTPメソッドをサポートしています。

メソッド	説明
GET	指定したAPIリソースから情報取得する要求を行います。 SmartCSのCLIコマンドの、show系コマンドのオペレーションとなります。
POST	指定したAPIに新たなリソースを作成する要求を行います。また、特定のオペレーションの実行についての要求を行います。 SmartCSのCLIコマンドの、createコマンドなどのオペレーションとなります。
PUT	指定したAPIリソースについて変更/修正する要求を行います。 SmartCSのCLIコマンドの、set/unsetコマンドなどのオペレーションとなります。
DELETE	指定したAPIリソースについて削除する要求を行います。 SmartCSのCLIコマンドの、deleteコマンドなどのオペレーションとなります。

APIリソースとメソッド

SEIKO

分類	URL	メソッド	概要
SYSTEM	/system/version	GET	システム情報の取得
USERS	/users	GET	ユーザ情報（一覧）の取得
		POST	ユーザ作成
	/users/{username}	GET	ユーザ情報の取得
		PUT	ユーザ情報の編集
		DELETE	ユーザ削除
	/users/login	GET	ログインユーザ情報の取得
SERIAL	/serial/tty	GET	シリアル情報(一覧)の取得
	/serial/tty/{ttylist}	GET	シリアル情報の取得
		PUT	シリアル情報の編集
	/serial/hangup/tty/{ttylist}	POST	シリアルのhangup
TTYMANAGE	/ttymanage	POST	TTYマネージ機能を使ってシリアルポートに文字列の送受信スクリプトを実行
LOG/HISTORY	/log/history/console	GET	SmartCSのコンソールログ情報の取得
	/log/history/command	GET	SmartCSのコマンドログ情報の取得
	/log/history/ttysend	GET	SmartCSのttysendログ情報の取得
	/log/history/webapi	GET	SmartCSのwebapiログ情報の取得
LOG/SERIAL	/log/serial/tty/{ttyno}	GET	SmartCSのttyログ情報の取得
	/log/serial/files/tty/{ttyno}	GET	SmartCSのttyログ情報の取得（DL）
	/log/serial/search/tty/{ttyno}	GET	SmartCSのttyログ情報を検索

SmartCS側の設定

SEIKO

- ユーザの作成
  - ・extusrグループのユーザを作成する必要がある
  - ・ターゲット機器へのコマンド実行を行う場合、アクセス可能なportの設定が必要

```
# create user <username> group extusr port <port_number> password
```

- ユーザへの権限付与
  - ・SmartCSの設定変更や、ログ取得を行う場合、root権限の付与が必要

```
# set user <username> permission root on
```

  - ・ターゲット機器へのコマンド実行を行う場合、ttymanage権限の付与が必要

```
# set user <username> permission ttymanage on
```

- Webサーバの有効化

```
# enable http/https
```

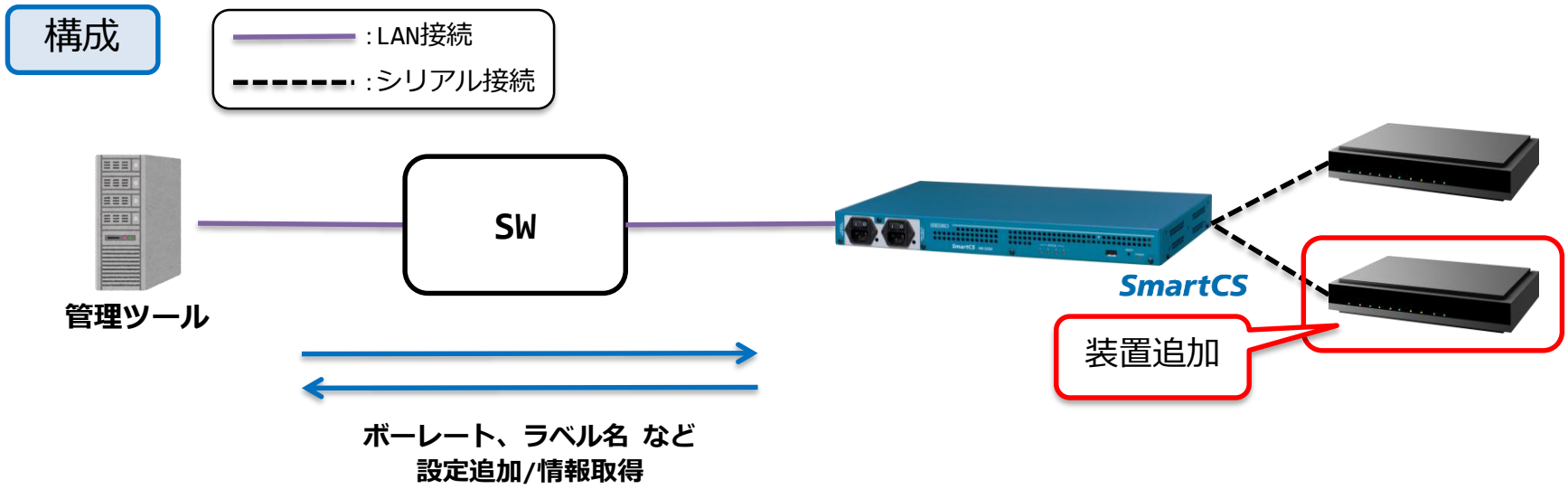
- ttyマネージ機能の有効化

```
# enable ttymanage
```

SEIKO

<SmartCSの設定変更、情報取得>

シリアルポートに接続される装置の追加時に、SmartCSのシリアルポート設定と情報取得を管理ツールから行います。



ユースケース① curlでSmartCSから情報を取得する

SEIKO

- tty設定情報取得

```
$ curl -Ss-u api:api -X GET http://<IP>:<PORT>/api/v1/serial/tty/1 | jq
```

実行結果

SmartCSのtty1の設定情報

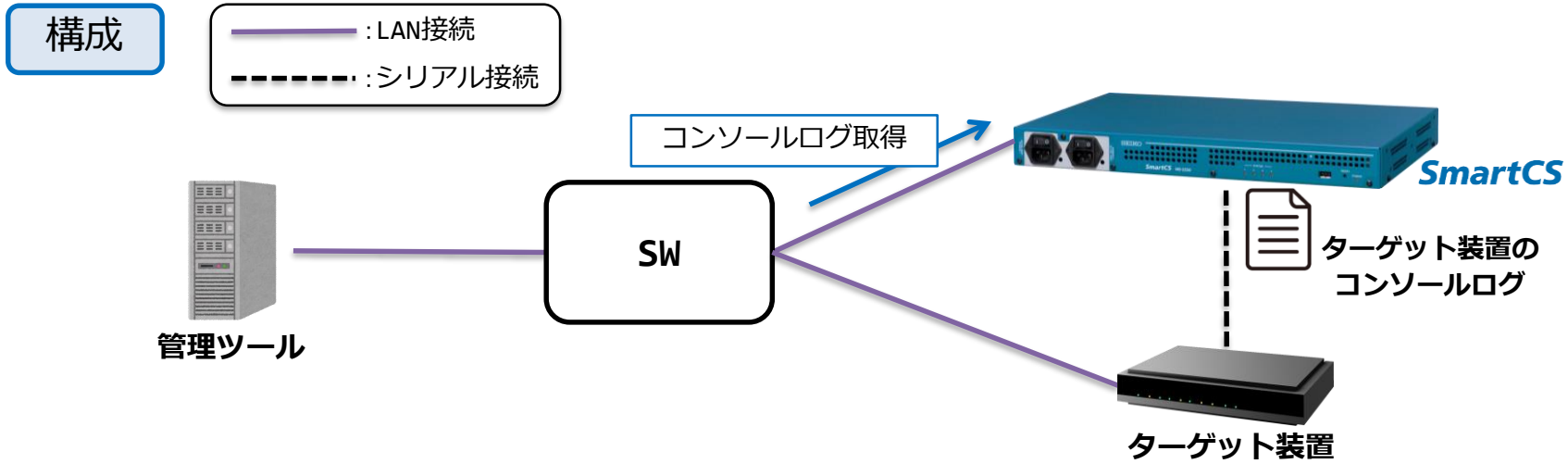
```
{
  "info": {                          ┐
    "result": 0,                     ├ 共通データ
    "message": ""                    ┘
  },
  "ttylist": [                       ┐
    {                                │
      "tty": 1,                      │
      "config": {                    │
        "baud": 9600,                │
        "bitchar": 8,                ├ レスポンスデータ
        "parity": "none",            │
        "stop": 1,                   │
        "flow": "none",              │
        "detect_dsr": "off",         │
        "label": "sw_01"             │
      },                             ┘
      "status": {                    ┐
        "DSR": "off",                │
        "CTS": "off",                │
        "DTR": "on",                 │
        "RTS": "on",                 │
        "CD": "off"                  │
      },                             │
      "stats": {                     ├ レスポンスデータ
        "TX_Octets": 1925,           │
        "RX_Octets": 32180,          │
        "Error_Parity": 0,           │
        "Error_Framing": 0,          │
        "Error_Overrun": 0,          │
        "Break_Count": 0             │
      }                              │
    }                                │
  ]                                  │
}                                    ┘
```

※実行例は整形して記載しています

SEIKO

<コンソールログ取得>

SmartCSに保存されているターゲット機器のコンソールログを**取得**。
テキストで取得するか、あるいはファイルとして取得し管理ツールに保存。



ユースケース② curlでSmartCSからコンソールログを取得する

SEIKO

- コンソールログ取得

```
$ curl -Ss -u api:api -X GET http://<IP>:<PORT>/api/v1/log/serial/tty/1 | jq
```

実行結果　tty1のコンソールログ

```
{
  "info": {
    "result": 0,
    "message": ""
  },
  "log": [
    "Technical Support: http://www.cisco.com/techsupport",
    "Copyright (c) 1986-2020 by Cisco Systems, Inc.",
    "Compiled Sun 06-Sep-20 10:50 by prod_rel_team",
    "*Jun 20 06:51:53.534: %STACKMGR-5-MASTER_READY: Master Switch 1 is READY",
    "*Jun 20 06:51:53.654: %SSH-5-ENABLED: SSH 1.99 has been enabled",
    "*Jun 20 06:51:55.098: %LINK-5-CHANGED: Interface GigabitEthernet1/0/2, changed state to
administratively down",
    "*Jun 20 06:51:55.184: %USB_CONSOLE-6-MEDIA_RJ45: Console media-type is RJ45.",
    "*Jun 20 06:51:58.949: %PNP-6-PNP_BEST_UDI_UPDATE: Best UDI [PID:C1000-8P-2G-L,VID:V01,SN:PSZ25231HF7]
identified via (master-registry)",
    "*Jun 20 06:51:58.949: %PNP-6-PNP_CDP_UPDATE: Device UDI [PID:C1000-8P-2G-L,VID:V01,SN:PSZ25231HF7]
identified for CDP",
    "*Jun 20 06:51:58.950: %PNP-6-PNP_DISCOVERY_STOPPED: PnP Discovery stopped (Startup Config Present)"
  ]
}
```

共通データ

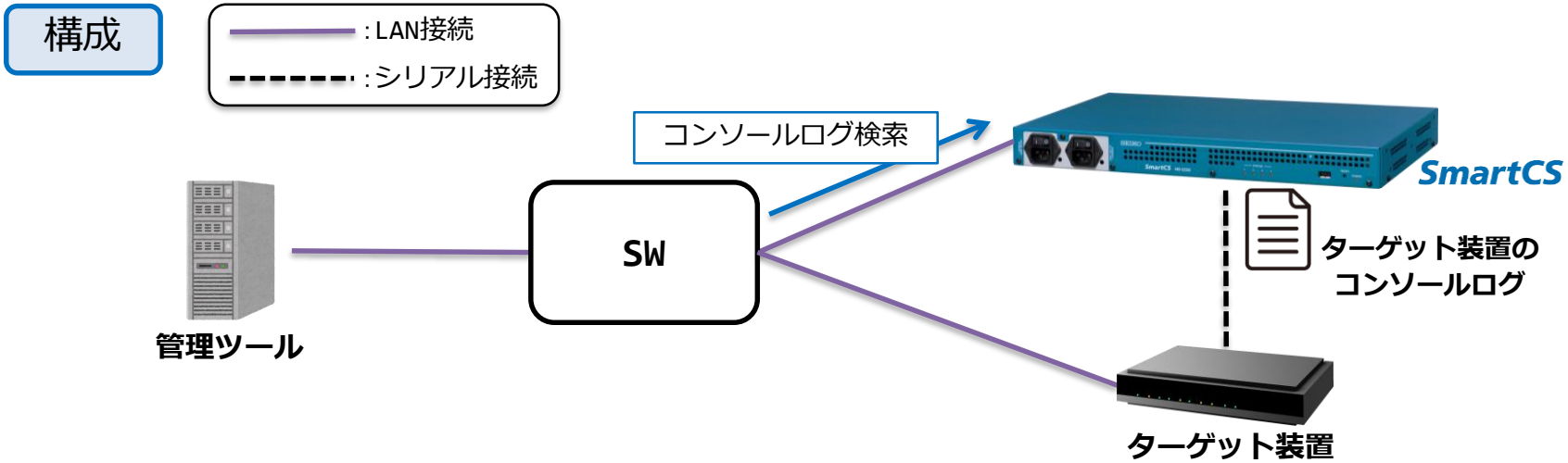
レスポンスデータ

※実行例は整形して記載しています

SEIKO

<コンソールログ検索>

SmartCSに保存されているターゲット機器のコンソールログを**検索**。
"error" や "reboot" などの文字列が含まれるログを検索し、
前後の行を含めて取得することで障害時の状況確認に繋がります。



ユースケース③ curlでSmartCSからコンソールログを検索する

SEIKO

- コンソールログ検索

```
$ curl -Ss -u api:api -X GET http://<IP>:<PORT>/api/v1/log/serial/search/tty/1?string=system&line=1 | jq
```

実行結果

tty1のコンソールログのうち、systemという文字列を含むログとその前後1行

```
{
  "info": {
    "result": 0,
    "message": ""
  },
  "tty": "1",
  "label": "C1000_1",
  "string": "system",
  "count": "2",
  "data": [
    [
      "DRAM Size: 512 MB",
      "Xmodem file system is available.",
      "USB EHCI 1.00"
    ],
    [
      "",
      "*** The system will autoboot in 5 seconds ***",
      "Send break character to prevent autobooting."
    ]
  ]
}
```

共通データ

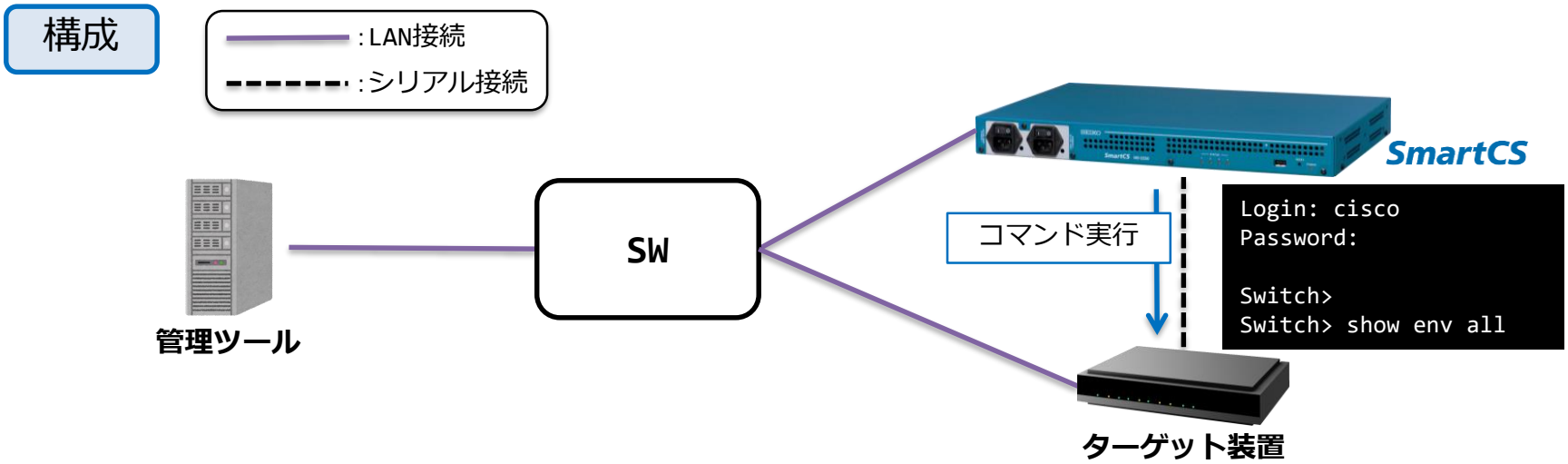
レスポンスデータ

※実行例は整形して記載しています

SEIKO

<ターゲット機器へのコマンド実行>

SmartCSに接続されているターゲット機器にAPI経由でコマンドを実行。
API経由でオペレーションできないリソースの取得などが可能に。



ユースケース④ curlでSmartCS接続機器から情報を取得する

SEIKO

- ttymanage機能を使用してシリアルポートに文字列の送受信を実行

```
$ curl -u api:api -X POST http://<IP>:<PORT>/api/v1/ttymanage -d @command.json | jq
```

リクエストボディのJSONデータ
■command.json

```
{
    "tty": "1",
    "cmd_timeout": 30,
    "recvchar": [
        "Press RETURN to get started."
    ],
    "recvchar_regex": [
        "(^|¥¥r|¥¥n|!)[a-zA-Z0-9_().-]*(>|#)",
        "[Pp]assword: ",
        "[Uu]sername: "
    ],
    "sendchar": [
        "__NL__",
        "ne",
        "network",
        "enable",
        "network",
        "terminal length 0",
        "show env all",
        "exit"
    ]
}
```

■tty
・文字列の送受信を行うシリアルポート番号

■recvchar
・sendchar送信後に受信を期待する文字列を指定
・指定した文字列のいずれかを受信すると次のsendcharを送信

■recvchar_regex
・recvcharの正規表現

■sendchar
・指定したttyに送信する文字列を指定
・リストの上から順番に送信

■JSONデータを1行にして送信することも可能です。

```
$ curl -u api:api -X GET http://<IP>:<PORT>/api/v1/ttymanage -d
{"tty":“1","cmd_timeout":30,"recvchar":["Press RETURN to get
started."],"recvchar_regex":["(^|¥¥r|¥¥n|!)[a-zA-Z0-9_().-
]*(>|#)","[Pp]assword: ","[Uu]sername:
"],"sendchar":["__NL__","ne","network","enable","network","termi
nal length 0","show env all","exit"]}
```

ユースケース④ curlでSmartCS接続機器から情報を取得する

SEIKO

実行結果

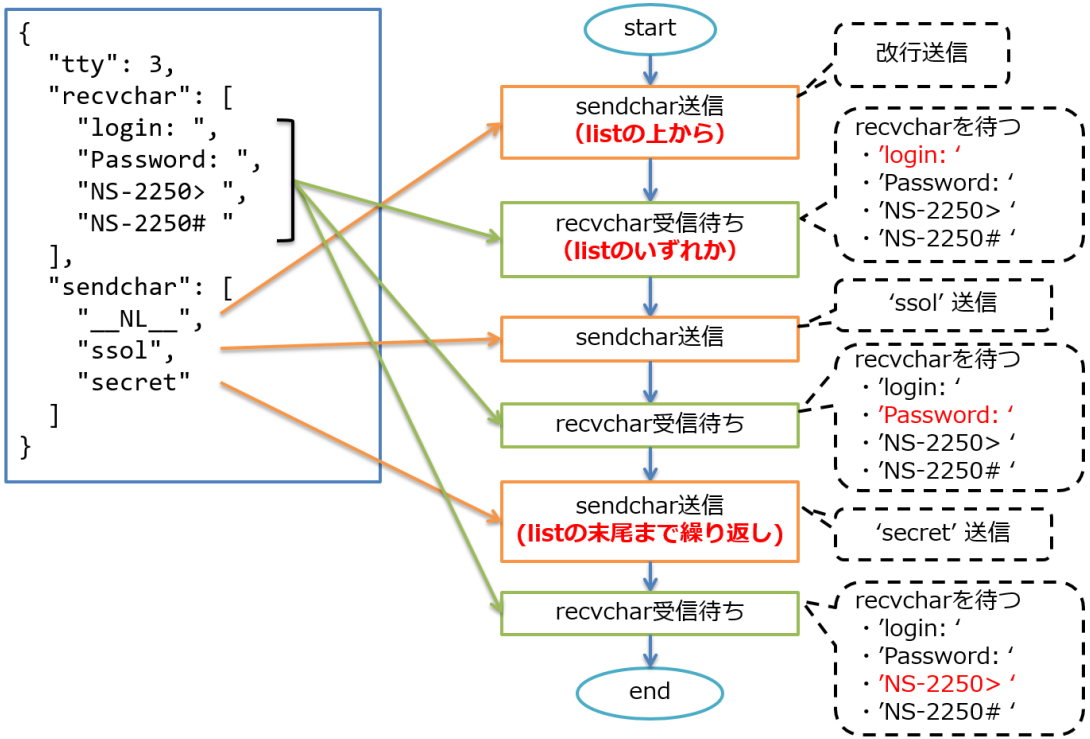

```
"info": {
   "result": 0,
   "message": ""
 },
 "request": {
   "tty": "3",
   "nl": "cr",
   "cmd_timeout": 30,
   "sendchar": [
     "__NL__",
     "ne",
     "network",
     "enable",
     "network",
     "terminal length 0",
     "show env all",
     "exit"
   ],
   "recvchar": [
     "Press RETURN to get started."
   ],
   "recvchar_regex": [
     "(^|¥¥r|¥¥n|!)[a-zA-Z0-9_().-]*(>|#)",
     "[Pp]assword: ",
     "[Uu]sername: "
   ],
   "error_recvchar_regex": [],
   "error_detect_on_sendchar": "cancel",
   "ttycmd_debug": "off"
 },

～～省略～～
```

```
  {
     "execute_command": "show env all",
     "response": [
       "",
       "SWITCH: 1",
       "SYSTEM TEMPERATURE is OK",
       "System Temperature Value: 49 Degree Celsius",
       "PHY Temperature Value: 44 Degree Celsius",
       "DDR Temperature Value: 50 Degree Celsius",
       "System Temperature State: GREEN",
       "Yellow Threshold : 79 Degree Celsius",
       "Red Threshold    : 82 Degree Celsius",
       "",
       "SWITCH: 1",
       "PID:  Built-in",
       "System Power:(Watts) 173",
       "Max Power Usage:(Watts) 81",
       "Maximum Heat Dissipation: (Watts) 22",
       "PoE Power extract:(Watts) 0.0",
       "Power Supply Status: Good",
       "",
       "C1000_2#"
     ]
   },

～～省略～～

  }
 ],
 "error": 0
}
```


※レスポンスデータの一部(data)については内容を省略しています。
data部分に、実際のシリアル通信のオペレーションが格納されます。

SEIKO

- sendchar、recvcharの動作イメージ

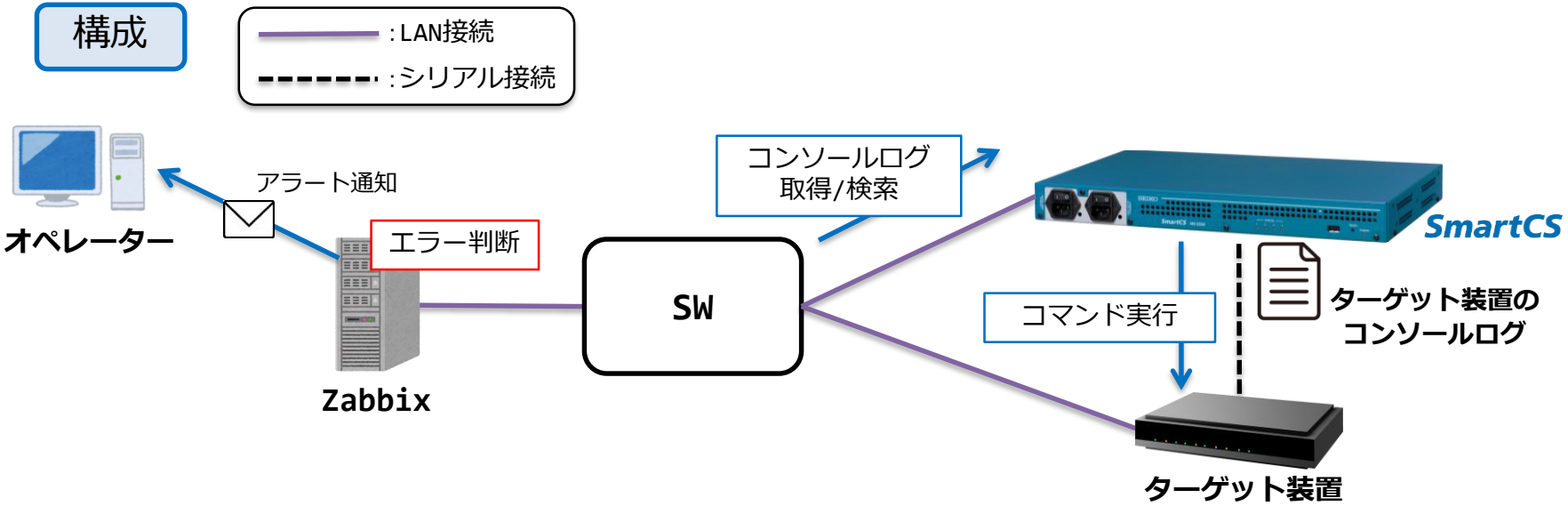


SEIKO

<監視ツールと連携>

Zabbixなどの監視ツールと連携し、障害発生をトリガーにターゲット機器へのコマンド実行などが可能。さらに、コンソールログの内容やコマンド実行結果をエラーと判断した際に、メール等でアラート情報を送信。

構成



今後検討している機能拡張について

tty、ユーザ、ログ 以外のリソース操作についても
今後拡張を予定しています。

例：

- SmartCSのconfig情報の取得やアップロード
  - config管理をシステム連携しやすく
- バージョンアップファイルのアップロード
  - 複数台のバージョンアップオペレーションを簡易に

お使いいただいてお気づきの点、要望等ありましたら是非ご連絡下さい

参考情報

- ttymanage機能を利用する場合のパラメーター
- ShowNet2022-2023での取り組み
- Zabbix経由でコマンド実行するサンプル
- Ansible playbookをJSONに変換する
- 参考情報

{ REST : API }

ttymanage機能を利用する場合のパラメーター

SEIKO

- リクエストボディで指定するオプション

オプション名	型	説明
tty (※必須)	数値(文字列)	sendcharで指定した文字列を送信するシリアルポート番号を指定。 1-10 の様なリスト指定は不可。“1” の様に文字列指定も可能。
cmd_timeout	数値(文字列)	sendcharで指定した文字列を送信する際のタイムアウト値を指定。 “30” の様に文字列指定も可能。
nl	文字列	sendcharで送信する改行コードを指定。(cr、lf、crlf)
sendchar (※必須)	配列	指定したttyに送信する文字列を指定。 一部文字列として指定不可な文字種があります。($ や ? など) 特殊な送信方法 ・__NL__　改行送信 ・__CTL__:hex　制御文字を1文字送信(0x00～0x1f、0x7f) ・__HEX__:hexs　制御文字(複数)送信(0x00～0x7f)
recvchar	配列	指定したAPIリソースについて変更/修正する要求を行います。 SmartCSのCLIコマンドの、set/unsetコマンドなどのオペレーションとなります。
recvchar_regex	配列	指定したAPIリソースについて削除する要求を行います。 SmartCSのCLIコマンドの、deleteコマンドなどのオペレーションとなります。

ttymanage機能を利用する場合のパラメーター

SEIKO

- リクエストボディで指定するオプション

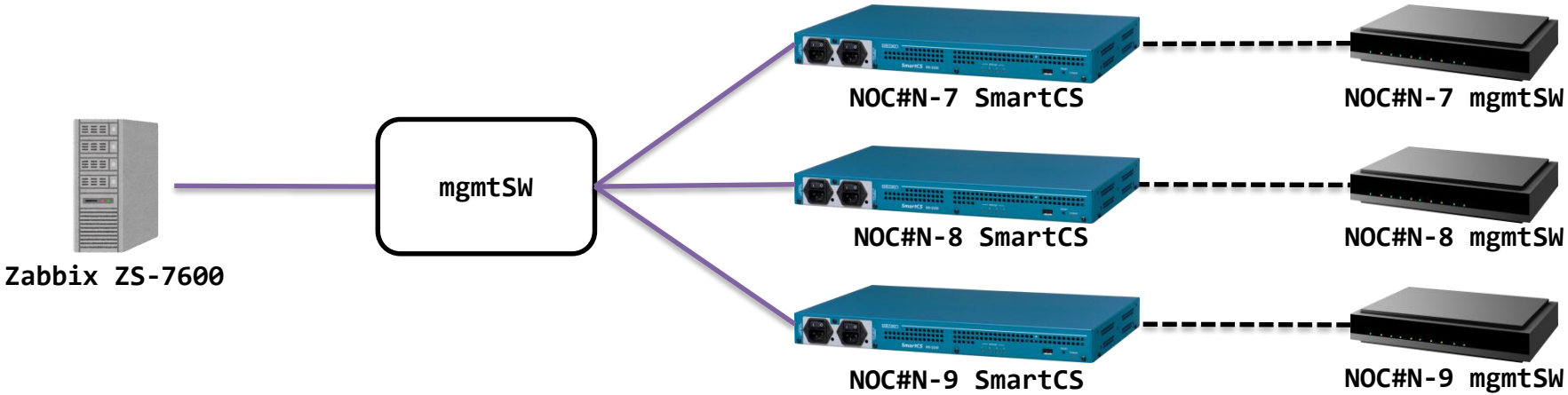
オプション名	型	説明
error_detect_on_sendchar	文字列	sendchar送信後にエラーが発生した場合、以降のsendcharを送信するか/しないかの設定を指定します。 ・exec エラーが発生した場合でも、sendcharを送信します。 ・cancel エラーが発生した場合、sendcharを送信しません。
error_recvchar_regex	配列	sendchar送信後、指定された文字列（正規表現）が含まれていたらエラーとなる文字列を指定します。
ttycmd_debug	文字列	デバッグ情報をレスポンスデータに含むかどうかの設定値を指定します。(off、on、detail)

Zabbixアプライアンス(ZS-7600)と連携
REST API機能を使用したコンソール経由のオペレーションを実施

- mgmtSWからLLDP neighbors情報の定期取得
- 伝送装置からスロットステータス情報の定期取得
- トラフィック増加時のインターフェース情報取得

- mgmtSWからLLDP neighbors情報の定期取得

SNMPで取得した際、IFの順番通りに取得できない場合があるため、
CLI経由で整形された情報として取得、ダッシュボードでの表示を実施

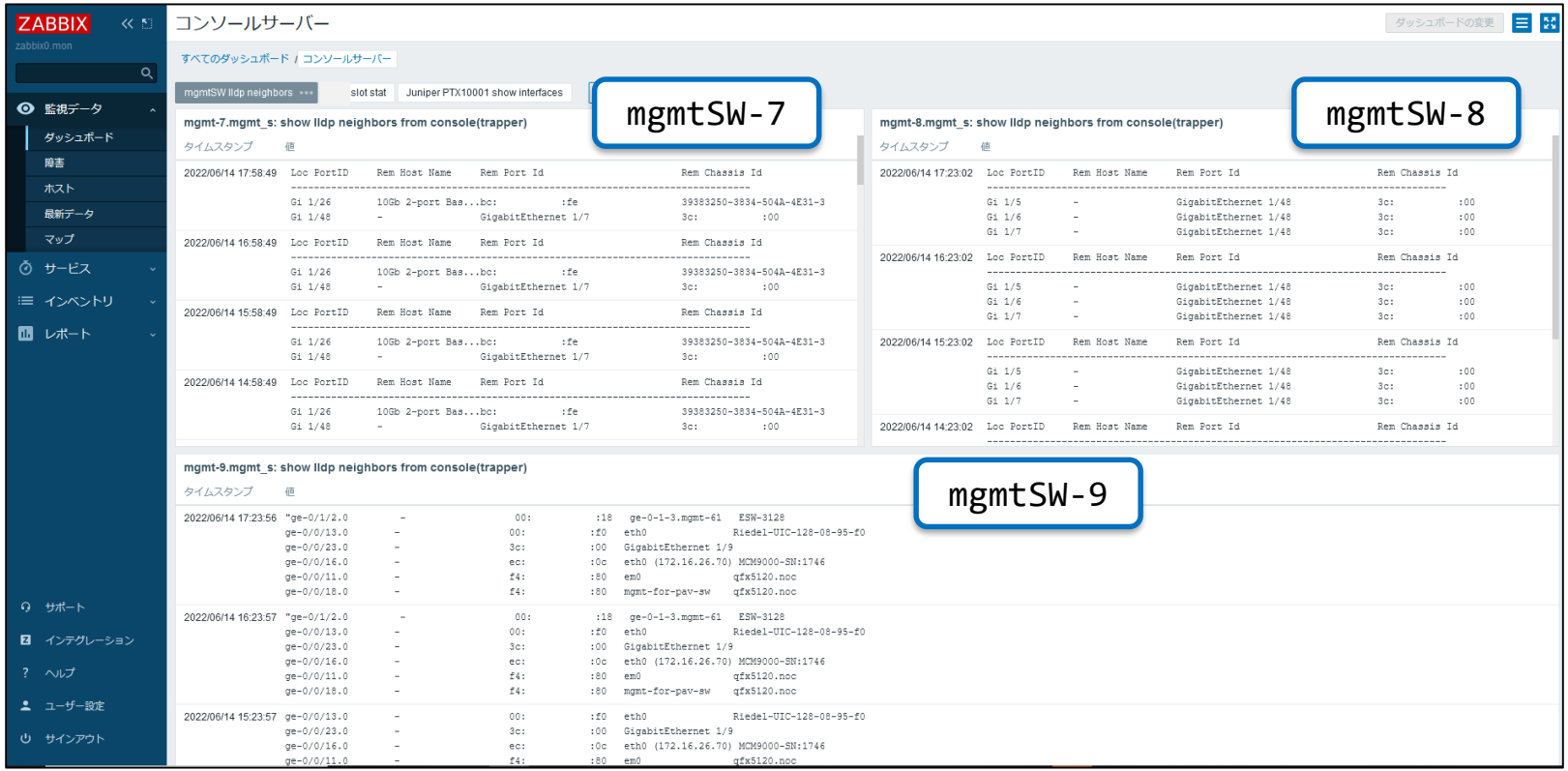


REST API(HTTP)
→コンソール経由でコマンド実行するリクエスト

シリアル通信
→show lldp neighborsを実行

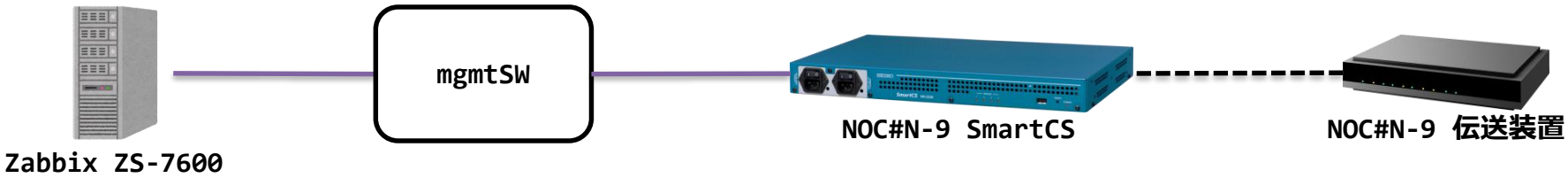
SEIKO

Zabbix ダッシュボード



- 伝送装置からスロットステータス情報の定期取得

一覧で各スロットの情報を見たいがシャーシが分かれておりSNMPでの取得が難しいため、CLI経由で各スロット情報を取得し、ダッシュボードでの表示を実施

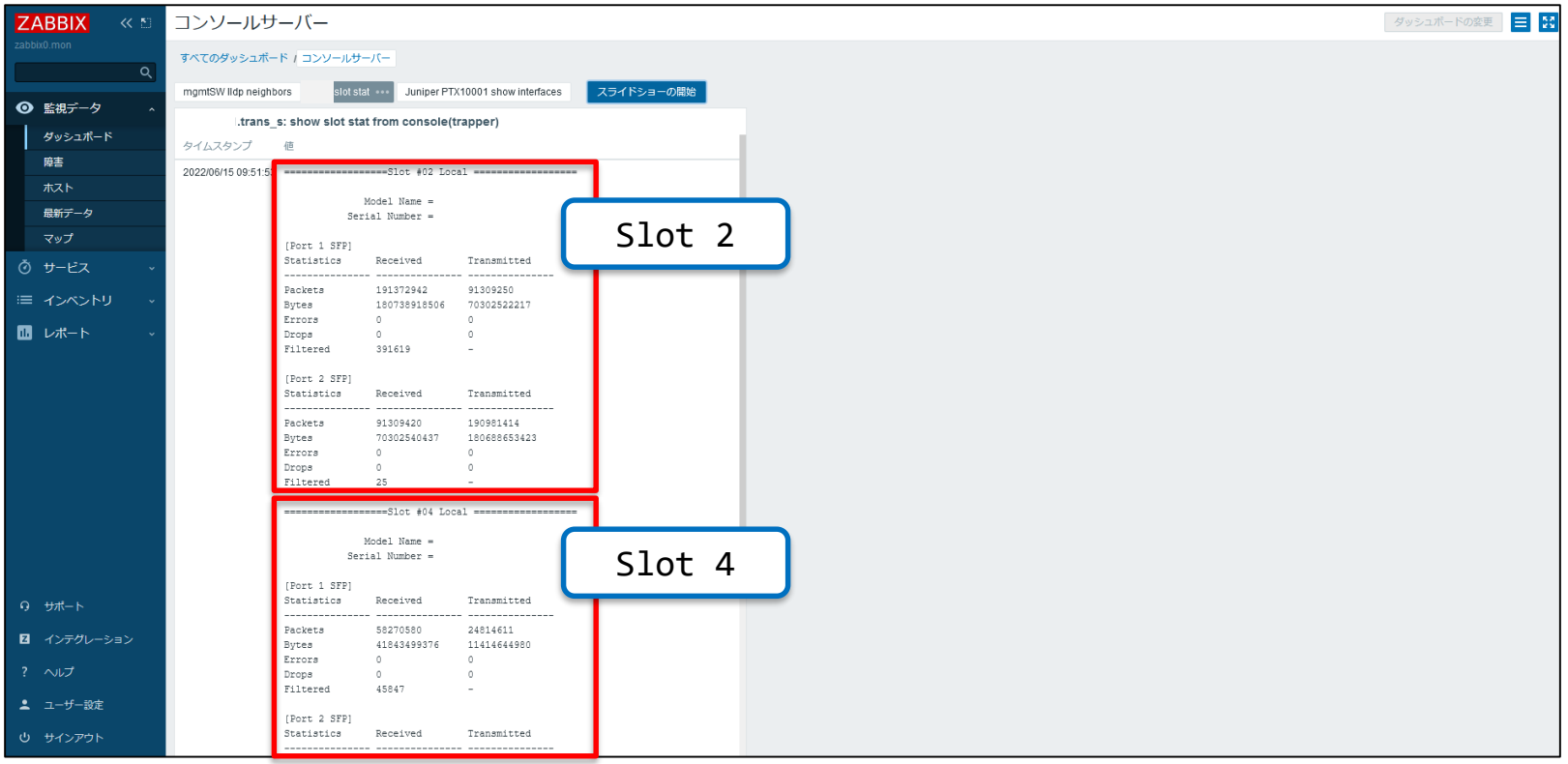


REST API
→コンソール経由でコマンド実行するリクエスト

シリアル通信
→show slot 2 stat
show slot 4 stat を実行

SEIKO

Zabbix ダッシュボード



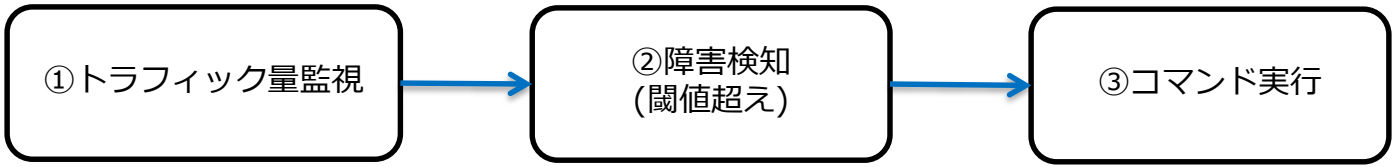
- トラフィック増加時のインターフェース情報取得

トラフィックが増加して特定の閾値を超えた場合に、
コンソール経由でインターフェース情報の取得を行いました。

REST API
→コンソール経由でコマンド実行するリクエスト

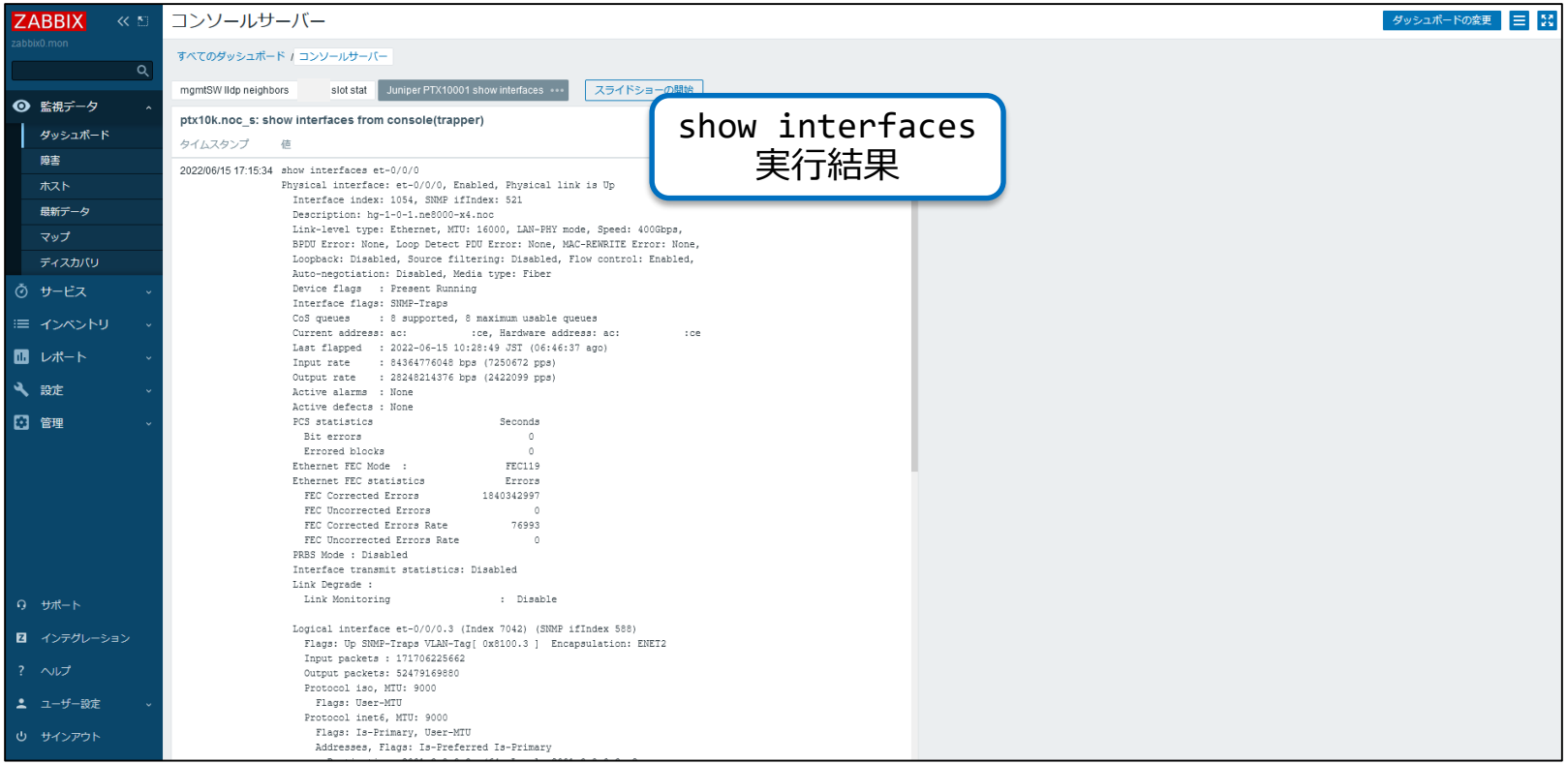
シリアル通信
→show interfaces を実行

フロー



SEIKO

Zabbix ダッシュボード



System Answer G3と連携

- 伝送装置からSFP受光レベルの定期取得

Zabbixアプライアンス(ZS-7600)と連携

- Boundary ClockからPTP情報の定期取得

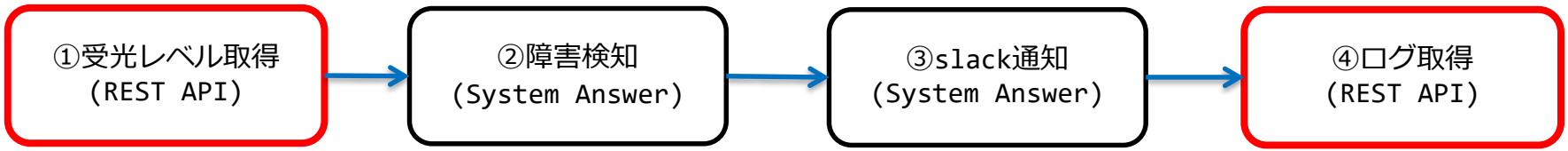
ShowNet2023での取り組み(IBC様/System Answer G3連携)

SEIKO

- 伝送 / SFP受光レベルの監視
  SNMP MIBで取得できないSFP受光レベルをconsoleから取得
  取得値をSystem Answer G3で監視、アラート発生時のログ取得を実施

REST API
→コンソール経由でコマンド実行するリクエスト

シリアル通信
→show slot xx info を実行



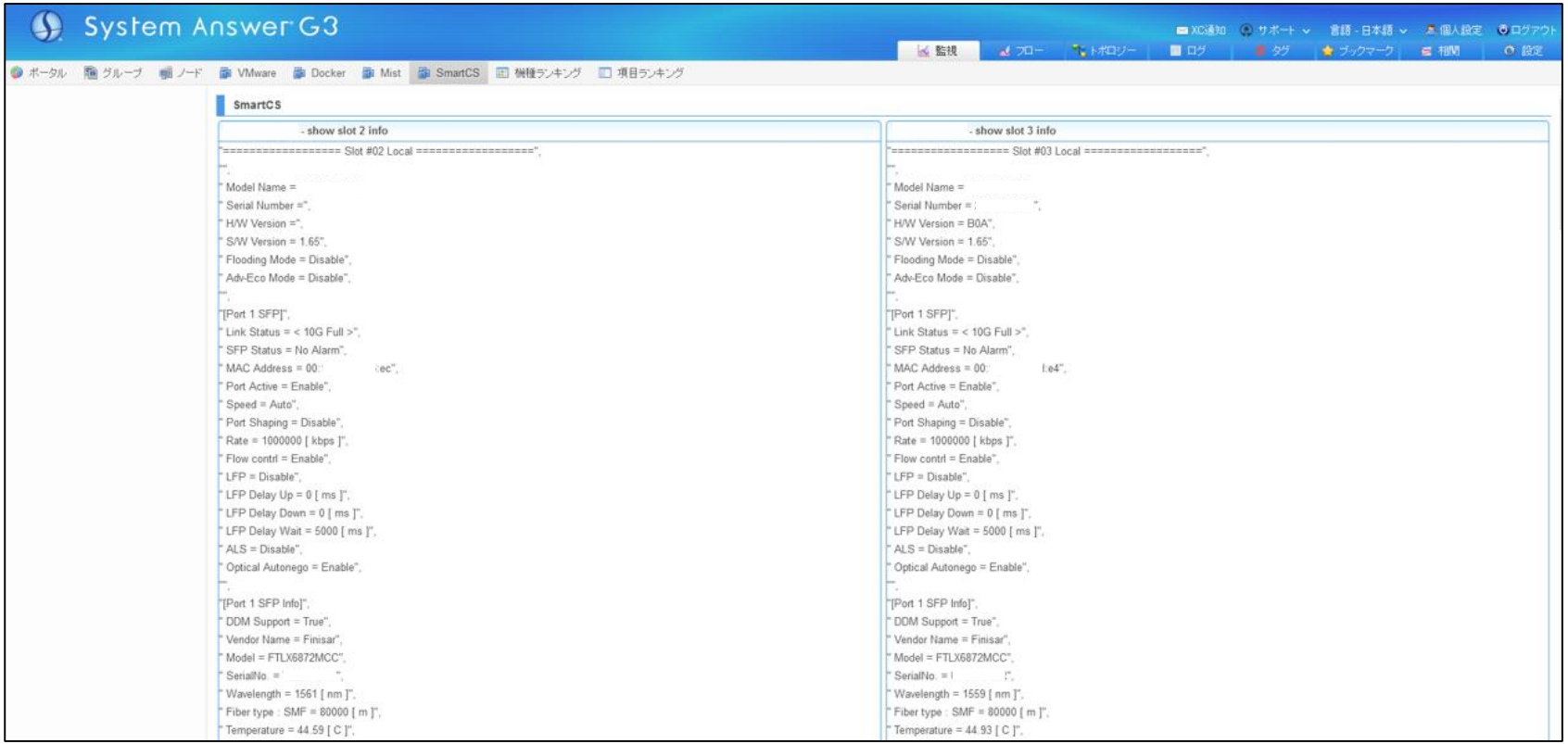
ShowNet2023での取り組み(IBC様/System Answer G3連携)

- SFP受光レベル 取得データのグラフ

ShowNet2023での取り組み(IBC様/System Answer G3連携)

SEIKO

- SFP受光レベル 取得コマンド実行結果

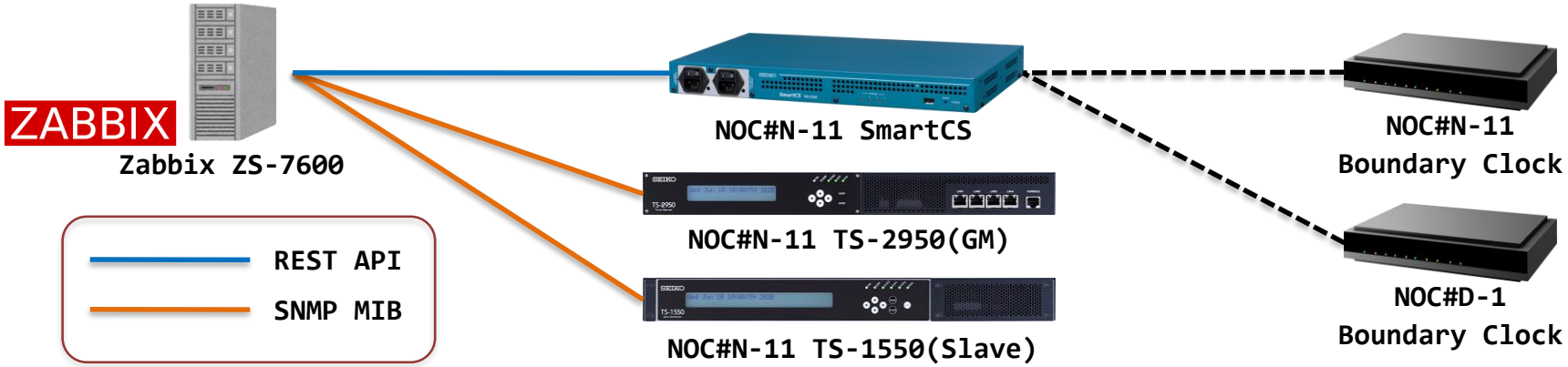


※SmartCSタブでのコマンド実行結果表示機能は、ShowNet2023限定機能です。

ShowNet2023での取り組み(Zabbix Japan様/Zabbix連携)

SEIKO

- Media over IP / PTP関連情報の監視
  SNMP MIBで取得できないPTP関連情報を、Boundary Clockのconsoleから取得
  SNMP監視可能なGM/Slaveと併せて、統合的な監視を実現

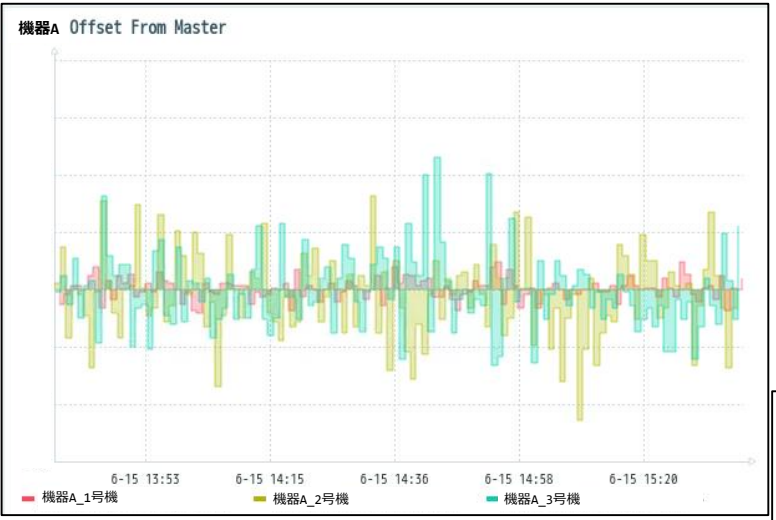


①PTP関連情報取得
(REST API)

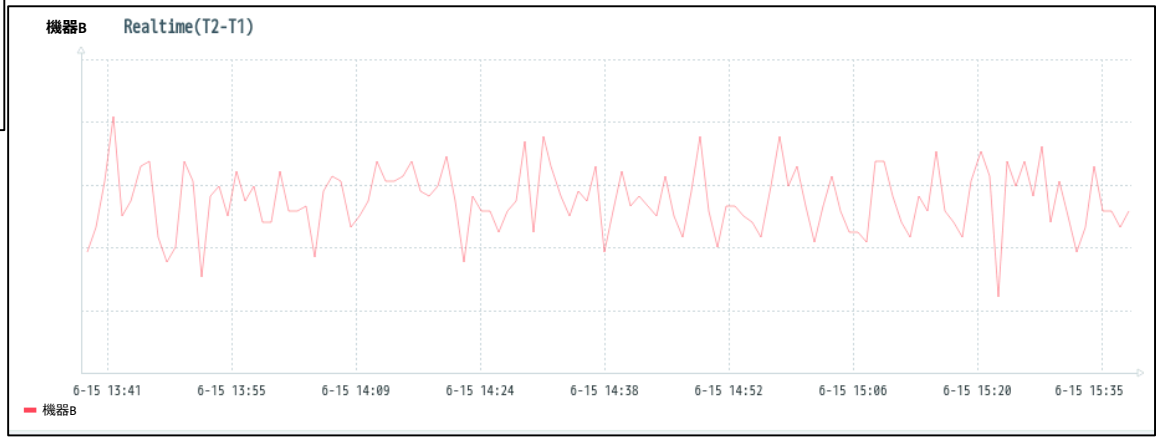
②トポロジ表示/グラフ表示/障害表示
(Zabbix)

ShowNet2023での取り組み(Zabbix Japan様/Zabbix連携)

- PTP offset from Master(PTP MasterとBCの時刻差)データのグラフ



・Boundary ClockとPTP Masterのoffset情報をグラフ化
・SNMP MIBでサポートしていないためSmartCS経由でconsoleから取得



Zabbix経由でSmartCS接続機器から情報を取得する

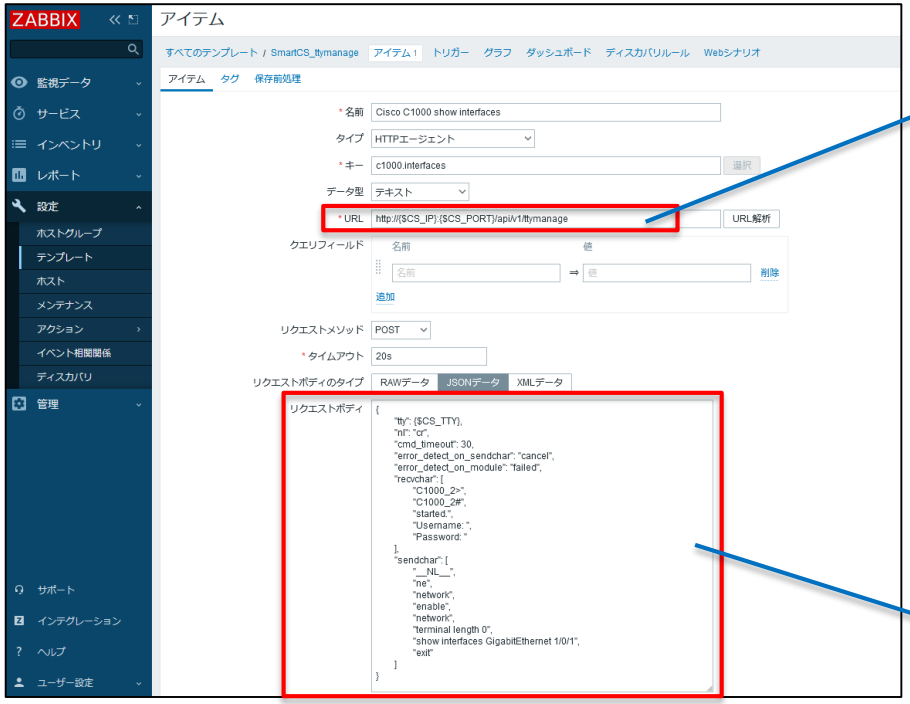
■テンプレート内のマクロ設定

Zabbix経由でSmartCS接続機器から情報を取得する

SEIKO

ZABBIX

■テンプレート内のアイテム設定




```
http://{$CS_IP}:{$CS_PORT}/api/v1/ttymanage
・ttymanage用のURLを指定
・IPやポートはマクロで指定した値を使用
```

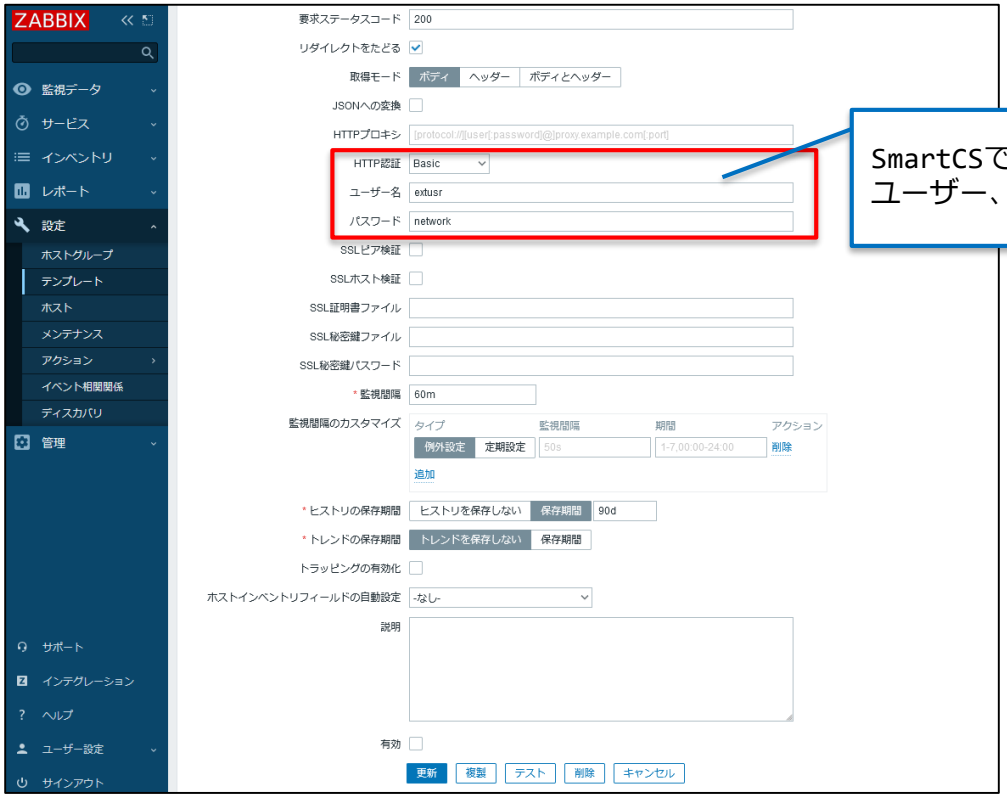
```
{
    "tty": {$CS_TTY},
    "nl": "cr",
    "cmd_timeout": 30,
    "error_detect_on_sendchar": "cancel",
    "error_detect_on_module": “failed",
    "recvchar": [
        "C1000_2>",
        "C1000_2#",
        "started.",
        "Username:",
        "Password:"
    ],
    "sendchar": [
        "__NL__",
        "ne",
        "network",
        "enable",
        "network",
        "terminal length 0",
        "show interfaces GigabitEthernet 1/0/1",
        "exit"
    ]
}
```

sendchar送信後に
待ち受ける文字列

送信する文字列

Zabbix経由でSmartCS接続機器から情報を取得する

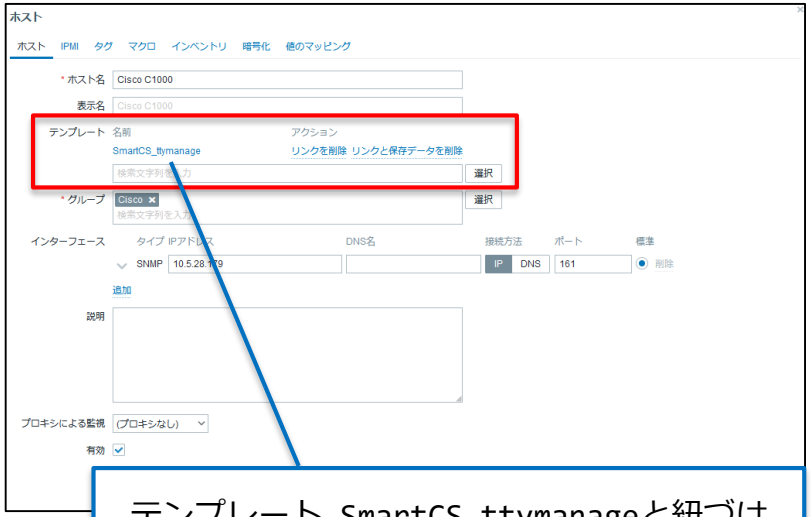
■テンプレート内のアイテム設定



SmartCSで作成した拡張ユーザーグループ(extusr)のユーザー、パスワードを指定

Zabbix経由でSmartCS接続機器から情報を取得する

■ホスト設定



テンプレート SmartCS_ttymanageと紐づけ
テンプレート内でマクロやアイテム設定を実施

■マクロ(変数)設定

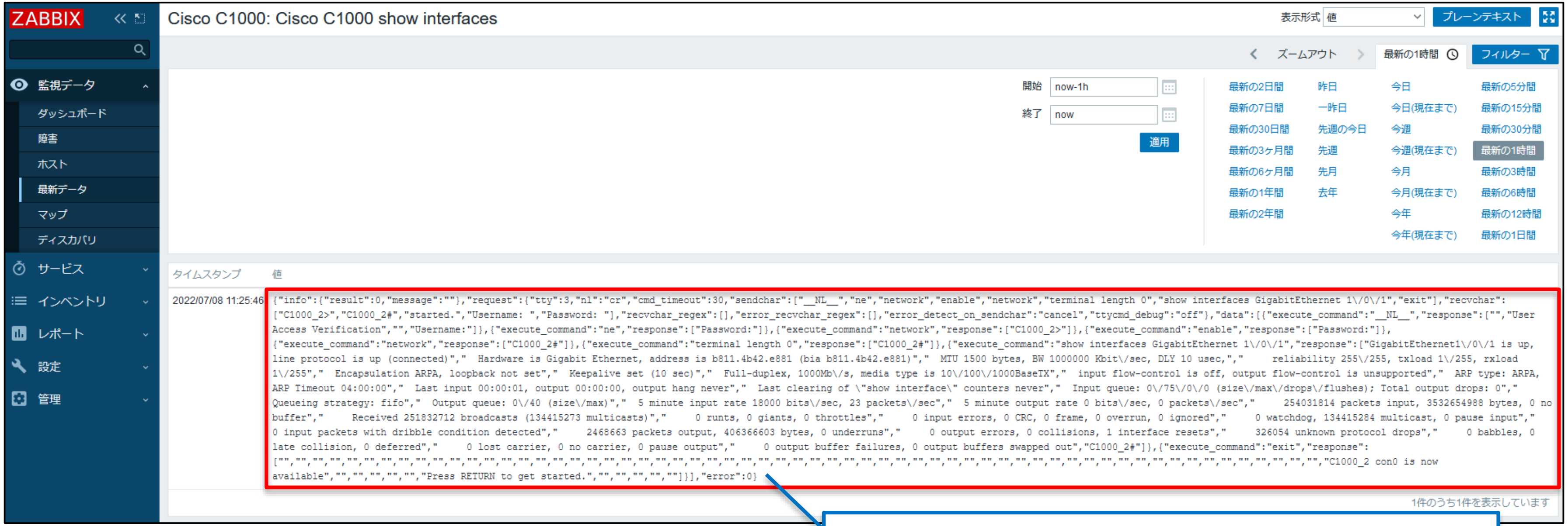
ホストごとに変数の変更が可能

テンプレート内でマクロ設定を継承
(SmartCSのIP、tty番号など)

SEIKO

ZABBIX

■実行結果



{"info":{"result":0,"message":""},"request":{"tty":3,"nl":"cr","cmd_timeout":30,"sendchar":["__NL__","ne","network","enable","network","terminal length 0","show interfaces GigabitEthernet 1\/0\/1","exit"],"recvchar":["C1000_2>","C1000_2#","started.","Username: ","Password: "],"recvchar_regex":[],"error_recvchar_regex":[],"error_detect_on_sendchar":"cancel","ttycmd_debug":"off"},"data":[{"execute_command":"__NL__","response":["","User Access Verification","","Username:"]},{"execute_command":"ne","response":["Password:"]},{"execute_command":"network","response":["C1000_2>"]},{"execute_command":"enable","response":["Password:"]},{"execute_command":"network","response":["C1000_2#"]},{"execute_command":"terminal length 0","response":["C1000_2#"]},{"execute_command":"show interfaces GigabitEthernet 1\/0\/1","response":["GigabitEthernet1\/0\/1 is up, line protocol is up (connected)"," Hardware is Gigabit Ethernet, address is b811.4b42.e881 (bia b811.4b42.e881)"," MTU 1500 bytes, BW 1000000 Kbit\/sec, DLY 10 usec,"," reliability 255\/255, txload 1\/255, rxload 1\/255"," Encapsulation ARPA, loopback not set"," Keepalive set (10 sec)"," Full-duplex, 1000Mb\/s, media type is 10\/100\/1000BaseTX"," input flow-control is off, output flow-control is unsupported"," ARP type: ARPA, ARP Timeout 04:00:00"," Last input 00:00:01, output 00:00:00, output hang never"," Last clearing of \"show interface\" counters never"," Input queue: 0\/75\/0\/0 (size\/max\/drops\/flushes); Total output drops: 0"," Queueing strategy: fifo"," Output queue: 0\/40 (size\/max)"," 5 minute input rate 18000 bits\/sec, 23 packets\/sec"," 5 minute output rate 0 bits\/sec, 0 packets\/sec"," 254031814 packets input, 3532654988 bytes, 0 no buffer"," Received 251832712 broadcasts (134415273 multicasts)"," 0 runts, 0 giants, 0 throttles"," 0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored"," 0 watchdog, 134415284 multicast, 0 pause input"," 0 input packets with dribble condition detected"," 2468663 packets output, 406366603 bytes, 0 underruns"," 0 output errors, 0 collisions, 1 interface resets"," 326054 unknown protocol drops"," 0 babbles, 0 late collision, 0 deferred"," 0 lost carrier, 0 no carrier, 0 pause output"," 0 output buffer failures, 0 output buffers swapped out","C1000_2#"]},{"execute_command":"exit","response":["","C1000_2 con0 is now available","","","","","","Press RETURN to get started.","","","","",""]}],"error":0}

作成したアイテム Cisco C1000 show interfaces の
最新データに実行結果(JSON)が格納される

Zabbix is a registered trademark of Zabbix LLC

Zabbix経由でSmartCS接続機器から情報を取得する

SEIKO

ZABBIX

■スクリプト設定から実行

Zabbixサーバー上で実行するコマンドを記載

```
curl -u extusr:network -X POST http://{$CS_IP}:{$CS_PORT}/api/v1/ttymanage -d @/tmp/cisco_show_interfaces_ge101.json | jq
```

/tmp/cisco_show_interfaces_ge101.json

```
{
    "tty": 3,
    "nl": "cr",
    "cmd_timeout": 30,
    "error_detect_on_sendchar": "cancel",
    "error_detect_on_module": "failed",
    "recvchar": [
        "C1000_2>",
        "C1000_2#",
        "started.",
        "Username:",
        "Password:"
    ],
    "sendchar": [
        "__NL__",
        "ne",
        "network",
        "enable",
        "network",
        "terminal length 0",
        "show interfaces GigabitEthernet 1/0/1",
        "exit"
    ]
}
```

Ansible playbookをJSONに変換する

SEIKO

- SmartCSxAnisble連携で利用しているPlaybookが既に存在する場合、yamlファイルからJSONファイルを生成することが可能です。
- 変換用サイト
  https://web-toolbox.dev/tools/json-yaml-converter

■Ansible Playbook(YAML)

```yaml
tasks:
  - name: show running config
    seiko.smartcs.smartcs_tty_command:
      tty: '1'
      error_detect_on_module: failed
      recvchar:
      - 'Password: '
      - '>'
      - '#'
      sendchar:
      - '__NL__'
      - 'network'
      - 'enable'
      - 'network'
      - 'show running-config interface'
      - 'GigabitEthernet 0/1'
```

■リクエストボディ(JSON)

```json
{
  "tty": "1",
  "error_detect_on_module": "failed",
  "recvchar": [
    "Password: ",
    ">",
    "#"
  ],
  "sendchar": [
    "__NL__",
    "network",
    "enable",
    "network",
    "show running-config interface",
    "GigabitEthernet 0/1"
  ]
}
```

参考情報

- ShowNet2022 ShowNetスタジオ
  運用自動化に効く！コンソールサーバー SmartCSで自動化の第一歩を！
  https://www.youtube.com/watch?v=za-oTV13w48

- SmartCSのAPI対応の解説動画（show intチャンネル）
  https://www.youtube.com/watch?v=UfkZRWGU0CY&t=3s

- SmartCSxRESTAPIの使用例(show int様より)
  https://github.com/taijiji/SmartCS_API