

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
- ・ SmartCSと運用自動化
 - REST API機能概要
 - URL、APIリソース
 - ユースケース例とcurlコマンド実行例
- ・ 参考情報



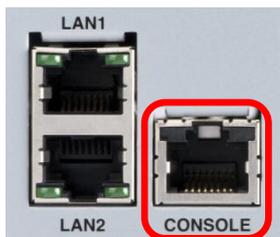
SmartCSの概要

- ・ 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



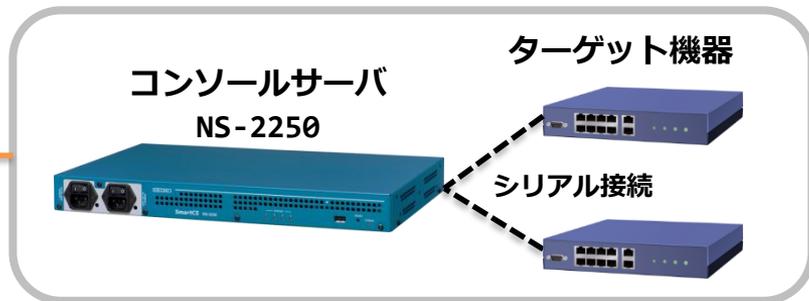
DB9

- 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、ファームウェアアップデート など

オペレーションセンター



遠隔地のDC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ェア 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/>

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

- ・ 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**

ターゲット機器へのコマンド実行

- ・ 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

- 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

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

- 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

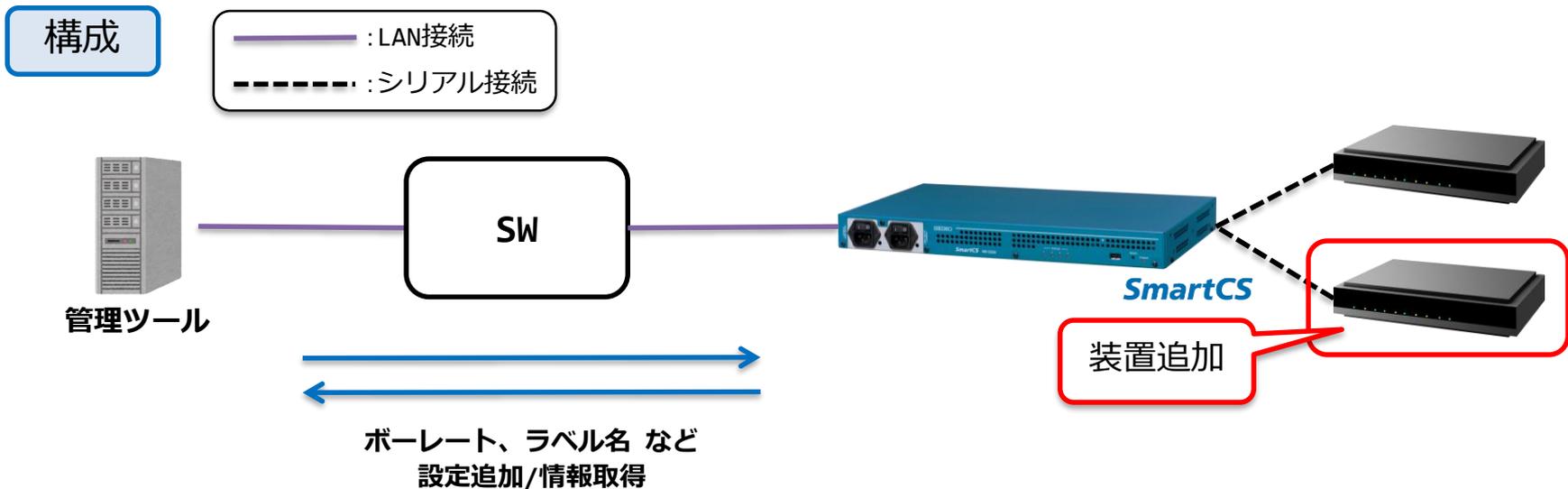
```
# enable http/https
```

- ttyマネージ機能の有効化

```
# enable ttymanage
```

＜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

● 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,
    "message": ""
  },
  "ttylist": [
    {
      "tty": 1,
      "config": {
        "baud": 9600,
        "bitchar": 8,
        "parity": "none",
        "stop": 1,
        "flow": "none",
        "detect_dsr": "off",
        "label": "sw_01"
      }
    }
  ]
}
```

} 共通データ

} レスポンスデータ

```

"status": {
  "DSR": "off",
  "CTS": "off",
  "DTR": "on",
  "RTS": "on",
  "CD": "off"
},
"stats": {
  "TX_Octets": 1925,
  "RX_Octets": 32180,
  "Error_Parity": 0,
  "Error_Framing": 0,
  "Error_Overrun": 0,
  "Break_Count": 0
}
}
]
```

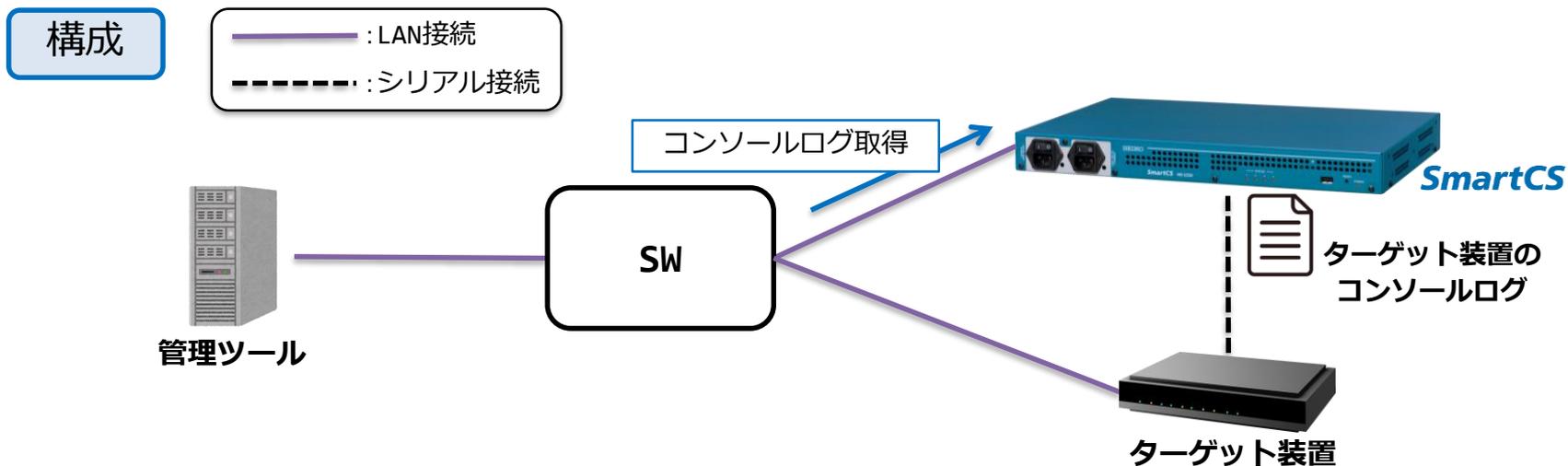
} レスポンスデータ

※実行例は整形して記載しています

<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

● 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
```



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)"
  ]
}
```

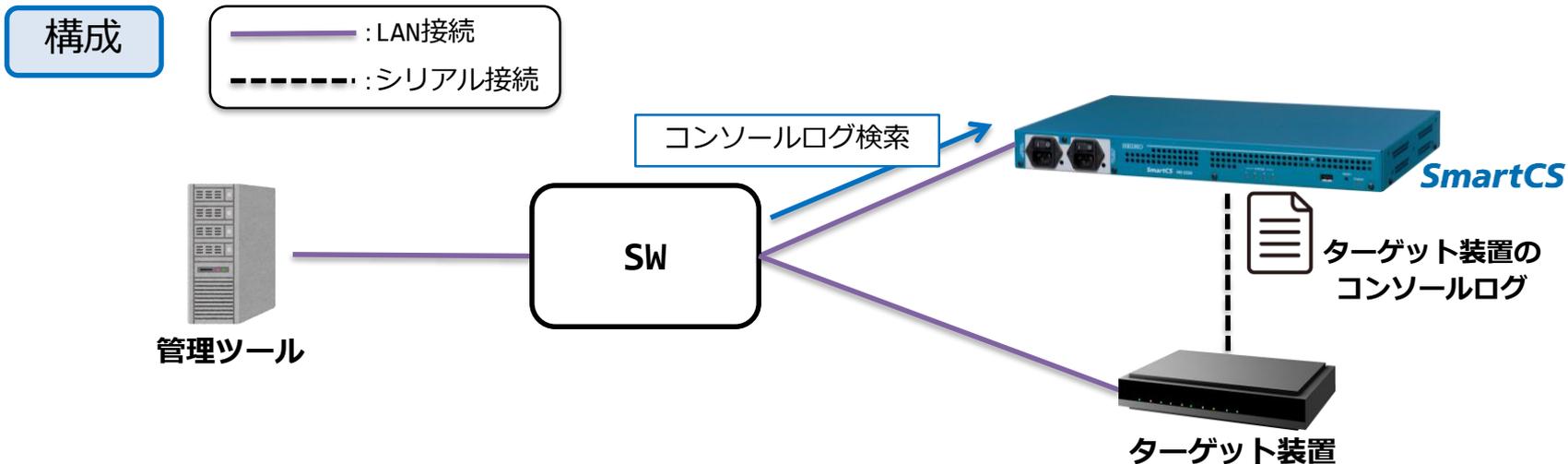
共通データ

レスポンスデータ

※実行例は整形して記載しています

<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

● 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."
    ]
  ]
}
```

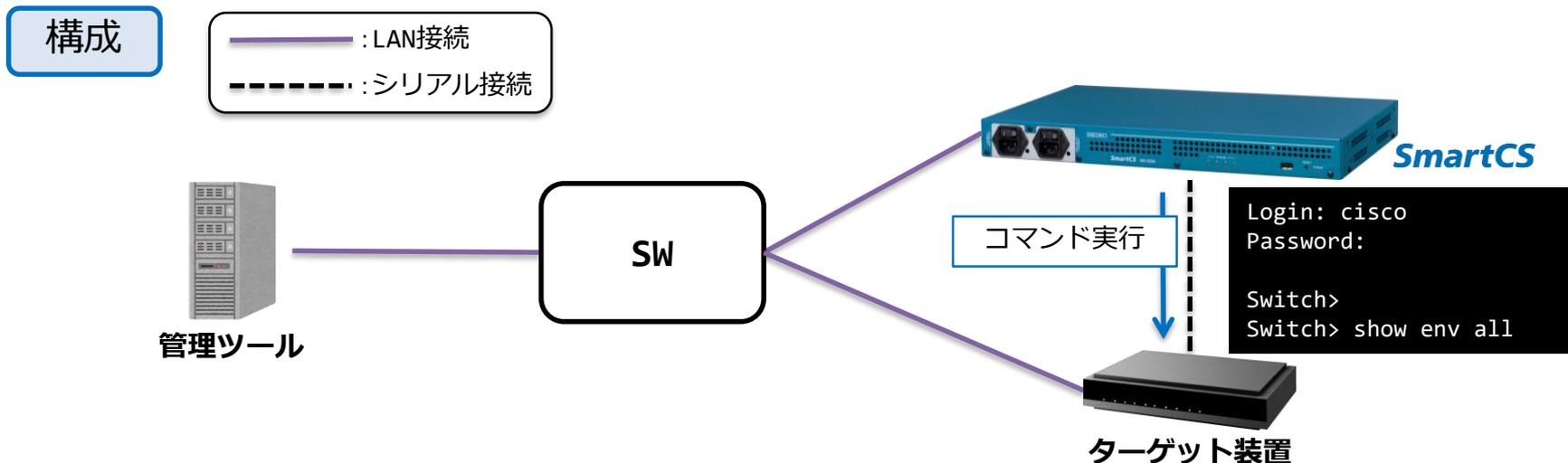
共通データ

レスポンスデータ

※実行例は整形して記載しています

<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

- 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

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~~

```

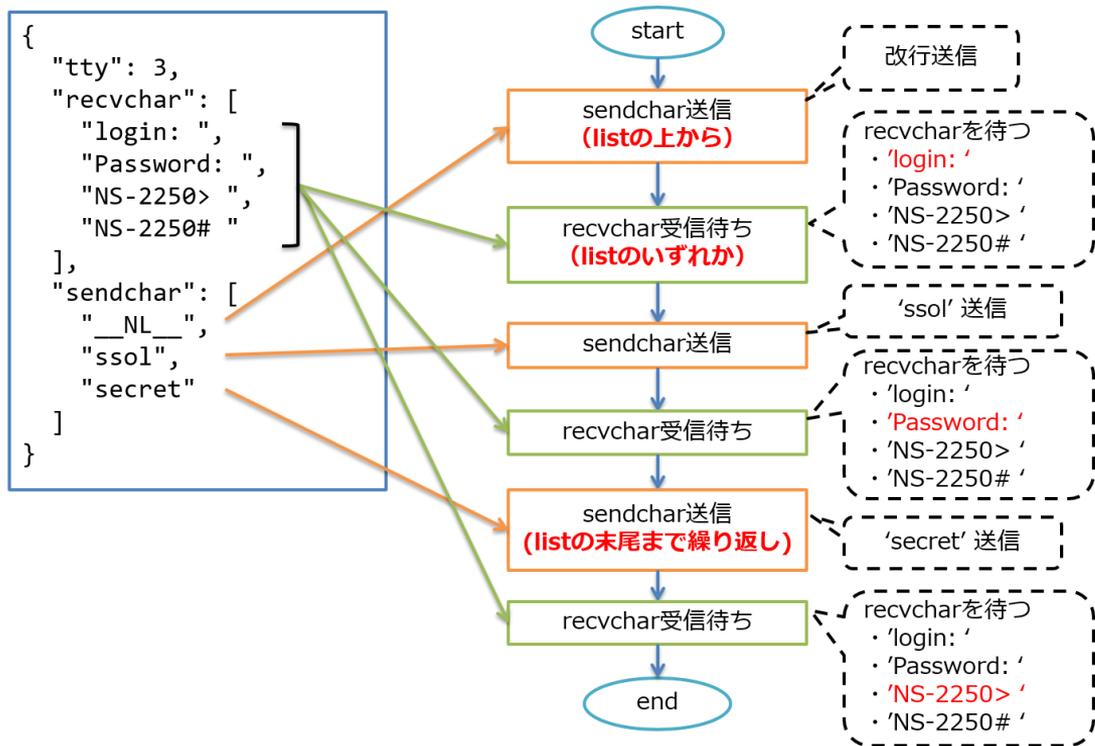
```

{
  "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
}

```

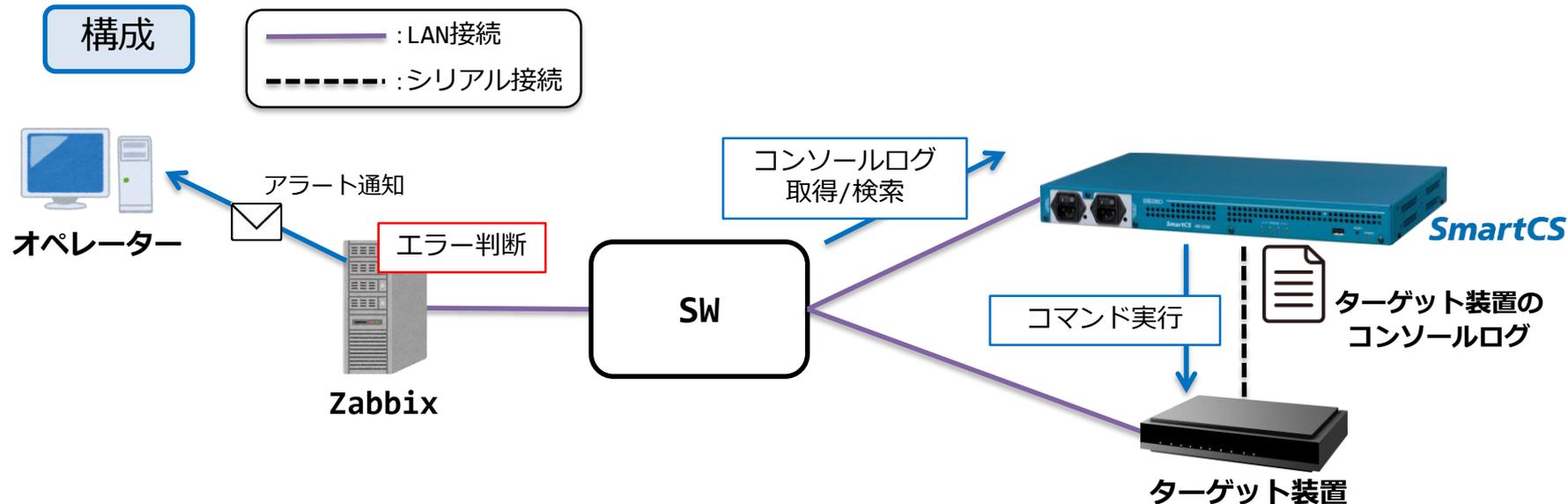
※レスポンスデータの一部(data)については内容を省略しています。
data部分に、実際のシリアル通信のオペレーションが格納されます。

- sendchar、recvcharの動作イメージ



<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**。



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ります。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 }

- 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 <ul style="list-style-type: none"> ・ __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。

- 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。 <ul style="list-style-type: none">・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)

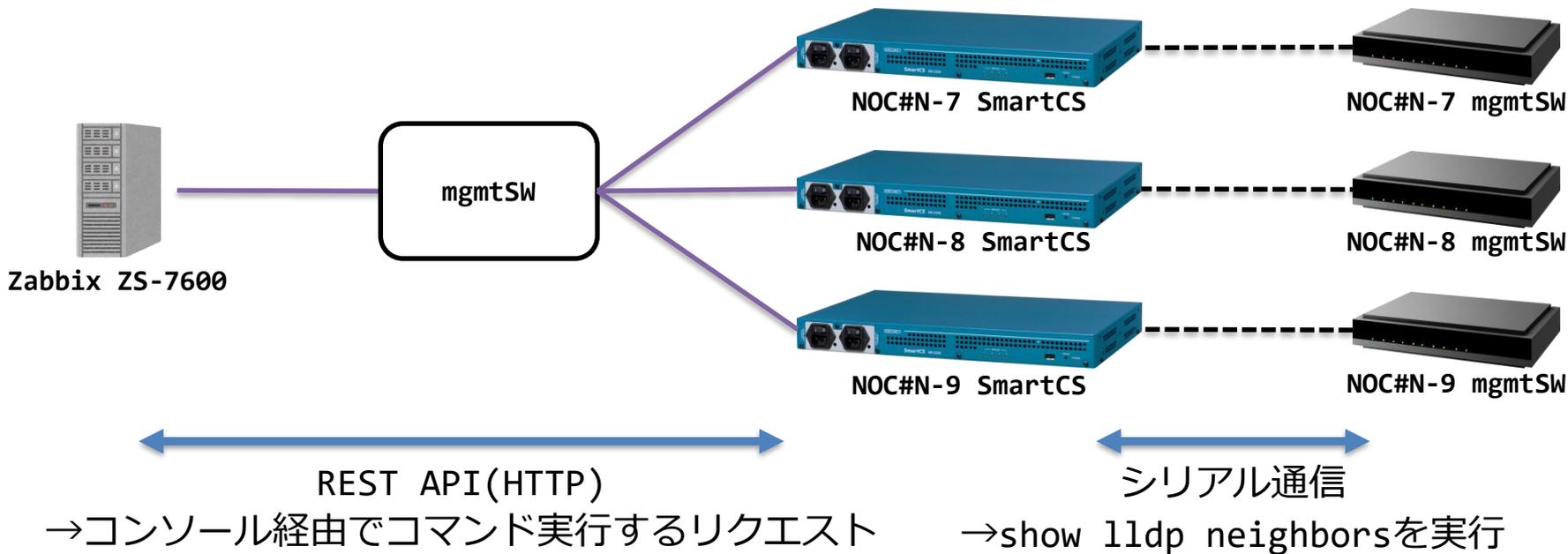
Zabbixアプライアンス(ZS-7600)と連携

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



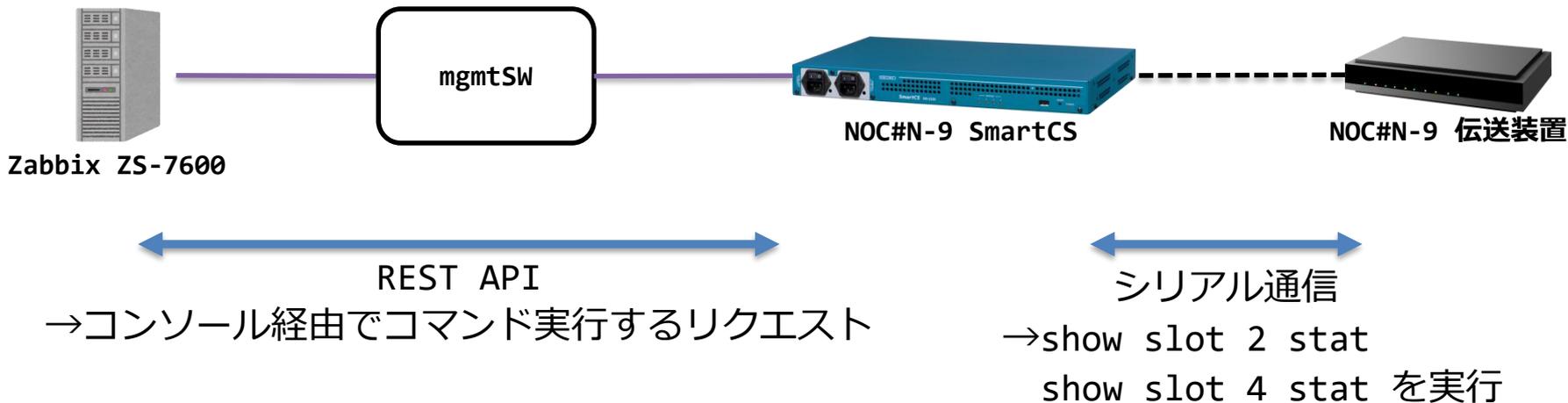
Zabbix ダッシュボード

The screenshot displays the Zabbix web interface for a console server. The left sidebar contains navigation options like '監視データ', 'ダッシュボード', '障害', 'ホスト', '最新データ', 'マップ', 'サービス', 'インベントリ', 'レポート', 'サポート', 'インテグレーション', 'ヘルプ', 'ユーザー設定', and 'サインアウト'. The main content area shows three panels, each displaying the output of a 'show lldp neighbors from console(trapper)' command for a specific switch:

- mgmtSW-7**: Shows LLDP neighbor information for ports Gi 1/26 and Gi 1/48. Neighbors include 39383250-3834-504A-4E31-3.
- mgmtSW-8**: Shows LLDP neighbor information for ports Gi 1/5, Gi 1/6, and Gi 1/7. Neighbors include GigabitEthernet 1/48 with MAC 3c:00:00:00:00:00.
- mgmtSW-9**: Shows LLDP neighbor information for ports ge-0/1/2.0, ge-0/0/13.0, ge-0/0/23.0, ge-0/0/16.0, ge-0/0/11.0, and ge-0/0/18.0. Neighbors include ESN-3128, Riedel-UTC-128-08-95-f0, GigabitEthernet 1/9, MCM9000-SN11746, and qfx5120.noc.

- 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



Zabbix ダッシュボード

The screenshot shows the Zabbix web interface for a console server. The main content area displays the output of the command `.trans_s: show slot stat from console(trapper)`. Two specific slot statistics are highlighted with red boxes and labeled with blue callouts: Slot 2 and Slot 4.

Slot 2 Statistics:

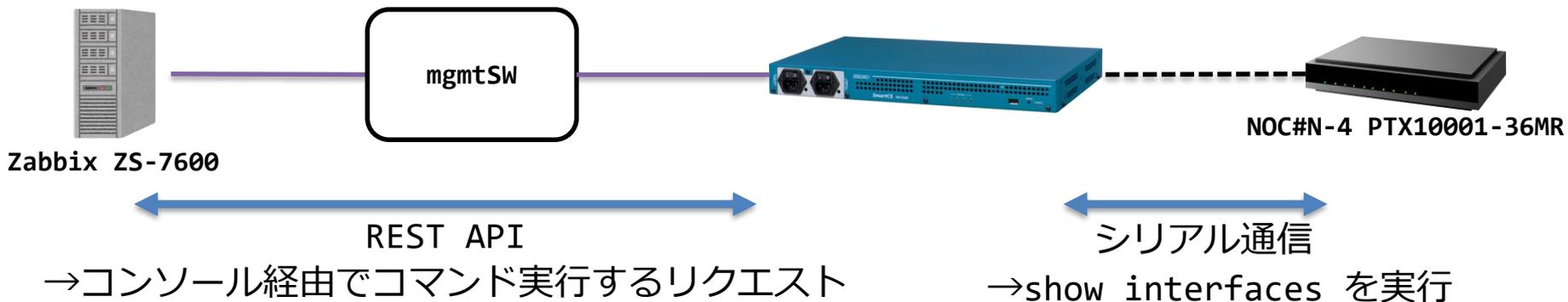
Statistics	Received	Transmitted
Packets	191372942	91309250
Bytes	180738918506	70302522217
Errors	0	0
Drops	0	0
Filtered	391619	-

Slot 4 Statistics:

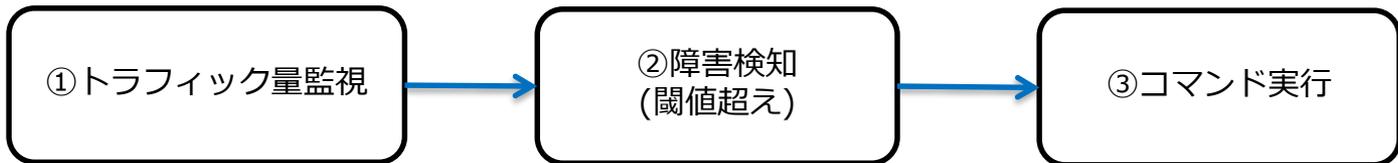
Statistics	Received	Transmitted
Packets	58270580	24814611
Bytes	41843499376	11414644980
Errors	0	0
Drops	0	0
Filtered	45847	-

- 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。



フロー



Zabbix ダッシュボード

The screenshot displays the Zabbix web interface. On the left is a navigation sidebar with options like '監視データ', 'ダッシュボード', '障害', 'ホスト', '最新データ', 'マップ', 'ティスカバリ', 'サービス', 'インベントリ', 'レポート', '設定', '管理', 'サポート', 'インテグレーション', 'ヘルプ', 'ユーザー設定', and 'サインアウト'. The main content area is titled 'コンソールサーバー' and shows a search bar with 'show interfaces' entered. Below the search bar, there are tabs for 'mgmtSW', 'lldp neighbors', 'slot stat', and 'Juniper PTX10001 show interfaces'. A blue callout box with the text 'show interfaces 実行結果' points to the command output. The output shows details for the physical interface 'et-0/0/0' and the logical interface 'ptx10k.noc_s'. The physical interface is up and has a speed of 400Gbps. The logical interface is also up and has a speed of 9000. The output includes various statistics such as input/output rates, error rates, and FEC statistics.

```
ptx10k.noc_s: show interfaces from console(trapper)
タイムスタンプ 値
2022/06/15 17:15:34 show interfaces et-0/0/0
Physical interface: et-0/0/0, Enabled, Physical link is Up
Interface index: 1054, SNMP ifIndex: 521
Description: hg-1-0-1-ne8000-rx4.noc
Link-level type: Ethernet, MTU: 16000, LAN-PHY mode, Speed: 400Gbps,
BFD Error: None, Loop Detect FDU Error: None, MAC-REWRITE Error: None,
Loopback: Disabled, Source filtering: Disabled, Flow control: Enabled,
Auto-negotiation: Disabled, Media type: Fiber
Device flags : Present Running
Interface flags: SNMP-Traps
CoS queues : 8 supported, 8 maximum usable queues
Current address: ac: :ce, Hardware address: ac: :ce
Last flapped : 2022-06-15 10:28:49 JST (06:46:37 ago)
Input rate : 84364776048 bps (7250672 pps)
Output rate : 28248214376 bps (2422099 pps)
Active alarms : None
Active defects : None
FCS statistics Seconds
Bit errors 0
Errored blocks 0
Ethernet FEC Mode : FEC119
Ethernet FEC statistics Errors
FEC Corrected Errors 1840342997
FEC Uncorrected Errors 0
FEC Corrected Errors Rate 76993
FEC Uncorrected Errors Rate 0
FRBS Mode : Disabled
Interface transmit statistics: Disabled
Link Degraded :
Link Monitoring : Disable

Logical interface et-0/0/0.3 (Index 7042) (SNMP ifIndex 588)
Flags: Up SNMP-Traps VLAN-Tag[ 0x8100.3 ] Encapsulation: ENET2
Input packets : 171706225662
Output packets: 52479169880
Protocol iso, MTU: 9000
Flags: User-MTU
Protocol inet6, MTU: 9000
Flags: Is-Primary, User-MTU
Addresses, Flags: Is-Preferred Is-Primary
```

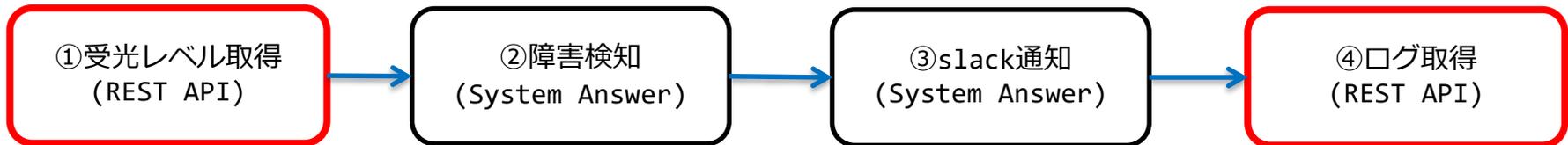
System Answer G3と連携

- 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

- 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



- SFP受光レベル 取得データのグラフ



- SFP受光レベル 取得コマンド実行結果

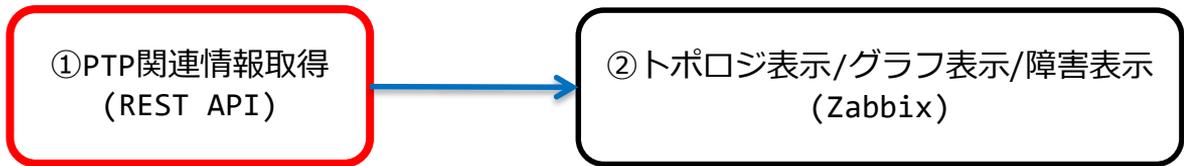
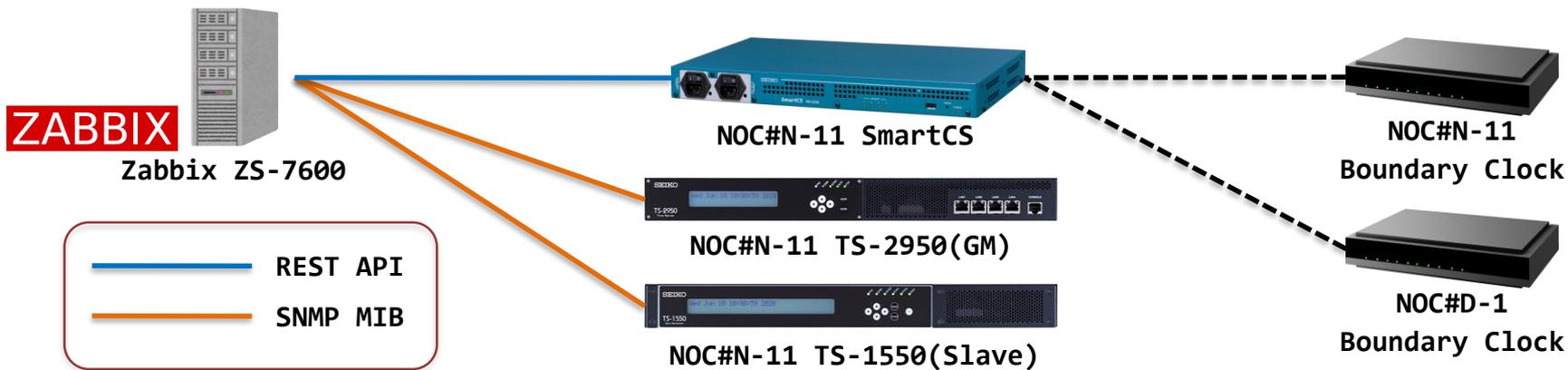
The screenshot displays the SmartCS interface for System Answer G3. The main content area is divided into two panels, each showing the output of a 'show slot' command. The left panel shows Slot #02 Local, and the right panel shows Slot #03 Local. Both panels display detailed SFP information, including model name, serial number, hardware and software versions, and port status. The temperature for Slot 2 is 44.59 [C], and for Slot 3, it is 44.93 [C].

```
- show slot 2 info
----- Slot #02 Local -----
Model Name =
Serial Number =
HW Version =
SW Version = 1.65
Flooding Mode = Disable
Adv-Eco Mode = Disable
[Port 1 SFP]
Link Status = < 10G Full >
SFP Status = No Alarm
MAC Address = 00:
Port Active = Enable
Speed = Auto
Port Shaping = Disable
Rate = 1000000 [ kbps ]
Flow contrl = Enable
LFP = Disable
LFP Delay Up = 0 [ ms ]
LFP Delay Down = 0 [ ms ]
LFP Delay Wait = 5000 [ ms ]
ALS = Disable
Optical Autonego = Enable
[Port 1 SFP Info]
DDM Support = True
Vendor Name = Finisar
Model = FTLX6872MCC
SerialNo =
Wavelength = 1561 [ nm ]
Fiber type : SMF = 80000 [ m ]
Temperature = 44.59 [ C ]

- show slot 3 info
----- Slot #03 Local -----
Model Name =
Serial Number =
HW Version = B0A
SW Version = 1.65
Flooding Mode = Disable
Adv-Eco Mode = Disable
[Port 1 SFP]
Link Status = < 10G Full >
SFP Status = No Alarm
MAC Address = 00:
Port Active = Enable
Speed = Auto
Port Shaping = Disable
Rate = 1000000 [ kbps ]
Flow contrl = Enable
LFP = Disable
LFP Delay Up = 0 [ ms ]
LFP Delay Down = 0 [ ms ]
LFP Delay Wait = 5000 [ ms ]
ALS = Disable
Optical Autonego = Enable
[Port 1 SFP Info]
DDM Support = True
Vendor Name = Finisar
Model = FTLX6872MCC
SerialNo =
Wavelength = 1559 [ nm ]
Fiber type : SMF = 80000 [ m ]
Temperature = 44.93 [ C ]
```

※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。

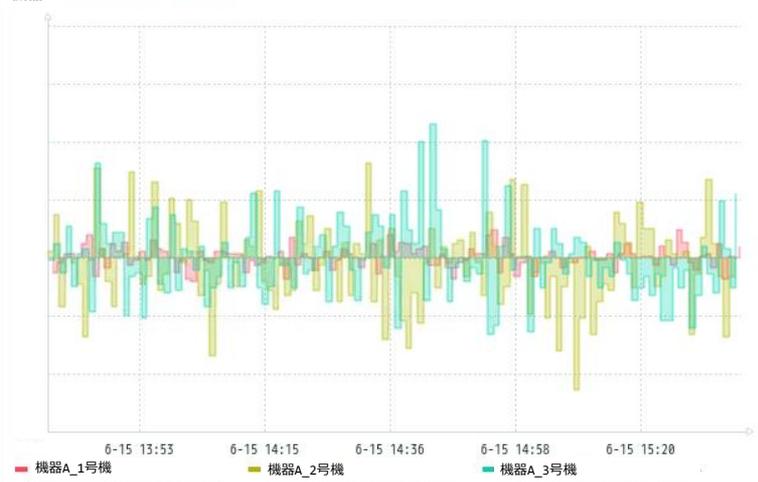
- Media over IP / PTP関連情報の監視
SNMP MIBで取得できないPTP関連情報を、Boundary Clockのconsoleから取得
SNMP監視可能なGM/Slaveと併せて、統合的な監視を実現



Zabbix is a registered trademark of Zabbix LLC

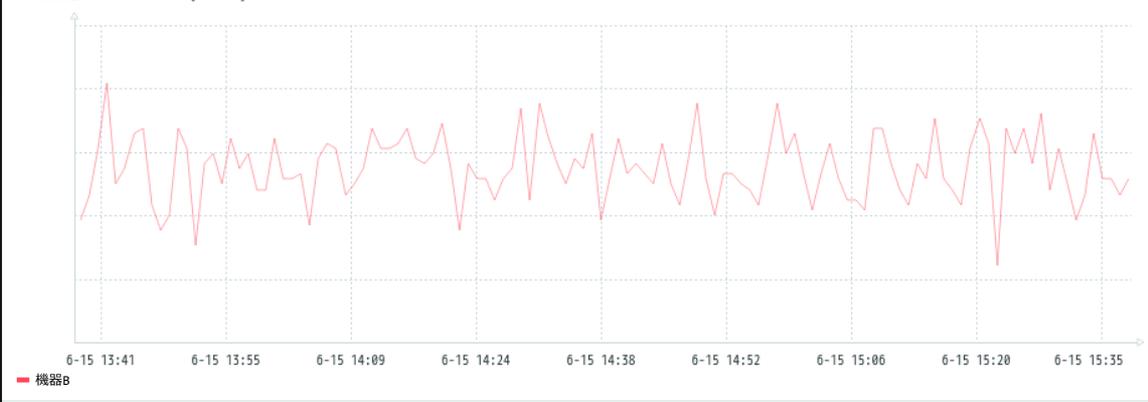
- PTP offset from Master(PTP MasterとBCの時刻差)データのグラフ

機器A Offset From Master



- 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

機器B Realtime(T2-T1)



■ テンプレート内のマクロ設定

テンプレート内のマクロ設定画面のスクリーンショット。マクロ設定表が赤い枠で囲われ、青い吹き出しで「IP、ポート、tty番号などを変数化して登録」と説明されている。

マクロ	値		説明	
{\$CS_IP}	10.5.28.177	T	説明	削除
{\$CS_PORT}	10080	T	説明	削除
{\$CS_TTY}	3	T	説明	削除

追加

更新 複製 すべて複製 削除 削除とクリア キャンセル

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

アイテム設定画面のスクリーンショット。URLフィールドとリクエストボディフィールドが赤い枠で囲われ、それぞれ右側のコードブロックとリンクしている。

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",
    "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 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

■ テンプレート内のアイテム設定

要求ステータスコード 200

リダイレクトをたどる

取得モード ボディ ヘッダー ボディとヘッダー

JSONへの変換

HTTPプロキシ [protocol://user[:password]@proxy.example.com[:port]]

HTTP認証 Basic

ユーザー名 extusr

パスワード network

SSLピア検証

SSLホスト検証

SSL証明書ファイル

SSL秘密鍵ファイル

SSL秘密鍵/パスワード

* 監視間隔 60m

監視間隔のカスタマイズ

タイプ	監視間隔	期間	アクション
例外設定	定期設定	60s	1-7,00:00-24:00
			削除

追加

* ヒストリの保存期間 ヒストリを保存しない 保存期間 90d

* トレンドの保存期間 トレンドを保存しない 保存期間

トラッピングの有効化

ホストインベントリフィールドの自動設定 なし

説明

有効

更新 複製 テスト 削除 キャンセル

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 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

■ ホスト設定

ホスト

ホスト名 Cisco C1000

表示名 Cisco C1000

テンプレート	名前	アクション
SmartCS_ttymanage	SmartCS_ttymanage	リンクを削除 リンクと保存データを削除

グループ Cisco

インターフェース

タイプ	IPアドレス	DNS名	接続方法	ポート	標準	
SNMP	10.5.28.177		IP	DNS	161	削除

説明

プロキシによる監視 (プロキシなし)

有効

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

ホストマクロ 継承したマクロとホストマクロ

マクロ	実効値	テンプレート値	グローバル値 (グローバルマクロ設定)
[\$CS_IP]	10.5.28.177	変更 SmartCS_ttymanage: "10.5.28.177"	
[\$CS_PORT]	10080	変更 SmartCS_ttymanage: "10080"	
[\$CS_TTY]	3	変更 SmartCS_ttymanage: "3"	
[\$SNMP_COMMUNITY]	public	変更	⇐ "public"

追加

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)

Zabbix経由でSmartCS接続機器から情報を取得する

■ 実行結果

The screenshot shows the Zabbix interface for monitoring a Cisco C1000 device. The command executed is 'Cisco C1000 show interfaces'. The result is displayed as JSON data in a red-bordered box:

```

{"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_detect_on_sendchar":"cancel","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":{"","Press RETURN to get started."},"error":0}}

```

The JSON data is highlighted in a red box in the screenshot. A blue arrow points from the text box below to the JSON data.

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経由でSmartCS接続機器から情報を取得する

■ 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
```

スクリーンショット: Zabbixの「スクリプト」設定画面。コマンド欄に以下のコマンドが記載されている。

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

その他の設定: 名前: cisco show interfaces, タイプ: Webhook, スクリプト, SSH, Telnet, IPMI, 次で実行: Zabbixエージェント, Zabbixサーバーまたはプロキシ, Zabbixサーバー。

```
/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"
  ]
}
```

Zabbix is a registered trademark of Zabbix LLC

Ansible playbookをJSONに変換する

- SmartCSxAnsible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)

```
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)

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

- 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