Specific Simple Network Management Tools

Jürgen Schönwälder

University of Osnabrück
Albrechtstr. 28
49069 Osnabrück
Germany

Email: <schoenw@informatik.uni-osnabrueck.de>
Web: <http://www.informatik.uni-osnabrueck.de/>

SNMP in a Nutshell

- The Simple Network Management Protocol (SNMP) is used to access and manipulate typed variables organized in conceptual tables or groups of scalars.
- The semantics of the variables are specified in MIB modules which are written in the SMI data definition language (Structure of Management Information).
- Each variable (either a scalar or a cell in a conceptual table) is uniquely identified by an OID value (a sequence of numbers defining a path in a global registration tree).
- SNMP operates on a (lexicographically) ordered list of variables (varbind list).
- Each varbind list element contains an OID value identifying a variable and it’s value.
- SNMP usually runs over UDP (stateless, retransmision control).

Generic vs. Specific SNMP Tools

- What we have available today:
  1. Generic low-level SNMP tools (snmpget, snmpwalk, ...)
  2. Generic low-level SNMP APIs (WinSnmp, SNMP++, Tnm, ...)
  3. Generic MIB browser (sgmospy, tkmib, sbrowser, ...)
  4. Generic monitoring tools (mrtg, ...)
  5. Generic management platforms (OpenView, Spectrum, ...)
- What we really wanted:
  - Specific SNMP tools that focus on doing one thing and doing it right

SNMP Command Line Interface (scli)

- Features:
  - Command line interface that works on devices produced by different vendors
  - Runs locally and communicates with the device using standard SNMP interactions
  - Commands are structured in a hierarchy (recursive command evaluation)
  - Related commands are logically grouped into modes
  - Default output format is optimized for human readability
  - Selecting objects using names and regular expressions
  - Support for simple online monitoring activities
  - Command line editing, command history and command aliases
  - Additional XML output format which is optimized for machine readability

show entity containment

local

```
Agent Boot Time:  2001-09-24 17:21:51 +02:00
Interfaces:       12
Bridge Type:      source route transparent (SRT)
(ciscobs.rz) scli > show entity containment
ENTITY CLASS     CONTAINMENT
     1 chassis     7206VXR chassis, Hw Serial#: 21275454, Hw Revision: D
     2 module      |- NPE 300 Card, Hw Serial#: 21275454, Hw Revision: D
     3 container   |- Chassis Slot
     4 module      |  `- I/O FastEthernet (TX-ISL)
     5 port        |     `- DEC21140A
     6 container   |- Chassis Slot
     7 module      |  `- 2 Port Fast Ethernet/ISL 100BaseTX Port Adapter
     8 port        |     |- AmdFE
     9 port        |     `- AmdFE
    10 container   |- Chassis Slot
    11 module      |  `- POS Port Adapter (SM)
    12 port        |     `- Packet over Sonet
    13 container   |- Chassis Slot
    14 container   |- Chassis Slot
    15 module      |  `- ATM Lite Port Adaptor (SM)
    16 port        |     `- TI1570 ATM
    17 container   |- Chassis Slot
    18 container   `- Chassis Slot
(ciscobs.rz) scli >
```

monitor interface stats

local

```
Agent:    ciscobs.rz:161 up 9 days 23:34:35                              15:56:26
Descr:    Cisco Internetwork Operating System Software   IOS (tm) 7200 Software
IPv4:     6435 pps in 6408 pps out 6399 pps fwd    0 pps rasm    0 pps frag
UDP:         8 pps in    6 pps out
TCP:         0 sps in    0 sps out    0 con est    0 con aopn    0 con popn
Command:  monitor interface stats
```

INTERFACE	STATUS	I-BPS	O-BPS	I-PPS	O-PPS	I-ERR	O-ERR	DESCRIPTION
1	UUCN	1m	2m	3270	3152	0	0	FastEthernet0/0
2	UUCN	0	23	0	0	0	0	FastEthernet1/0
3	UUCN	10k	10k	50	50	0	0	FastEthernet1/1
4	UUCN	2m	1m	3197	3254	0	0	POS2/0
5	UDCN	0	0	0	0	0	0	ATM4/0
6	UD--	----	----	----	----	----	----	ATM4/0-atm layer
7	UD--	----	----	----	----	----	----	ATM4/0.0-atm subif
8	UDNN	0	----	0	----	----	----	ATM4/0-aal5 layer
9	UDNN	0	----	0	----	----	----	ATM4/0.0-aal5 layer
10	UUNN	0	2105	0	18	0	0	Null0
11	UUNN	0	0	0	0	0	0	Loopback0
12	UUNN	0	0	0	0	0	0	Tunnel34

scli interface mode

```
set interface status <regexp> <status>
set interface alias <regexp> <string>
set interface notifications <regexp> <value>
set interface promiscuous <regexp> <bool>

show interface info [<regexp>]
show interface details [<regexp>]
show interface stack [<regexp>]
show interface stats [<regexp>]

monitor interface stats [<regexp>]

dump interface
```

- Provides commands to configure, display and monitor network interfaces.
- Interfaces are identified by regular expressions matched against names.

scli nortel mode

```
create nortel bridge vlan <vlanid> <name>
delete nortel bridge vlan <regexp>

set nortel bridge vlan ports <regexp> <ports>
set nortel bridge vlan default <string> <ports>

show nortel bridge vlan info [<regexp>]
show nortel bridge vlan details [<regexp>]
show nortel bridge vlan ports

dump nortel bridge vlan
```

- Provides commands to configure and display VLAN configurations on Nortel bridges.
- VLANs are identified by regular expressions matched against names.
- Port lists can be specified in a human readable easy to use format.

Configuring VLANs using `scli` and `m4`

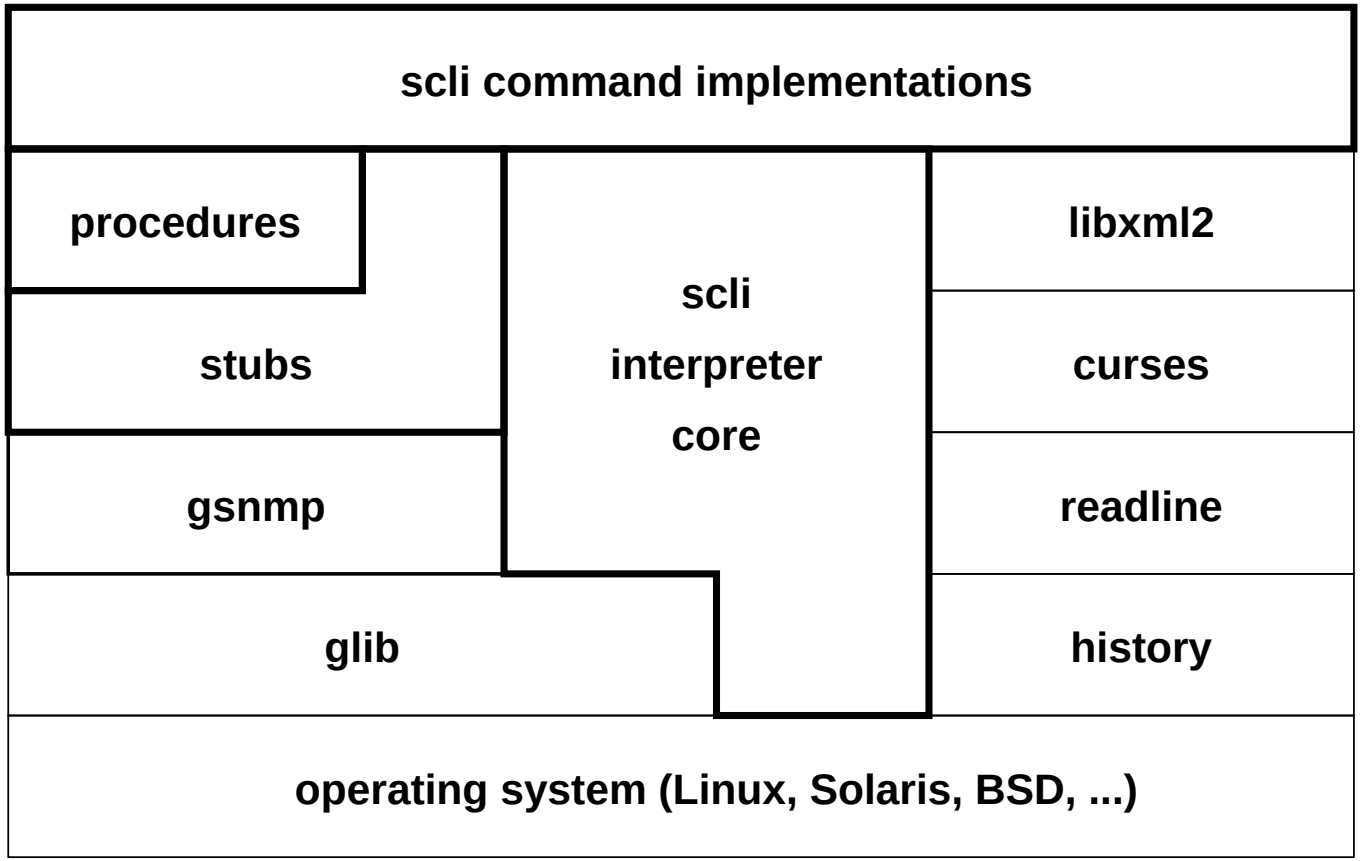
```
delete nortel bridge vlan "^(134|ibr-)"     # regexps are cool :-)

create nortel bridge vlan 544 ibr-core
create nortel bridge vlan 545 ibr-cip
create nortel bridge vlan 546 ibr-test
create nortel bridge vlan 547 ibr-wlan
```

```
define(UP,‘25,185’)                 # uplink ports
define(WLAN,‘2,56’)                 # wireless vlan
define(CORE,‘1,3-24,33-55,65-88’) # core vlan

include(vlan-all.scli)              # create the vlans

set nortel bridge vlan ports ibr-core UP,CORE
set nortel bridge vlan default ibr-core CORE
set nortel bridge vlan ports ibr-wlan UP,WLAN
set nortel bridge vlan default ibr-wlan UP,WLAN
```

Software Design

- Extensibility:
  - Make it relatively easy to add new features (for the average programmer)
  - Hide low-level SNMP communication details as much as possible
- Robustness:
  - Ensure that errors are detected and handled gracefully where possible
  - Abort if coders forget to check for error conditions
- Maintainability:
  - The software will evolve over time, which includes internal API changes
  - Ensure that the documentation is available and in sync with the implementation
- Efficiency:
  - Fast startup times so that the tools can be used efficiently in scripts
- Portability:
  - The tools should run on all major Unix systems
  - A port to Win32 platforms should be possible (at reasonable costs)

Software Architecture



- The `stubs` are generated from MIB modules by using a specialized MIB compiler.

Stub Code Generator

- Generates stub functions for retrieving and/or modifying
  - complete conceptual tables
  - rows of conceptual tables
  - groups of scalars
- Hides all low-level SNMP details such as OID naming and (de-)serialization
- Additional features of the generated code:
  - type checking
  - automatic packing and unpacking of complex instance identifiers
  - handling of so called “holes” in conceptual tables
- Generated data structures use pointers (forces programmers to validate data members)
- Implementation integrated into `smidump`, available as part of the `libsmi` package

Stub Interface for Printer-MIB::prtConsoleLightEntry

```
typedef struct {
    gint32   hrDeviceIndex;
    gint32   prtConsoleLightIndex;
    gint32   *prtConsoleOnTime;
    gint32   *prtConsoleOffTime;
    gint32   *prtConsoleColor;
    guchar   *prtConsoleDescription;
    gsize    _prtConsoleDescriptionLength;
} printer_mib_prtConsoleLightEntry_t;

extern void
printer_mib_get_prtConsoleLightTable(GSnmpSession *s,
        printer_mib_prtConsoleLightEntry_t ***prtConsoleLightEntry, gint mask);

extern void
printer_mib_free_prtConsoleLightTable(printer_mib_prtConsoleLightEntry_t **prtConsoleLightEntry);

extern printer_mib_prtConsoleLightEntry_t *
printer_mib_new_prtConsoleLightEntry(void);

extern void
printer_mib_get_prtConsoleLightEntry(GSnmpSession *s,
        printer_mib_prtConsoleLightEntry_t **prtConsoleLightEntry,
        gint32 hrDeviceIndex, gint32 prtConsoleLightIndex, gint mask);

extern void
printer_mib_set_prtConsoleLightEntry(GSnmpSession *s,
        printer_mib_prtConsoleLightEntry_t *prtConsoleLightEntry, gint mask);

extern void
printer_mib_free_prtConsoleLightEntry(printer_mib_prtConsoleLightEntry_t *prtConsoleLightEntry);
```

Command Implementation

```
static int show_printer_console_lights(scli_interp_t *interp, int argc, char **argv)
{
    printer_mib_prtConsoleLightEntry_t **lightTable;
    int i, light_width = 12;

    if (argc > 1) return SCLI_SYNTAX;

    printer_mib_get_prtConsoleLightTable(interp->peer, &lightTable, 0);
    if (interp->peer->error_status) return SCLI_SNMP;

    if (lightTable) {
        for (i = 0; lightTable[i]; i++) {
            if (lightTable[i]->_prtConsoleDescriptionLength > light_width)
                light_width = lightTable[i]->_prtConsoleDescriptionLength;
        }
        if (! scli_interp_xml(interp)) {
            g_string_sprintfa(interp->header, "PRINTER LIGHT %-*s STATUS COLOR",
                                              light_width, "DESCRIPTION");
        }
        for (i = 0; lightTable[i]; i++) {
            if (scli_interp_xml(interp))
                xml_printer_console_light(interp->xml_node, lightTable[i]);
            else
                fmt_printer_console_light(interp->result, lightTable[i], light_width);
        }
    }

    if (lightTable) printer_mib_free_prtConsoleLightTable(lightTable);
    return SCLI_OK;
}
```

Formatting Function

```
static void
fmt_printer_console_light(GString *s, printer_mib_prtConsoleLightEntry_t *lightEntry, int light_width)
{
    const char *state = "off", *e;

    g_string_sprintfa(s, "%6d  ", lightEntry->hrDeviceIndex);
    g_string_sprintfa(s, "%4d  ", lightEntry->prtConsoleLightIndex);

    if (lightEntry->prtConsoleDescription) {
        g_string_sprintfa(s, "%-*.*s ", light_width,
                   (int) lightEntry->_prtConsoleDescriptionLength,
                         lightEntry->prtConsoleDescription);
    } else {
        g_string_sprintfa(s, "%*s", light_width, "");
    }

    if (*lightEntry->prtConsoleOnTime && !*lightEntry->prtConsoleOffTime) {
        state = "on";
    } else if (!*lightEntry->prtConsoleOnTime && *lightEntry->prtConsoleOffTime) {
        state = "off";
    } else if (*lightEntry->prtConsoleOnTime && *lightEntry->prtConsoleOffTime) {
        state = "blink";
    }
    g_string_sprintfa(s, " %-*s ", 5, state);

    e = fmt_enum(printer_mib_enums_prtConsoleColor, lightEntry->prtConsoleColor);
    g_string_sprintfa(s, "%s\n", e ? e : "");
}
```

Command Registration

```
void scli_init_printer_mode(scli_interp_t * interp)
{
    static scli_cmd_t cmds[] = {
        { "show printer console lights", NULL,
          "The show printer console lights command shows the current\n"
          "status of the printer's lights. [...]",
          SCLI_CMD_FLAG_NEED_PEER | SCLI_CMD_FLAG_XML,
          "printer console",
          "<xsd> <!-- ... --> </xsd>",
          show_printer_console_lights },
        { "monitor printer console lights", NULL,
          "The monitor printer console lights command shows the same\n"
          "information as the show printer console lights command. The\n"
          "information is updated periodically.",
          SCLI_CMD_FLAG_NEED_PEER | SCLI_CMD_FLAG_MONITOR,
          NULL, NULL,
          show_printer_console_lights },
        { NULL, NULL, NULL, 0, NULL, NULL, NULL }
    };

    static scli_mode_t printer_mode = {
        "printer",
        "The scli printer mode is based on the Printer-MIB as published\n"
        "in RFC 1759 and some updates currently being worked on in the\n"
        "IETF Printer MIB working group.",
        cmds
    };

    scli_register_mode(interp, &printer_mode);
}
```

Ideas for Future Work

- More command implementations for additional device types and MIBs
- Support for SNMPv3 security in the `gsnmp` engine
- Adopt `gnet` for low-level portable network access?
- Better code generation, e.g. support for spin-lock scalars
- Automatic caching schemes with intelligent cache validation
- Code generation for MIB “procedures” from formal MIB annotation
- Perhaps a `gtk` user interface in addition to the scli and XML output formats

Try `scli` yourself!

The source code and documentation is available from:

http://www.ibr.cs.tu-bs.de/projects/scli/

http://www.ibr.cs.tu-bs.de/projects/libsmi/