Using Lua for BACnet OEM solutions in building automation

How to brew coffee with Lua and BACnet

Robert Schlephorst
schlephorst@se-elektronic.de

SE Elektronic GmbH

Lua Workshop 2013
about SE Elektronic GmbH

- company founded 1983 (30 years)
- develops building automation products: sensors, actors, building controller
- development & production in south Germany
What is BACnet?

- Building Automation and Control Networks
- ASHRAE/ANSI 135-2010, ISO 16484-5
- Communication between devices and building management system

- Provide BMS with information
- Sensors, Actors
- AirConditioning, Elevators, Escalators, ...

<table>
<thead>
<tr>
<th>BACnet Application Layer</th>
<th>BACnet Network Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>BVLC</td>
<td>MS/TP</td>
</tr>
<tr>
<td>UDP/IP</td>
<td>RS-485</td>
</tr>
<tr>
<td></td>
<td>LON</td>
</tr>
<tr>
<td></td>
<td>PTP</td>
</tr>
<tr>
<td></td>
<td>RS-232</td>
</tr>
</tbody>
</table>

Using Lua for BACnet OEM solutions in building automation
Why BACnet?
BACnet compared to other protocols

e.g. Modbus
- designed for resource limited devices
- master/slave communication
- read/write registers/file records
- datatypes limited (simple like integer, float)
- no data descriptions readable
  → need device documentation for integration
Why BACnet?
BACnet compared to other protocols

BACnet
- data/features structured as objects
- developers like objects
- searches possible, objects browsable
- Event & ChangeOfValue Notifications
- BACnet network routable
- conformance tests → cross-vendor interoperability
Why BACnet?

BACnet compared to other protocols

- Data/features structured as objects
- Developers like objects
- Searches possible, objects browsable
- Event & ChangeOfValue Notifications
- BACnet network routable
- Conformance tests → cross-vendor interoperability
- High complexity
- Many optional features
- Stack partitioned into functional blocks
BACnet is all about objects

- analog-input, 42
  - object
    - present-value
    - object-name
    - object-identifier
    - out-of-service
    - unit
    - EventTimeStamps
      - properties

Datatype
- REAL
- STRING
- OBJID
- BOOLEAN
- ENUM
- TimeStamp[3]

BACnet is all about objects
BACnet is all about objects

- analog-input, 42
- object
- value @ physical input
- present-value
- object-name
- object-identifier
- out-of-service
- unit
- EventTimeStamps
- properties

Datatype
- REAL
- STRING
- OBJID
- BOOLEAN
- ENUM
- TimeStamp[3]

SE Elektronic GmbH - www.se-elektronic.de

Using Lua for BACnet OEM solutions in building automation
BACnet is all about objects

**analog-input, 42**

- **present-value**: REAL
- **object-name**: STRING
- **object-identifier**: OBJID
- **out-of-service**: BOOLEAN
- **unit**: ENUM
- **EventTimeStamps**: TimeStamp[3]

```
properties
```

---

SE Elektronic GmbH - www.se-elektronic.de

Using Lua for BACnet OEM solutions in building automation
BACnet is all about objects

- **analog-input, 42**: value @ physical input
- **object**: present-value
- **object-name**: REAL
- **object-identifier**: STRING
- **out-of-service**: OBJID
- **unit**: BOOLEAN
- **EventTimeStamps**: ENUM
- **properties**: TimeStamp[3]

SE Elektronic GmbH - www.se-elektronic.de

Using Lua for BACnet OEM solutions in building automation
BACnet is all about objects

- analog-input, 42
- object
- "temperature"
- °C

Properties:
- present-value (REAL)
- object-name (STRING)
- object-identifier (OBJID)
- out-of-service (BOOLEAN)
- unit (ENUM)
- EventTimeStamps (TimeStamp[3])

Array of complex datatype
Communication Between Devices

- Object Access Services (Read, Write, Create, ...)
- Alarm & Event Services (EventNotification, ...)
- Device Management (Backup, Restore, ...)

Using Lua for BACnet OEM solutions in building automation
BACnet Requirements

- **full BACnet** integration (Coffee Management System)
- **control** water temperature → Loop (PID)
- **log** coffee consumption → Trendlog
BACnet Requirements

- full BACnet integration (Coffee Management System)
- control water temperature → Loop (PID)
- log coffee consumption → Trendlog

OEM Application

- send SMS on low coffee level
The BACnet Coffee Machine
UberCoffee 2013™

BACnet Requirements

- full BACnet integration (Coffee Management System)
- control water temperature → Loop (PID)
- log coffee consumption → Trendlog

OEM Application

- send SMS on low coffee level
- no more beans: send Email to Java™ support
The BACnet Coffee Machine
UberCoffee 2013™

BACnet Requirements
- full BACnet integration (Coffee Management System)
- control water temperature → Loop (PID)
- log coffee consumption → Trendlog

OEM Application
- send SMS on low coffee level
- no more beans: send Email to procurement
The BACnet Coffee Machine
UberCoffee 2013™

BACnet Requirements
- full BACnet integration (Coffee Management System)
- control water temperature → Loop (PID)
- log coffee consumption → Trendlog

OEM Application
- send SMS on low coffee level
- no more beans: send Email to procurement
- web frontend
The BACnet Coffee Machine
UberCoffee 2013™

BACnet Requirements
- full BACnet integration (Coffee Management System)
- control water temperature → Loop (PID)
- log coffee consumption → Trendlog

OEM Application
- send SMS on low coffee level
- no more beans: send Email to procurement
- web frontend
- integrated display with customizable content
BACnet demands new processing strategies

Legacy Programmable Logic

- application w/ cyclic I/O: read - process - write
- data points (digital & analog)

\[ \text{in}^{a/d} \xrightarrow{\text{process}} \text{out}^{a/d} \]
BACnet demands new processing strategies

Legacy Programmable Logic

Using Lua for BACnet OEM solutions in building automation
BACnet demands new processing strategies

Legacy Programmable Logic

- application w/ cyclic I/O: read - process - write
- data points (digital & analog)

Problems

- translate between BACnet datatypes and data points
- impossible to read and/or modify complex properties
- can’t use BACnet Notifications: ChangeOfValue & Events
Legacy Solution

- create special data points/modules which trigger actions
- special (OEM) logic **hardcoded** in firmware

`send Email`

`enable`  `ok`

`magic?`

Content  Recipient
Solution

- Lua application for OEM tasks
- BACnet API for Lua
- BACnet datatypes for Lua
- asynchronous processing
Using Lua for BACnet OEM solutions in building automation
Lua Program Object

Runtime

- Lua 5.2 VM
- Memory Quota → restrict memory consumption
- Watchdog → detect e.g. infinite loops
- Shell access and execution control via Telnet
- Loader for own package format
- is BACnet object → configuration and control with BACnet
Lua Program Object
C-Libs/APIs

Libs
- Baselibs
- LuaSocket
- LuaFileSystem

APIs
- Data Points
- Timer
- System Configuration
- User Accounts
- GUI
- BACnet

Using Lua for BACnet OEM solutions in building automation
Using Lua for BACnet OEM solutions in building automation
Asynchronous Processing

- Messagequeue
- Lua VM

Callback Functions
- Timers
- System Events
- BACnet COV & BACnet Events
- BACnet Service Responses

Using Lua for BACnet OEM solutions in building automation
using lua for bacnet oem solutions in building automation
Asynchronous Processing

BACnet Event 'low coffee'

AnalogInput 'coffee level'

Callback Functions

- Timers
- System Events
- BACnet COV & BACnet Events
- BACnet Service Responses
Asynchronous Processing

BACnet Event
'low coffee'

AnalogInput
'coffee level'

Callback Functions
- Timers
- System Events
- BACnet COV & BACnet Events
- BACnet Service Responses

send SMS

update display
Example
BACnet Datatype Construction in Lua

```lua
local time_a = bacnet.data.daytime(10, 20, 42)
```

userdata
type constructor
Example

BACnet Datatype Construction in Lua

local time_a = bacnet.data.daytime(10, 20, 42)
local time_b = bacnet.data.daytime("10:19:42")

overloaded for convenience
Example

BACnet Datatype Construction in Lua

```lua
local time_a = bacnet.data.daytime(10, 20, 42)
local time_b = bacnet.data.daytime("10:19:42")
local h,m,s = time_a()
```

convert to Lua type(s)
Example
BACnet Datatype Construction in Lua

```lua
local time_a = bacnet.data.daytime(10, 20, 42)
local time_b = bacnet.data.daytime("10:19:42")
local h,m,s = time_a()
local early = time_b < time_a

metatable magic
```
local time_a = bacnet.data.daytime(10, 20, 42)
local time_b = bacnet.data.daytime("10:19:42")
local h,m,s = time_a()
local early = time_b < time_a
local datetime = bacnet.data.date_time(
    bacnet.data.daytime("10:41:00"), bacnet.data.date(2013,11,24))

complex type is composition of data

time member
date member
local time_a = bacnet.data.daytime(10, 20, 42)
local time_b = bacnet.data.daytime("10:19:42")
local h,m,s = time_a()
local early = time_b < time_a
local datetime = bacnet.data.date_time(
    bacnet.data.daytime("10:41:00"), bacnet.data.date(2013,11,24))
for member, value in pairs(datetime) do
    print(member, value)
end
again metatable magic
Example
BACnet Datatype Construction in Lua

```lua
local time_a = bacnet.data.daytime(10, 20, 42)
local time_b = bacnet.data.daytime("10:19:42")
local h,m,s = time_a()
local early = time_b < time_a
local datetime = bacnet.data.date_time(
    bacnet.data.daytime("10:41:00"), bacnet.data.date(2013,11,24))
for member, value in pairs(datetime) do
    print(member, value) end
local constr_datetime = datetime:typefunc()
```

retrieve constructor from type
Example

BACnet Datatype Construction in Lua

```lua
local time_a = bacnet.data.daytime(10, 20, 42)
local time_b = bacnet.data.daytime(“10:19:42”)
local h,m,s = time_a()
local early = time_b < time_a
local datetime = bacnet.data.date_time(
    bacnet.data.daytime(“10:41:00”), bacnet.data.date(2013,11,24))
for member, value in pairs(datetime) do
    print(member, value) end
local constr_datetime = datetime:typefunc()
local member_info = constr_datetime()
```

empty call delivers metainfo on members
Example
BACnet Object Access

Object Handle [OBJECT]

local av, err = bacnet.object.get("AnalogValue", 42)
local av, err = bacnet.object.get("AnalogValue", 42)
local pv, err = av:pv()

Value [REAL]  read 'present-value' property
Example
BACnet Object Access

local av, err = bacnet.object.get("AnalogValue", 42)
local pv, err = av:pv()
local ok, err = av:pv(nil, pv + 12)

write 'present-value' property
metatable operation
optional 'array-index'
Example
BACnet Object Access

```lua
local av, err = bacnet.object.get("AnalogValue", 42)
local pv, err = av:pv()
local ok, err = av:pv(nil, pv + 12)
local ok, err = av:property("low-limit", nil, bacnet.data.real(15.0))
```

- property identifier
- read/write arbitrary property
- datatype constructor
Example

Timers

```lua
Timer = timer.new(function () [...] end)
Timer:start(50)
```

- start timer with timeout 50 ms
- timer constructor
- callback
Example

BACnet Remote Object Access

local ok, err = bacnet.service.write(2013, function (error) [...] end, bacnet.data.objid("AnalogValue", 42), "present-value", nil, bacnet.data.real(42))

remote device ID
response callback (async)
object ID
property ID
array index
write data
Application Examples

Display Unit (HMI)
- browse objects
- modify objects
- alarm notifications
Application Examples

Display Unit (HMI)
- browse objects
- modify objects
- alarm notifications

Webserver
- LuaSocket
- e.g. Xavante
- browse objects
- modify objects
- show alarms
Using Lua for BACnet OEM solutions in building automation
Application Examples

### Display Unit (HMI)
- browse objects
- modify objects
- alarm notifications

### Control Application
- control algorithm in Lua
- based on BACnet objects
- optionally event based

### Webserver
- LuaSocket
- e.g. Xavante
- browse objects
- modify objects
- show alarms

### One-Time-Tasks
- initial startup operations
- diagnosis
- maintenance tasks
Easy access to BACnet objects and properties.

Versatile high-level API for customized BACnet applications.

Possibility for rapid development, rapid deployment and easy in-target debugging of applications.

No Firmware extension in C needed for OEM applications.
Introduction
BACnet Lua Solutions
Summary

Benefits

- Easy access to BACnet objects and properties.
- Versatile high-level API for customized BACnet applications.
- Possibility for rapid development, rapid deployment and easy in-target debugging of applications.
- No Firmware extension in C needed for OEM applications.

QUESTIONS?