

Layered Syslog Architecture (syslog-protocol draft)

Rainer Gerhards

Adiscon

rgerhards@hq.adiscon.com

2004-02-24

Status around November 2003

- RFC 3164, 3195, syslog-sign and -international each specify message format, transport specifics and on top of that some specific functionality.
- Each RFC/ID makes slight changes to the format, so there are minor inconsistencies.
- This makes it hard to
 - support all standards in a single program
 - e.g. transport a syslog-sign message over a 3195 channel
 - build relay chains with different RFCs “in them”

A Solution

- Experience in protocol design tells us that forming multiple layers, each one with defined “interface” to its lower any upper layers as well as a defined functionality helps to keep things
 - simple
 - extensible & powerful (new functionality “immediately” available to older parts)
 - “change-ignorant” (new functionality does not break existing code)
- Many successful protocols use this approach (e.g. IP stack itself, SMTP/MIME)

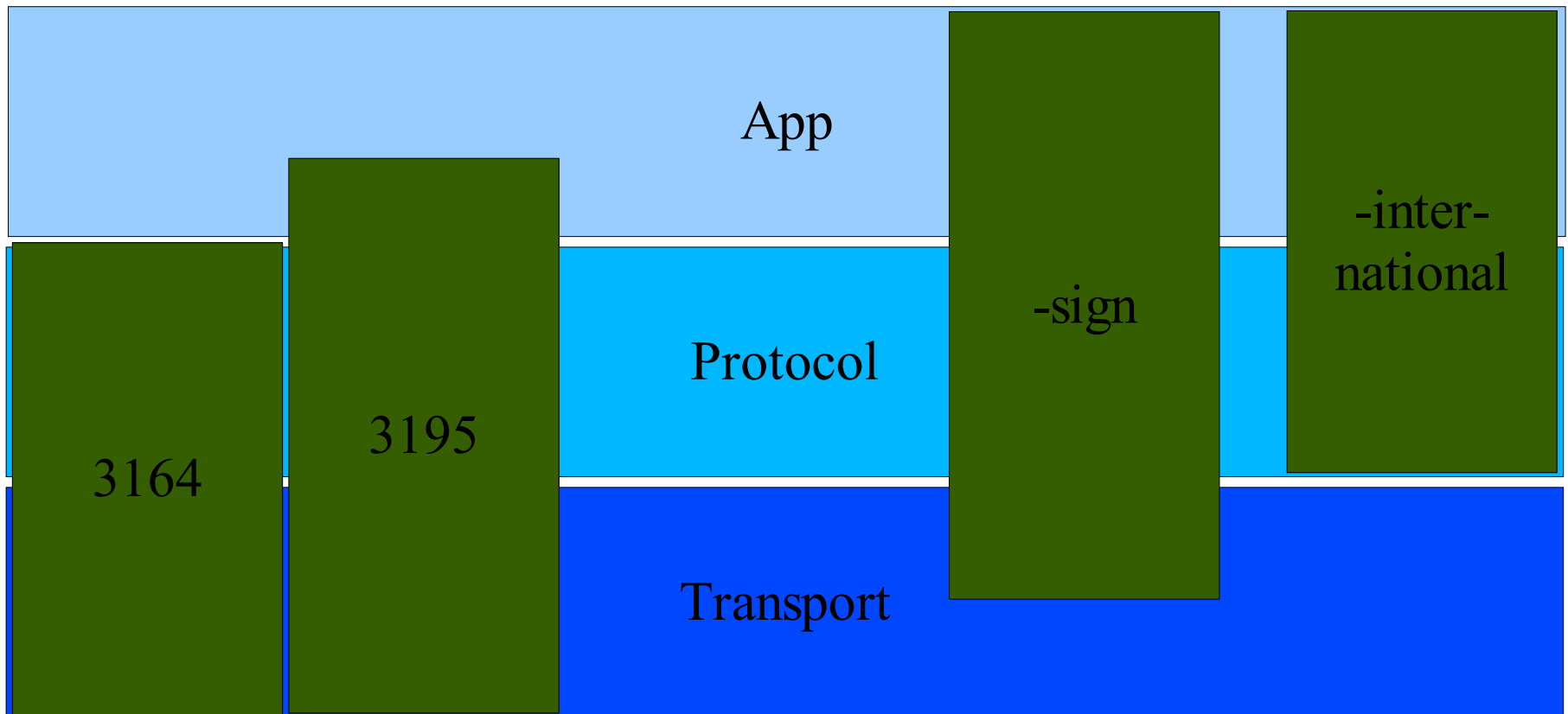
Syslog Protocol Layers

“Applications”, e. g. Signatures, International Characters

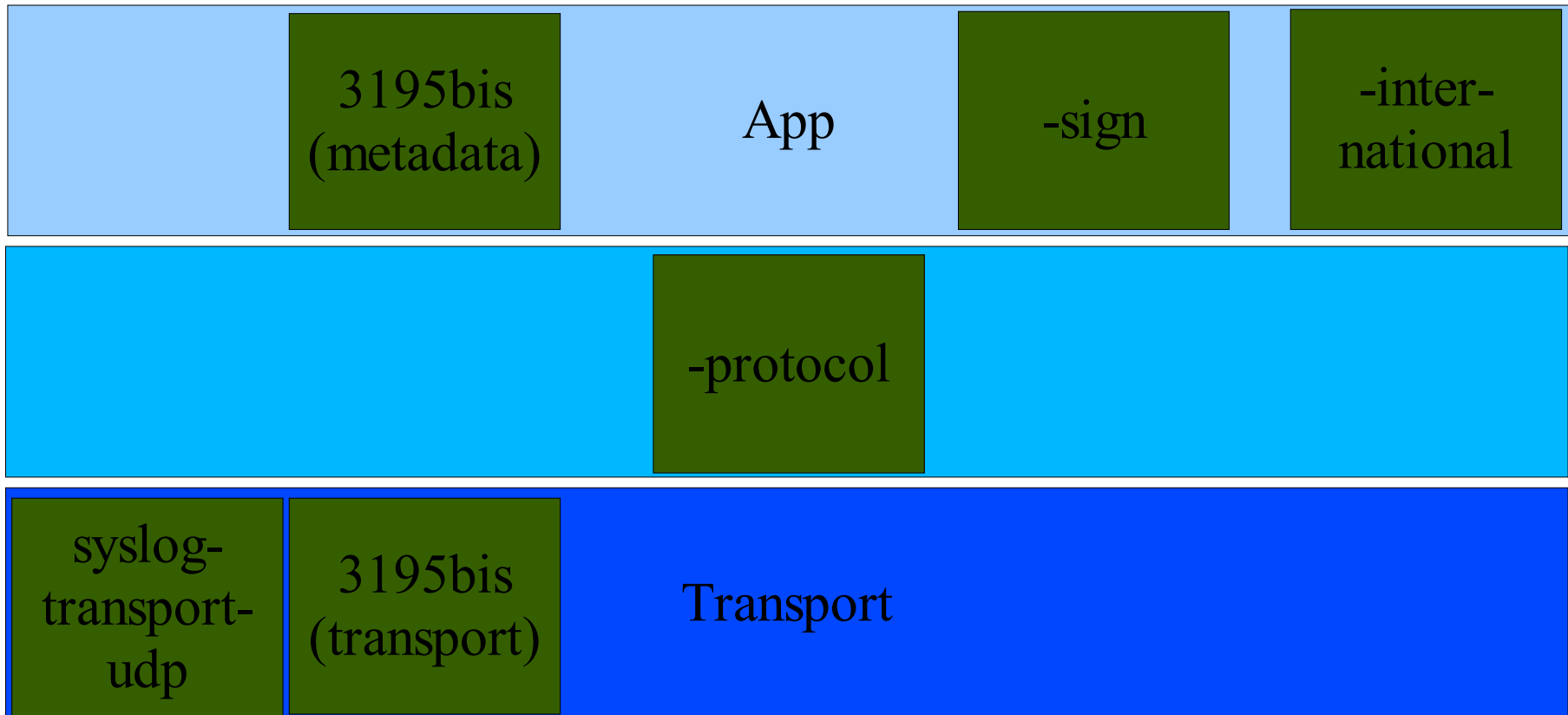
Message Format & Syslog Semantics

Transport Mappings

RFCs/IDs as of 11/2003



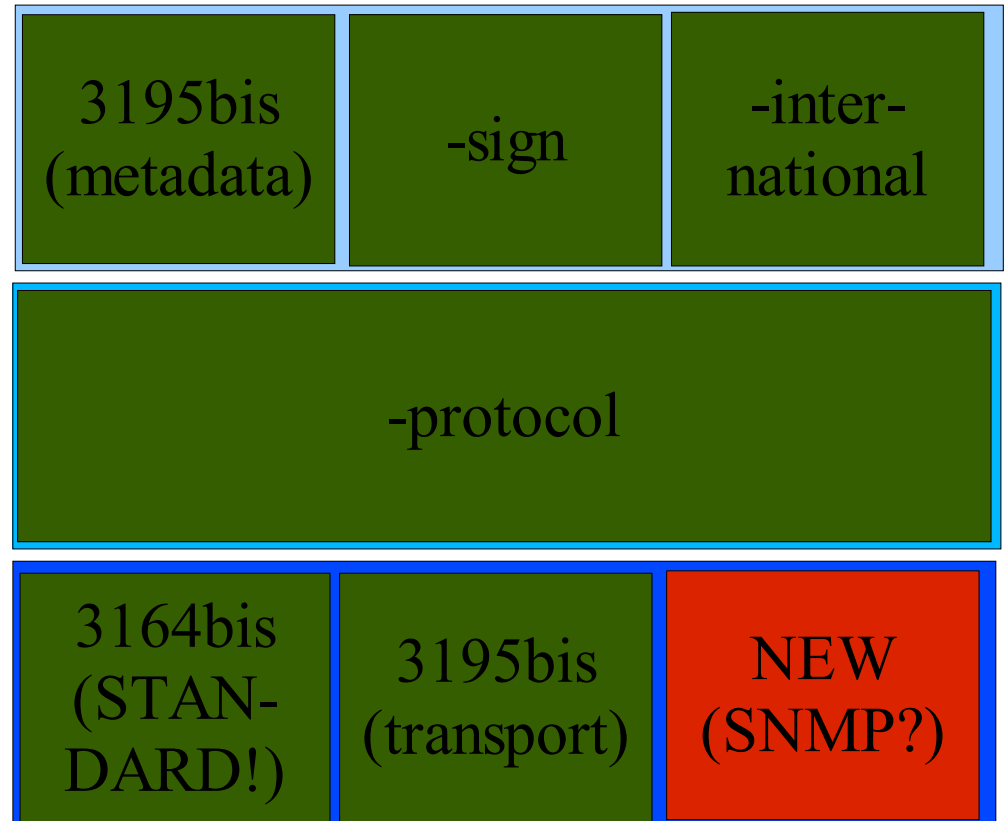
Existing documents in new Concept...



The layered architecture means 6 RFCs instead of 4, but each one shorter and interoperating. Thanks to consistency, they would hopefully be easier to implement and manage (operations management).

What happens if Something New is added?

- Everything else would continue to work
- The new functionality would be available to all existing syslog “parts”



What is in -protocol?

- Syslog “system” components
 - protocol layers described
 - devices, relays, collectors
 - simplex communication structure
- Message format
 - common header
 - structured data elements (concept & some defined)
 - “classical” freeform message
 - all message text (not header) in Unicode (**UTF-8**)
- General security considerations

A word on the simplex structure...

- -protocol inherits the simplex communications structure from legacy syslog
 - There is **NO** two-way communication
 - The sender never knows what happened to the message
 - RFC3195 provides a partial duplex communication, but not end-to-end (other transports may provide this, too)
- Redesigning syslog to be [end-to-end] duplex
 - would require a totally different protocol
 - thus we have decided to stick with the simplex structure (knowing the issues this causes)
 - after all, legacy syslog has proven fine in reality.

Transport Layer/Mapping

- Abstracted transport interface
- Mapping to UDP
 - currently being written by Anton Okmianski
 - will be a required transport (to ensure “least common denominator” interoperability) – the only one required
- Mapping to TCP – later, transport RFC 3195bis can eventually serve as this
- Potential for other mappings
 - SNMP (“syslogish” messages are send via SNMP TRAPS by some devices)
 - “unreliable”, raw TCP for those who really need it [I don't say it is necessarily a good idea, though ;-)]

Backward Compatibility

- -protocol is NOT backward compatible to classical syslog
- If a -protocol message is received by a RFC 3164 syslogd, it will be treated as a message without header, which is moderately convenient (at least, it is a defined case and its handling makes sense).
- WG concensus was that it is more important to build a solid base than to preserve backward compatibility at all costs (<http://www.syslog.cc/ietf/protocol/issue4.html>)

A new HEADER

- More HEADER fields, including a version number
- Other HEADER fields (e.g. TIMESTAMP) more precisely defined
- Modelled in a way that legacy and RFC 3164 syslog will see messages as “messages without header” - their resulting processing is probably the best that we can get from legacy implementations (at least nothing is messed up and everything from the -protocol message is kept intact).

Structured Data Elements I

- Used to implement “extended functionality”.
- Build on idea of cookies introduced in syslog-sign
- So far, have a simple form:
 - [`@#ID name="value"`]
 - Multiple name/value pairs possible
 - Can NOT be structured in a tree/container
 - May eventually be moved to a specific field (in -protocol-03, they can be everywhere in the message – see <http://www.syslog.cc/ietf/protocol/issue8.html>)

Structured Data Elements II

- Potential for vendor-/experimental extensions
- Can carry everything that has a precise structure
- Currently defined IDs:
 - msgpart – multi-part messaging
 - time – describe TZ and clock accuracy
 - origin – describe originator
 - More to come (“official” ones to be IANA-assigned)

Multi-Part Messaging

- -protocol specifies a maximum message size of 1280 octets
- Larger messages can be transmitted via multi-part messaging
 - a concept also known as fragmentation or segmentation
 - optional – must only be used if message is too large
 - A transport mapping may reduce the max size and thus a relay may eventually split a message into multiple parts (still being discussed)

Draft Status/Issue Tracking

- All open issues are tracked at <http://www.syslog.cc/ietf/protocol.html>
- <http://www.syslog.cc/ietf/> also contains information on other syslog-related topics
- The following pages address some important topics currently being discussed.

Semantics

- Should we describe the facility semantics?
<http://www.syslog.cc/ietf/protocol/issue5.html>
- Should we describe specific semantics for TAG?
<http://www.syslog.cc/ietf/protocol/issue16.html>

Issue 13: Precise or Lax?

- Syslog-protocol currently “dictates” a lot of things that an implementation has to do
- Goal: more security by less ambiguity
- Some discussion posts indicates this goes overboard and makes it unnecessary complex
- Currently included:
 - Precise instructions on how to process message
 - Implementor's notes pointing out known problem areas and potential solutions
- See also: <http://www.syslog.cc/ietf/why-indepth.html>

Issue 13 – Possible Solutions I

- Leave as as
 - It's lengthy, but ensures that each implementor reads the notes
- Move notes to a separate, informational RFC
 - keeps -protocol focussed on the bare essentials
 - Implementor recommendations still available in the RFC series (and thus likely to be seen)
 - **As of now, this is my personal recommendation** (looks like a good compromise)

Issue 13 – Possible Solutions II

- Remove these Notes
 - Streamlines -protocol
 - Leaves a lot of suggestions and clarifications out, thus more potential for misunderstanding
 - Of course, these could be put on a web page – but who will find that page... (so its more consequent to remove it – hiding on a web page is a false compromise)
- What do do?
 - I asked for external advise on various mailing list (currently in progress)
 - What does the meeting audience think?

Relay Operations

- Describe relay operations in -protocol or in a separate RFC? (<http://www.syslog.cc/ietf/protocol/issue15.html>)
- I think we should
 - describe a few basic facts about relays in -protocol
 - leave the details for another draft
- Acceptable, because
 - Relay mode actually not implemented by the majority of software
 - Allows us to publish -protocol in a more timely manner
 - Keeps a complex aspect in its own context

Message Size

- Currently -protocol specifies a max size of 1280 octets (well... it should... due to rush-editing to meet the meeting deadline, I accidentally deleted that paragraph – sorry)
- Currently discussed is if we should NOT set an upper limit and leave it to the transport mapping to specify how large messages are transmitted.
 - pro: keeps -protocol clean and really transport-ignorant
 - con: I personally have the “feeling” that this is not in the “spirit” of the original syslog protocol (and will eventually complicate things – in most cases, messages are very short...)

What's next?

- Comments and recommendations from the meeting are highly recommended.
- Feedback on the draft is highly appreciated.
- A good starting point is <http://www.syslog.cc/ietf/protocol.html>