SMIng Overview

Jürgen Schönwälder

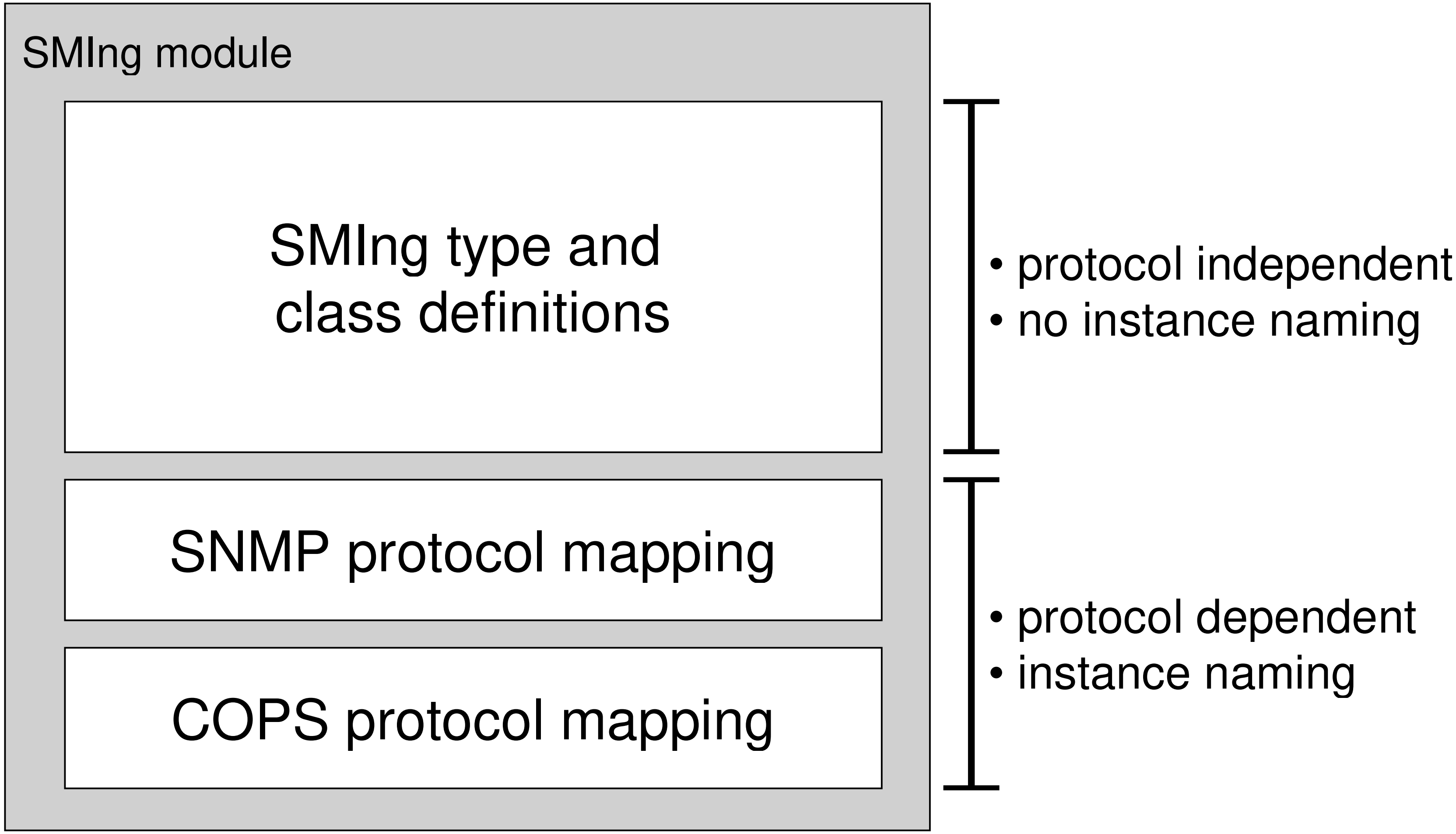
Technical University of Braunschweig
Bültenweg 74/75
38106 Braunschweig, Germany

SMIng WG Meeting, 49th IETF, San Diego, 2000-12-13

Document Overview

- core language <draft-irtf-nmrg-sming-04.txt>
  - type system
  - class system
  - identities
  - module structure
  - syntax and language extensibility
  - versioning and extension rules
- core data definitions <draft-irtf-nmrg-sming-modules-01.txt>
- Internet data definitions <draft-irtf-nmrg-sming-inet-modules-01.txt>
- protocol mappings <draft-irtf-nmrg-sming-snmp-01.txt>

SMIng Module Structure



SMIng Modules

- related SMIng definitions are contained in a module
- modules provide a naming scope for all contained definitions
- module names must be globally unique
- implicit export of all definitions contained in a module
- explicit import of definitions from other modules
- modules record some basic meta information (contact, description, revisions)
- module organization must follow a predefined structure to simplify implementations

SMIng Type System

- base data types:
  - Integer32, Unsigned32, Integer64, Unsigned64,
  - Float32, Float64, Float128,
  - Enumeration, Bits,
  - OctetString,
  - Pointer
- new types with additional semantics can be derived from any existing type
- range and size restrictions can be used to formally constrain the set of legal values
- optional display format and units definitions

SMIng Class System

- classes encapsulate attributes and events
- attributes have an associated type which can be
  - a base type
  - a derived type
  - a class (containment)
- classes can be derived from other classes (single inheritance)
- classes can have associated events
- classes provide their own naming scope for attributes and events

SMIng Identities

- identities provide a mechanism to identify constants without the need for a central authority
- names of identities are scoped by the module name
- the Pointer base type can point to identities

SMIng Syntax

- programmer friendly syntax
  - look & feel similar to Java, C++, ...
  - improved compactness in the normal cases
  - consistent structure of statements (easy to memorize)
- easy to implement and efficient to parse
  - consistent syntactic structure simplifies grammar
  - no forward references (except in cases where unavoidable)
  - statement separators help to recover from parse errors
  - complete grammar specified in ABNF
- language extensibility
  - declaration of new statements, unknown statements skipped

SMIng Versioning

- once published, definitions can not be changed
- all definitions have an associated status:
  - current
  - deprecated
  - obsolete
- updated current definitions may not depend on deprecated or obsolete definitions
- module versions are identified by the time-stamp of the last revision
- extension rules define allowed changes between revisions (ensure forward & backward compatibility)

Core Data Definitions

- fundamental type definitions
  - Gauge32, Counter32, TimeTicks [Unsigned32]
  - Gauge64, Counter64 [Unsigned64]
  - TimeStamp [TimeTicks]
  - TimeInterval [Integer32 (0..2147483647)]
  - DateAndTime [OctetString (8|11)]
  - PhysAddress [OctetString]
  - MacAddress [OctetString (6)]
  - TruthValue [Enumeration]
  - DisplayString, Utf8String [OctetString]
  - DisplayString255 [DisplayString (0..255)]
  - Utf8String255 [Utf8String (0..255)]

Core Data Definitions (cont.)

- historic type definitions
  - Opaque [OctetString] obsolete
  - IpAddress [OctetString (4)] deprecated

- fundamental identity:
  - null

Core Internet Data Definitions

- fundamental type definitions
  - InetPortNumber [Unsigned32 (0..65535)]
  - InetProtocolNumber [Unsigned32 (0..255)]
  - InetDiffServCodePoint [Unsigned32 (-1,0..63)]

  - InetAddressType [Enumeration]
  - InetAddressMask [Unsigned32]
  - InetAddress [OctetString]
  - InetAddressIPv4 [InetAddress (4)]
  - InetAddressIPv6 [InetAddress (16|20)]
  - InetAddressDNS [InetAddress (1..255)]

Core Internet Data Definitions (cont.)

- fundamental class definitions
  - InetNetworkEndpoint [InetAddressType, InetAddress]
  - InetTransportEndpoint [InetNetworkEndpoint, InetPortNumber]
  - InetSubnet [InetNetworkEndpoint, InetAddressMask]
  - InetPortNumberRange [InetPortNumber, InetPortNumber]

- filter definitions
  - generic and IP specific packet filter definitions

SNMP Data Definitions

- SNMP specific type definitions
  - TestAndIncr [Integer32 (0..2147483647)]
  - AutonomousType [Pointer]
  - VariablePointer [Pointer]
  - RowPointer [Pointer]
  - RowStatus [Enumeration]
  - StorageType [Enumeration]
  - TDomain [Pointer]
  - TAddressOrZero [OctetString (0..255)]
  - TAddress [TAddressOrZero (1..255)]

SNMP Data Type Mapping

- base data type mapping

– OctetString	⇒	OCTET STRING	
– Pointer	⇒	OBJECT IDENTIFIER	
– Integer32	⇒	Integer32	
– Integer64	⇒	Opaque(Integer64)	(*)
– Unsigned32	⇒	Unsigned32	
– Unsigned64	⇒	Opaque(Unsigned64)	(*)
– Float32	⇒	Opaque(Float32)	(*)
– Float64	⇒	Opaque(Float64)	(*)
– Float128	⇒	Opaque(Float128)	(*)
– Enumeration	⇒	INTEGER	
– Bits	⇒	OCTET STRING	

SNMP Data Type Mapping (cont.)

- derived data type mapping
  - derived types are in general encoded according to their underlying base type
  - types derived from the derived SMIng types Counter32, Counter64, TimeTicks, IpAddress, Opaque are encoded as the corresponding SNMP data type with the same name
- ASN.1 definitions are provided for the new base types that do not exist in SNMPv2c/SNMPv3 so that they can be wrapped using Opaque
- future versions of SNMP may support the new ASN.1 base types natively

SNMP Naming

- instance naming
  - node statement to build the OID tree and to assign OIDs to identities
  - scalars and table statements to assign OIDs to class attributes
  - notification statement to assign OIDs to events
- flattening
  - attributes of classes are mapped to scalars in a group of scalars or columns in a table
  - nested classes (containment) are "flattened" into a flat table structure or a group of scalars with a common parent node

SNMP Conformance Definitions

- conformance definitions
  - single group statement to group notifications or objects into conformance groups
  - compliance statement to define implementation requirements

Confused?

Stay tuned for some examples!