Network Working Group
Request for Comments: 5388
Category: Standards Track

S. Niccolini
S. Tartarelli
J. Quittek
T. Dietz
NEC
M. Swany
UDel

December 2008

Information Model and XML Data Model for Traceroute Measurements

Status of This Memo

Copyright Notice

Abstract

This document describes a standard way to store the configuration and the results of traceroute measurements. This document first describes the terminology used in this document and the traceroute tool itself; afterwards, the common information model is defined, dividing the information elements into two semantically separated groups (configuration elements and results elements). Moreover, an additional element is defined to relate configuration elements and results elements by means of a common unique identifier. On the basis of the information model, a data model based on XML is defined to store the results of traceroute measurements.

Table of Contents

1. Introduction

Traceroutes are used by lots of measurement efforts, either as independent measurements or as a means of getting path information to support other measurement efforts. That is why there is the need to standardize the way the configuration and the results of traceroute measurements are stored. The standard metrics defined by the IPPM group in matters of delay, connectivity, and losses do not apply to the metrics returned by the traceroute tool. Therefore, in order to compare results of traceroute measurements, the only possibility is to add to the stored results a specification of the operating system as well as the name and version for the traceroute tool used. This document, in order to store results of traceroute measurements and allow comparison of them, defines a standard way to store them using an XML schema.

The document is organized as follows: Section 2 defines the terminology used in this document; Section 3 describes the traceroute tool; Section 4 describes the results of a traceroute measurement as displayed to the screen from which the traceroute tool was launched; Section 5 and Section 6, respectively, describe the information model and data model for storing configuration and results of the traceroute measurements; Section 7 contains the XML schema to be used as a template for storing and/or exchanging traceroute measurement information; the document ends with security considerations and IANA considerations in Section 8 and Section 9 respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Terminology Used in This Document

The terminology used in this document is defined as follows:

- traceroute tool: a software tool for network diagnostic that behaves as described in Section 3;

- traceroute measurement: an instance of the traceroute tool launched, with specific configuration parameters (traceroute measurement configuration parameters), from a specific host (initiator of the traceroute measurement) giving as output specific traceroute measurement results;

- traceroute probe: one of many IP packets sent out by the traceroute tool during a traceroute measurement;

- traceroute measurement configuration parameters: the configuration parameters of a traceroute measurement;
- traceroute measurement results: the results of a traceroute measurement;
- traceroute measurement information: both the results and the configuration parameters of a traceroute measurement;
- traceroute measurement path: a sequence of hosts transited in order by traceroute probes during a traceroute measurement.

3. The Traceroute Tool and Its Operations

Traceroute is a network diagnostic tool used to determine the hop-by-hop path from a source to a destination and the Round Trip Time (RTT) from the source to each hop. Traceroute can be therefore used to discover some information (hop counts, delays, etc.) about the path between the initiator of the traceroute measurement and other hosts.

Typically, the traceroute tool attempts to discover the path to a destination by sending UDP probes with specific time-to-live (TTL) values in the IP packet header and trying to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host.

In more detail, a first set of probes with TTL equal to 1 is sent by the traceroute tool from the host initiating the traceroute measurement (some tool implementations allow setting the initial TTL to a value equal to "n" different from 1, so that the first "n-1" hops are skipped and the first hop that will be traced is the "n-th" in the path). Upon receiving a probe, the first hop host decreases the TTL value (by one or more). By observing a TTL value equal to zero, the host rejects the probe and typically returns an ICMP message with a TIME_EXCEEDED value. The traceroute tool can therefore derive the IP address of the first hop from the header of the ICMP message and evaluate the RTT between the host initiating the traceroute measurement and the first hop. The next hops are discovered following the same procedure, taking care to increase at each step the TTL value of the probes by one. The TTL value is increased until either an ICMP PORT_UNREACHABLE message is received, meaning that the destination host has been reached, or the maximum configured number of hops has been hit.

Some implementations use ICMP Echoes, instead of UDP datagrams. However, many routers do not return ICMP messages about ICMP messages, i.e., no ICMP TIME_EXCEEDED is returned for an ICMP Echo.

Therefore, this document recommends to base implementations on UDP datagrams. Considerations on TCP-based implementations of the traceroute tool are reported in Appendix A.1.

4. Results of Traceroute Measurements

The following list reports the information fields provided as results by all traceroute tool implementations considered. The order in which they are reported here is not relevant and changes in different implementations. For each hop, the following information is reported:

- the hop index;
- the host symbolic address, provided that at least one of the probes received a response, the symbolic address could be resolved at the corresponding host, and the option to display only numerical addresses was not set;
- the host IP address, provided that at least one of the probes received a response;
- the RTT for each response to a probe.

Depending on the traceroute tool implementation, additional information might be displayed in the output (for instance, MPLS-related information).

It might happen that some probes do not receive a response within the configured timeout (for instance, if the probe is filtered out by a firewall). In this case, an "*" is displayed in place of the RTT. The information model reflects this using a string with the value of "RoundTripTimeNotAvailable", meaning either the probe was lost because of a timeout or it was not possible to transmit a probe. It may also happen that some implementations print the same line multiple times when a router decreases the TTL by more than one, thus looking like multiple hops. The information model is not impacted by this since each line is handled separately; it is left to the applications handling the XML file how to deal with it. Moreover, for delays below 1 ms, some implementations report 0 ms (e.g., UNIX and LINUX), while WINDOWS reports "< 1 ms".

5. Information Model for Traceroute Measurements

The information model is composed of information elements; for defining these information elements, a template is used. Such template is specified in the list below:

- name - A unique and meaningful name for the information element. The preferred spelling for the name is to use mixed case if the name is compound, with an initial lower-case letter, e.g., "sourceIpAddress".

- description - The semantics of this information element.

- dataType - One of the types listed in Section 5.1 of this document or in an extension of the information model. The type space for attributes is constrained to facilitate implementation.

- units - If the element is a measure of some kind, the units identify what the measure is.

5.1. Data Types

This section describes the set of basic valid data types of the information model.

- string - The type "string" represents a finite-length string of valid characters from the Unicode character encoding set. Unicode allows for ASCII and many other international character sets to be used. It is expected that strings will be encoded in UTF-8 format, which is identical in encoding for US-ASCII characters but which also accommodates other Unicode multi-byte characters.

- string255 - Same type as "string" but with the restriction of 255 characters.

- inetAddressType - The type "inetAddressType" represents a type of Internet address. The allowed values are imported from [RFC4001] (where the intent was to import only some of the values); additional allowed values are "asnumber" and "noSpecification".

- inetAddress - The type "inetAddress" denotes a generic Internet address. The allowed values are imported from [RFC4001] (the values imported are unknown, ipv4, ipv6, and dns), while non-global IPv4/IPv6 addresses (e.g., ipv4z and ipv6z) are excluded; an additional allowed value is the AS number, indicated as the actual number plus the indication of how the mapping from IP address to AS number was performed. "Unknown" is used to indicate an IP address that is not in one of the formats defined.

- ipASNumberMappingType - The type "ipASNumberMappingType" represents a type of mapping from IP to AS number, it indicates the method that was used to do get the mapping (allowed values are "bgptables", "routingregistries", "nslookup", "others" or "unknown").

- boolean - The type "boolean" represents a boolean value according to XML standards [W3C.REC-xmlschema-2-20041028].

- unsignedInt - The type "unsignedInt" represents a value in the range (0..4294967295).

- unsignedShort - The type "unsignedShort" represents a value in the range (0..65535).

- unsignedByte - The type "unsignedByte" represents a value in the range (0..255).

- u8nonzero - The type "u8nonzero" represents a value in the range (1..255).

- probesType - The type "probesType" represents a way of indicating the protocol used for the traceroute probes. Values defined in this document are UDP, TCP, and ICMP.

- operationResponseStatus - The type "operationResponseStatus" is used to report the result of an operation. The allowed values are imported from [RFC4560].

- dateTime - The type "dateTime" represents a date-time specification according to XML standards [W3C.REC-xmlschema-2-20041028] but is restricted to the values defined in [RFC3339].

5.2. Information Elements

This section describes the elements related to the storing of a traceroute measurement. The elements are grouped in two groups (configuration and results) according to their semantics. In order to relate configuration and results elements by means of a common unique identifier, an additional element is defined belonging to both groups.

5.2.1. Relationships between the Information Elements

Every traceroute measurement is represented by an instance of the "traceRoute" element. This class provides a standardized representation for traceroute measurement data. The "traceroute" element is an element that can be composed of (depending on the nature of the traceroute measurement):

- 1 optional "RequestMetadata" element;

- 0..2147483647 "Measurement" elements.

Each "Measurement" element contains:

- 1 optional "MeasurementMetadata" element;
- 0..2147483647 "MeasurementResult" elements.

The "RequestMetadata" element can be used for specifying parameters of a traceroute measurement to be performed at one or more nodes by one or more traceroute implementations. Depending on the capabilities of a traceroute implementation, not all requested parameters can be applied. Which parameters have actually been applied for a specific traceroute measurement is specified in a "MeasurementMetadata" element.

The "RequestMetadata" element is a sequence that contains:

- 1 "TestName" element;
- 1 optional "ToolVersion" element;
- 1 optional "ToolName" element;
- 1 "CtlTargetAddress" element;
- 1 optional "CtlBypassRouteTable" element;
- 1 optional "CtlProbeDataSize" element;
- 1 optional "CtlTimeOut" element;
- 1 optional "CtlProbesPerHop" element;
- 1 optional "CtlPort" element;
- 1 optional "CtlMaxTtl" element;
- 1 optional "CtlDSField" element;
- 1 optional "CtlSourceAddress" element;
- 1 optional "CtlIfIndex" element;
- 1 optional "CtlMiscOptions" element;
- 1 optional "CtlMaxFailures" element;
- 1 optional "CtlDontFragment" element;

- 1 optional "CtlInitialTtl" element;
- 1 optional "CtlDescr" element;
- 1 "CtlType" element.

If the "RequestMetadata" element is omitted from an XML file, it means that the traceroute measurement configuration parameters requested were all used and the "MeasurementMetadata" element lists them in detail.

The "MeasurementMetadata" element is a sequence that contains:

- 1 "TestName" element;
- 1 "OSName" element;
- 1 "OSVersion" element;
- 1 "ToolVersion" element;
- 1 "ToolName" element;
- 1 "CtlTargetAddressType" element;
- 1 "CtlTargetAddress" element;
- 1 "CtlBypassRouteTable" element;
- 1 "CtlProbeDataSize" element;
- 1 "CtlTimeOut" element;
- 1 "CtlProbesPerHop" element;
- 1 "CtlPort" element;
- 1 "CtlMaxTtl" element;
- 1 "CtlDSField" element;
- 1 "CtlSourceAddressType" element;
- 1 "CtlSourceAddress" element;
- 1 "CtlIfIndex" element;
- 1 optional "CtlMiscOptions" element;

- 1 "CtlMaxFailures" element;
- 1 "CtlDontFragment" element;
- 1 "CtlInitialTtl" element;
- 1 optional "CtlDescr" element;
- 1 "CtlType" element.

Configuration information elements can describe not just traceroute measurements that have already happened ("MeasurementMetadata" elements), but also the configuration to be used when requesting a measurement to be made ("RequestMetadata" element). This is quite different semantically, even if the individual information elements are similar. Due to this similarity, both "RequestMetadata" and "MeasurementMetadata" are represented by the same type in the XML schema. All elements that are missing from the "RequestMetadata" or marked as optional in the "RequestMetadata" but mandatory in the "MeasurementMetadata" must be specified as empty elements. Specifying them as empty elements means use the default value. The "CtlType" element could have been optional in the "RequestMetadata", but since default values cannot be specified for complex types in an XML schema, the element is mandatory in the "RequestMetadata".

The "MeasurementResult" element is a sequence that contains:

- 1 "TestName" element;
- 1 "ResultsStartDateAndTime" element;
- 1 "ResultsIpTgtAddrType" element;
- 1 "ResultsIpTgtAddr" element;
- 1 "ProbeResults" elements;
- 1 "ResultsEndDateAndTime" element.

Additionally, it is important to say that each "ProbeResults" element is a sequence that contains:

- 1..255 "hop" elements.

Each "hop" element is a sequence that contains:

- 1..10 "probe" elements;

- 1 optional "HopRawOutputData" element.

Each "probe" element contains:

- 1 "HopAddrType" element;
- 1 "HopAddr" element;
- 1 optional "HopName" element;
- 0..255 optional "MPLSLabelStackEntry" elements;
- 1 "ProbedRoundTripTime" element;
- 1 "ResponseStatus" element;
- 1 "Time" element.

Different numbers of appearances of the three basic elements in the XML file are meant for different scopes:

- a file with only 1 "RequestMetadata" element represents a file containing the traceroute measurement configuration parameters of a traceroute measurement; it can be used to distribute the traceroute measurement configuration parameters over multiple nodes asked to run the same traceroute measurement;
- a file with 1 "Measurement" element containing 1 "MeasurementMetadata" and 1 "MeasurementResult" element represents a file containing the traceroute measurement information of a traceroute measurement;
- a file with 1 "Measurement" element containing 1 "MeasurementMetadata" and n "MeasurementResult" elements represents a file containing the traceroute measurement information of a set of traceroute measurements run over different times with always the same traceroute measurement configuration parameters;
- a file with 1 "RequestMetadata" and 1 "Measurement" element containing 1 "MeasurementMetadata" and 1 "Measurement" element represents a file containing the traceroute measurement information of a traceroute measurement (containing both the requested traceroute measurement configuration parameters and the ones actually used);
- other combinations are possible to store multiple traceroute measurements all in one XML file.

5.2.2. Configuration Information Elements

This section describes the elements specific to the configuration of the traceroute measurement (belonging to both the "RequestMetadata" and "MeasurementMetadata" elements).

5.2.2.1. CtlTargetAddressType

- name - CtlTargetAddressType
- description - Specifies the type of address in the corresponding "CtlTargetAddress" element. This element is not directly reflected in the XML schema of Section 7. The host address type can be determined by examining the inetAddress type name and the corresponding element value.
- dataType - inetAddressType
- units - N/A

5.2.2.2. CtlTargetAddress

- name - CtlTargetAddress
- description - In the "RequestMetadata" element, it specifies the host address requested to be used in the traceroute measurement. In the "MeasurementMetadata" element, it specifies the host address used in the traceroute measurement.
- dataType - inetAddress
- units - N/A

5.2.2.3. CtlBypassRouteTable

- name - CtlBypassRouteTable
- description - In the "RequestMetadata" element, specifies if the optional bypassing of the route table was enabled or not. In the "MeasurementMetadata" element, specifies if the optional bypassing of the route table was enabled or not. If enabled, the normal routing tables will be bypassed and the probes will be sent directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to perform the traceroute measurement to a local host through an interface that has no route defined. This object can be used when the setsockopt SOL_SOCKET SO_DONTROUTE option is supported and set (see [IEEE.1003-1G.1997]).

- dataType - boolean
- units - N/A

5.2.2.4. CtlProbeDataSize

- name - CtlProbeDataSize
- description - Specifies the size of the probes of a traceroute measurement in octets (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element). If UDP datagrams are used as probes, then the value contained in this object is exact. If another protocol is used to transmit probes (i.e., TCP or ICMP), for which the specified size is not appropriate, then the implementation can use whatever size (appropriate to the method) is closest to the specified size. The maximum value for this object is computed by subtracting the smallest possible IP header size of 20 octets (IPv4 header with no options) and the UDP header size of 8 octets from the maximum IP packet size. An IP packet has a maximum size of 65535 octets (excluding IPv6 jumbograms).
- dataType - unsignedShort
- units - octets

5.2.2.5. CtlTimeOut

- name - CtlTimeOut
- description - Specifies the timeout value, in seconds, for each probe of a traceroute measurement (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element).
- dataType - unsignedByte
- units - seconds

5.2.2.6. CtlProbesPerHop

- name - CtlProbesPerHop
- description - Specifies the number of probes with the same time-to-live (TTL) value that are sent for each host (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element).

- dataType - unsignedByte
- units - probes

5.2.2.7. CtlPort

- name - CtlPort
- description - Specifies the base port used by the traceroute measurement (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element).
- dataType - unsignedShort
- units - port number

5.2.2.8. CtlMaxTtl

- name - CtlMaxTtl
- description - Specifies the maximum TTL value for the traceroute measurement (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element).
- dataType - u8nonzero
- units - time-to-live value

5.2.2.9. CtlDSField

- name - CtlDSField
- description - Specifies the value that was requested to be stored in the Differentiated Services (DS) field in the traceroute probe (if in the "RequestMetadata" element). Specifies the value that was stored in the Differentiated Services (DS) field in the traceroute probe (if in the "MeasurementMetadata" element). The DS field is defined as the Type of Service (TOS) octet in an IPv4 header or as the Traffic Class octet in an IPv6 header (see Section 7 of [RFC2460]). The value of this object must be a decimal integer in the range from 0 to 255. This option can be used to determine what effect an explicit DS field setting has on a traceroute measurement and its probes. Not all values are legal or meaningful. Useful TOS octet values are probably 16 (low delay) and 8 (high throughput). Further references can be found in [RFC2474] for the definition of the Differentiated Services (DS) field and in [RFC1812] Section 5.3.2 for Type of Service (TOS).

- dataType - unsignedByte
- units - N/A

5.2.2.10. CtlSourceAddressType

- name - CtlSourceAddressType
- description - Specifies the type of address in the corresponding "CtlSourceAddress" element. This element is not directly reflected in the XML schema of Section 7. The host address type can be determined by examining the "inetAddress" type name and the corresponding element value. DNS names are not allowed for the "CtlSourceAddress".
- dataType - inetAddressType
- units - N/A

5.2.2.11. CtlSourceAddress

- name - CtlSourceAddress
- description - Specifies the IP address (which has to be given as an IP number, not a hostname) as the source address in traceroute probes (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element). On hosts with more than one IP address, this option can be used in the "RequestMetadata" element to force the source address to be something other than the primary IP address of the interface the probe is sent on; the value "unknown" means the default address will be used.
- dataType - inetAddress
- units - N/A

5.2.2.12. CtlIfIndex

- name - CtlIfIndex
- description - Specifies the interface index as defined in [RFC2863] that is requested to be used in the traceroute measurement for sending the traceroute probes (if in the "RequestMetadata" element). A value of 0 indicates that no specific interface is requested. Specifies the interface index actually used (if in the "MeasurementMetadata" element).

- dataType - unsignedInt
- units - N/A

5.2.2.13. CtlMiscOptions

- name - CtlMiscOptions
- description - Specifies implementation-dependent options (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element).
- dataType - string255
- units - N/A

5.2.2.14. CtlMaxFailures

- name - CtlMaxFailures
- description - Specifies the maximum number of consecutive timeouts allowed before terminating a traceroute measurement (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element). A value of either 255 (maximum hop count/possible TTL value) or 0 indicates that the function of terminating a remote traceroute measurement when a specific number of consecutive timeouts are detected was disabled. This element is included to give full compatibility with [RFC4560]. No known implementation of traceroute currently supports it.
- dataType - Unsigned8
- units - timeouts

5.2.2.15. CtlDontFragment

- name - CtlDontFragment
- description - Specifies if the don’t fragment (DF) flag in the IP header for a probe was enabled or not (if in the "MeasurementMetadata" element). If in the "RequestMetadata", it specifies if the flag was requested to be enabled or not. Setting the DF flag can be used for performing a manual PATH MTU test.
- dataType - boolean
- units - N/A

5.2.2.16. CtlInitialTtl

- name - CtlInitialTtl
- description - Specifies the initial TTL value for a traceroute measurement (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element). Such TTL setting is intended to bypass the initial (often well-known) portion of a path.
- dataType - u8nonzero
- units - N/A

5.2.2.17. CtlDescr

- name - CtlDescr
- description - Provides a description of the traceroute measurement.
- dataType - string255
- units - N/A

5.2.2.18. CtlType

- name - CtlType
- description - Specifies the implementation method used for the traceroute measurement (requested if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element). It specifies if the traceroute is using TCP, UDP, ICMP, or other types of probes. It is possible to specify other types of probes by using an element specified in another schema with a different namespace.
- dataType - probesType
- units - N/A

5.2.3. Results Information Elements

This section describes the elements specific to the results of the traceroute measurement.

5.2.3.1. ResultsStartDateAndTime

- name - ResultsStartDateAndTime
- description - Specifies the date and start time of the traceroute measurement. This is the time when the first probe was seen at the sending interface.
- dataType - DateTime
- units - N/A

5.2.3.2. ResultsIpTgtAddrType

- name - ResultsIpTgtAddrType
- description - Specifies the type of address in the corresponding "ResultsIpTgtAddr" element. This element is not directly reflected in the XML schema of Section 7. The host address type can be determined by examining the "inetAddress" type name and the corresponding element value.
- dataType - inetAddressType
- units - N/A

5.2.3.3. ResultsIpTgtAddr

- name - ResultsIpTgtAddr
- description - Specifies the IP address associated with a "CtlTargetAddress" value when the destination address is specified as a DNS name. The value of this object should be "unknown" if a DNS name is not specified or if a specified DNS name fails to resolve.
- dataType - inetAddress
- units - N/A

5.2.3.4. HopAddrType

- name - HopAddrType
- description - Specifies the type of address in the corresponding "HopAddr" element. This element is not directly reflected in the XML schema of Section 7. The host address type can be determined

by examining the "inetAddress" type name and the corresponding element value. DNS names are not allowed for "HopAddr".

- dataType - inetAddressType
- units - N/A

5.2.3.5. HopAddr

- name - HopAddr
- description - Specifies the address of a hop in the traceroute measurement path. This object is not allowed to be a DNS name.
- dataType - inetAddress
- units - N/A

5.2.3.6. HopName

- name - HopName
- description - Specifies the DNS name of the "HopAddr" if it is available. If it is not available, the element is omitted.
- dataType - inetAddress
- units - N/A

5.2.3.7. MPLSLabelStackEntry

- name - MPLSLabelStackEntry
- description - Specifies entries of the MPLS label stack of a probe observed when the probe arrived at the hop that replied to the probe. This object contains one MPLS label stack entry as a 32-bit value as it is observed on the MPLS label stack. Contained in this single number are the MPLS label, the Exp field, the S flag, and the MPLS TTL value as specified in [RFC3032]. If more than one MPLS label stack entry is reported, then multiple instances of elements of this type are used. They must be ordered in the same order as on the label stack with the top label stack entry being reported first.
- dataType - unsignedInt
- units - N/A

5.2.3.8. ProbeRoundTripTime

- name - ProbeRoundTripTime
- description - If this element contains the element "roundTripTime", this specifies the amount of time measured in milliseconds from when a probe was sent to when its response was received or when it timed out. The value of this element is reported as the truncation of the number reported by the traceroute tool (the output "< 1 ms" is therefore encoded as 0 ms). If it contains the element "roundTripTimeNotAvailable", it means either the probe was lost because of a timeout or it was not possible to transmit a probe.
- dataType - unsignedShort or string
- units - milliseconds or N/A

5.2.3.9. ResponseStatus

- name - ResponseStatus
- description - Specifies the result of a traceroute measurement made by the host for a particular probe.
- dataType - operationResponseStatus
- units - N/A

5.2.3.10. Time

- name - Time
- description - Specifies the timestamp for the time the response to the probe was received at the interface.
- dataType - DateTime
- units - N/A

5.2.3.11. ResultsEndDateAndTime

- name - ResultsEndDateAndTime
- description - Specifies the date and end time of the traceroute measurement. It is either the time when the response to the last probe of the traceroute measurement was received or the time when

the last probe of the traceroute measurement was sent plus the relative timeout (in case of a missing response).

- dataType - DateTime
- units - N/A

5.2.3.12. HopRawOutputData

- name - HopRawOutputData
- description - Specifies the raw output data returned by the traceroute measurement for a certain hop in a traceroute measurement path. It is an implementation-dependent, printable string, expected to be useful for a human interpreting the traceroute results.
- dataType - string
- units - N/A

5.2.4. Information Element Correlating Configuration and Results Elements

This section defines an additional element belonging to both previous groups (configuration elements and results elements) named "TestName". This element is defined in order to relate configuration and results elements by means of a common unique identifier (to be chosen in accordance to the specification of [RFC4560]).

5.2.4.1. TestName

- name - TestName
- description - Specifies the name of a traceroute measurement. This is not necessarily unique within any well-defined scope (e.g., a specific host, initiator of the traceroute measurement).
- dataType - string255
- units - N/A

5.2.5. Information Elements to Compare Traceroute Measurement Results with Each Other

This section defines additional elements belonging to both previous groups (configuration elements and results elements); these elements were defined in order to allow traceroute measurement results comparison among different traceroute measurements.

5.2.5.1. OSName

- name - OSName
- description - Specifies the name of the operating system on which the traceroute measurement was launched. This element is ignored if used in the "RequestMetadata".
- dataType - string255
- units - N/A

5.2.5.2. OSVersion

- name - OSVersion
- description - Specifies the OS version on which the traceroute measurement was launched. This element is ignored if used in the "RequestMetadata".
- dataType - string255
- units - N/A

5.2.5.3. ToolVersion

- name - ToolVersion
- description - Specifies the version of the traceroute tool (requested to be used if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element).
- dataType - string255
- units - N/A

5.2.5.4. ToolName

- o name - ToolName
- o description - Specifies the name of the traceroute tool (requested to be used if in the "RequestMetadata" element, actually used if in the "MeasurementMetadata" element).
- o dataType - string255
- o units - N/A

6. Data Model for Storing Traceroute Measurements

For storing and transmitting information according to the information model defined in the previous section, a data model is required that specifies how to encode the elements of the information model.

There are several design choices for a data model. It can use a binary or textual representation and it can be defined from scratch or use already existing frameworks and data models. In general, the use of already existing frameworks and models should be preferred.

Binary and textual representations both have advantages and disadvantages. Textual representations are (with some limitations) human-readable, while a binary representation consumes less resources for storing, transmitting, and parsing data.

An already existing and closely related data model is the DISMAN-TRACEROUTE-MIB module [RFC4560], which specifies a Structure of Management Information version 2 (SMIv2) encoding [RFC2578], [RFC2579], and [RFC2580] for transmitting traceroute measurement information (configuration and results). This data model is well suited and supported within network management systems, but as a general format for storing and transmitting traceroute results, it is not easily applicable.

Another binary representation would be an extension of traffic-flow information encodings as specified for the IP Flow Information Export (IPFIX) protocol [RFC5101], [RFC5102]. The IPFIX protocol is extensible. However, the architecture behind this protocol [IPFIX] is targeted at exporting passively measured flow information. Therefore, some obstacles are expected when trying to use it for transmitting traceroute measurement information.

For textual representations, using the eXtensible Markup Language (XML) [W3C.REC-xml-20060816] is an obvious choice. XML supports clean structuring of data and syntax checking of records. With some

limitations, it is human-readable. It is supported well by a huge pool of tools and standards for generating, transmitting, parsing, and converting it to other data formats. Its disadvantages are the resource consumption for processing, storing, and transmitting information. Since the expected data volumes related to traceroute measurement in network operation and maintenance are not expected to be extremely high, the inefficient usage of resources is not a significant disadvantage. Therefore, XML was chosen as a basis for the traceroute measurement information model that is specified in this memo.

Section 7 contains the XML schema to be used as a template for storing and/or exchanging traceroute measurement information. The schema was designed in order to use an extensible approach based on templates (pretty similar to how the IPFIX protocol is designed) where the traceroute configuration elements (both the requested parameters, "RequestMetadata", and the actual parameters used, "MeasurementMetadata") are metadata to be referenced by results information elements (data) by means of the "TestName" element (used as a unique identifier, chosen in accordance to the specification of [RFC4560]). Currently Open Grid Forum (OGF) is also using this approach and cross-requirements have been analyzed. As a result of this analysis, the XML schema contained in Section 7 is compatible with the OGF schema since both were designed in a way that limits the unnecessary redundancy and a simple one-to-one transformation between the two exists.

7. XML Schema for Traceroute Measurements

This section presents the XML schema to be used as a template for storing and/or exchanging traceroute measurement information. The schema uses UTF-8 encoding as defined in [RFC3629]. In documents conforming to the format presented here, an XML declaration SHOULD be present specifying the version and the character encoding of the XML document. The document should be encoded using UTF-8. Since some of the strings can span multiple lines, [RFC5198] applies. XML processing instructions and comments MUST be ignored. Mind that whitespace is significant in XML when writing documents conforming to this schema. Documents using the presented format must be valid according to the XML schema shown in this section. Since elements of type "_CtlType" may contain elements from unknown namespaces, those elements MUST be ignored if their namespace is unknown to the processor. Values for elements using the XML schema type "dateTime" MUST be restricted to values defined in [RFC3339]. Future versions of this format MAY extend this schema by creating a new schema that redefines all or some of the data types and elements defined in this version or by establishing a complete new schema.

Due to the limited line length some lines appear wrapped.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
           targetNamespace="urn:ietf:params:xml:ns:traceroute-1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:tr="urn:ietf:params:xml:ns:traceroute-1.0">
  <xs:simpleType name="string255">
    <xs:annotation>
      <xs:documentation>String restricted to 255
      characters.</xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:string">
      <xs:maxLength value="255"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="u8nonzero">
    <xs:annotation>
      <xs:documentation>unsignedByte with non zero
      value.</xs:documentation>
    </xs:annotation>

    <xs:restriction base="xs:unsignedByte">
      <xs:minInclusive value="1"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="_roundTripTime">
    <xs:choice>
      <xs:element name="roundTripTime">
        <xs:simpleType>
          <xs:restriction base="xs:unsignedInt"/>
        </xs:simpleType>
      </xs:element>

      <xs:element name="roundTripTimeNotAvailable">
        <xs:complexType/>
      </xs:element>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="_inetAddressUnknown"/>

  <xs:simpleType name="_inetAddressIpv4">
    <xs:restriction base="xs:string">
      <xs:pattern value="(([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5
```

```
]).){3}([1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="_inetAddressIpv6">
    <xs:restriction base="xs:string">
      <xs:pattern value="(([\dA-Fa-f]{1,4}:){7}[\dA-Fa-f]{1,4})(:([\d
]{1,3}.){3}[\d]{1,3})?"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="_inetAddressDns">
    <xs:restriction base="xs:string">
      <xs:maxLength value="256"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="_inetAddressASNumber">
    <xs:annotation>
      <xs:documentation>Specifies the AS number of a hop in the
      traceroute path as a 32-bit number and indicates how the
      mapping from IP address to AS number was
      performed.</xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="asNumber" type="xs:unsignedInt"/>

      <xs:element name="ipASNumberMappingType">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="bgptables"/>

            <xs:enumeration value="routingregistries"/>

            <xs:enumeration value="nslookup"/>

            <xs:enumeration value="others"/>

            <xs:enumeration value="unknown"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="inetAddress">
    <xs:choice>
```

```
      <xs:element name="inetAddressUnknown"
                  type="tr:_inetAddressUnknown"/>

      <xs:element name="inetAddressIpv4" type="tr:_inetAddressIpv4"/>

      <xs:element name="inetAddressIpv6" type="tr:_inetAddressIpv6"/>

      <xs:element name="inetAddressASNumber"
                  type="tr:_inetAddressASNumber"/>

      <xs:element minOccurs="0" name="inetAddressDns"
                  type="tr:_inetAddressDns"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="inetAddressWithoutDns">
    <xs:sequence>
      <xs:choice>
        <xs:element name="inetAddressUnknown"
                    type="tr:_inetAddressUnknown"/>

        <xs:element name="inetAddressIpv4"
                    type="tr:_inetAddressIpv4"/>

        <xs:element name="inetAddressIpv6"
                    type="tr:_inetAddressIpv6"/>

        <xs:element name="inetAddressASNumber"
                    type="tr:_inetAddressASNumber"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="operationResponseStatus">
    <xs:restriction base="xs:string">
      <xs:enumeration value="responseReceived"/>

      <xs:enumeration value="unknown"/>

      <xs:enumeration value="internalError"/>

      <xs:enumeration value="requestTimedOut"/>

      <xs:enumeration value="unknownDestinationAddress"/>

      <xs:enumeration value="noRouteToTarget"/>

      <xs:enumeration value="interfaceInactiveToTarget"/>
```

```
      <xs:enumeration value="arpFailure"/>

      <xs:enumeration value="maxConcurrentLimitReached"/>

      <xs:enumeration value="unableToResolveDnsName"/>

      <xs:enumeration value="invalidHostAddress"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="_CtlType">
    <xs:choice>
      <xs:element name="TCP">
        <xs:complexType/>
      </xs:element>

      <xs:element name="UDP">
        <xs:complexType/>
      </xs:element>

      <xs:element name="ICMP">
        <xs:complexType/>
      </xs:element>

      <xs:any namespace="##other"/>
    </xs:choice>
  </xs:complexType>

  <xs:complexType name="_ProbeResults">
    <xs:sequence>
      <xs:element maxOccurs="255" name="hop">
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="10" name="probe">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="HopAddr"
                              type="tr:inetAddressWithoutDns">
                    <xs:annotation>
                      <xs:documentation>Specifies the address of a
                      hop in the traceroute measurement path.  This
                      object is not allowed to be a DNS name.  The
                      address type can be determined by examining the
                      "inetAddress" type name and the corresponding
                      element value.</xs:documentation>
                    </xs:annotation>
                  </xs:element>
```

```
                <xs:element minOccurs="0" name="HopName"
                            type="tr:_inetAddressDns">
                  <xs:annotation>
                    <xs:documentation>Specifies the DNS name of
                    the "HopAddr" if it is available.  If it is
                    not available, the element is
                    omitted.</xs:documentation>
                  </xs:annotation>
                </xs:element>

                <xs:element maxOccurs="255" minOccurs="0"
                            name="MPLSLabelStackEntry">
                  <xs:annotation>
                    <xs:documentation>Specifies entries of the
                    MPLS label stack of a probe observed when the
                    probe arrived at the hop that replied to the
                    probe.  This object contains one MPLS label stack
                    entry as a 32-bit value as it is observed on the
                    MPLS label stack.  Contained in this single
                    number are the MPLS label, the Exp field, the S
                    flag, and the MPLS TTL value as specified in
                    [RFC3032].  If more than one MPLS label stack
                    entry is reported, then multiple instances of
                    elements of this type are used.  They must be
                    ordered in the same order as on the label stack
                    with the top label stack entry being reported
                    first.</xs:documentation>
                  </xs:annotation>

                  <xs:simpleType>
                    <xs:restriction base="xs:unsignedInt">
                      <xs:maxInclusive value="4294967295"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>

                <xs:element name="ProbeRoundTripTime"
                            type="tr:_roundTripTime">
                  <xs:annotation>
                    <xs:documentation>If this element contains the
                    element "roundTripTime", this specifies the
                    amount of time measured in milliseconds from
                    when a probe was sent to when its response was
                    received or when it timed out.  The value of
                    this element is reported as the truncation of
                    the number reported by the traceroute tool (the
                    output "&lt; 1 ms" is therefore encoded as 0 ms).
                    If it contains the element
```

```
                    "roundTripTimeNotAvailable", it means either
                    the probe was lost because of a timeout or it
                    was not possible to transmit a probe.
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>

                <xs:element name="ResponseStatus"
                            type="tr:operationResponseStatus">
                  <xs:annotation>
                    <xs:documentation>Specifies the result of a
                    traceroute measurement made by the host for a
                    particular probe.</xs:documentation>
                  </xs:annotation>
                </xs:element>

                <xs:element name="Time" type="xs:dateTime">
                  <xs:annotation>
                    <xs:documentation>Specifies the timestamp for
                    the time the response to the probe was
                    received at the interface.</xs:documentation>
                  </xs:annotation>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>

          <xs:element minOccurs="0" name="HopRawOutputData"
                      type="tr:string255">
            <xs:annotation>
              <xs:documentation>Specifies the raw output data
              returned by the traceroute measurement for a
              certain hop in a traceroute measurement path.  It is
              an implementation-dependent, printable string,
              expected to be useful for a human interpreting the
              traceroute results.</xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_Metadata">
  <xs:annotation>
    <xs:documentation>Specifies the metadata for a traceroute
    operation -- the parameters requested if used in
```

```
      "RequestMetadata" or the actual parameters used if used in
      "MeasurementMetadata".</xs:documentation>
    </xs:annotation>

    <xs:sequence>
      <xs:element name="TestName" type="tr:string255">
        <xs:annotation>
          <xs:documentation>Specifies the name of a traceroute
          measurement.  This is not necessarily unique within any
          well-defined scope (e.g., a specific host, initiator of
          the traceroute measurement).</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="" name="OSName" type="tr:string255">
        <xs:annotation>
          <xs:documentation>Specifies the name of the operating
          system on which the traceroute measurement was launched.
          This element is ignored if used in the
          "RequestMetadata".</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="" name="OSVersion" type="tr:string255">
        <xs:annotation>
          <xs:documentation>Specifies the OS version on which the
          traceroute measurement was launched.  This element is
          ignored if used in the
          "RequestMetadata".</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="" name="ToolVersion" type="tr:string255">
        <xs:annotation>
          <xs:documentation>Specifies the version of the traceroute
          tool (requested to be used if in the "RequestMetadata"
          element, actually used if in the "MeasurementMetadata"
          element).</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="" name="ToolName" type="tr:string255">
        <xs:annotation>
          <xs:documentation>Specifies the name of the traceroute
          tool (requested to be used if in the "RequestMetadata"
          element, actually used if in the "MeasurementMetadata"
          element).</xs:documentation>
        </xs:annotation>
```

```
      </xs:element>

      <xs:element name="CtlTargetAddress" type="tr:inetAddress">
        <xs:annotation>
          <xs:documentation>In the "RequestMetadata" element, it
          specifies the host address requested to be used in the
          traceroute measurement.  In the "MeasurementMetadata"
          element, it specifies the host address used in the
          traceroute measurement.  The host address type can be
          determined by examining the "inetAddress" type name and
          the corresponding element value.</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="false" name="CtlBypassRouteTable"
                  type="xs:boolean">
        <xs:annotation>
          <xs:documentation>In the "RequestMetadata" element
          specifies if the optional bypassing of the route
          table was enabled or not.  In the "MeasurementMetadata"
          element, specifies if the optional bypassing of the route
          table was enabled or not.  If enabled, the normal routing
          tables will be bypassed and the probes will be sent
          directly to a host on an attached network.  If the host is
          not on a directly attached network, an error is returned.
          This option can be used to perform the traceroute
          measurement to a local host through an interface that has
          no route defined.  This object can be used when the
          setsockopt SOL_SOCKET SO_DONTROUTE option is supported and
          set (see the POSIX standard IEEE.1003-1G.1997).
          </xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="0" name="CtlProbeDataSize">
        <xs:annotation>
          <xs:documentation>Specifies the size of the probes of a
          traceroute measurement in octets (requested if in the
          "RequestMetadata" element, actually used if in the
          "MeasurementMetadata" element).  If UDP datagrams are used
          as probes, then the value contained in this object is
          exact.  If another protocol is used to transmit probes
          (i.e., TCP or ICMP) for which the specified size is not
          appropriate, then the implementation can use whatever
          size (appropriate to the method) is closest to the
          specified size.  The maximum value for this object is
          computed by subtracting the smallest possible IP header
          size of 20 octets (IPv4 header with no options) and the
```

```
          UDP header size of 8 octets from the maximum IP packet
          size.  An IP packet has a maximum size of 65535 octets
          (excluding IPv6 jumbograms).</xs:documentation>
        </xs:annotation>

        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:maxInclusive value="65507"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>

      <xs:element default="3" name="CtlTimeOut">
        <xs:annotation>
          <xs:documentation>Specifies the timeout value, in
          seconds, for each probe of a traceroute measurement
          (requested if in the "RequestMetadata" element, actually
          used if in the "MeasurementMetadata"
          element).</xs:documentation>
        </xs:annotation>

        <xs:simpleType>
          <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="1"/>

            <xs:maxInclusive value="60"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>

      <xs:element default="3" name="CtlProbesPerHop">
        <xs:annotation>
          <xs:documentation>Specifies the number of probes with the
          same time-to-live (TTL) value that are sent for each host
          (requested if in the "RequestMetadata" element, actually
          used if in the "MeasurementMetadata"
          element).</xs:documentation>
        </xs:annotation>

        <xs:simpleType>
          <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="1"/>

            <xs:maxInclusive value="10"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
```

```
      <xs:element default="33434" name="CtlPort">
        <xs:annotation>
          <xs:documentation>Specifies the base port used by the
          traceroute measurement (requested if in the
          "RequestMetadata" element, actually used if in the
          "MeasurementMetadata" element).</xs:documentation>
        </xs:annotation>

        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:minInclusive value="1"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>

      <xs:element default="30" name="CtlMaxTtl" type="tr:u8nonzero">
        <xs:annotation>
          <xs:documentation>Specifies the maximum TTL value for the
          traceroute measurement (requested if in the
          "RequestMetadata" element, actually used if in the
          "MeasurementMetadata" element).</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="0" name="CtlDSField"
                  type="xs:unsignedByte">
        <xs:annotation>
          <xs:documentation>Specifies the value that was requested
          to be stored in the Differentiated Services (DS) field in
          the traceroute probe (if in the "RequestMetadata"
          element).  Specifies the value that was stored in the
          Differentiated Services (DS) field in the traceroute
          probe (if in the "MeasurementMetadata" element).  The DS
          field is defined as the Type of Service (TOS) octet in an
          IPv4 header or as the Traffic Class octet in an IPv6
          header (see Section 7 of [RFC2460]).  The value of this
          object must be a decimal integer in the range from 0 to
          255.  This option can be used to determine what effect an
          explicit DS field setting has on a traceroute measurement
          and its probes.  Not all values are legal or meaningful.
          Useful TOS octet values are probably 16 (low delay) and
          8 (high throughput).  Further references can be found in
          [RFC2474] for the definition of the Differentiated
          Services (DS) field and in [RFC1812] Section 5.3.2 for
          Type of Service (TOS).</xs:documentation>
        </xs:annotation>
      </xs:element>
```

```
      <xs:element name="CtlSourceAddress"
                  type="tr:inetAddressWithoutDns">
        <xs:annotation>
          <xs:documentation>Specifies the IP address (which has to
          be given as an IP number, not a hostname) as the source
          address in traceroute probes (requested if in the
          "RequestMetadata" element, actually used if in the
          "MeasurementMetadata" element).  On hosts with more than
          one IP address, this option can be used in the
          "RequestMetadata" element to force the source address to
          be something other than the primary IP address of the
          interface the probe is sent on; the value "unknown" means
          the default address will be used.  The address type can be
          determined by examining the "inetAddress" type name and the
          corresponding element value.</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="0" name="CtlIfIndex"
                  type="xs:unsignedInt">
        <xs:annotation>
          <xs:documentation>Specifies the interface index as
          defined in [RFC2863] that is requested to be used in the
          traceroute measurement for sending the traceroute probes
          (if in the "RequestMetadata" element).  A value of 0
          indicates that no specific interface is requested.
          Specifies the interface index actually used (if in the
          "MeasurementMetadata" element).</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element minOccurs="0" name="CtlMiscOptions"
                  type="tr:string255">
        <xs:annotation>
          <xs:documentation>Specifies implementation-dependent
          options (requested if in the "RequestMetadata" element,
          actually used if in the "MeasurementMetadata"
          element).</xs:documentation>
        </xs:annotation>
      </xs:element>

      <xs:element default="5" name="CtlMaxFailures"
                  type="xs:unsignedByte">
        <xs:annotation>
          <xs:documentation>Specifies the maximum number of
          consecutive timeouts allowed before terminating a
          traceroute measurement (requested if in the
          "RequestMetadata" element, actually used if in the
```

```
      "MeasurementMetadata" element).  A value of either 255
      (maximum hop count/possible TTL value) or 0 indicates
      that the function of terminating a remote traceroute
      measurement when a specific number of consecutive
      timeouts are detected was disabled.  This element is
      included to give full compatibility with [RFC4560].  No
      known implementation of traceroute currently supports
      it.</xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element default="false" name="CtlDontFragment"
              type="xs:boolean">
    <xs:annotation>
      <xs:documentation>Specifies if the don't fragment (DF)
      flag in the IP header for a probe was enabled or not (if
      in the "MeasurementMetadata" element).  If in the
      "RequestMetadata", it specifies if the flag was requested
      to be enabled or not.  Setting the DF flag can be used for
      performing a manual PATH MTU test.</xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element default="1" name="CtlInitialTtl"
              type="tr:u8nonzero">
    <xs:annotation>
      <xs:documentation>Specifies the initial TTL value for a
      traceroute measurement (requested if in the
      "RequestMetadata" element, actually used if in the
      "MeasurementMetadata" element).  Such TTL setting is
      intended to bypass the initial (often well-known) portion
      of a path.</xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element maxOccurs="1" minOccurs="0" name="CtlDescr"
              type="tr:string255">
    <xs:annotation>
      <xs:documentation>Provides a description of the traceroute
      measurement.</xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="CtlType" type="tr:_CtlType">
    <xs:annotation>
      <xs:documentation>Specifies the implementation method
      used for the traceroute measurement (requested if in the
      "RequestMetadata" element, actually used if in the
```

```
            "MeasurementMetadata" element).  It specifies if the
            traceroute is using TCP, UDP, ICMP, or other types of
            probes.  It is possible to specify other types of probes
            by using an element specified in another schema with a
            different namespace.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="_Measurement">
      <xs:annotation>
        <xs:documentation>Contains the actual traceroute measurement
        results.</xs:documentation>
      </xs:annotation>

      <xs:sequence>
        <xs:element name="TestName" type="tr:string255">
          <xs:annotation>
            <xs:documentation>Specifies the name of a traceroute
            measurement.  This is not necessarily unique within any
            well-defined scope (e.g., a specific host, initiator of
            the traceroute measurement).</xs:documentation>
          </xs:annotation>
        </xs:element>

        <xs:element name="ResultsStartDateAndTime" type="xs:dateTime">
          <xs:annotation>
            <xs:documentation>Specifies the date and start time of
            the traceroute measurement.  This is the time when the
            first probe was seen at the sending
            interface.</xs:documentation>
          </xs:annotation>
        </xs:element>

        <xs:element name="ResultsIpTgtAddr"
                    type="tr:inetAddressWithoutDns">
          <xs:annotation>
            <xs:documentation>Specifies the IP address associated
            with a "CtlTargetAddress" value when the destination
            address is specified as a DNS name.  The value of this
            object should be "unknown" if a DNS name is not specified
            or if a specified DNS name fails to resolve.  The
            address type can be determined by examining the "inetAddress"
            type name and the corresponding element
            value.</xs:documentation>
          </xs:annotation>
        </xs:element>
```

```
      <xs:element name="ProbeResults" type="tr:_ProbeResults"/>

      <xs:element name="ResultsEndDateAndTime" type="xs:dateTime">
        <xs:annotation>
          <xs:documentation>Specifies the date and end time of the
          traceroute measurement.  It is either the time when the
          response to the last probe of the traceroute measurement
          was received or the time when the last probe of the
          traceroute measurement was sent plus the relative timeout
          (in case of a missing response).</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="traceRoute">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="RequestMetadata"
                    type="tr:_Metadata"/>

        <xs:element maxOccurs="2147483647" minOccurs="0"
                    name="Measurement">
          <xs:complexType>
            <xs:sequence>
              <xs:element minOccurs="0" name="MeasurementMetadata"
                          type="tr:_Metadata"/>

              <xs:element maxOccurs="2147483647" minOccurs="0"
                          name="MeasurementResult"
                          type="tr:_Measurement"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8. Security Considerations

Security considerations discussed in this section are grouped into considerations related to conducting traceroute measurements and considerations related to storing and transmitting traceroute measurement information.

This memo does not specify an implementation of a traceroute tool. Neither does it specify a certain procedure for storing traceroute measurement information. Still, it is considered desirable to discuss related security issues below.

8.1. Conducting Traceroute Measurements

Conducting Internet measurements can raise both security and privacy concerns. Traceroute measurements, in which traffic is injected into the network, can be abused for denial-of-service attacks disguised as legitimate measurement activity.

Measurement parameters MUST be carefully selected so that the measurements inject trivial amounts of additional traffic into the networks they measure. If they inject "too much" traffic, they can skew the results of the measurement, and in extreme cases cause congestion and denial of service.

The measurements themselves could be harmed by routers giving measurement traffic a different priority than "normal" traffic, or by an attacker injecting artificial measurement traffic. If routers can recognize measurement traffic and treat it separately, the measurements will not reflect actual user traffic. If an attacker injects artificial traffic that is accepted as legitimate, the loss rate will be artificially lowered. Therefore, the measurement methodologies SHOULD include appropriate techniques to reduce the probability that measurement traffic can be distinguished from "normal" traffic.

Authentication techniques, such as digital signatures, may be used where appropriate to guard against injected traffic attacks.

8.2. Securing Traceroute Measurement Information

Traceroute measurement information is not considered highly sensitive. Still, it may contain sensitive information on network paths, routing states, used IP addresses, and roundtrip times that operators of networks may want to protect for business or security reasons.

It is thus important to control access to information acquired by conducting traceroute measurements, particularly when transmitting it over a networks but also when storing it. It is RECOMMENDED that a transmission of traceroute measurement information over a network uses appropriate protection mechanisms for preserving privacy, integrity, and authenticity. It is further RECOMMENDED that secure authentication and authorization are used for protecting stored traceroute measurement information.

9. IANA Considerations

This document uses URNs to describe an XML namespace and an XML schema for traceroute measurement information storing and transmission, conforming to a registry mechanism described in [RFC3688]. Two URI assignments have been made.

1. Registration for the IPPM traceroute measurements namespace

   * URI: urn:ietf:params:xml:ns:traceroute-1.0

   * Registrant Contact: IESG

   * XML: None. Namespace URIs do not represent an XML.

2. Registration for the IPPM traceroute measurements schema

   * URI: urn:ietf:params:xml:schema:traceroute-1.0

   * Registrant Contact: IESG

   * XML: See Section 7 of this document.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

[RFC4560] Quittek, J. and K. White, "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", RFC 4560, June 2006.

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, March 2008.

10.2. Informative References

[IEEE.1003-1G.1997] Institute of Electrical and Electronics Engineers, "Protocol Independent Interfaces", IEEE Standard 1003.1G, March 1997.

[IPFIX] Sadasivan, G., "Architecture for IP Flow Information Export", Work in Progress, September 2006.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.

[RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.

[W3C.REC-xml-20060816]
Bray, T., Paoli, J., Maler, E., Sperberg-McQueen, C., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", World Wide Web Consortium FirstEdition REC-xml-20060816, August 2006, <http://www.w3.org/TR/2006/REC-xml-20060816>.

[W3C.REC-xmlschema-2-20041028]
Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

Appendix A. Traceroute Default Configuration Parameters

This section lists traceroute measurement configuration parameters as well as their defaults on various platforms and illustrates how widely they may vary. This document considers four major traceroute tool implementations and compares them based on configurable parameters and default values. The LINUX (SUSE 9.1), BSD (FreeBSD 7.0), and UNIX (SunOS 5.9) implementations are based on UDP datagrams, while the WINDOWS (XP SP2) one uses ICMP Echoes. The comparison is summarized in the following table, where an N/A in the option column means that such parameter is not configurable for the specific implementation. A comprehensive comparison of available implementations is outside the scope of this document; however, by sampling a few different implementations, it can be observed that they can differ quite significantly in terms of configurable parameters and also default values. Note that in the following table only those options that are available in at least two of the considered implementations are reported.

OS	Option	Description	Default
LINUX	-m	Specify the maximum TTL used in traceroute probes.	30
FreeBSD	-m		OS var
UNIX	-m		30
WINDOWS	-h		30
LINUX	-n	Display hop addresses numerically rather than symbolically.	-
FreeBSD	-n		-
UNIX	-n		-
WINDOWS	-d		-
LINUX	-w	Set the time to wait for a response to a probe.	3 sec
FreeBSD	-w		5 sec
UNIX	-w		5 sec
WINDOWS	-w		4 sec

LINUX	N/A	Specify a loose source route gateway (to direct the traceroute probes through routers not necessarily in the path).	-
FreeBSD	-g		-
UNIX	-g		-
WINDOWS	-g		-
LINUX	-p	Set the base UDP port number used in traceroute probes (UDP port = base + nhops - 1).	33434
FreeBSD	-p		33434
UNIX	-p		33434
WINDOWS	N/A		-
LINUX	-q	Set the number of probes per TTL.	3
FreeBSD	-q		3
UNIX	-q		3
WINDOWS	N/A		3
LINUX	-S	Set the IP source address in outgoing probes to the specified value.	IP address of the out interface
FreeBSD	-s		
UNIX	-s		
WINDOWS	N/A		
LINUX	-t	Set the Type of Service (TOS) in the probes to the specified value.	0
FreeBSD	-t		0
UNIX	-t		0
WINDOWS	N/A		0
LINUX	-v	Verbose output: received ICMP packets other than TIME_EXCEEDED and UNREACHABLE are listed.	-
FreeBSD	-v		-
UNIX	-v		-
WINDOWS	N/A		-

LINUX	N/A	Set the time (in msec) to pause between probes.	-
FreeBSD	-z		0
UNIX	-P		0
WINDOWS	N/A		-
LINUX	-r	Bypass the normal routing tables and send directly to a host on attached network.	-
FreeBSD	-r		-
UNIX	-r		-
WINDOWS	N/A		-
LINUX	-f	Set the initial TTL for the first probe.	1
FreeBSD	-f		1
UNIX	-f		1
WINDOWS	N/A		1
LINUX	-F	Set the "don’t fragment" bit.	-
FreeBSD	-F		-
UNIX	-F		-
WINDOWS	N/A		-
LINUX	N/A	Enable socket level debugging.	-
FreeBSD	-d		-
UNIX	-d		-
WINDOWS	N/A		-
LINUX	N/A	Use ICMP Echoes instead of UDP datagrams.	-
FreeBSD	-I		-
UNIX	-I		-
WINDOWS	N/A		-

LINUX	-I	Specify a network interface to obtain the IP address for outgoing IP packets (alternative to option -s).	-
FreeBSD	-i		-
UNIX	-i		-
WINDOWS	N/A		-
LINUX	N/A	Toggle checksum.	-
FreeBSD	-x		-
UNIX	-x		-
WINDOWS	N/A		-
LINUX	-	As optional last parameter, LINUX, FreeBSD, and UNIX implementations allow specifying the probe datagram length for outgoing probes.	Depends on implementation.
FreeBSD	-		
UNIX	-		
WINDOWS	N/A		

A.1. Alternative Traceroute Implementations

As stated above, the widespread use of firewalls might prevent UDP- or ICMP-based traceroutes to completely trace the path to a destination since traceroute probes might end up being filtered. In some cases, such limitation might be overcome by sending instead TCP packets to specific ports that hosts located behind the firewall are listening for connections on. TCP-based implementations use TCP, SYN, or FIN probes and listen for TIME_EXCEEDED messages, TCP RESET, and other messages from firewalls and gateways on the path. On the other hand, some firewalls filter out TCP SYN packets to prevent denial-of-service attacks; therefore, the actual advantage of using TCP instead of UDP traceroute depends mainly on firewall configurations, which are not known in advance. A detailed analysis of TCP-based traceroute tools and measurements is outside the scope of this document; regardless, for completeness reasons, the information model also supports the storing of TCP-based traceroute measurements.

Appendix B. Known Problems with Traceroute

B.1. Compatibility between Traceroute Measurement Results and IPPM Metrics

Because of implementation choices, a known inconsistency exists between the round-trip delay metric defined by the IPPM working group in RFC 2681 and the results returned by the current traceroute tool implementations. Unfortunately, it is unlikely that the traceroute tool implementations will implement the standard definition in the near future. The only possibility is therefore to compare results of different traceroute measurements with each other; in order to do this, specifications both of the operating system (name and version) and of the traceroute tool version used were added to the metadata elements in order to help in comparing metrics between two different traceroute measurement results (if run using the same operating system and the same version of the tool). Moreover, the traceroute tool has built-in configurable mechanisms like timeouts and can experience problems related to the crossing of firewalls; therefore, some of the packets that traceroute sends out end up being timeout or filtered. As a consequence, it might not be possible to trace the path to a node or there might not be a complete enough set of probes describing the RTT to reach it.

Appendix C. Differences to DISMAN-TRACEROUTE-MIB

For performing remote traceroute operations at managed node, the IETF has standardized the DISMAN-TRACEROUTE-MIB module in [RFC4560]. This module allows:

- retrieving capability information of the traceroute tool implementation at the managed node;
- configuring traceroute measurements to be performed;
- retrieving information about ongoing and completed traceroute measurements;
- retrieving traceroute measurement statistics.

The traceroute storage format described in this document has significant overlaps with this MIB module. Particularly, the models for the traceroute measurement configuration and for the results from completed measurements are almost identical. But for other parts of the DISMAN-TRACEROUTE MIB module there is no need to model them in a traceroute measurement storage format. Particularly, the capability information, information about ongoing measurements, and measurement statistics are not covered by the DISMAN traceroute storage model.

Concerning traceroute measurements and their results, there are structural differences between the two models caused by the different choices for the encoding of the specification. For DISMAN-TRACEROUTE-MIB, the Structure of Management Information (SMIv2, STD 58, RFC 2578 [RFC2578]) was used, while the IPPM traceroute measurement data model is encoded using XML.

This difference in structure implies that the DISMAN-TRACEROUTE-MIB module contains SMI-specific information elements (managed objects) that concern tables of managed objects (specification, entry creation and deletion, status retrieval) that are not required for the XML-encoded traceroute measurement data model.

But for most of the remaining information elements that concern configuration of traceroute measurements and results of completed measurements, the semantics are identical between the DISMAN-TRACEROUTE-MIB module and the traceroute measurement data model. There are very few exceptions to this; these are listed below. Also, naming of information elements is identical between both models with a few exceptions. For the traceroute measurement data model, a few information elements have been added, some because of the different structure and some to provide additional information on completed measurements.

C.1. Scope

There are some basic differences in nature and application between MIB modules and XML documents. This results in two major differences of scope between the DISMAN-TRACEROUTE-MIB module and the traceroute measurement data model.

The first difference is the "traceRouteResultsTable" contained in the DISMAN-TRACEROUTE-MIB module. This table allows online observation of status and progress of an ongoing traceroute measurement. This highly dynamic information is not included in the traceroute measurement data model because it has not been envisioned to use the model for dynamically reporting progress of individual traceroute measurements. The traceroute measurement data model is rather intended to be used for reporting completed traceroute measurements.

The second difference is due to the fact that information in a MIB is typically tied to a local node hosting the MIB instance. The "RequestMetadata" element specified in the traceroute measurement data model can be used for specifying a measurement request that may be applied to several probes in a network. This concept does not exist in the DISMAN-TRACEROUTE-MIB module.

For the remaining elements in the DISMAN-TRACEROUTE-MIB module and in the traceroute measurement data model, there is a very good match between the two worlds. The "traceRouteCtlTable" corresponds to the "MeasurementMetadata" element, and the combination of the "traceRouteProbeHistoryTable" and the "traceRouteHopsTable" corresponds to a collection of "MeasurementResult" elements.

C.2. Naming

Basically, names in both models are chosen using the same naming conventions.

For the traceroute measurement configuration information, all names, such as "CtlProbesPerHop", are identical in both models except for the traceRoute prefix that was removed to avoid unnecessary redundancy in the XML file and for "CtlDataSize", which was renamed to "CtlProbeDataSize" for clarification in the traceroute measurement data model.

Results of measurements in the DISMAN-TRACEROUTE-MIB modules are distributed over two tables, the "traceRouteResultsTable" contains mainly information about ongoing measurements and the "traceRouteProbeHistoryTable" contains only information about completed measurements. According to the SMIv2 naming conventions, names of information elements in these tables have different prefixes ("traceRouteResults" and "traceRouteProbeHistory"). Since the traceroute measurement data model only reports on completed measurements, this separation is not needed anymore and the prefix "Results" is used for all related information elements.

Beyond that, there are only a few changes in element names. The renaming actions include:

- "traceRouteProbeHistoryResponse" to "ProbeRoundTripTime";
- "traceRouteProbeHistoryHAddr" to "HopAddr";
- "traceRouteProbeHistoryTime" to "ResultsEndDateAndTime";
- "traceRouteProbeHistoryLastRC" to "ResultsHopRawOutputData".

C.3. Semantics

The semantics were changed for two information elements only.

For "traceRouteProbeHistoryResponse" in the DISMAN-TRACEROUTE-MIB, a value of 0 indicates that it is not possible to transmit a probe. For the traceroute measurement data model, a value of 0 for element

"RoundTripTime" indicates that the measured time was less than one millisecond. For the case that it was not possible to transmit a probe, a string is used that indicates the problem.

For "traceRouteCtlIfIndex" in the DISMAN-TRACEROUTE-MIB, a value of 0 indicates that the option to set the index is not available. This was translated to the traceroute measurement data model, such that a value of 0 for this element indicates that the used interface is unknown.

The element "traceRouteProbeHistoryLastRC" in the DISMAN-TRACEROUTE-MIB was replaced by element "ResultsHopRawOutputData". While "traceRouteProbeHistoryLastRC" just reports a reply code, "ResultsHopRawOutputData" reports the full raw output data (per hop) produced by the traceroute measurement that was used.

C.4. Additional Information Elements

Only a few information elements have been added to the model of the DISMAN-TRACEROUTE-MIB module.

- For providing information on the MPLS label stack entries of a probe in the traceroute measurement path, "MPLSLabelStackEntry" was added.

- For providing additional timestamp beyond "ResultsEndDateAndTime", "ResultsStartDateAndTime" and "Time" were added.

- For providing DNS names at the time of the execution of the traceroute for each "HopAddr" (which may change over time), "HopName" was added.

Appendix D. Traceroute Examples with XML Representation

This section shows some examples of traceroute applications. In addition, the encoding of requests and results is shown for some of those examples. Also, note that in these XML examples some lines appear wrapped due to the limited length of line.

A typical traceroute on a LINUX system looks like the following:

```
# traceroute  -f 4 www.example 1500
traceroute to ww.example (192.0.2.42), 30 hops max, 1500-byte packets
 5  out.host1.example (192.0.2.254)  6.066 ms   5.625 ms   6.095 ms
 6  rtr4.host6.example (192.0.2.142)  6.979 ms   6.221 ms   7.368 ms
 7  hop7.rtr9.example (192.0.2.11)  16.165 ms   15.347 ms   15.514 ms
 8  192.0.2.222 (192.0.2.222)  32.796 ms   28.723 ms   26.988 ms
 9  in.example (192.0.2.123)  15.861 ms   16.262 ms   17.610 ms
10  in.example (192.0.2.123)(N!)  17.391 ms * *
```

This traceroute ignores the first 4 hops and uses 1500-byte packets including the header. It does not reach its goal since the last listed hop says that the network is not reachable (N!). The XML representation for this trace follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<traceRoute xmlns="urn:ietf:params:xml:ns:traceroute-1.0">
  <RequestMetadata>
    <TestName>Example 1</TestName>
    <OSName/>
    <OSVersion/>
    <ToolVersion/>
    <ToolName/>
    <CtlTargetAddress>
      <inetAddressDns>www.example</inetAddressDns>
    </CtlTargetAddress>
    <CtlBypassRouteTable/>
    <CtlProbeDataSize>1472</CtlProbeDataSize>
    <CtlTimeOut/>
    <CtlProbesPerHop/>
    <CtlPort/>
    <CtlMaxTtl/>
    <CtlDSField/>
    <CtlSourceAddress>
      <inetAddressUnknown/>
    </CtlSourceAddress>
    <CtlIfIndex/>
    <CtlMiscOptions/>
    <CtlMaxFailures/>
    <CtlDontFragment/>
    <CtlInitialTtl>4</CtlInitialTtl>
    <CtlDescr>Show how it encodes in XML</CtlDescr>
    <CtlType><UDP/></CtlType>
  </RequestMetadata>
  <Measurement>
    <MeasurementMetadata>
      <TestName>Example 1</TestName>
      <OSName>Linux</OSName>
      <OSVersion>2.6.16.54-0.2.5-smp i386</OSVersion>
      <ToolVersion>1.0</ToolVersion>
      <ToolName>traceroute</ToolName>
      <CtlTargetAddress>
        <inetAddressDns>www.example</inetAddressDns>
      </CtlTargetAddress>
      <CtlBypassRouteTable/>
      <CtlProbeDataSize>1472</CtlProbeDataSize>
      <CtlTimeOut/>
      <CtlProbesPerHop/>
      <CtlPort/>
```

```
        <CtlMaxTtl/>
        <CtlDSField/>
        <CtlSourceAddress>
          <inetAddressIpv4>192.0.2.1</inetAddressIpv4>
        </CtlSourceAddress>
        <CtlIfIndex>2</CtlIfIndex>
        <CtlMiscOptions/>
        <CtlMaxFailures/>
        <CtlDontFragment/>
        <CtlInitialTtl>4</CtlInitialTtl>
        <CtlDescr>Show how it encodes in XML</CtlDescr>
        <CtlType><UDP/></CtlType>
      </MeasurementMetadata>
      <MeasurementResult>
        <TestName>Example 1</TestName>
        <ResultsStartDateAndTime>2008-05-16T14:22:34+02:00</ResultsStar
   tDateAndTime>
        <ResultsIpTgtAddr>
          <inetAddressIpv4>192.0.2.42</inetAddressIpv4>
        </ResultsIpTgtAddr>
        <ProbeResults>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.254</inetAddressIpv4>
              </HopAddr>
              <HopName>out.host1.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>6</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:35+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.254</inetAddressIpv4>
              </HopAddr>
              <HopName>out.host1.example</HopName>
              <ProbeRoundTripTime><roundTripTime>5</roundTripTime></Pro
   beRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:35+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.254</inetAddressIpv4>
              </HopAddr>
              <HopName>out.host1.example</HopName>
```

```
              <ProbeRoundTripTime>
                <roundTripTime>6</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:35+02:00</Time>
            </probe>
            <HopRawOutputData> 5  out.host1.example (192.0.2.254)  6.06
   6 ms   5.625 ms   6.095 ms</HopRawOutputData>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.142</inetAddressIpv4>
              </HopAddr>
              <HopName>rtr4.host6.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>6</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:36+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.142</inetAddressIpv4>
              </HopAddr>
              <HopName>rtr4.host6.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>6</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:36+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.142</inetAddressIpv4>
              </HopAddr>
              <HopName>rtr4.host6.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>7</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:37+02:00</Time>
            </probe>
            <HopRawOutputData> 6  rtr4.host6.example (192.0.2.142)  6.9
   79 ms   6.221 ms   7.368 ms</HopRawOutputData>
          </hop>
          <hop>
            <probe>
```

```
              <HopAddr>
                <inetAddressIpv4>192.0.2.11</inetAddressIpv4>
              </HopAddr>
              <HopName>hop7.rtr9.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>16</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:37+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.11</inetAddressIpv4>
              </HopAddr>
              <HopName>hop7.rtr9.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>15</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:38+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.11</inetAddressIpv4>
              </HopAddr>
              <HopName>hop7.rtr9.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>15</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:38+02:00</Time>
            </probe>
            <HopRawOutputData> 7  hop7.rtr9.example (192.0.2.11)  16.16
   5 ms   15.347 ms   15.514 ms</HopRawOutputData>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.222</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>32</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:39+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
```

```
                <inetAddressIpv4>192.0.2.222</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>38</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:39+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.222</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>26</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:39+02:00</Time>
            </probe>
            <HopRawOutputData> 8  192.0.2.222 (192.0.2.222)  32.796 ms
     28.723 ms   26.988 ms</HopRawOutputData>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <HopName>in.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>15</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:40+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <HopName>in.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>16</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:40+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
```

```
              <HopName>in.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>17</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-16T14:22:40+02:00</Time>
            </probe>
            <HopRawOutputData> 9  in.example (192.0.2.123)  15.861 ms
    16.262 ms   17.610 ms</HopRawOutputData>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <HopName>in.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>17</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>noRouteToTarget</ResponseStatus>
              <Time>2008-05-16T14:22:41+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <HopName>in.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTimeNotAvailable/>
              </ProbeRoundTripTime>
              <ResponseStatus>requestTimedOut</ResponseStatus>
              <Time>2008-05-16T14:22:44+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <HopName>in.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTimeNotAvailable/>
              </ProbeRoundTripTime>
              <ResponseStatus>requestTimedOut</ResponseStatus>
              <Time>2008-05-16T14:22:44+02:00</Time>
            </probe>
            <HopRawOutputData>10  in.example (192.0.2.123)(N!)  17.391
   ms * *</HopRawOutputData>
          </hop>
        </ProbeResults>
```

```
      <ResultsEndDateAndTime>2008-05-16T14:22:44+02:00</ResultsEndDat
eAndTime>
    </MeasurementResult>
  </Measurement>
</traceRoute>
```

The second example was generated on an OpenBSD system. On that system, the traceroute looks like the following:

```
# traceroute -P tcp w2.example 128

traceroute to w2.example (192.0.2.254), 64 hops max, 160-byte packets
 1  router1.example.org (192.0.2.22)  0.486 ms  0.486 ms  0.482 ms
 2  router7.example.org (192.0.2.1)  3.27 ms  1.420 ms  1.873 ms
 3  hop0.c.example (192.0.2.105)  3.177 ms  3.258 ms  3.859 ms
 4  hop6.c.example (192.0.2.107)  5.994 ms  4.607 ms  5.678 ms
 5  hop3.c.example (192.0.2.111)  20.341 ms  20.732 ms  19.505 ms
 6  in.example.net (192.0.2.222)  20.333 ms  19.174 ms  19.856 ms
 7  egress.example.net (192.0.2.227)  20.268 ms  21.79 ms  19.992 ms
 8  routerin.example (192.0.2.253)  19.983 ms  19.931 ms  19.894 ms
 9  routerdmz.example (192.0.2.249)  20.943 ms !X *  19.829 ms !X
```

It was executed with the TCP protocol and 128-byte packets (plus header). The traceroute ended at hop 9 because the packets are administratively filtered (!X). A corresponding XML representation follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<traceRoute xmlns="urn:ietf:params:xml:ns:traceroute-1.0">
  <RequestMetadata>
    <TestName>Example 2</TestName>
    <OSName/>
    <OSVersion/>
    <ToolVersion/>
    <ToolName/>
    <CtlTargetAddress>
      <inetAddressDns>w2.example</inetAddressDns>
    </CtlTargetAddress>
    <CtlBypassRouteTable/>
    <CtlProbeDataSize>128</CtlProbeDataSize>
    <CtlTimeOut/>
    <CtlProbesPerHop/>
    <CtlPort/>
    <CtlMaxTtl/>
    <CtlDSField/>
    <CtlSourceAddress>
      <inetAddressUnknown/>
    </CtlSourceAddress>
    <CtlIfIndex/>
    <CtlMiscOptions/>
```

```
      <CtlMaxFailures/>
      <CtlDontFragment/>
      <CtlInitialTtl/>
      <CtlDescr>Show how it encodes in XML</CtlDescr>
      <CtlType><TCP/></CtlType>
    </RequestMetadata>
    <Measurement>
      <MeasurementMetadata>
        <TestName>Example 2</TestName>
        <OSName>OpenBSD</OSName>
        <OSVersion>4.1 i386</OSVersion>
        <ToolVersion></ToolVersion>
        <ToolName>traceroute</ToolName>
        <CtlTargetAddress>
          <inetAddressDns>w2.example</inetAddressDns>
        </CtlTargetAddress>
        <CtlBypassRouteTable/>
        <CtlProbeDataSize>128</CtlProbeDataSize>
        <CtlTimeOut/>
        <CtlProbesPerHop/>
        <CtlPort/>
        <CtlMaxTtl/>
        <CtlDSField/>
        <CtlSourceAddress>
          <inetAddressIpv4>192.0.2.42</inetAddressIpv4>
        </CtlSourceAddress>
        <CtlIfIndex>1</CtlIfIndex>
        <CtlMiscOptions/>
        <CtlMaxFailures/>
        <CtlDontFragment/>
        <CtlInitialTtl/>
        <CtlDescr>Show how it encodes in XML</CtlDescr>
        <CtlType><TCP/></CtlType>
      </MeasurementMetadata>
      <MeasurementResult>
        <TestName>Example 2</TestName>
        <ResultsStartDateAndTime>2008-05-14T09:57:11+02:00</ResultsStar
   tDateAndTime>
        <ResultsIpTgtAddr>
          <inetAddressIpv4>192.0.2.254</inetAddressIpv4>
        </ResultsIpTgtAddr>
        <ProbeResults>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.22</inetAddressIpv4>
              </HopAddr>
              <HopName>router1.example.org</HopName>
```

```
            <ProbeRoundTripTime>
              <roundTripTime>0</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:13+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.22</inetAddressIpv4>
            </HopAddr>
            <HopName>router1.example.org</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>0</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:13+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.22</inetAddressIpv4>
            </HopAddr>
            <HopName>router1.example.org</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>0</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:13+02:00</Time>
          </probe>
        </hop>
        <hop>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.1</inetAddressIpv4>
            </HopAddr>
            <HopName>router7.example.org</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>3</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:13+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.1</inetAddressIpv4>
            </HopAddr>
            <HopName>router7.example.org</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>1</roundTripTime>
```

```
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:13+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.1</inetAddressIpv4>
            </HopAddr>
            <HopName>router7.example.org</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>1</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:14+02:00</Time>
          </probe>
        </hop>
        <hop>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.105</inetAddressIpv4>
            </HopAddr>
            <HopName>hop0.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>3</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:14+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.105</inetAddressIpv4>
            </HopAddr>
            <HopName>hop0.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>3</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:14+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.105</inetAddressIpv4>
            </HopAddr>
            <HopName>hop0.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>3</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
```

```
            <Time>2008-05-14T09:57:14+02:00</Time>
          </probe>
        </hop>
        <hop>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.107</inetAddressIpv4>
            </HopAddr>
            <HopName>hop6.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>5</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:15+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.107</inetAddressIpv4>
            </HopAddr>
            <HopName>hop6.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>4</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:16+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.107</inetAddressIpv4>
            </HopAddr>
            <HopName>hop6.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>5</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:16+02:00</Time>
          </probe>
        </hop>
        <hop>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.111</inetAddressIpv4>
            </HopAddr>
            <HopName>hop3.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>20</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
```

```
            <Time>2008-05-14T09:57:17+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.111</inetAddressIpv4>
            </HopAddr>
            <HopName>hop3.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>20</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:18+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.111</inetAddressIpv4>
            </HopAddr>
            <HopName>hop3.c.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>19</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:19+02:00</Time>
          </probe>
        </hop>
        <hop>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.222</inetAddressIpv4>
            </HopAddr>
            <HopName>in.example.net</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>20</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:20+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.222</inetAddressIpv4>
            </HopAddr>
            <HopName>in.example.net</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>19</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>responseReceived</ResponseStatus>
            <Time>2008-05-14T09:57:20+02:00</Time>
          </probe>
```

```
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.222</inetAddressIpv4>
              </HopAddr>
              <HopName>in.example.net</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>19</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T09:57:21+02:00</Time>
            </probe>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.227</inetAddressIpv4>
              </HopAddr>
              <HopName>egress.example.net</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>20</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T09:57:22+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.227</inetAddressIpv4>
              </HopAddr>
              <HopName>egress.example.net</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>21</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T09:57:22+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.227</inetAddressIpv4>
              </HopAddr>
              <HopName>egress.example.net</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>19</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T09:57:23+02:00</Time>
            </probe>
          </hop>
          <hop>
```

```
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.253</inetAddressIpv4>
              </HopAddr>
              <HopName>routerin.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>19</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T09:57:24+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.253</inetAddressIpv4>
              </HopAddr>
              <HopName>routerin.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>19</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T09:57:24+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.253</inetAddressIpv4>
              </HopAddr>
              <HopName>routerin.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>19</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T09:57:25+02:00</Time>
            </probe>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.249</inetAddressIpv4>
              </HopAddr>
              <HopName>routerdmz.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>20</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>unknown</ResponseStatus>
              <Time>2008-05-14T09:57:26+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
```

```
              <inetAddressIpv4>192.0.2.249</inetAddressIpv4>
            </HopAddr>
            <HopName>routerdmz.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTimeNotAvailable/>
            </ProbeRoundTripTime>
            <ResponseStatus>requestTimedOut</ResponseStatus>
            <Time>2008-05-14T09:57:26+02:00</Time>
          </probe>
          <probe>
            <HopAddr>
              <inetAddressIpv4>192.0.2.249</inetAddressIpv4>
            </HopAddr>
            <HopName>routerdmz.example</HopName>
            <ProbeRoundTripTime>
              <roundTripTime>19</roundTripTime>
            </ProbeRoundTripTime>
            <ResponseStatus>unknown</ResponseStatus>
            <Time>2008-05-14T09:57:30+02:00</Time>
          </probe>
        </hop>
      </ProbeResults>
      <ResultsEndDateAndTime>2008-05-14T09:57:30+02:00</ResultsEndDat
eAndTime>
    </MeasurementResult>
  </Measurement>
</traceRoute>
```

The third and last example is based on the Microsoft Windows pendant of traceroute. On an MS Windows system, the command is called "tracert" and typically looks as follows:

```
# tracert -h 10 www.example.org

Tracing route to www.example.org [192.0.2.11]
over a maximum of 10 hops:

  1     1 ms     1 ms     8 ms  192.0.2.99
  2    <1 ms    <1 ms    <1 ms  r1.provider4.example [192.0.2.102]
  3    <1 ms    <1 ms    <1 ms  rtr8.provider8.example [192.0.2.254]
  4     1 ms     1 ms     1 ms  hop11.hoster7.example [192.0.2.4]
  5     2 ms     3 ms     1 ms  sw6.provider2.example [192.0.2.201]
  6     3 ms     3 ms     3 ms  out.provider2.example [192.0.2.111]
  7     *        6 ms     5 ms  192.0.2.123
  8     5 ms     5 ms     5 ms  192.0.2.42
  9    94 ms    95 ms    95 ms  ingress.example.org [192.0.2.199]
 10   168 ms   169 ms   169 ms  192.0.2.44
```

Trace complete.

In this example, the trace was limited to 10 hops, so the tenth and last hop of this example was not the final destination. Applying the XML schema defined in this document, the trace could look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<traceRoute xmlns="urn:ietf:params:xml:ns:traceroute-1.0">
  <RequestMetadata>
    <TestName>Example 3</TestName>
    <OSName/>
    <OSVersion/>
    <ToolVersion/>
    <ToolName/>
    <CtlTargetAddress>
      <inetAddressDns>www.example.org</inetAddressDns>
    </CtlTargetAddress>
    <CtlBypassRouteTable/>
    <CtlProbeDataSize/>
    <CtlTimeOut/>
    <CtlProbesPerHop/>
    <CtlPort/>
    <CtlMaxTtl>10</CtlMaxTtl>
    <CtlDSField/>
    <CtlSourceAddress>
      <inetAddressUnknown/>
    </CtlSourceAddress>
    <CtlIfIndex/>
    <CtlMiscOptions/>
    <CtlMaxFailures/>
    <CtlDontFragment/>
    <CtlInitialTtl/>
    <CtlDescr>Show how it encodes in XML</CtlDescr>
    <CtlType><TCP/></CtlType>
  </RequestMetadata>
  <Measurement>
    <MeasurementMetadata>
      <TestName>Example 3</TestName>
      <OSName>Windows</OSName>
      <OSVersion>XP SP2 32-bit</OSVersion>
      <ToolVersion></ToolVersion>
      <ToolName>tracert</ToolName>
      <CtlTargetAddress>
        <inetAddressDns>www.example.org</inetAddressDns>
      </CtlTargetAddress>
      <CtlBypassRouteTable/>
      <CtlProbeDataSize/>
      <CtlTimeOut/>
      <CtlProbesPerHop/>
```

```
        <CtlPort/>
        <CtlMaxTtl>10</CtlMaxTtl>
        <CtlDSField/>
        <CtlSourceAddress>
          <inetAddressIpv4>192.0.2.142</inetAddressIpv4>
        </CtlSourceAddress>
        <CtlIfIndex>3</CtlIfIndex>
        <CtlMiscOptions/>
        <CtlMaxFailures/>
        <CtlDontFragment/>
        <CtlInitialTtl/>
        <CtlDescr>Show how it encodes in XML</CtlDescr>
        <CtlType><TCP/></CtlType>
      </MeasurementMetadata>
      <MeasurementResult>
        <TestName>Example 3</TestName>
        <ResultsStartDateAndTime>2008-05-14T11:03:09+02:00</ResultsStar
  tDateAndTime>
        <ResultsIpTgtAddr>
          <inetAddressIpv4>192.0.2.11</inetAddressIpv4>
        </ResultsIpTgtAddr>
        <ProbeResults>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.99</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>1</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:09+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.99</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>1</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:09+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.99</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
```

```
                &lt;roundTripTime&gt;8&lt;/roundTripTime&gt;
              &lt;/ProbeRoundTripTime&gt;
              &lt;ResponseStatus&gt;responseReceived&lt;/ResponseStatus&gt;
              &lt;Time&gt;2008-05-14T11:03:09+02:00&lt;/Time&gt;
            &lt;/probe&gt;
          &lt;/hop&gt;
          &lt;hop&gt;
            &lt;probe&gt;
              &lt;HopAddr&gt;
                &lt;inetAddressIpv4&gt;192.0.2.102&lt;/inetAddressIpv4&gt;
              &lt;/HopAddr&gt;
              &lt;HopName&gt;r1.provider4.example&lt;/HopName&gt;
              &lt;ProbeRoundTripTime&gt;
                &lt;roundTripTime&gt;0&lt;/roundTripTime&gt;
              &lt;/ProbeRoundTripTime&gt;
              &lt;ResponseStatus&gt;responseReceived&lt;/ResponseStatus&gt;
              &lt;Time&gt;2008-05-14T11:03:09+02:00&lt;/Time&gt;
            &lt;/probe&gt;
            &lt;probe&gt;
              &lt;HopAddr&gt;
                &lt;inetAddressIpv4&gt;192.0.2.102&lt;/inetAddressIpv4&gt;
              &lt;/HopAddr&gt;
              &lt;HopName&gt;r1.provider4.example&lt;/HopName&gt;
              &lt;ProbeRoundTripTime&gt;
                &lt;roundTripTime&gt;0&lt;/roundTripTime&gt;
              &lt;/ProbeRoundTripTime&gt;
              &lt;ResponseStatus&gt;responseReceived&lt;/ResponseStatus&gt;
              &lt;Time&gt;2008-05-14T11:03:09+02:00&lt;/Time&gt;
            &lt;/probe&gt;
            &lt;probe&gt;
              &lt;HopAddr&gt;
                &lt;inetAddressIpv4&gt;192.0.2.102&lt;/inetAddressIpv4&gt;
              &lt;/HopAddr&gt;
              &lt;HopName&gt;r1.provider4.example&lt;/HopName&gt;
              &lt;ProbeRoundTripTime&gt;
                &lt;roundTripTime&gt;0&lt;/roundTripTime&gt;
              &lt;/ProbeRoundTripTime&gt;
              &lt;ResponseStatus&gt;responseReceived&lt;/ResponseStatus&gt;
              &lt;Time&gt;2008-05-14T11:03:09+02:00&lt;/Time&gt;
            &lt;/probe&gt;
          &lt;/hop&gt;
          &lt;hop&gt;
            &lt;probe&gt;
              &lt;HopAddr&gt;
                &lt;inetAddressIpv4&gt;192.0.2.254&lt;/inetAddressIpv4&gt;
              &lt;/HopAddr&gt;
              &lt;HopName&gt;rtr8.provider8.example&lt;/HopName&gt;
              &lt;ProbeRoundTripTime&gt;
```

```
            <roundTripTime>0</roundTripTime>
          </ProbeRoundTripTime>
          <ResponseStatus>responseReceived</ResponseStatus>
          <Time>2008-05-14T11:03:09+02:00</Time>
        </probe>
        <probe>
          <HopAddr>
            <inetAddressIpv4>192.0.2.254</inetAddressIpv4>
          </HopAddr>
          <HopName>rtr8.provider8.example</HopName>
          <ProbeRoundTripTime>
            <roundTripTime>0</roundTripTime>
          </ProbeRoundTripTime>
          <ResponseStatus>responseReceived</ResponseStatus>
          <Time>2008-05-14T11:03:09+02:00</Time>
        </probe>
        <probe>
          <HopAddr>
            <inetAddressIpv4>192.0.2.254</inetAddressIpv4>
          </HopAddr>
          <HopName>rtr8.provider8.example</HopName>
          <ProbeRoundTripTime>
            <roundTripTime>0</roundTripTime>
          </ProbeRoundTripTime>
          <ResponseStatus>responseReceived</ResponseStatus>
          <Time>2008-05-14T11:03:09+02:00</Time>
        </probe>
      </hop>
      <hop>
        <probe>
          <HopAddr>
            <inetAddressIpv4>192.0.2.4</inetAddressIpv4>
          </HopAddr>
          <HopName>hop11.hoster7.example</HopName>
          <ProbeRoundTripTime>
            <roundTripTime>1</roundTripTime>
          </ProbeRoundTripTime>
          <ResponseStatus>responseReceived</ResponseStatus>
          <Time>2008-05-14T11:03:09+02:00</Time>
        </probe>
        <probe>
          <HopAddr>
            <inetAddressIpv4>192.0.2.4</inetAddressIpv4>
          </HopAddr>
          <HopName>hop11.hoster7.example</HopName>
          <ProbeRoundTripTime>
            <roundTripTime>1</roundTripTime>
          </ProbeRoundTripTime>
```

```
        <ResponseStatus>responseReceived</ResponseStatus>
        <Time>2008-05-14T11:03:10+02:00</Time>
      </probe>
      <probe>
        <HopAddr>
          <inetAddressIpv4>192.0.2.4</inetAddressIpv4>
        </HopAddr>
        <HopName>hop11.hoster7.example</HopName>
        <ProbeRoundTripTime>
          <roundTripTime>1</roundTripTime>
        </ProbeRoundTripTime>
        <ResponseStatus>responseReceived</ResponseStatus>
        <Time>2008-05-14T11:03:10+02:00</Time>
      </probe>
    </hop>
    <hop>
      <probe>
        <HopAddr>
          <inetAddressIpv4>192.0.2.201</inetAddressIpv4>
        </HopAddr>
        <HopName>sw6.provider2.example</HopName>
        <ProbeRoundTripTime>
          <roundTripTime>2</roundTripTime>
        </ProbeRoundTripTime>
        <ResponseStatus>responseReceived</ResponseStatus>
        <Time>2008-05-14T11:03:10+02:00</Time>
      </probe>
      <probe>
        <HopAddr>
          <inetAddressIpv4>192.0.2.201</inetAddressIpv4>
        </HopAddr>
        <HopName>sw6.provider2.example</HopName>
        <ProbeRoundTripTime>
          <roundTripTime>3</roundTripTime>
        </ProbeRoundTripTime>
        <ResponseStatus>responseReceived</ResponseStatus>
        <Time>2008-05-14T11:03:11+02:00</Time>
      </probe>
      <probe>
        <HopAddr>
          <inetAddressIpv4>192.0.2.201</inetAddressIpv4>
        </HopAddr>
        <HopName>sw6.provider2.example</HopName>
        <ProbeRoundTripTime>
          <roundTripTime>1</roundTripTime>
        </ProbeRoundTripTime>
        <ResponseStatus>responseReceived</ResponseStatus>
        <Time>2008-05-14T11:03:11+02:00</Time>
```

```
            </probe>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.111</inetAddressIpv4>
              </HopAddr>
              <HopName>out.provider2.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>3</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:11+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.111</inetAddressIpv4>
              </HopAddr>
              <HopName>out.provider2.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>3</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:11+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.111</inetAddressIpv4>
              </HopAddr>
              <HopName>out.provider2.example</HopName>
              <ProbeRoundTripTime>
                <roundTripTime>3</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:12+02:00</Time>
            </probe>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTimeNotAvailable/>
              </ProbeRoundTripTime>
              <ResponseStatus>requestTimedOut</ResponseStatus>
              <Time>2008-05-14T11:03:14+02:00</Time>
            </probe>
```

```
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>6</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:15+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.123</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>5</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:16+02:00</Time>
            </probe>
          </hop>
          <hop>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.42</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>5</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:17+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.42</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>5</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:17+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.42</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>5</roundTripTime>
```

```
      </ProbeRoundTripTime>
      <ResponseStatus>responseReceived</ResponseStatus>
      <Time>2008-05-14T11:03:17+02:00</Time>
    </probe>
  </hop>
  <hop>
    <probe>
      <HopAddr>
        <inetAddressIpv4>192.0.2.199</inetAddressIpv4>
      </HopAddr>
      <HopName>ingress.example.org</HopName>
      <ProbeRoundTripTime>
        <roundTripTime>94</roundTripTime>
      </ProbeRoundTripTime>
      <ResponseStatus>responseReceived</ResponseStatus>
      <Time>2008-05-14T11:03:19+02:00</Time>
    </probe>
    <probe>
      <HopAddr>
        <inetAddressIpv4>192.0.2.199</inetAddressIpv4>
      </HopAddr>
      <HopName>ingress.example.org</HopName>
      <ProbeRoundTripTime>
        <roundTripTime>95</roundTripTime>
      </ProbeRoundTripTime>
      <ResponseStatus>responseReceived</ResponseStatus>
      <Time>2008-05-14T11:03:19+02:00</Time>
    </probe>
    <probe>
      <HopAddr>
        <inetAddressIpv4>192.0.2.199</inetAddressIpv4>
      </HopAddr>
      <HopName>ingress.example.org</HopName>
      <ProbeRoundTripTime>
        <roundTripTime>95</roundTripTime>
      </ProbeRoundTripTime>
      <ResponseStatus>responseReceived</ResponseStatus>
      <Time>2008-05-14T11:03:19+02:00</Time>
    </probe>
  </hop>
  <hop>
    <probe>
      <HopAddr>
        <inetAddressIpv4>192.0.2.44</inetAddressIpv4>
      </HopAddr>
      <ProbeRoundTripTime>
        <roundTripTime>168</roundTripTime>
      </ProbeRoundTripTime>
```

```
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:20+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.44</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>169</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:21+02:00</Time>
            </probe>
            <probe>
              <HopAddr>
                <inetAddressIpv4>192.0.2.44</inetAddressIpv4>
              </HopAddr>
              <ProbeRoundTripTime>
                <roundTripTime>169</roundTripTime>
              </ProbeRoundTripTime>
              <ResponseStatus>responseReceived</ResponseStatus>
              <Time>2008-05-14T11:03:23+02:00</Time>
            </probe>
          </hop>
        </ProbeResults>
        <ResultsEndDateAndTime>2008-05-14T11:03:23+02:00</ResultsEndDat
   eAndTime>
      </MeasurementResult>
    </Measurement>
   </traceRoute>
```

The three examples given in this section are intended to give an impression of how a trace could be represented in XML. The representation generated by an implementation may differ from the examples here depending on the system and the capabilities of the traceroute implementation.

Authors' Addresses

Saverio Niccolini
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany
Phone: +49 (0) 6221 4342 118
EMail: saverio.niccolini@nw.neclab.eu
URI: http://www.nw.neclab.eu

Sandra Tartarelli
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany
Phone: +49 (0) 6221 4342 132
EMail: sandra.tartarelli@nw.neclab.eu
URI: http://www.nw.neclab.eu

Juergen Quittek
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany
Phone: +49 (0) 6221 4342 115
EMail: quittek@nw.neclab.eu
URI: http://www.nw.neclab.eu

Thomas Dietz
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany
Phone: +49 (0) 6221 4342 128
EMail: thomas.dietz@nw.neclab.eu
URI: http://www.nw.neclab.eu

Martin Swany
Dept. of Computer and Information Sciences
University of Delaware
Newark DE 19716
U.S.A.
EMail: swany@UDel.Edu