

IPPM Working Group
Internet-Draft
Expires: September 6, 2006

S. Niccolini
S. Tartarelli
J. Quittek
NEC
M. Swany
UDel
March 5, 2006

**How to store traceroute measurements and related metrics
draft-niccolini-ippm-storetracroutes-03**

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on September 6, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This memo describes a standard way to store traceroute measurements and related metrics. To better address the traceroute measurements storing issue, the authors first of all give a definition of the traceroute tool, describe the tool itself as well as its parameters and the default values on the most common operating systems and the

output results that can be stored. Afterwards, the common information model with the base elements of the traceroute measurement storing is defined dividing the information elements in two semantically separated groups (configuration elements and results ones). Moreover an additional element is defined to relate configuration elements and results ones by means of a common unique identifier. On the basis of the information model a data model is then proposed in order to actually store the traceroute measurements. Finally considerations on how to extract relevant metrics related to traceroute measurements are given.

Table of Contents

1.	Introduction	4
2.	Definition of traceroute	5
2.1.	Traceroute configuration parameters	5
3.	Known problems with traceroute	9
4.	Reports/results	10
5.	Information model for storing traceroutes measurements	11
5.1.	Data types	11
5.2.	Information Elements	12
5.2.1.	Configuration Information Elements	12
5.2.2.	Results Information Elements	18
5.2.3.	Information Element correlating configuration and results elements	23
6.	Data model for storing traceroutes measurements	24
6.1.	XML schema for traceroute measurements	25
6.2.	Design considerations	40
7.	Metrics considerations for traceroute measurements	41
8.	Conclusions	42
9.	Security considerations	43
10.	IANA Considerations	44
11.	Open Issues	45
12.	Normative References	45
	Authors' Addresses	47

Intellectual Property and Copyright Statements	48
--	--------------------

1. Introduction

Traceroutes are being used by lots of measurement efforts, either as an independent measurement or to get path information to support other measurement efforts. That is why there is the need to standardize the way traceroute measurements are stored and the related metrics associated with such measurements. Since traceroute is a tool that has built-in configurable mechanisms like time-outs and can experience problems related to the crossing of firewalls thus experiencing fake losses or incomplete delay information, the standard metrics defined by IPPM working group in matter of delay, connectivity and losses may not be complete enough to define univocally the metrics retrieved by the traceroute tool. In such a context, already standardized metrics have to be closely examined in order to check for their applicability. Moreover there is a need to better define the traceroute tool as well as its parameters and the results it outputs since, to the authors' knowledge, there is so far no standard describing these. These are the motivations that moved the authors to write this draft in the context of the IPPM working group even if other working groups (like DISMAN) have already addressed similar issues related to the definition of the MIB for configuring and retrieving traceroute measurements results. The rest of the draft is organised as follows: [Section 2](#) gives the definition of traceroute used as reference in the rest of the draft as well as the traceroute configurable parameters and their default values for the most common operating systems. [Section 3](#) reports on both known problems of traceroute and existing alternatives. [Section 4](#) describes the results available from the traceroute output screen. [Section 5](#) and [Section 6](#) respectively describe our proposed Information model and Data model for storing traceroute measurements. [Section 7](#) reports on considerations on how to extract relevant metrics related to traceroute measurements while [Section 8](#) gives our conclusions. The draft ends with security considerations, IANA considerations and open issues in [Section 9](#), [Section 10](#) and [Section 11](#) respectively.

2. Definition of traceroute

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 therefore be used to discover where and how a host is connected to the Internet and can be usefully employed to troubleshoot network connections.

Typically, traceroute 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.

More in detail, the host running the traceroute sends the first set of probes with TTL equal to one (some implementations allow setting the initial TTL to a value equal to "n" different from one, 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. By observing a TTL value equal to zero, the host rejects the probe and typically returns an ICMP message with a TIME_EXCEEDED value. Traceroute can therefore derive the IP address of the first hop from the header of the ICMP message and evaluate the RTT between the source and the first hop. The next hops are discovered following the same procedure, taking care of increasing at each step the TTL value of the outgoing probes by one. The TTL value is increased until either an ICMP PORT_UNREACHABLE message is received, meaning that the destination has been reached, or the maximum configured number of hops has been hit.

Some implementations, use ICMP Echos, 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, in this draft we RECOMMEND to base implementations on UDP datagrams.

2.1. Traceroute configuration parameters

In order to define an information model for storing traceroutes, we first investigated which configuration parameters are relevant when running traceroute. We considered three major traceroute implementations and compared them based on configurable parameters and default values. The LINUX (SUSE 9.1) and UNIX (multiple platform) implemetations are based on UDP datagrams, while the WINDOWS (XP SP2) one uses ICMP Echos. The following table summarizes such comparison. A N/A in the option column, means that such option is not configurable for the specific implementation.

OS	Option	Description	Default
LINUX	-m	Specify the maximum TTL used in outgoing probes.	30
UNIX	-m		30
WINDOWS	-h		30
LINUX	-n	Display hop addresses numerically rather than symbolically.	-
UNIX	-n		-
WINDOWS	-d		-
LINUX	-w	Set the time to wait for a response to a probe.	3 sec
UNIX	-w		5 sec
WINDOWS	-w		4 sec
LINUX	N/A	Specify a loose source route gateway (to direct the traceroute through routers not necessarily in the path).	-
UNIX	-g		-
WINDOWS	-g		-
LINUX	-p	Set the base UDP port number used in probes	33434
UNIX	-p	(UDP port = base + nhops - 1).	33434
WINDOWS	N/A		-
LINUX	-q	Set the number of probes per TTL.	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
UNIX	-s		
WINDOWS	N/A		
LINUX	-t	Set the type-of-service (TOS) in the probes to the specified value.	0
UNIX	-t		0

	-----+-----		-----+-----
	WINDOWS N/A		0
+	-----+-----+	+	-----+-----+
	LINUX -v		Verbose output: received ICMP
	-----+-----		packets other than TIME_EXCEEDED
	UNIX -v		and UNREACHABLE are listed.
	-----+-----		-----+-----
	WINDOWS N/A		-
+	-----+-----+	+	-----+-----+
	LINUX -l		Display the TTL value of
	-----+-----		returned packets. This is useful
	UNIX N/A		for checking for asymmetric
	-----+-----		routing.
	WINDOWS N/A		-----+-----
+	-----+-----+	+	-----+-----+
	LINUX -r		Bypass the normal routing tables
	-----+-----		and send directly to a host on
	UNIX -r		attached network.
	-----+-----		-----+-----
	WINDOWS N/A		-
+	-----+-----+	+	-----+-----+
	LINUX N/A		Set the initial TTL for the
	-----+-----		first probe.
	UNIX -f		-----+-----
	-----+-----		1
	WINDOWS N/A		-----+-----
+	-----+-----+	+	-----+-----+
	LINUX N/A		Set the "don't fragment" bit.
	-----+-----		-----+-----
	UNIX -F		-
	-----+-----		-----+-----
	WINDOWS N/A		-
+	-----+-----+	+	-----+-----+
	LINUX N/A		Enables socket level debugging.
	-----+-----		-----+-----
	UNIX -d		-
	-----+-----		-----+-----
	WINDOWS N/A		-
+	-----+-----+	+	-----+-----+
	LINUX N/A		Use ICMP ECHO instead of UDP
	-----+-----		datagrams.
	UNIX -I		-----+-----
	-----+-----		-
	WINDOWS N/A		-----+-----
+	-----+-----+	+	-----+-----+
	LINUX N/A		Specify a network interface to
	-----+-----		obtain the IP address for
	UNIX -i		outgoing IP packets (alternative
	-----+-----		-----+-----
	-----+-----		-

-----+-----	to option -s).	-----
WINDOWS N/A		-
+-----+-----+	-----+	-----+
LINUX N/A	Toggle checksum.	-
-----+-----		-----
UNIX -x		-
-----+-----		-----
WINDOWS N/A		-
+-----+-----+	-----+	-----+
LINUX -	As optional last paramater,	Depends
-----+-----	LINUX and UNIX implementations	on
UNIX -	allow specifying the probe	implement
-----+-----	datagram length for outgoing	ation.
WINDOWS N/A	probes.	
+-----+-----+	-----+	-----+

3. Known problems with traceroute

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 traceroutes is outside the scope of this draft, therefore in the sequel, we will restrict our focus to the most commonly implemented UDP based traceroute.

4. Reports/results

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

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

Depending on the traceroute 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 time-out (for instance if the probe is filtered out by a firewall). In this case, an "*" is displayed in place of the RTT. Besides, for delays below 1 ms, some implementations reports 0 ms (f.i. UNIX and LINUX) while WINDOWS tracert reports "< 1 ms".

5. Information model for storing traceroutes measurements

This section describes the information model for the traceroute measurements data storing. 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:

- o 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".
- o description - The semantics of this information element.
- o 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.
- o units - If the element is a measure of some kind, the units identify what the measure is.
- o default value - The default value for the element (where applicable).

5.1. Data types

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

- o 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 USASCII characters, but also accomodates other Unicode multibyte characters.
- o InetAddressType - The type "InetAddressType" represents a type of Internet address. The allowed values are to be intended as imported from [\[1\]](#).
- o InetAddress - The type "InetAddress" denotes a generic Internet address. The allowed values are to be intended as imported from [\[1\]](#).
- o TruthValue - The type "TruthValue" represents a boolean value. The allowed values are to be intended as imported from [\[2\]](#).

- o Unsigned32 - The type "Unsigned32" represents a value in the range (0..4294967295).
- o InterfaceIndexOrZero - The type "InterfaceIndexOrZero" is an extension of the InterfaceIndex convention. The latter defines a greater than zero value used to identify an interface or interface sub-layer in the system. This extension permits the additional value of zero. Examples of the usage of zero might include situations where interface was unknown, or when none or all interfaces need to be referenced. The allowed values are to be intended as imported from [5].
- o ProbesType - The type "ProbesType" represents a way of indicating the protocol used for the traceroute probes. Allowed values are UDP, TCP, ICMP.
- o DateAndTime - The type "DateAndTime" represents a date-time specification. The allowed values are to be intended as imported from [2] apart from the fact that in this document there is the need to use a milli-second resolution instead a deci-second one.
- o OperationResponseStatus - The type "OperationResponseStatus" is used to report the result of an operation. The allowed values are to be intended as imported from [6] or from its recent update ([8]).

5.2. Information Elements

This section describes the elements of the 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 the two groups.

5.2.1. Configuration Information Elements

This section describes the elements of the traceroute measurement that are specific to traceroute configuration.

5.2.1.1. traceRouteCtlTargetAddressType

- o name - traceRouteCtlTargetAddressType
- o description - Specifies the type of host address used in the traceroute command.
- o dataType - InetAddressType

- o units - N/A
- o default value - N/A

5.2.1.2. traceRouteCtlTargetAddress

- o name - traceRouteCtlTargetAddress
- o description - Specifies the host address used in the traceroute command. The host address type can be determined by the examining the value of the corresponding traceRouteCtlTargetAddressType.
- o dataType - InetAddress
- o units - N/A
- o default value - N/A

5.2.1.3. traceRouteCtlByPassRouteTable

- o name - traceRouteCtlByPassRouteTable
- o description - Specifies if the optional bypassing of the route table was enabled or not. If enabled, the traceroute will bypass the normal routing tables and send 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 operation to a local host through an interface that has no route defined.
- o dataType - TruthValue
- o units - N/A
- o default value - false

5.2.1.4. traceRouteCtlProbeDataSize

- o name - traceRouteCtlProbeDataSize
- o description - Specifies the size of the data portion of a traceroute operation in octets. If the RECOMMENDED traceroute method (UDP datagrams as probes) is used, then the value contained in this object is exact. If another traceroute method is used for which the specified size is not appropriate, then the implementation should have used whatever size (appropriate to the method) is closest to the specified size. The maximum value for this object was 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).

- o dataType - Unsigned32
- o units - octets
- o default value - 0

5.2.1.5. traceRouteCtlTimeOut

- o name - traceRouteCtlTimeOut
- o description - Specifies the time-out value, in seconds, for the traceroute operation.
- o dataType - Unsigned32
- o units - seconds
- o default value - 3

5.2.1.6. traceRouteCtlProbesPerHop

- o name - traceRouteCtlProbesPerHop
- o description - Specifies the number of times to reissue a traceroute request with the same time-to-live (TTL) value.
- o dataType - Unsigned32
- o units - probes
- o default value - 3

5.2.1.7. traceRouteCtlPort

- o name - traceRouteCtlPort
- o description - Specifies the base UDP port used by the traceroute operation. Need to specify a port that is not in use at the destination (target) host. The default value for this object is the IANA assigned port, 33434, for the traceroute function.
- o dataType - Unsigned32

- o units - UDP Port
- o default value - 33434

5.2.1.8. traceRouteCtlMaxTtl

- o name - traceRouteCtlMaxTtl
- o description - Specifies the maximum TTL value for the traceroute operation.
- o dataType - Unsigned32
- o units - time-to-live value
- o default value - 30

5.2.1.9. traceRouteCtlDSField

- o name - traceRouteCtlDSField
- o description - Specifies the value that was stored in the Differentiated Services (DS) field in the IP packet used to encapsulate the traceroute probe. The DS Field is defined as the Type of Service (TOS) octet in a IPv4 header or as the Traffic Class octet in a IPv6 header. 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 response. Not all values are legal or meaningful. DS field usage is often not supported by IP implementations. A value of 0 means that the function represented by this option is not supported. Useful TOS octet values are probably '16' (low delay) and '8' (high throughput). Further references can be found in [3] for the definition of the Differentiated Services (DS) field and to [4] [Section 5.3.2](#) for Type of Service (TOS).
- o dataType - Unsigned32
- o units - N/A
- o default value - 0

5.2.1.10. traceRouteCtlSourceAddressType

- o name - traceRouteCtlSourceAddressType
- o description - Specifies the type of the source address, traceRouteCtlSourceAddress, used when performing the traceroute

operation.

- o dataType - InetAddressType
- o units - N/A
- o default value - N/A

5.2.1.11. traceRouteCtlSourceAddress

- o name - traceRouteCtlSourceAddress
- o description - Specifies the IP address (which has to be given as an IP number, not a hostname) as the source address used in outgoing probe packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the primary IP address of the interface the probe packet is sent on. A zero length octet string value for this object means that source address specification was disabled. The address type (InetAddressType) that relates to this object is specified by the corresponding value of traceRouteCtlSourceAddressType.
- o dataType - InetAddress
- o units - N/A
- o default value - N/A

5.2.1.12. traceRouteCtlIfIndex

- o name - traceRouteCtlIfIndex
- o description - Specifies the interface index used in the traceroute operation for sending the traceroute probes. A value of zero for this object implies that the interface was unknown.
- o dataType - InterfaceIndexOrZero
- o units - N/A
- o default value - 0

5.2.1.13. traceRouteCtlMiscOptions

- o name - traceRouteCtlMiscOptions

- o description - Specifies implementation dependent options.
- o dataType - String
- o units - N/A
- o default value - N/A

5.2.1.14. traceRouteCtlMaxFailures

- o name - traceRouteCtlMaxFailures
- o description - Specifies the maximum number of consecutive timeouts allowed before terminating a traceroute operation. A value of either 255 (maximum hop count/possible TTL value) or a 0 indicates that the function of terminating a remote traceroute operation when a specific number of consecutive timeouts are detected was disabled. This element is included to give full compatibility with [6] and with its recent update ([8]). No known implementation of traceroute currently supports it.
- o dataType - Unsigned32
- o units - timeouts
- o default value - 5

5.2.1.15. traceRouteCtlDontFragment

- o name - traceRouteCtlDontFragment
- o description - Specifies if the don't fragment flag (DF) in the IP header for a probe was enabled or not. The use of the DF flag is intended to be relative to a manual PATH MTU test.
- o dataType - TruthValue
- o units - N/A
- o default value - false

5.2.1.16. traceRouteCtlInitialTtl

- o name - traceRouteCtlInitialTtl
- o description - Specifies the initial TTL value uses in a traceroute operation. The use of initial TTL setting is intended to be used when bypassing the initial (often well known) portion of a path is

to be achieved.

- o dataType - Unsigned32
- o units - N/A
- o default value - 1

5.2.1.17. traceRouteCtlDescr

- o name - traceRouteCtlDescr
- o description - The purpose of this element is to provide a description of the traceroute test.
- o dataType - String
- o units - N/A
- o default value - N/A

5.2.1.18. traceRouteCtlType

- o name - traceRouteCtlType
- o description - Specifies the implementation method used for the traceroute operation.
- o dataType - ProbesType
- o units - N/A
- o default value - UDP

5.2.2. Results Information Elements

This section describes the elements of the traceroute measurement that are specific to the results of a traceroute operation.

5.2.2.1. traceRouteResultsStartDateAndTime

- o name - traceRouteResultsStartDateAndTime
- o description - Specifies the date and start time of the traceroute operation. This is the time when the first probe was seen at the sending interface.

- o dataType - DateAndTime
- o units - N/A
- o default value - N/A

5.2.2.2. traceRouteResultsIpTgtAddrType

- o name - traceRouteResultsIpTgtAddrType
- o description - Specifies the type of address stored in the corresponding traceRouteResultsIpTgtAddr element.
- o dataType - InetAddressType
- o units - N/A
- o default value - N/A

5.2.2.3. traceRouteResultsIpTgtAddr

- o name - traceRouteResultsIpTgtAddr
- o description - Specifies the IP address associated with a traceRouteCtlTargetAddress value when the destination address is specified as a DNS name. The value of this object should be a zero length octet string when a DNS name is not specified or when a specified DNS name fails to resolve.
- o dataType - InetAddress
- o units - N/A
- o default value - N/A

5.2.2.4. traceRouteResultsProbeIndex

- o name - traceRouteResultsProbeIndex
- o description - Specifies the progressive index of a probe within the traceroute operation. The maximum value here is the number resulting from the operation $\text{traceRouteCtlMaxTtl} * \text{traceRouteCtlProbesPerHop}$.
- o dataType - Unsigned32
- o units - N/A

- o default value - N/A

5.2.2.5. traceRouteResultsProbeHopIndex

- o name - traceRouteResultsProbeHopIndex
- o description - Specifies which hop in a traceroute path that the probe's results are for. The value of this element is initially determined by the value of traceRouteCtlInitialTtl.
- o dataType - Unsigned32
- o units - N/A
- o default value - N/A

5.2.2.6. traceRouteResultsProbeIndexPerHop

- o name - traceRouteResultsProbeIndexPerHop
- o description - Specifies the index of a probe for a particular hop in a traceroute path. The number of probes per hop is determined by the value of the corresponding traceRouteCtlProbesPerHop element.
- o dataType - Unsigned32
- o units - N/A
- o default value - N/A

5.2.2.7. traceRouteResultsProbeHopAddrType

- o name - traceRouteResultsProbeHopAddrType
- o description - Specifies the type of address stored in the corresponding traceRouteResultsProbeHopAddr element.
- o dataType - InetAddressType
- o units - N/A
- o default value - N/A

5.2.2.8. traceRouteResultsProbeHopAddr

- o name - traceRouteResultsProbeHopAddr
- o description - Specifies the address of a hop in the traceroute path. This object is not allowed to be a DNS name. The value of the corresponding object, traceRouteResultsProbeHopAddrType, indicates this object's IP address type.
- o dataType - InetAddress
- o units - N/A
- o default value - N/A

5.2.2.9. traceRouteResultsProbeHopASNumber

- o name - traceRouteResultsProbeHopASNumber
- o description - Specifies the AS number of a hop in the traceroute path.
- o dataType - Unsigned32
- o units - N/A
- o default value - N/A

5.2.2.10. traceRouteResultsProbeHopGeoLocation

- o name - traceRouteResultsProbeHopGeoLocation
- o description - Specifies the geo location of a hop in the traceroute path.
- o dataType - String
- o units - N/A
- o default value - N/A

5.2.2.11. traceRouteResultsProbeRoundTripTime

- o name - traceRouteResultsProbeRoundTripTime
- o description - 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 0 when it was not possible to transmit a probe.

- o dataType - Unsigned32
- o units - milliseconds
- o default value - N/A

5.2.2.12. traceRouteResultsProbeResponseStatus

- o name - traceRouteResultsProbeResponseStatus
- o description - Specifies the result of a traceroute operation made by the host for a particular probe.
- o dataType - OperationResponseStatus
- o units - N/A
- o default value - N/A

5.2.2.13. traceRouteResultsProbeTime

- o name - traceRouteResultsProbeTime
- o description - Specifies the timestamp for when the response to the probe was received interface.
- o dataType - DateAndTime
- o units - N/A
- o default value - N/A

5.2.2.14. traceRouteResultsHopRawOutputData

- o name - traceRouteResultsHopRawOutputData
- o description - Specifies the raw output data returned by the traceroute operation for a certain hop in a traceroute path.
- o dataType - String
- o units - N/A
- o default value - N/A

5.2.2.15. traceRouteResultsEndDateAndTime

- o name - traceRouteResultsEndDateAndTime
- o description - Specifies the date and end time of the traceroute operation. It is either the time when the response to the last probe of the traceroute operation was received or the time when the last probe of the traceroute operation was sent plus the relative timeout (in case of missing response).
- o dataType - DateAndTime
- o units - N/A
- o default value - N/A

5.2.3. Information Element correlating configuration and results elements

This section defines an additional element belonging to both the two previous groups. This element is defined in order to relate configuration elements and results ones by means of a common unique identifier.

5.2.3.1. traceRouteTestName

- o name - traceRouteTestName
- o description - Specifies the name of a traceroute test. This is locally unique.
- o dataType - String
- o units - N/A
- o default value - N/A

6. Data model for storing traceroutes 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 representation 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 [6], that specifies a BER encoding [9] used by the Simple Network Management Protocol (SNMP) [10] for transmitting traceroute information. 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 specified for the NetFlow version 9 [11] protocol and the IPFIX protocol [12], [13]. For these protocols standard templates could be defined as it has been done for another purpose in [14]. The architectures behind these protocols have been designed for the export of passively measured flow information. For following this approach, it needs to be investigated whether or not the respective architectures can be extended to active route measurement.

For textual representations, using the eXtensible Markup Language (XML) [15] 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 is the resource consumption for processing, storing, and transmitting information. Since the expected data volumes of traceroute data in network operation and maintenance is not expected to be extremely high, the inefficient usage of resources is not a significant disadvantage. Therefore, XML was chosen as basis for the traceroute information model that is specified in this section.

6.1. XML schema for traceroute measurements

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="inetAddressType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unknown"/>
      <xs:enumeration value="ipv4"/>
      <xs:enumeration value="ipv6"/>
      <xs:enumeration value="dns"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="inetAddressTypeWithoutDns">
    <xs:restriction base="xs:string">
      <xs:enumeration value="unknown"/>
      <xs:enumeration value="ipv4"/>
      <xs:enumeration value="ipv6"/>
      <xs:enumeration value="noSpecification"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="_zeroLengthString">
    <xs:restriction base="xs:string">
      <xs:maxLength value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <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="([1-9A-Fa-f]{1,4}:){7}[1-9A-Fa-f]{1,4}
        (:|[1-9A-Fa-f]{1,4}){3}:[1-9A-Fa-f]{1,4}"/>
    </xs:restriction>
  </xs:simpleType>

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



```
        </xs:restriction>
</xs:simpleType>

<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:simpleType name="_dateAndTimeUpToSeconds">
  <xs:restriction base="xs:dateTime"/>
</xs:simpleType>

<xs:simpleType name="_timeMilliseconds">
  <xs:restriction base="xs:unsignedShort">
    <xs:maxExclusive value="1000"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteTestName">
  <xs:annotation>
    <xs:documentation>Specifies the name of a traceroute
      test. This is locally unique.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="32"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteCtlByPassRouteTable">
  <xs:annotation>
    <xs:documentation>Specifies if the optional bypassing
      of the route table was enabled or not. If enabled,
      the traceroute will bypass the normal routing tables
      and send 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 operation to a local host
        through an interface that has no route defined.
    </xs:documentation>
</xs:annotation>
<xs:restriction base="xs:boolean"/>
</xs:simpleType>

<xs:simpleType name="_traceRouteCtlProbeDataSize">
    <xs:annotation>
        <xs:documentation>Specifies the size of the data
        portion of a traceroute operation in octets. If the
        RECOMMENDED traceroute method (UDP datagrams as probes)
        is used, then the value contained in this object is
        exact. If another traceroute method is used for which
        the specified size is not appropriate, then the
        implementation should have used whatever size (appropriate
        to the method) is closest to the specified size.
        The maximum value for this object was 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). Units are: octets.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:unsignedShort">
        <xs:maxExclusive value="65508"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteCtlTimeOut">
    <xs:annotation>
        <xs:documentation>Specifies the time-out value, in
        seconds, for the traceroute operation.
        Units are: seconds.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:unsignedByte">
        <xs:minExclusive value="0"/>
        <xs:maxExclusive value="61"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteCtlProbesPerHop">
    <xs:annotation>
        <xs:documentation>Specifies the number of times to
        reissue a traceroute request with the same time-to-live
        (TTL) value. Units are: probes.
```



```
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:unsignedByte">
        <xs:minExclusive value="0"/>
        <xs:maxExclusive value="11"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="_traceRouteCtlPort">
      <xs:annotation>
        <xs:documentation>Specifies the base UDP port used by
          the traceroute operation. Need to specify a port that
          is not in use at the destination (target) host.
          The default value for this object is the IANA assigned
          port, 33434, for the traceroute function.
          Units are: UDP port.
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:unsignedShort">
        <xs:minExclusive value="0"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="_traceRouteCtlMaxTtl">
      <xs:annotation>
        <xs:documentation>Specifies the maximum TTL value for
          the traceroute operation.
          Units are: time-to-live value.
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:unsignedByte">
        <xs:minExclusive value="0"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="_traceRouteCtlDSField">
      <xs:annotation>
        <xs:documentation>Specifies the value that was stored in
          the Differentiated Services (DS) field in the IP packet
          used to encapsulate the traceroute probe. The DS Field
          is defined as the Type of Service (TOS) octet in a IPv4
          header or as the Traffic Class octet in a IPv6 header.
          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 response. Not all values are legal or
          meaningful. DS field usage is often not supported by IP
          implementations. A value of 0 means that the function
```


represented by this option is not supported. Useful TOS octet values are probably '16' (low delay) and '8' (high throughput). Further references can be found in the [RFC 2474](#) for the definition of the Differentiated Services (DS) field and to the [RFC 1812 Section 5.3.2](#) for Type of Service (TOS).

```
</xs:documentation>
</xs:annotation>
<xs:restriction base="xs:unsignedByte"/>
</xs:simpleType>

<xs:simpleType name="_traceRouteCtlIfIndex">

  <xs:annotation>
    <xs:documentation>Specifies the interface index used in the
      traceroute operation for sending the traceroute
      probes. A value of zero for this object implies
      that the interface was unknown.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:unsignedByte"/>
</xs:simpleType>

<xs:simpleType name="_traceRouteCtlMiscOptions">
  <xs:annotation>
    <xs:documentation>Specifies implementation dependent
      options.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="100"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteCtlMaxFailures">
  <xs:annotation>
    <xs:documentation>Specifies the maximum number of
      consecutive timeouts allowed before terminating a
      traceroute operation. A value of either 255
      (maximum hop count/possible TTL value) or a 0 indicates
      that the function of terminating a remote traceroute
      operation when a specific number of consecutive
      timeouts are detected was disabled. This element is
      included to give full compatibility with DISMAN working
      group documents. No known implementation of traceroute
      currently supports it.
      Units are: timeouts.
    </xs:documentation>
```



```
        </xs:annotation>
        <xs:restriction base="xs:unsignedByte"/>
    </xs:simpleType>

    <xs:simpleType name="_traceRouteCtlDontFragment">
        <xs:annotation>
            <xs:documentation>Specifies if the don't fragment flag
            (DF) in the IP header for a probe was enabled or not.
            The use of the DF flag is intended to be relative to a
            manual PATH MTU test.
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:boolean"/>
    </xs:simpleType>

    <xs:simpleType name="_traceRouteCtlInitialTtl">
        <xs:annotation>
            <xs:documentation>Specifies the initial TTL value uses
            in a traceroute operation. The use of initial TTL
            setting is intended to be used when bypassing the
            initial (often well known) portion of a path is to be
            achieved.
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:unsignedByte">
            <xs:minExclusive value="0"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="_traceRouteCtlDescr">
        <xs:annotation>
            <xs:documentation>The purpose of this element is to
            provide a description of the traceroute test.
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:maxLength value="100"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="_traceRouteCtlType">
        <xs:annotation>
            <xs:documentation>Specifies the implementation method
            used for the traceroute operation.
            </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:enumeration value="TCP"/>
        </xs:restriction>
    </xs:simpleType>
```



```
        <xs:enumeration value="UDP"/>
        <xs:enumeration value="ICMP"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteResultsProbeIndex">
    <xs:annotation>
        <xs:documentation>Specifies the progressive index of
        a probe within the traceroute operation. The maximum
        value here is the number resulting from the operation
        traceRouteCtlMaxTtl*traceRouteCtlProbesPerHop.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:unsignedShort">
        <xs:minExclusive value="0"/>
        <xs:maxExclusive value="2551"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteResultsProbeHopIndex">
    <xs:annotation>
        <xs:documentation>Specifies which hop in a traceroute
        path that the probe's results are for. The value of
        this element is initially determined by the value of
        traceRouteCtlInitialTtl.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:unsignedByte">
        <xs:minExclusive value="0"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteResultsProbeIndexPerHop">
    <xs:annotation>
        <xs:documentation>Specifies the index of a probe for
        a particular hop in a traceroute path. The number of
        probes per hop is determined by the value of the
        corresponding traceRouteCtlProbesPerHop element.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:unsignedByte">
        <xs:minExclusive value="0"/>
        <xs:maxExclusive value="11"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteResultsProbeHopASNumber">
    <xs:annotation>
```



```

    <xs:documentation>Specifies the AS number of a hop in
    the traceroute path.
  </xs:documentation>
</xs:annotation>
  <xs:restriction base="xs:unsignedInt"/>
</xs:simpleType>

<xs:simpleType name="_traceRouteResultsProbeHopGeoLocation">
  <xs:annotation>
    <xs:documentation>Specifies the geo location of a hop
    in the traceroute path.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="100"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteResultsProbeRoundTripTime">
  <xs:annotation>
    <xs:documentation>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 0 when it was not possible
    to transmit a probe.
    Units are: milliseconds.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:unsignedShort">
    <xs:maxExclusive value="60001"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="_traceRouteResultsHopRawOutputData">
  <xs:annotation>
    <xs:documentation>Specifies the raw output data returned
    by the traceroute operation for a certain hop in a
    traceroute path.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="200"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="_inetAddress">
  <xs:choice>
    <xs:element name="inetAddressUnknown">
```



```
        type="_zeroLengthString"/>
    <xs:element name="inetAddressIpv4"
        type="_inetAddressIpv4"/>
    <xs:element name="inetAddressIpv6"
        type="_inetAddressIpv6"/>
    <xs:element name="inetAddressDns"
        type="_inetAddressDns"/>
</xs:choice>
</xs:complexType>

<xs:complexType name="_inetAddressWithoutDns">
    <xs:choice>
        <xs:element name="inetAddressUnknown"
            type="_zeroLengthString"/>
        <xs:element name="inetAddressIpv4"
            type="_inetAddressIpv4"/>
        <xs:element name="inetAddressIpv6"
            type="_inetAddressIpv6"/>
        <xs:element name="zeroLengthString"
            type="_zeroLengthString"/>
    </xs:choice>
</xs:complexType>

<xs:complexType name="_dateAndTime">
    <xs:sequence>
        <xs:element name="dateAndTimeUpToSeconds"
            type="_dateAndTimeUpToSeconds"/>
        <xs:element name="timeMilliseconds"
            type="_timeMilliseconds"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteCtlTargetAddressType">
    <xs:annotation>
        <xs:documentation>Specifies the type of host address
            used in the traceroute command.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="targetAddressType"
            type="inetAddressType"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteCtlTargetAddress">
    <xs:annotation>
        <xs:documentation>Specifies the host address used in
            the traceroute command. The host address type can be
```



```
        determined by the examining the value of the
        corresponding traceRouteCtlTargetAddressType.
      </xs:documentation>
    </xs:annotation>
  <xs:sequence>
    <xs:element name="targetAddress" type="_inetAddress"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteCtlSourceAddressType">
  <xs:annotation>
    <xs:documentation>Specifies the type of the source address,
    traceRouteCtlSourceAddress, used when performing the
    traceroute operation.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="sourceAddressType"
                  type="inetAddressTypeWithoutDns"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteCtlSourceAddress">
  <xs:annotation>
    <xs:documentation>Specifies the IP address (which has
    to be given as an IP number, not a hostname) as the
    source address used in outgoing probe packets. On hosts
    with more than one IP address, this option can be used
    to force the source address to be something other than
    the primary IP address of the interface the probe packet
    is sent on. A zero length octet string value for this
    object means that source address specification was disabled.
    The address type (InetAddressType) that relates to this
    object is specified by the corresponding value of
    traceRouteCtlSourceAddressType.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="sourceAddress"
                  type="_inetAddressWithoutDns"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteResultsStartDateAndTime">
  <xs:annotation>
    <xs:documentation>Specifies the date and start time of the
    traceroute operation. This is the time when the first probe
    was sent.
```



```
        </xs:documentation>
      </xs:annotation>
    <xs:sequence>
      <xs:element name="dateAndTime" type="_dateAndTime"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="_traceRouteResultsIpTgtAddrType">
    <xs:annotation>
      <xs:documentation>Specifies the type of address stored
        in the corresponding traceRouteResultsIpTgtAddr element.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ipTgtAddrType"
        type="inetAddressTypeWithoutDns"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="_traceRouteResultsIpTgtAddr">
    <xs:annotation>
      <xs:documentation>Specifies the IP address associated
        with a traceRouteCtlTargetAddress value when the
        destination address is specified as a DNS name.
        The value of this object should be a zero length octet
        string when a DNS name is not specified or when a
        specified DNS name fails to resolve.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ipTgtAddr"
        type="_inetAddressWithoutDns"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="_traceRouteResultsProbeHopAddrType">
    <xs:annotation>
      <xs:documentation>Specifies the type of address stored
        in the corresponding traceRouteResultsProbeHopAddr
        element.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="probeHopAddrType"
        type="inetAddressTypeWithoutDns"/>
    </xs:sequence>
  </xs:complexType>
```



```
<xs:complexType name="_traceRouteResultsProbeHopAddr">
  <xs:annotation>
    <xs:documentation>Specifies the address of a hop in
      the traceroute path. This object is not allowed to
      be a DNS name. The value of the corresponding object,
      traceRouteResultsProbeHopAddrType, indicates this
      object's IP address type.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="probeHopAddr"
      type="_inetAddressWithoutDns"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteResultsProbeResponseStatus">
  <xs:annotation>
    <xs:documentation>Specifies the result of a traceroute
      operation made by the host for a particular probe.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="probeResponseStatus"
      type="operationResponseStatus"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteResultsProbeTime">
  <xs:annotation>
    <xs:documentation>Specifies the timestamp when the
      response to the probe was received.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="dateAndTime" type="_dateAndTime"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRouteResultsProbe">
  <xs:sequence>
    <xs:element name="traceRouteResultsProbeIndex"
      type="_traceRouteResultsProbeIndex"/>
    <xs:element name="traceRouteResultsProbeHopIndex"
      type="_traceRouteResultsProbeHopIndex"/>
    <xs:element name="traceRouteResultsProbeIndexPerHop"
      type="_traceRouteResultsProbeIndexPerHop"/>
    <xs:element name="traceRouteResultsProbeHopAddrType"
      type="_traceRouteResultsProbeHopAddrType"/>
  </xs:sequence>
</xs:complexType>
```



```
<xs:element name="traceRouteResultsProbeHopAddr"
  type="_traceRouteResultsProbeHopAddr"/>
<xs:element name="traceRouteResultsProbeHopASNumber"
  type="_traceRouteResultsProbeHopASNumber"
  minOccurs="0" maxOccurs="1"/>
<xs:element name="traceRouteResultsProbeHopGeoLocation"
  type="_traceRouteResultsProbeHopGeoLocation"
  minOccurs="0" maxOccurs="1"/>
<xs:element name="traceRouteResultsProbeRoundTripTime"
  type="_traceRouteResultsProbeRoundTripTime"/>
<xs:element name="traceRouteResultsProbeResponseStatus"
  type="_traceRouteResultsProbeResponseStatus"/>
<xs:element name="traceRouteResultsProbeTime"
  type="_traceRouteResultsProbeTime"/>
</xs:sequence>
</xs:complexType>

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

<xs:complexType name="_traceRouteMetadata">
  <xs:annotation>
    <xs:documentation>Specifies the metadata for a
      traceroute operation. In a request, these are the
      requested parameters. In a response, they are the
      actual parameters used.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="traceRouteTestName"
      type="_traceRouteTestName"/>
    <xs:element name="traceRouteCtlTargetAddressType"
      type="_traceRouteCtlTargetAddressType"/>
    <xs:element name="traceRouteCtlTargetAddress"
      type="_traceRouteCtlTargetAddress"/>
    <xs:element name="traceRouteCtlByPassRouteTable"
```



```
        type="_traceRouteCtlByPassRouteTable"
        minOccurs="0" maxOccurs="1"
        default="false"/>
<xs:element name="traceRouteCtlProbeDataSize"
    type="_traceRouteCtlProbeDataSize"
    minOccurs="0" maxOccurs="1" default="0"/>
<xs:element name="traceRouteCtlTimeOut"
    type="_traceRouteCtlTimeOut" minOccurs="0"
    maxOccurs="1" default="3"/>
<xs:element name="traceRouteCtlProbesPerHop"
    type="_traceRouteCtlProbesPerHop"
    minOccurs="0" maxOccurs="1" default="3"/>
<xs:element name="traceRouteCtlPort"
    type="_traceRouteCtlPort" minOccurs="0"
    maxOccurs="1" default="33434"/>
<xs:element name="traceRouteCtlMaxTtl"
    type="_traceRouteCtlMaxTtl" minOccurs="0"
    maxOccurs="1" default="30"/>
<xs:element name="traceRouteCtlDSField"
    type="_traceRouteCtlDSField" minOccurs="0"
    maxOccurs="1" default="0"/>
<xs:element name="traceRouteCtlSourceAddressType"
    type="_traceRouteCtlSourceAddressType"/>
<xs:element name="traceRouteCtlSourceAddress"
    type="_traceRouteCtlSourceAddress"/>
<xs:element name="traceRouteCtlIfIndex"
    type="_traceRouteCtlIfIndex" minOccurs="0"
    maxOccurs="1" default="0"/>
<xs:element name="traceRouteCtlMiscOptions"
    type="_traceRouteCtlMiscOptions" minOccurs="0"
    maxOccurs="1"/>
<xs:element name="traceRouteCtlMaxFailures"
    type="_traceRouteCtlMaxFailures" minOccurs="0"
    maxOccurs="1" default="5"/>
<xs:element name="traceRouteCtlDontFragment"
    type="_traceRouteCtlDontFragment" minOccurs="0"
    maxOccurs="1" default="false"/>
<xs:element name="traceRouteCtlInitialTtl"
    type="_traceRouteCtlInitialTtl" minOccurs="0"
    maxOccurs="1" default="1"/>
<xs:element name="traceRouteCtlDescr"
    type="_traceRouteCtlDescr" minOccurs="0"
    maxOccurs="1"/>
<xs:element name="traceRouteCtlType"
    type="_traceRouteCtlType" minOccurs="0"
    maxOccurs="1"
    default="UDP"/>
</xs:sequence>
```



```
</xs:complexType>

<xs:complexType name="_traceRouteMeasurement">
  <xs:annotation>
    <xs:documentation>
      Contains the actual traceroute measurement.
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="traceRouteTestName"
      type="_traceRouteTestName"/>
    <xs:element name="traceRouteResultsStartDateAndTime"
      type="_traceRouteResultsStartDateAndTime"/>
    <xs:element name="traceRouteResultsIpTgtAddrType"
      type="_traceRouteResultsIpTgtAddrType"/>
    <xs:element name="traceRouteResultsIpTgtAddr"
      type="_traceRouteResultsIpTgtAddr"/>
    <xs:element name="traceRouteResultsProbe"
      type="_traceRouteResultsProbe"
      minOccurs="1" maxOccurs="2550"/>
    <xs:element name="traceRouteResultsHopRawOutputData"
      type="_traceRouteResultsHopRawOutputData"
      minOccurs="0" maxOccurs="255"/>
    <xs:element name="traceRouteResultsEndDateAndTime"
      type="_traceRouteResultsEndDateAndTime"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="_traceRoute">
  <xs:choice>
    <xs:element name="traceRouteRequest"
      type="_traceRouteMetadata"/>
    <xs:element name="traceRouteMeasurementMetadata"
      type="_traceRouteMetadata"/>
    <xs:element name="traceRouteMeasurement"
      type="_traceRouteMeasurement"/>
  </xs:choice>
</xs:complexType>

<!--Reference to "traceRoute" element-->
<xs:element name="traceRoute" type="_traceRoute"/>

</xs:schema>
```


6.2. Design considerations

In [Section 6.1](#) we proposed an example of a possible XML schema as a template for storing and/or exchanging traceroute measurements. We designed the schema in order to use an extensible approach based on templates (pretty similar to how IPFIX protocol is designed) where the traceroute configuration elements (both the requested parameters, `traceRouteRequest`, and the actual parameters used, `traceRouteMeasurementMetadata`) are metadata to be referenced by results information elements (data) by means of the `traceRouteTestName` element (used as unique identifier). Currently Global Grid Forum (GGF) is also using this approach and cross-requirements have been analyzed as well as standardization opportunities of a joint work.

Looking at the state of the art of the XML schema design, it is likely that naming issues will have to be solved in order to have a common work shared among the IETF and the GGF. A possibility on how to solve this is to standardize a common XML schema with different naming instances making possible a one-to-one name conversion that could be used for IETF-GGF interoperability purposes.

7. Metrics considerations for traceroute measurements

The previous sections of this draft have been dealing with the traceroute tool description, its parameters and the output that is going to be stored; in this section we want to highlight the relevant metrics that are of interest when storing traceroute measurements.

Since the traceroute has built-in configurable mechanisms like time-outs and can experience problems related to the crossing of firewalls, some of the packets that traceroute sends out end up being time-out or filtered. Sometimes it is impossible to trace the path to a node or there is not a complete set of probes describing the RTT to reach it. Moreover, a known inconsistency exists between the round-trip delay metric defined by the IPPM working group and the results returned by the current traceroute implementations. Unfortunately, it is unlikely that the traceroute implementations will change accordingly in the near future therefore. In order to compare results of traceroute measurements, a possible solution would consist in adding to the metadata elements a specification of the operating system and version for the tool used in order to help in comparing metrics.

8. Conclusions

This draft attempts to define a standard way to store traceroute measurements and related metrics. The document starts by examining a set of traceroute tools from the most common operating systems and provides a summary of configurable and default input parameters. This investigation served as base for the definition of an information model and related data model. Furthermore, this draft tackles the issue of metrics resulting from traceroute measurements. Metrics already standardized by the IPPM are indeed not directly applicable to traceroute results because of how traceroute tools are currently implemented and where it is necessary to take into account, for instance, for missing responses. A number of open issues that still need to be discussed in the IPPM mailing list was also drafted.

9. Security considerations

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. This memo does not specify an implementation of the metrics, so it does not directly affect the security of the Internet nor of applications which run on the Internet. However, implementations of these metrics must be mindful of security and privacy concerns. Since traceroute measurements involve metrics of different types, each of the metric defined for storing traceroute measurements needs separate security considerations that are basically already covered in the related RFCs published by the IPPM working group. The 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 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.

10. IANA Considerations

This document will need to register a new XML schema according to the guidelines in [\[7\]](#).

The URN Sub-Namespace Registration will be for `urn:ietf:params:xml:schema:<id>`. Where id has to be provided by IANA. NOTE: in order for a URN of this type to be assigned, the item being registered MUST have been through the IETF consensus process. Basically, this means that it must be documented in a RFC.

URI: The URI for this schema will be `urn:ietf:params:xml:schema:<id>`.

Registrant Contact: TBD.

XML: TBD.

11. Open Issues

This section is intended to keep track of the open issues that are to be discussed in the IPPM working group.

- o Consider improving section on metrics considerations.
- o Consider improving section on security considerations.
- o Complete IANA considerations.
- o Consider rewriting the description for ResultsProbeHopIndex, right now it is written in pure SNMP language and might be not clear.
- o Consider encoding the traceroute output when the delay is less than 1 ms or when the host was not reachable (update element traceRouteResultsProbeRoundTripTime). Right now a 0 is inserted when it was not possible to transmit a probe. Think a better encoding.
- o Consider keeping into account different versions of the traceroute tool instead of just UNIX, LINUX, WINDOWS. Update table in [Section 2](#) accordingly.
- o Consider adding another probe result with the MPLS label clarifying if it is per hop or per probe and keeping it optional.
- o Consider including the AS number in the address type definition.
- o Double check references and divide them in normative and informational.
- o Consider updating the title and the text of the draft referring to the fact that the actual schema can be used not only to store traceroute measurements but also to request/configure a measurement.
- o Check metadata/data separation in the traceRoute element, right now it is exclusive but metadata and data could also be stored together.

12. Normative References

- [1] Daniele et al., M., "Textual Conventions for Internet Network Addresses", [RFC 3291](#), May 2002.
- [2] McCloghrie et al., K., "Textual Conventions for SMIV2",

[RFC 2579](#), April 1999.

- [3] Nichols et al., K., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", [RFC 2474](#), December 1998.
- [4] Baker, F., "Requirements for IP Version 4 Routers", [RFC 1812](#), June 1995.
- [5] McCloghrie et al., K., "The Interfaces Group MIB", [RFC 2863](#), June 2000.
- [6] White, K., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", [RFC 2925](#), September 2000.
- [7] Mealling, M., "The IETF XML registry", [RFC 3688](#), January 2004.
- [8] Quittek, J., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations",
DRAFT [draft-ietf-disman-remops-mib-v2-06.txt](#), December 2004.
- [9] Presuhn et al., R., "Transport Mappings for the Simple Network Management Protocol (SNMP)", [RFC 3417](#), December 2002.
- [10] Case et al., J., "Introduction and Applicability Statements for Internet Standard Management Framework", [RFC 3410](#), December 2002.
- [11] Claise, B., "Cisco Systems NetFlow Services Export Version 9", [RFC 3954](#), October 2004.
- [12] Claise, B., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations",
DRAFT [draft-ietf-ipfix-protocol-17.txt](#), July 2005.
- [13] Quittek et al., J., "Information Model for IP Flow Information Export", DRAFT [draft-ietf-ipfix-info-09.txt](#), July 2005.
- [14] Stephan et al., E., "IPFIX templates for common ISP usages",
DRAFT [draft-stephan-isp-templates-00.txt](#), July 2005.
- [15] Yergeau et al., F., "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation, February 2004.

Authors' Addresses

Saverio Niccolini
Network Laboratories, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 905 11 18
Email: saverio.niccolini@netlab.nec.de
URI: <http://www.netlab.nec.de>

Sandra Tartarelli
Network Laboratories, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 905 11 32
Email: sandra.tartarelli@netlab.nec.de
URI: <http://www.netlab.nec.de>

Juergen Quittek
Network Laboratories, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 905 11 15
Email: quittek@netlab.nec.de
URI: <http://www.netlab.nec.de>

Martin Swany
Dept. of Computer and Information Sciences, University of Delaware
Newark DE 19716
U.S.A.

Email: swany@UDel.Edu

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

