



**Implementation Agreement
for Bandwidth Manager TC-2 interface**

MSF-IA-SNMP.001-FINAL

MultiService Forum Implementation Agreement

Contribution Number: msf2006.074.05
Document Filename: MSF-IA-SNMP.001-FINAL
Working Group: Protocol and Control

Title: Implementation Agreement for Bandwidth Manager TC-2 interface

Editors: Cisco Systems
John Evans Dave Melton
joevans @cisco.com dmelton@cisco.com

Working Group Chairperson: Chris Gallon (c.gallon@ftel.co.uk)

Date: 14 August 2006

Abstract: This IA defines an inception of the R3 TC-2 interface (Bandwidth Manager to network element), which can be used to audit the provisioned tunnels on a head-end router and to receive notifications of tunnel status. It is realisable with SNMPv2c or v3 and defined MIBs and hence is realisable within the GMI2006 timeframe.

The goal of the MSF is to promote multi-vendor interoperability as part of a drive to accelerate the deployment of next generation networks. To this end the MSF looks to adopt pragmatic solutions in order to maximize the chances for early deployment in real world networks.

To date the MSF has defined a number of detailed Implementation Agreements and detailed Test Plans for the signaling protocols between network components and is developing additional Implementation Agreements and Test Plans addressing some of the other technical issues such as QoS and Security to assist vendors and operators in deploying interoperable solutions.

The MSF welcomes feedback and comment and would encourage interested parties to get involved in this work program. Information about the MSF and membership options can be found on the MSF website <http://www.msforum.org/>

Note: Attention is called to the possibility that use or implementation of this MSF Implementation Agreement may require use of subject matter covered by intellectual property rights owned by parties who have not authorized such use. By publication of this Implementation Agreement, no position is taken by MSF as its Members with respect to the existence or validity of any intellectual property rights in connection therewith, nor does any warranty, express or implied, arise by reason of the publication by MSF of this Implementation Agreement. Moreover, the MSF shall not have any responsibility whatsoever for determining the existence of IPR for which a license may be required for the use or implementation of an MSF Implementation Agreement, or for conducting inquiries into the legal validity or scope of such IPR that is brought to its attention.

DISCLAIMER

The information in this publication is believed to be accurate as of its publication date. Such information is subject to change without notice and the MultiService Forum is not responsible for any errors or omissions. The MultiService Forum does not assume any responsibility to update

or correct any information in this publication. Notwithstanding anything to the contrary, neither the MultiService Forum nor the publisher make any representation or warranty, expressed or implied, concerning the completeness, accuracy, or applicability of any information contained in this publication. No liability of any kind whether based on theories of tort, contract, strict liability or otherwise, shall be assumed or incurred by the MultiService Forum, its member companies, or the publisher as a result of reliance or use by any party upon any information contained in this publication. All liability for any implied or express warranty of merchantability or fitness for a particular purpose is hereby disclaimed.

The receipt or any use of this document or its contents does not in any way create by implication or otherwise:

Any express or implied license or right to or under any MultiService Forum member company's patent, copyright, trademark or trade secret rights which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor

Any warranty or representation that any MultiService Forum member companies will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor

Any commitment by a MultiService Forum company to purchase or otherwise procure any product(s) and/or service(s) that embody any or all of the ideas, technologies, or concepts contained herein; nor

Any form of relationship between any MultiService Forum member companies and the recipient or user of this document.

Implementation or use of specific MultiService Forum Implementation Agreements, Architectural Frameworks or recommendations and MultiService Forum specifications will be voluntary, and no company shall agree or be obliged to implement them by virtue of participation in the MultiService Forum.

For addition information contact:

MultiService Forum
39355 California Street, Suite 307
Fremont, CA 94538
USA

Phone: +1 510 608-5922

Fax: +1 510 608-5917

info@msforum.org

<http://www.msforum.org>

1 Introduction

The bandwidth management technical report [1] outlines the issues surrounding bandwidth management for PSTN grade voice and multi-media services over packet networks within the context of the MSF QOS solution. The MSF Bandwidth Manager functionality is an example of an off-path¹ or path-decoupled topology-aware connection admission control (CAC) system, which could be used to provide admission control in these scenarios.

Within [1] the assumption was made that, where the Bandwidth Manager is used in the context of an MPLS Traffic Engineering (TE) tunnel deployment, the Bandwidth Manager was required to specify to a TE tunnel head-end the end-to-end path of tunnels that should be used for a particular call, by way of specifying a label stack. MSF2006.007 [2] subsequently described two deployment models for scaling TE tunnel deployments which do not require that the Bandwidth Manager specify to a TE tunnel source (known as head-end) the end-to-end path of tunnels that should be used for a particular call, by way of specifying a label stack. Rather, in these deployment models the head-end determines which tunnel will be used for each call, and the Bandwidth Manager needs to be able to resolve a particular call to the TE tunnels that it will use, and to track the status of the tunnels.

These TE deployment models can result in a significant simplification of the requirements for the R3 TC-2 interface (previously the R2 IF-4 interface) – the interface between the Bandwidth Manager and the TE tunnel head-end router described in [1] – by not necessitating that the TC-2 interface must be used to provision tunnels.

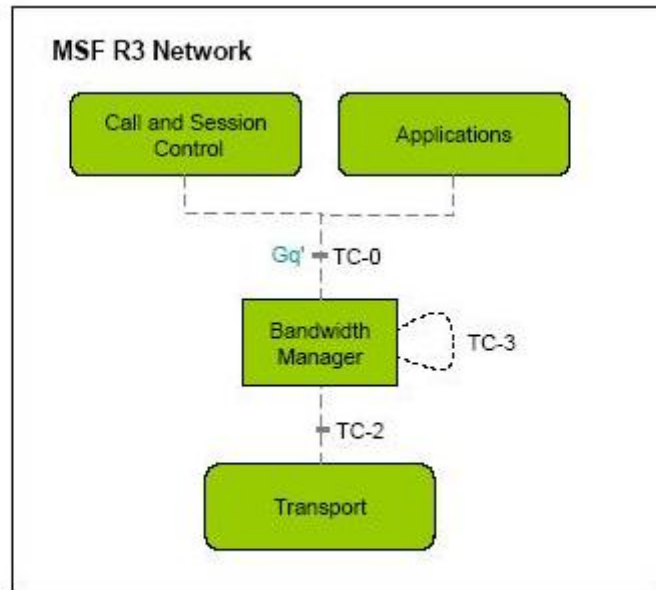


Figure 1. Generalised MSF R3 architecture from bandwidth manager perspective

Hence, this IA defines an inception of the TC-2 interface that can be used to discover the core topology (abstracted as TE tunnels) and for dynamically tracking the state of that topology. This IA uses SNMPv2c or v3 and defined MIBs to audit the provisioned tunnels on a head-end router

¹ With off-path approaches the messages used for QOS signaling are routed through nodes that are not assumed to be on the data (bearer) path.

and to receive notifications of tunnel status, and hence is realisable within the GMI2006 timeframe.

Alternative IAs would be required to address non MPLS-TE deployments; for example, by participating in interior gateway routing protocols (IGPs), such as OSPF or ISIS. This approach would, however, suffer the issue of transient congestion following network element failures, as discussed in section 2.1 of MSF-TR-ARCH-008-FINAL.

There are a number of interfaces that could be used to track the access topology, in terms of the location of end system addressing with respect to the core topology, which is used to resolve core requests to the impacted core topology, such as the TISPAN E4 interface from NASS to RACS. For GMI2006 It is recommended that the access topology is statically defined.

2 Deployment Context and Interface Requirements

In a Bandwidth Manager deployment, in order to provide deterministic service and ensure that transient congestion cannot occur following network failure cases, it is critical that traffic cannot be rerouted before a new admission control decision is made based upon an up-to-date network view. This capability is implicit in the PSTN, but not natively in IP networks. On the presumption that CAC is needed to cover network element failure case conditions and that transient congestion following network failures is to be avoided the “MPLS TE with specified tunnel bandwidth” network connectivity model described in section 2.3 of MSF2006.007 is assumed in this IA. This model may be augmented with the addition of MPLS TE Fast Reroute (FRR), in order to reduce the loss of connectivity and service disruption caused to those calls directly impacted by the failed network element to in the order of 50ms, as described in section 2.4 of MSF2006.007. We consider the TC-2 interface in the context of a deployment where TE tunnels are deployed edge-to-edge, at least for the traffic for which the Bandwidth Manager is providing admission control.

In such a deployment model the Bandwidth Manager is aware of the tunnel head-end routers (possibly via pre-configuration) and is able to audit the head-end routers in order find out what tunnels originate from each head-end, which router is the tail-end of each tunnel, and what bandwidth is provisioned for each tunnel. The Bandwidth Manager then tracks tunnel and head-end status and manages admission control decisions into available TE tunnel bandwidth accordingly.

In this model, the TE tunnel head-end routers, or an offline system (i.e. tunnel server or Path Computation Element [3]) are responsible for tunnel path calculation, both for CAC and for bandwidth optimisation (offload routing); they perform a constraint based shortest path first (CSPF) computation to pick the least cost path that satisfies the configured tunnel constraints, including available bandwidth. The Bandwidth Manager controls admission to the TE tunnel bandwidth and needs to track only the status of TE tunnels and head-ends. When a request is received, the Bandwidth Manager resolves the request to the underlying tunnel impacted by the request, confirms that the tunnel has sufficient available bandwidth to support the call, admits or denies the call accordingly, and updates the state table of available bandwidth if the request is admitted. A system other than the Bandwidth Manager may be responsible for tunnel configuration and sizing, or the tunnels may be statically configured (even if they are dynamically signalled).

Consider the example network shown in Figure 2, where gateway A and gateway B each connect to respective edge routers (for simplicity two unidirectional tunnels are represented with a single bidirectional arrow in the diagram).

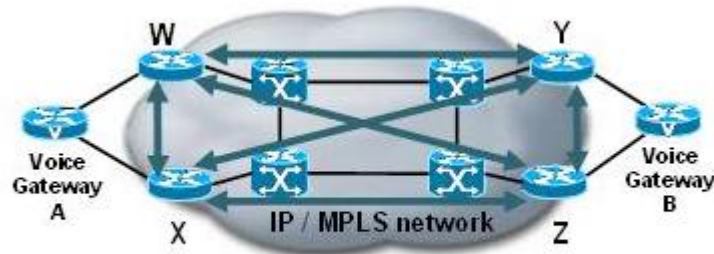


Figure 2. Example network topology

The Bandwidth Manager will maintain a map of the key network bandwidth resources needed to make valid admission control decisions, which in this example are the core TE tunnels. This approach abstracts the Bandwidth Manager from the detail of modelling the entire network state; it need not know the actual core network topology, but just needs to track the availability TE tunnel bandwidth; TE enables this abstraction for the core. For example, a possible abstracted Bandwidth Manager representation of the core network in Figure 2 is shown in Figure 3.

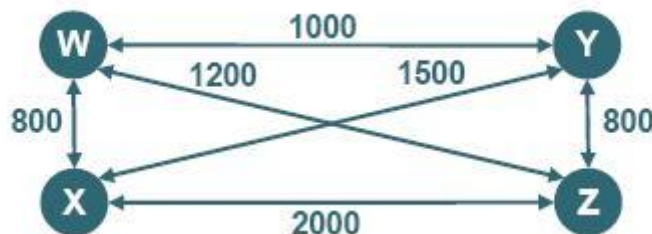


Figure 3. Bandwidth manager tunnel bandwidth representation²

Given the network above, consider the following Bandwidth Manager sequence of events:

1. Tunnels are configured from each edge router to every other edge router. A system other than the Bandwidth Manager may be responsible for tunnel configuration and sizing, or the tunnels may be statically configured (even if they are dynamically signalled).

Tunnel paths can either be dynamically calculated online in a distributed fashion by the TE tunnel head-ends themselves (i.e. using a dynamic path option) or can be calculated by an offline centralised function (i.e. tunnel server or path computation element), which then specifies the explicit tunnel path a head-end should use for a particular tunnel (using an explicit path option). With either approach, a CSPF calculation is used to determine the path that a particular tunnel will take. This

² This representation shows symmetrical bidirectional bandwidth resources, however, there is no requirement for bandwidth symmetry

CSPF calculation is similar to a conventional IGP SPF calculation, but it also takes into account bandwidth and administrative constraints, to determine the shortest path to a destination, which also satisfies those constraints. Whether online or offline path calculation is used, the output is an explicit route object (ERO) which defines the hop-by-hop tunnel path and which is handed to RSVP in order to signal the tunnel label switched path (LSP).

2. The Bandwidth Manager may use an auditing capability to determine what tunnels originate from each head-end, which router is the tail-end of each tunnel, and what bandwidth is provisioned for each tunnel.
3. The call control layer requests admission for a call to be setup from IP address A to IP address B
4. The Bandwidth Manager resolves the call to the impacted network resources, i.e. TE tunnels, and responds depending upon the status of those tunnels and any configured policies.
5. Having determined which network resources are impacted by the call, a number of potential models could be applied, with respect to the processing of admission control decisions and any required network interaction. These models differ in the level of network control and complexity required of the Bandwidth Manager, and the required interaction with the head-ends. For the purposes of this IA, we assume a simple model (which we call **Model #1**), where the Bandwidth Manager does not change configuration or size of any tunnels, but rather the Bandwidth Manager simply tracks tunnel and head- end status and manages admission control decisions into available TE tunnel bandwidth accordingly. The impacts of more complicated models on the TC-2 interface requirements are discussed in section 4.

In the context of this model, possible responses are:

- a. IF the head-end of a tunnel is down, the tunnel must be down; the call should be rejected and the call control layer informed
- b. ELSE IF a tunnel itself is down, e.g. as a result of a network failure there are no paths available which meet the tunnel bandwidth constraint, then the call should be rejected and the call control layer informed
- c. ELSE IF the tail-end of a tunnel is down, the tunnel must be down, and the call should be rejected and the call control layer informed
- d. ELSE IF FRR is used and a tunnel is rerouted using FRR somewhere along the path, i.e. due to a failed network element, AND IF bandwidth protection is not used (e.g. bandwidth is not specified for the FRR protection tunnel) which would be a network design decision, then even though the tunnel is up there may be insufficient bandwidth for the call on the FRR backup path and the call should be rejected and the call control layer informed. There are a number of potential techniques which can be used to reduce or mitigate this possibility, but they are beyond the scope of this IA definition.

- e. ELSE IF there is insufficient bandwidth available on the tunnel (i.e. insufficient bandwidth unused by other calls) to support the call or if policies dictate, the call should be rejected and the call control layer informed
- f. ELSE the call should be accepted and the call control layer informed. The bandwidth manager updates its state table of available bandwidth accordingly.

3 TC-2 Interface IA

A TC-2 interface, capable of supporting the Bandwidth Manager deployment model described in section 2, could be realised using SNMP v2c or v3 and the MPLS-TE-STD-MIB defined in RFC3812 [4]. Earlier versions of SNMP are not recommended for this interface because they do not support a reliable notification capability; such reliable notification is considered essential for the TC-2 interface.

The following network information and status indications would be communicated from the network to the Bandwidth Manager over the TC-2 interface, using SNMP.

3.1 Tunnel Auditing

Auditing of tunnel head-ends to determine what tunnels originate from each head-end, which router is the tail-end of each tunnel, and what bandwidth is provisioned for each tunnel, can be achieved using the MPLS-TE-STD-MIB.

3.1.1 Tunnels on head-end

The Tunnel table (*mplsTunnelTable*) can be used to determine which tunnels originate from a particular head-end. The Tunnel table shows both transit and terminated tunnels. The *mplsTunnelRole* value can be used to determine the role of the tunnel on the queried router; a value of 1 indicates that this router is the head-end of the respective tunnel.

3.1.2 Tail-end of each tunnel

The *mplsTunnelEgressLSRID* entry with the Tunnel table (*mplsTunnelTable*) identifies the tail-end of each tunnel; the IP address of the tunnel tail-end can be retrieved from the tunnel specified hop table (*mplsTunnelHopTable*).

3.1.3 Bandwidth of each tunnel

The bandwidth of each tunnel can be retrieved from the Resource table (*mplsTunnelResourceTable*) using the *mplsTunnelResourcePointer* of the respective tunnel.

An example walk of the TE MIB is included in Appendix A, with the above attributes identified.

3.2 Tunnel status

3.2.1 Tunnel Up / down

Tunnel up / down status can be communicated from the head-end using SNMP informs³ (for reliability) of the *mplsTunnelUp* / *mplsTunnelDown* notifications from the MPLS-TE-STD-MIB.

In addition, if the head-end router implementation treats the tunnel as a logical interface, tunnel up / down status may be communicated from the head-end using SNMP informs of the *linkup* / *linkDown* notifications from the Interfaces MIB defined in RFC2863 [5].

3.2.2 Tunnel Rerouted

Indication that a tunnel is rerouted, i.e. by FRR due to a network element failure, is provided via SNMP informs of the *mplsTunnelRerouted* notification from the MPLS-TE-STD-MIB. This indication is needed because when tunnels are in the rerouted state, prior to re-optimisation, if bandwidth protection (e.g. bandwidth is not specified for the backup tunnel) is not used there may not be sufficient bandwidth on the rerouted path to support the rerouted tunnel.

3.2.3 Tunnel Re-optimised

The *mplsTunnelReoptimized* notification from the MPLS-TE-STD-MIB can be used to determine when a tunnel has been successfully re-optimised subsequent to having been rerouted, and hence when new calls can be accepted into that tunnel.

3.3 Head-end status

Availability of the tunnel head-ends can be verified via the use of a Bidirectional Forwarding Detection (BFD) [6] or a simple ping keep-alives. This is required in order to determine when a tunnel head-end itself has failed – as clearly there are no traps which explicitly provide this notification – and hence to be able to deny all call requests which resolve to the impacted head-end.

A simple liveness check could send keep-alives to each head-end with a period X ms, and consider the head-end unavailable when no replies are received for a multiple Y of X , i.e. $X * Y$ ms. The head-end would be considered available again when Z consecutive replies are received from the head-end.

3.4 Tail-end status

No explicit means is required to determine tail-end status, as if the tail-end fails, any tunnels terminating on that tail-end will fail also, and hence by tracking tunnel status, tail-end failure is detected by implication.

³ SNMP informs (supported in v2c and v3) are acknowledged unlike the unreliable traps, which are the only form of notification supported in v1.

4 TC-2 Interface Evolutions

In section 2, a simple deployment model (**Model #1**) was assumed where the Bandwidth Manager does not change the configuration or size of any tunnels, but rather the Bandwidth Manager simply tracks tunnel and head-end status and manages admission control decisions into available TE tunnel bandwidth accordingly. Other deployment models are possible and may differ in the level of network control and complexity required of the Bandwidth Manager; consequently they may impose different requirements on the TC-2 interface. Some possible more complicated models and their impact on the TC-2 interface requirements are discussed below:

- **Model #2.** A more complicated deployment model could see the Bandwidth Manager attempt to dynamically resize the requested tunnel bandwidth specified on the tunnel head-end, based upon the received bandwidth requests. In this model, the TE tunnel head-end routers are also responsible for tunnel path calculation (using a dynamic path option), both for CAC and for bandwidth optimisation (offload routing), whilst the Bandwidth Manager controls both the specified TE tunnel bandwidth and admission to that bandwidth.

This model would require a TC-2 interface which provided the capability for the Bandwidth Manager to change the bandwidth of a particular tunnel at a tunnel head-end. This capability is provided by the Resource table (*mplsTunnelResourceTable*) within the MPLS-TE-STD-MIB.

- **Model #3.** In a still more complicated model, the Bandwidth Manager could act as a tunnel server or Path Computation Element (PCE) [6]; in this model it is aware of the network topology and determines tunnel explicit paths based upon the received bandwidth requests and configures the tunnel head-ends accordingly, using an explicit path option. In this model, the Bandwidth Manager is responsible for explicit path calculation, both for network level CAC and for bandwidth optimisation (offload routing), and for admission control into the tunnel bandwidth; TE tunnels are used simply to define explicitly routed paths through the network.

This model would require a TC-2 interface which provided the capability for the Bandwidth Manager to configure tunnels and their explicit paths at a tunnel head-end. This capability is provided by the Tunnel specified hop table (*mplsTunnelHopTable*) within the MPLS-TE-STD-MIB.

If models #2 and #3 described above are required, they could potentially be supported by the MPLS-TE-STD-MIB defined in RFC3812, although it is recommended that they be considered outside the scope of GMI2006.

5 Compliance

To be compliant with the TC-2 interface as defined in this IA:

1. The bandwidth manager and network element MUST support the RFC 3812 MPLS-TE-STD-MIB, for auditing head-ends and receiving notifications of network status as described in section 3.1

2. The bandwidth manager and network element **MUST** support SNMPv2c or v3, in order to support informs for reliable notification of network status as described in section 3.2
3. The bandwidth manager **MUST** support the use of a ping keepalive mechanism for tracking head-end status as described in section 3.3
4. The bandwidth manager and network element **SHOULD** support the use of BFD for tracking head-end status as described in section 3.3

6 References

- [1] "Bandwidth Management in Next Generation Packet Networks", MSF-TR-ARCH-005-FINAL
- [2] [MSF-TR-ARCH-008-FINAL] "Network Engineering to Support the Bandwidth Manager Architecture", MSF White Paper, May 2006
- [3] Adrian Farrel, Jean-Philippe Vasseur, Jerry Ash, "A Path Computation Element (PCE) Based Architecture", IETF draft-ietf-pce-architecture-04.txt, January 2006
- [4] T. Nadeau et al, "Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) Management Information Base (MIB)", IETF RFC 3812, June 2004
- [5] K. McCloghrie, F. Kastenholz, "The Interfaces Group MIB", IETF RFC 2863
- [6] D. Katz, D. Ward, "Bidirectional Forwarding Detection", IETF draft-ietf-bfd-base-04.txt, October 2005

APPENDIX A: EXAMPLE TE MIB Walk

```
**** SNMP QUERY STARTED ****
1: mplsTunnelConfigured.0 (gauge) 1
2: mplsTunnelActive.0 (gauge) 1
3: mplsTunnelTEDistProto.0 (octet string) @ [ospf(1)]
4: mplsTunnelMaxHops.0 (gauge) 65535
5: mplsTunnelIndexNext.0 (integer) 0
6: mplsTunnelName.10.0.168430081.168430084 (octet string) Tunnel10 [54.75.6E.6E.65.6C.31.30 (hex)] << Tunnel ID
7: mplsTunnelName.10.145.168430081.168430084 (octet string) (zero-length)
8: mplsTunnelDescr.10.0.168430081.168430084 (octet string) PE1_t10 [50.45.31.5F.74.31.30 (hex)]
9: mplsTunnelDescr.10.145.168430081.168430084 (octet string) PE1_t10 [50.45.31.5F.74.31.30 (hex)]
10: mplsTunnelsIf.10.0.168430081.168430084 (integer) true(1)
11: mplsTunnelsIf.10.145.168430081.168430084 (integer) false(2)
12: mplsTunnelfIndex.10.0.168430081.168430084 (integer) 13 [13]
13: mplsTunnelfIndex.10.145.168430081.168430084 (integer) 0 [0]
14: mplsTunnelXCPointer.10.0.168430081.168430084 (object identifier) (null-oid) zeroDotZero
15: mplsTunnelXCPointer.10.145.168430081.168430084 (object identifier) (null-oid) zeroDotZero
16: mplsTunnelSignallingProto.10.0.168430081.168430084 (integer) rsvp(2)
17: mplsTunnelSignallingProto.10.145.168430081.168430084 (integer) rsvp(2)
18: mplsTunnelSetupPrio.10.0.168430081.168430084 (integer) 0
19: mplsTunnelSetupPrio.10.145.168430081.168430084 (integer) 0
20: mplsTunnelHoldingPrio.10.0.168430081.168430084 (integer) 0
21: mplsTunnelHoldingPrio.10.145.168430081.168430084 (integer) 0
22: mplsTunnelSessionAttributes.10.0.168430081.168430084 (octet string) 08 (hex) [isComputed(4)]
23: mplsTunnelSessionAttributes.10.145.168430081.168430084 (octet string) B0 (hex) [fastReroute(0) | isPersistent(2) | isPinned(3)]
24: mplsTunnelOwner.10.0.168430081.168430084 (integer) admin(1)
25: mplsTunnelOwner.10.145.168430081.168430084 (integer) other(5)
26: mplsTunnelLocalProtectInUse.10.0.168430081.168430084 (integer) false(2)
27: mplsTunnelLocalProtectInUse.10.145.168430081.168430084 (integer) false(2)
28: mplsTunnelResourcePointer.10.0.168430081.168430084 (object identifier) (null-oid) zeroDotZero
29: mplsTunnelResourcePointer.10.145.168430081.168430084 (object identifier) mplsTunnelResourceMaxRate.28186368
30: mplsTunnelInstancePriority.10.0.168430081.168430084 (gauge) 0
31: mplsTunnelInstancePriority.10.145.168430081.168430084 (gauge) 0
32: mplsTunnelHopTableIndex.10.0.168430081.168430084 (gauge) 10
33: mplsTunnelHopTableIndex.10.145.168430081.168430084 (gauge) 0
34: mplsTunnelARHopTableIndex.10.0.168430081.168430084 (gauge) 0
35: mplsTunnelARHopTableIndex.10.145.168430081.168430084 (gauge) 28186368
36: mplsTunnelCHopTableIndex.10.0.168430081.168430084 (gauge) 0
37: mplsTunnelCHopTableIndex.10.145.168430081.168430084 (gauge) 28186368
38: mplsTunnelPrimaryInstance.10.0.168430081.168430084 (gauge) 145
39: mplsTunnelPrimaryInstance.10.145.168430081.168430084 (gauge) 0
40: mplsTunnelPrimaryTimeUp.10.0.168430081.168430084 (timeticks) 0 days 00h:06m:52s.96th (41296)
41: mplsTunnelPrimaryTimeUp.10.145.168430081.168430084 (timeticks) 0 days 00h:00m:00s.00th (0)
42: mplsTunnelPathChanges.10.0.168430081.168430084 (counter) 11
43: mplsTunnelPathChanges.10.145.168430081.168430084 (counter) 0
44: mplsTunnelLastPathChange.10.0.168430081.168430084 (timeticks) 0 days 00h:06m:50s.35th (41035)
45: mplsTunnelLastPathChange.10.145.168430081.168430084 (timeticks) 0 days 00h:00m:00s.00th (0)
46: mplsTunnelCreationTime.10.0.168430081.168430084 (timeticks) 0 days 00h:00m:02s.63th (263)
47: mplsTunnelCreationTime.10.145.168430081.168430084 (timeticks) 0 days 00h:00m:00s.00th (0)
48: mplsTunnelStateTransitions.10.0.168430081.168430084 (counter) 7
49: mplsTunnelStateTransitions.10.145.168430081.168430084 (counter) 0
50: mplsTunnelIncludeAnyAffinity.10.0.168430081.168430084 (gauge) 0
51: mplsTunnelIncludeAnyAffinity.10.145.168430081.168430084 (gauge) 0
52: mplsTunnelIncludeAllAffinity.10.0.168430081.168430084 (gauge) 0
53: mplsTunnelIncludeAllAffinity.10.145.168430081.168430084 (gauge) 0
54: mplsTunnelExcludeAllAffinity.10.0.168430081.168430084 (gauge) 65535
55: mplsTunnelExcludeAllAffinity.10.145.168430081.168430084 (gauge) 0
56: mplsTunnelPathInUse.10.0.168430081.168430084 (gauge) 1
57: mplsTunnelPathInUse.10.145.168430081.168430084 (gauge) 0
58: mplsTunnelRole.10.0.168430081.168430084 (integer) head(1) << Identifies tunnel as head-end
59: mplsTunnelRole.10.145.168430081.168430084 (integer) head(1)
60: mplsTunnelTotalUpTime.10.0.168430081.168430084 (timeticks) 0 days 02h:04m:44s.14th (748414)
61: mplsTunnelTotalUpTime.10.145.168430081.168430084 (timeticks) 0 days 00h:00m:00s.00th (0)
62: mplsTunnelInstanceUpTime.10.0.168430081.168430084 (timeticks) 0 days 00h:00m:00s.00th (0)
63: mplsTunnelInstanceUpTime.10.145.168430081.168430084 (timeticks) 0 days 00h:00m:00s.00th (0)
64: mplsTunnelAdminStatus.10.0.168430081.168430084 (integer) up(1)
65: mplsTunnelAdminStatus.10.145.168430081.168430084 (integer) up(1)
66: mplsTunnelOperStatus.10.0.168430081.168430084 (integer) up(1)
67: mplsTunnelOperStatus.10.145.168430081.168430084 (integer) up(1)
68: mplsTunnelRowStatus.10.0.168430081.168430084 (integer) active(1)
69: mplsTunnelRowStatus.10.145.168430081.168430084 (integer) active(1)
70: mplsTunnelStorageType.10.0.168430081.168430084 (integer) readOnly(5)
71: mplsTunnelStorageType.10.145.168430081.168430084 (integer) readOnly(5)
```

72: mplsTunnelHopListIndexNext.0 (gauge) 0
73: mplsTunnelHopAddrType.10.1.1 (integer) ipV4(1)
74: mplsTunnelHopIpv4Addr.10.1.1 (octet string) 10.10.10.4 [0A.0A.0A.04 (hex)]
75: mplsTunnelHopIpv4PrefixLen.10.1.1 (gauge) 32 <<< IP addr of tunnel tail-end
76: mplsTunnelHopIpv6Addr.10.1.1 (octet string) (zero-length)
77: mplsTunnelHopIpv6PrefixLen.10.1.1 (gauge) 0
78: mplsTunnelHopAsNumber.10.1.1 (gauge) 0
79: mplsTunnelHopLspld.10.1.1 (octet string) (zero-length)
80: mplsTunnelHopType.10.1.1 (integer) strict(1)
81: mplsTunnelHopRowStatus.10.1.1 (integer) active(1)
82: mplsTunnelHopStorageType.10.1.1 (integer) readOnly(5)
83: mplsTunnelResourceIndexNext.0 (gauge) 0
84: mplsTunnelResourceMaxRate.28186368 (integer) 999000 [999000] <<< Tunnel bandwidth
85: mplsTunnelResourceMeanRate.28186368 (integer) 999000 [999000]
86: mplsTunnelResourceMaxBurstSize.28186368 (integer) 1000 [1000]
87: mplsTunnelResourceRowStatus.28186368 (integer) active(1)
88: mplsTunnelResourceStorageType.28186368 (integer) readOnly(5)
89: mplsTunnelARHopAddrType.28186368.1 (integer) ipV4(1)
90: mplsTunnelARHopAddrType.28186368.2 (integer) ipV4(1)
91: mplsTunnelARHopAddrType.28186368.3 (integer) ipV4(1)
92: mplsTunnelARHopAddrType.28186368.4 (integer) ipV4(1)
93: mplsTunnelARHopIpv4Addr.28186368.1 (octet string) 10.10.10.10 [0A.0A.0A.0A (hex)]
94: mplsTunnelARHopIpv4Addr.28186368.2 (octet string) 10.10.10.11 [0A.0A.0A.0B (hex)]
95: mplsTunnelARHopIpv4Addr.28186368.3 (octet string) 10.10.10.13 [0A.0A.0A.0D (hex)]
96: mplsTunnelARHopIpv4Addr.28186368.4 (octet string) 10.10.10.4 [0A.0A.0A.04 (hex)]
97: mplsTunnelARHopIpv4PrefixLen.28186368.1 (gauge) 32
98: mplsTunnelARHopIpv4PrefixLen.28186368.2 (gauge) 32
99: mplsTunnelARHopIpv4PrefixLen.28186368.3 (gauge) 32
100: mplsTunnelARHopIpv4PrefixLen.28186368.4 (gauge) 32
101: mplsTunnelARHopIpv6Addr.28186368.1 (octet string) (zero-length)
102: mplsTunnelARHopIpv6Addr.28186368.2 (octet string) (zero-length)
103: mplsTunnelARHopIpv6Addr.28186368.3 (octet string) (zero-length)
104: mplsTunnelARHopIpv6Addr.28186368.4 (octet string) (zero-length)
105: mplsTunnelARHopIpv6PrefixLen.28186368.1 (gauge) 0
106: mplsTunnelARHopIpv6PrefixLen.28186368.2 (gauge) 0
107: mplsTunnelARHopIpv6PrefixLen.28186368.3 (gauge) 0
108: mplsTunnelARHopIpv6PrefixLen.28186368.4 (gauge) 0
109: mplsTunnelARHopAsNumber.28186368.1 (gauge) 0
110: mplsTunnelARHopAsNumber.28186368.2 (gauge) 0
111: mplsTunnelARHopAsNumber.28186368.3 (gauge) 0
112: mplsTunnelARHopAsNumber.28186368.4 (gauge) 0
113: mplsTunnelARHopType.28186368.1 (integer) strict(1)
114: mplsTunnelARHopType.28186368.2 (integer) strict(1)
115: mplsTunnelARHopType.28186368.3 (integer) strict(1)
116: mplsTunnelARHopType.28186368.4 (integer) strict(1)
117: mplsTunnelCHopAddrType.28186368.1 (integer) ipV4(1)
118: mplsTunnelCHopAddrType.28186368.2 (integer) ipV4(1)
119: mplsTunnelCHopAddrType.28186368.3 (integer) ipV4(1)
120: mplsTunnelCHopAddrType.28186368.4 (integer) ipV4(1)
121: mplsTunnelCHopAddrType.28186368.5 (integer) ipV4(1)
122: mplsTunnelCHopAddrType.28186368.6 (integer) ipV4(1)
123: mplsTunnelCHopAddrType.28186368.7 (integer) ipV4(1)
124: mplsTunnelCHopAddrType.28186368.8 (integer) ipV4(1)
125: mplsTunnelCHopIpv4Addr.28186368.1 (octet string) 1.0.0.2 [01.00.00.02 (hex)]
126: mplsTunnelCHopIpv4Addr.28186368.2 (octet string) 1.0.1.1 [01.00.01.01 (hex)]
127: mplsTunnelCHopIpv4Addr.28186368.3 (octet string) 1.0.1.2 [01.00.01.02 (hex)]
128: mplsTunnelCHopIpv4Addr.28186368.4 (octet string) 3.0.1.2 [03.00.01.02 (hex)]
129: mplsTunnelCHopIpv4Addr.28186368.5 (octet string) 3.0.1.1 [03.00.01.01 (hex)]
130: mplsTunnelCHopIpv4Addr.28186368.6 (octet string) 2.0.2.1 [02.00.02.01 (hex)]
131: mplsTunnelCHopIpv4Addr.28186368.7 (octet string) 2.0.2.2 [02.00.02.02 (hex)]
132: mplsTunnelCHopIpv4Addr.28186368.8 (octet string) 10.10.10.4 [0A.0A.0A.04 (hex)]
133: mplsTunnelCHopIpv4PrefixLen.28186368.1 (gauge) 32
134: mplsTunnelCHopIpv4PrefixLen.28186368.2 (gauge) 32
135: mplsTunnelCHopIpv4PrefixLen.28186368.3 (gauge) 32
136: mplsTunnelCHopIpv4PrefixLen.28186368.4 (gauge) 32
137: mplsTunnelCHopIpv4PrefixLen.28186368.5 (gauge) 32
138: mplsTunnelCHopIpv4PrefixLen.28186368.6 (gauge) 32
139: mplsTunnelCHopIpv4PrefixLen.28186368.7 (gauge) 32
140: mplsTunnelCHopIpv4PrefixLen.28186368.8 (gauge) 32
141: mplsTunnelCHopIpv6Addr.28186368.1 (octet string) (zero-length)
142: mplsTunnelCHopIpv6Addr.28186368.2 (octet string) (zero-length)
143: mplsTunnelCHopIpv6Addr.28186368.3 (octet string) (zero-length)
144: mplsTunnelCHopIpv6Addr.28186368.4 (octet string) (zero-length)
145: mplsTunnelCHopIpv6Addr.28186368.5 (octet string) (zero-length)
146: mplsTunnelCHopIpv6Addr.28186368.6 (octet string) (zero-length)
147: mplsTunnelCHopIpv6Addr.28186368.7 (octet string) (zero-length)
148: mplsTunnelCHopIpv6Addr.28186368.8 (octet string) (zero-length)

149: mplsTunnelCHopIpv6PrefixLen.28186368.1 (gauge) 0
150: mplsTunnelCHopIpv6PrefixLen.28186368.2 (gauge) 0
151: mplsTunnelCHopIpv6PrefixLen.28186368.3 (gauge) 0
152: mplsTunnelCHopIpv6PrefixLen.28186368.4 (gauge) 0
153: mplsTunnelCHopIpv6PrefixLen.28186368.5 (gauge) 0
154: mplsTunnelCHopIpv6PrefixLen.28186368.6 (gauge) 0
155: mplsTunnelCHopIpv6PrefixLen.28186368.7 (gauge) 0
156: mplsTunnelCHopIpv6PrefixLen.28186368.8 (gauge) 0
157: mplsTunnelCHopAsNumber.28186368.1 (gauge) 0
158: mplsTunnelCHopAsNumber.28186368.2 (gauge) 0
159: mplsTunnelCHopAsNumber.28186368.3 (gauge) 0
160: mplsTunnelCHopAsNumber.28186368.4 (gauge) 0
161: mplsTunnelCHopAsNumber.28186368.5 (gauge) 0
162: mplsTunnelCHopAsNumber.28186368.6 (gauge) 0
163: mplsTunnelCHopAsNumber.28186368.7 (gauge) 0
164: mplsTunnelCHopAsNumber.28186368.8 (gauge) 0
165: mplsTunnelCHopType.28186368.1 (integer) strict(1)
166: mplsTunnelCHopType.28186368.2 (integer) strict(1)
167: mplsTunnelCHopType.28186368.3 (integer) strict(1)
168: mplsTunnelCHopType.28186368.4 (integer) strict(1)
169: mplsTunnelCHopType.28186368.5 (integer) strict(1)
170: mplsTunnelCHopType.28186368.6 (integer) strict(1)
171: mplsTunnelCHopType.28186368.7 (integer) strict(1)
172: mplsTunnelCHopType.28186368.8 (integer) strict(1)
173: mplsTunnelPerfPackets.10.0.168430081.168430084 (counter) 15
174: mplsTunnelPerfPackets.10.145.168430081.168430084 (counter) 0
175: mplsTunnelPerfHCPackets.10.0.168430081.168430084 (counter64) 0
176: mplsTunnelPerfHCPackets.10.145.168430081.168430084 (counter64) 0
177: mplsTunnelPerfErrors.10.0.168430081.168430084 (counter) 0
178: mplsTunnelPerfErrors.10.145.168430081.168430084 (counter) 0
179: mplsTunnelPerfBytes.10.0.168430081.168430084 (counter) 1620
180: mplsTunnelPerfBytes.10.145.168430081.168430084 (counter) 0
181: mplsTunnelPerfHCBytes.10.0.168430081.168430084 (counter64) 0
182: mplsTunnelPerfHCBytes.10.145.168430081.168430084 (counter64) 0
183: mplsTunnelTrapEnable.0 (integer) true(1)
***** SNMP QUERY FINISHED *****