

Internet Engineering Task Force
Internet Draft
Intended status: Standards Track
Expires: September 2017

Kun Wang
Ericsson
Alex Wang
Ericsson
Chin Chen
Ericsson
Hua Lv
Ericsson
March 29, 2017

**Yang Data Model for CFM Protocol
draft-wang-netmod-cfm-yang-00.txt**

Abstract

This document defines a YANG data model that can be used to configure and manage CFM.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [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 29, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Overview.....	2
1.1.	Requirement language.....	2
2.	Design of Data Model.....	3
2.1.	Overview.....	3
2.2.	CFM stack.....	8
2.3.	CFM Default MD.....	9
2.4.	CFM VLAN.....	9
2.5.	CFM Config Error List.....	10
2.6.	CFM MD.....	10
2.7.	CFM MA.....	11
2.8.	CFM MEP.....	12
3.	CFM YANG Module.....	20
4.	Security Considerations.....	90
5.	IANA Considerations.....	90
6.	Normative References.....	90

[1. Overview](#)

This document defines a YANG [[RFC6020](#)] data model which can be used for configuring and managing CFM [[IEEE802.1ag](#)].

This data model includes configuration data and state data, for example the configuration of MD, MA, and MEP and the state of MEP.

[1.1. Requirement language](#)

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

2. Design of Data Model

The goal of this document is to define a data model that provides a common user interface to the CFM. There is very little information that is designated as "mandatory", providing freedom for vendors to adapt this data model to their respective product implementations.

2.1. Overview

The CFM YANG module defined in this document has all the common building blocks for the CFM protocol.

The CFM module is created based on CFM MIB with a slight change for meeting the YANG model requirement.

```

module: ietf-cfm
  +--ro dot1agCfmStack
    | +--ro dot1agCfmStackEntry* [dot1agCfmStackifIndex
dot1agCfmStackVlanIdOrNone dot1agCfmStackMdLevel
dot1agCfmStackDirection]
    |   +--ro dot1agCfmStackifIndex          cfm:if-index
    |   +--ro dot1agCfmStackVlanIdOrNone      cfm:VlanIdOrNone
    |   +--ro dot1agCfmStackMdLevel           cfm:Dot1agCfmMDLevel
    |   +--ro dot1agCfmStackDirection         cfm:Dot1agCfmMpDirection
    |   +--ro dot1agCfmStackMdIndex?          uint32
    |   +--ro dot1agCfmStackMaIndex?          uint32
    |   +--ro dot1agCfmStackMepId?            cfm:Dot1agCfmMepIdOrZero
    |   +--ro dot1agCfmStackMacAddress?       yang:mac-address
  +--rw dot1agCfmDefaultMd
    | +--rw dot1agCfmDefaultMdDefLevel?       cfm:Dot1agCfmMDLevel
    | +--rw dot1agCfmDefaultMdDefMhfCreation?
cfm:Dot1agCfmMhfCreation
    | +--rw dot1agCfmDefaultMdDefIdPermission?
cfm:Dot1agCfmIdPermission
    | +--rw dot1agCfmDefaultMdEntry* [dot1agCfmDefaultMdComponentId
dot1agCfmDefaultMdPrimaryVid]
    |   +--rw dot1agCfmDefaultMdComponentId
cfm:Dot1agCfmPbbComponentIdentifier
    |   +--rw dot1agCfmDefaultMdPrimaryVid     cfm:VlanId
    |   +--ro dot1agCfmDefaultMdStatus?        boolean
    |   +--rw dot1agCfmDefaultMdLevel?
cfm:Dot1agCfmMDLevelOrNone
    |   +--rw dot1agCfmDefaultMdMhfCreation?
cfm:Dot1agCfmMhfCreation
    |   +--rw dot1agCfmDefaultMdIdPermission?
cfm:Dot1agCfmIdPermission
  +--rw dot1agCfmVlan

```



```

    | +--rw dot1agCfmVlanEntry* [dot1agCfmVlanComponentId
dot1agCfmVlanVid]
    |   +--rw dot1agCfmVlanComponentId
cfm:Dot1agCfmPbbComponentIdentifier
    |   +--rw dot1agCfmVlanVid          cfm:VlanId
    |   +--rw dot1agCfmVlanPrimaryVid?  cfm:VlanId
    |   +--rw dot1agCfmVlanRowStatus?    cfm:RowStatus
+--ro dot1agCfmConfigErrorList
| +--ro dot1agCfmConfigErrorListEntry*
[dot1agCfmConfigErrorListVid dot1agCfmConfigErrorListIfIndex]
|   +--ro dot1agCfmConfigErrorListVid      cfm:VlanId
|   +--ro dot1agCfmConfigErrorListIfIndex  cfm:if-index
|   +--ro dot1agCfmConfigErrorListErrorType?
cfm:Dot1agCfmConfigErrors
+--rw dot1agCfmMd
| +--ro dot1agCfmMdTableNextIndex?
cfm:Dot1afCfmIndexIntegerNextFree
| +--rw dot1agCfmMdEntry* [dot1agCfmMdIndex]
|   +--rw dot1agCfmMdIndex          uint32
|   +--rw dot1agCfmMdFormat?
cfm:Dot1agCfmMaintDomainNameType
|   +--rw dot1agCfmMdName?
cfm:Dot1agCfmMaintDomainName
|   +--rw dot1agCfmMdMdLevel?          cfm:Dot1agCfmMDLevel
|   +--rw dot1agCfmMdMhfCreation?      cfm:Dot1agCfmMhfCreation
|   +--rw dot1agCfmMdMhfIdPermission?
cfm:Dot1agCfmIdPermission
|   +--ro dot1agCfmMdMaNextIndex?
cfm:Dot1afCfmIndexIntegerNextFree
|   +--rw dot1agCfmMdRowStatus?        cfm:RowStatus
+--rw dot1agCfmMa
| +--rw dot1agCfmMaNetEntry* [dot1agCfmMdIndex dot1agCfmMaIndex]
| | +--rw dot1agCfmMdIndex          ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
| | +--rw dot1agCfmMaIndex          uint32
| | +--rw dot1agCfmMaNetFormat?
cfm:Dot1agCfmMaintAssocNameType
| | +--rw dot1agCfmMaNetName?
cfm:Dot1agCfmMaintAssocName
| | +--rw dot1agCfmMaNetCcmInterval?  cfm:Dot1agCfmCcmInterval
| | +--rw dot1agCfmMaNetRowStatus?    cfm:RowStatus
| +--rw dot1agCfmMaCompEntry* [dot1agCfmMaComponentId
dot1agCfmMdIndex dot1agCfmMaIndex]
| | +--rw dot1agCfmMdIndex          ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
| | +--rw dot1agCfmMaIndex          ->
/dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex

```



```

    | | +--rw dot1agCfmMaComponentId
cfm:Dot1agCfmPbbComponentIdentifier
    | | +--rw dot1agCfmMaCompPrimaryVlanId?   cfm:VlanIdOrNone
    | | +--rw dot1agCfmMaCompMhfCreation?
cfm:Dot1agCfmMhfCreation
    | | +--rw dot1agCfmMaCompIdPermission?
cfm:Dot1agCfmIdPermission
    | | +--ro dot1agCfmMaCompNumberOfVids?    uint32
    | | +--rw dot1agCfmMaCompRowStatus?       cfm:RowStatus
    | +--rw dot1agCfmMaMepListEntry* [dot1agCfmMdIndex
dot1agCfmMaIndex dot1agCfmMaMepListIdentifier]
    |   +--rw dot1agCfmMdIndex                ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
    |   +--rw dot1agCfmMaIndex                ->
/dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex
    |   +--rw dot1agCfmMaMepListIdentifier    cfm:Dot1agCfmMepId
    |   +--rw dot1agCfmMaMepListRowStatus?   cfm:RowStatus
+--rw dot1agCfmMep
    +--rw dot1agCfmMepEntry* [dot1agCfmMdIndex dot1agCfmMaIndex
dot1agCfmMepIdentifier]
    | +--rw dot1agCfmMdIndex                  ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
    | +--rw dot1agCfmMaIndex                  ->
/dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex
    | +--rw dot1agCfmMepIdentifier
cfm:Dot1agCfmMepId
    | +--rw dot1agCfmMepIfIndex?              cfm:if-
index-or-zero
    | +--rw dot1agCfmMepDirection?
cfm:Dot1agCfmMpDirection
    | +--rw dot1agCfmMepPrimaryVid?           uint32
    | +--rw dot1agCfmMepActive?              boolean
    | +--ro dot1agCfmMepFngState?
cfm:Dot1agCfmFngState
    | +--rw dot1agCfmMepCciEnabled?           boolean
    | +--rw dot1agCfmMepCcmLtmPriority?       uint32
    | +--ro dot1agCfmMepMacAddress?           yang:mac-
address
    | +--rw dot1agCfmMepLowPrDef?
cfm:Dot1agCfmLowestAlarmPri
    | +--rw dot1agCfmMepFngAlarmTime?
cfm:TimeInterval
    | +--rw dot1agCfmMepFngResetTime?
cfm:TimeInterval
    | +--ro dot1agCfmMepHighestPrDefect?
cfm:Dot1agCfmHighestDefectPri

```



```

    | +--ro dot1agCfmMepDefects?
cfm:Dot1agCfmMepDefects
    | +--ro dot1agCfmMepErrorCcmLastFailure?        binary
    | +--ro dot1agCfmMepXconCcmLastFailure?        binary
    | +--ro dot1agCfmMepCcmSequenceErrors?
yang:counter32
    | +--ro dot1agCfmMepCciSentCcms?
yang:counter32
    | +--ro dot1agCfmMepNextLbmTransId?            uint32
    | +--ro dot1agCfmMepLbrIn?
yang:counter32
    | +--ro dot1agCfmMepLbrInOutOfOrder?
yang:counter32
    | +--ro dot1agCfmMepLbrBadMsdu?
yang:counter32
    | +--ro dot1agCfmMepLtmNextSeqNumber?          uint32
    | +--ro dot1agCfmMepUnexpLtrIn?
yang:counter32
    | +--ro dot1agCfmMepLbrOut?
yang:counter32
    | +--rw dot1agCfmMepTransmitLbmStatus?          boolean
    | +--rw dot1agCfmMepTransmitLbmDestMacAddress?  yang:mac-
address
    | +--rw dot1agCfmMepTransmitLbmDestMepId?
cfm:Dot1agCfmMepIdOrZero
    | +--rw dot1agCfmMepTransmitLbmDestIsMepId?    boolean
    | +--rw dot1agCfmMepTransmitLbmMessages?        int32
    | +--rw dot1agCfmMepTransmitLbmDataTlv?         binary
    | +--rw dot1agCfmMepTransmitLbmVlanPriority?    int32
    | +--rw dot1agCfmMepTransmitLbmVlanDropEnable?  boolean
    | +--ro dot1agCfmMepTransmitLbmResultOK?        boolean
    | +--ro dot1agCfmMepTransmitLbmSeqNumber?       uint32
    | +--ro dot1agCfmMepTransmitLtmStatus?          boolean
    | +--rw dot1agCfmMepTransmitLtmFlags?           bits
    | +--rw dot1agCfmMepTransmitLtmTargetMacAddress? yang:mac-
address
    | +--rw dot1agCfmMepTransmitLtmTargetMepId?
cfm:Dot1agCfmMepIdOrZero
    | +--rw dot1agCfmMepTransmitLtmTargetIsMepId?    boolean
    | +--rw dot1agCfmMepTransmitLtmTtl?             uint32
    | +--ro dot1agCfmMepTransmitLtmResult?          boolean
    | +--ro dot1agCfmMepTransmitLtmSeqNumber?       uint32
    | +--rw dot1agCfmMepTransmitLtmEgressIdentifier? binary
    | +--rw dot1agCfmMepRowStatus?
cfm:RowStatus

```



```

    +--rw dot1agCfmLtrEntry* [dot1agCfmMdIndex dot1agCfmMaIndex
dot1agCfmMepIdentifier dot1agCfmLtrSeqNumber
dot1agCfmLtrReceiveOrder]
      | +--rw dot1agCfmMdIndex ->
      /dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
      | +--rw dot1agCfmMaIndex ->
      /dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex
      | +--rw dot1agCfmMepIdentifier ->
      /dot1agCfmMep/dot1agCfmMepEntry/dot1agCfmMepIdentifier
      | +--rw dot1agCfmLtrSeqNumber uint32
      | +--rw dot1agCfmLtrReceiveOrder uint32
      | +--ro dot1agCfmLtrTtl? uint32
      | +--ro dot1agCfmLtrForwarded? boolean
      | +--ro dot1agCfmLtrTerminalMep? boolean
      | +--ro dot1agCfmLtrLastEgressIdentifier? binary
      | +--ro dot1agCfmLtrNextEgressIdentifier? binary
      | +--ro dot1agCfmLtrRelay?
cfm:Dot1agCfmRelayActionFieldValue
      | +--ro dot1agCfmLtrChassisIdSubtype?
cfm:LldpChassisIdSubtype
      | +--ro dot1agCfmLtrChassisId?
cfm:LldpChassisId
      | +--ro dot1agCfmLtrManAddressDomain? cfm:TDomain
      | +--ro dot1agCfmLtrManAddress? cfm:TAddress
      | +--ro dot1agCfmLtrIngress?
cfm:Dot1agCfmIngressActionFieldValue
      | +--ro dot1agCfmLtrIngressMac? yang:mac-
address
      | +--ro dot1agCfmLtrIngressPortIdSubtype?
cfm:LldpPortIdSubtype
      | +--ro dot1agCfmLtrIngressPortId? cfm:LldpPortId
      | +--ro dot1agCfmLtrEgress?
cfm:Dot1agCfmEgressActionFieldValue
      | +--ro dot1agCfmLtrEgressMac? yang:mac-
address
      | +--ro dot1agCfmLtrEgressPortIdSubtype?
cfm:LldpPortIdSubtype
      | +--ro dot1agCfmLtrEgressPortId? cfm:LldpPortId
      | +--ro dot1agCfmLtrOrganizationSpecificTlv? binary
    +--rw dot1agCfmMepDbEntry* [dot1agCfmMdIndex dot1agCfmMaIndex
dot1agCfmMepIdentifier dot1agCfmMepDbRMepIdentifier]
      +--rw dot1agCfmMdIndex ->
      /dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
      +--rw dot1agCfmMaIndex ->
      /dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex
      +--rw dot1agCfmMepIdentifier ->
      /dot1agCfmMep/dot1agCfmMepEntry/dot1agCfmMepIdentifier

```



```

    +---rw dot1agCfmMepDbRMepIdentifier          cfm:Dot1agCfmMepId
    +---ro dot1agCfmMepDbRMepState?
cfm:Dot1agCfmRemoteMepState
    +---ro dot1agCfmMepDbRMepFailedOkTime?      yang:timestamp
    +---ro dot1agCfmMepDbMacAddress?            yang:mac-address
    +---ro dot1agCfmMepDbRdi?                  boolean
    +---ro dot1agCfmMepDbPortStatusTlv?
cfm:Dot1agCfmPortStatus
    +---ro dot1agCfmMepDbInterfaceStatusTlv?
cfm:Dot1agCfmInterfaceStatus
    +---ro dot1agCfmMepDbChassisIdSubtype?
cfm:LldpChassisIdSubtype
    +---ro dot1agCfmMepDbChassisId?             cfm:LldpChassisId
    +---ro dot1agCfmMepDbManAddressDomain?      cfm:TDomain
    +---ro dot1agCfmMepDbManAddress?            cfm:TAddress
notifications:
    +---n dot1agCfmFaultAlarm
    +---ro dot1agCfmFaultAlarm-dot1agCfmMepHighestPrDefect
    +---ro dot1agCfmMdIndex?                    ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
    +---ro dot1agCfmMaIndex?                    ->
/dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex
    +---ro dot1agCfmMepIdentifier?              ->
/dot1agCfmMep/dot1agCfmMepEntry/dot1agCfmMepIdentifier
    +---ro dot1agCfmMepHighestPrDefect?
cfm:Dot1agCfmHighestDefectPri

```

2.2. CFM stack

It enables the network administrator to discover the information about the Maintenance Points configured on a port.

```
+--ro dot1agCfmStack
```

```

| +--ro dot1agCfmStackEntry* [dot1agCfmStackifIndex
dot1agCfmStackVlanIdOrNone dot1agCfmStackMdLevel
dot1agCfmStackDirection]

|   +---ro dot1agCfmStackifIndex          cfm:if-index

|   +---ro dot1agCfmStackVlanIdOrNone     cfm:VlanIdOrNone

|   +---ro dot1agCfmStackMdLevel          cfm:Dot1agCfmMDLevel

|   +---ro dot1agCfmStackDirection        cfm:Dot1agCfmMpDirection

```



```

|      +---ro dot1agCfmStackMdIndex?          uint32
|
|      +---ro dot1agCfmStackMaIndex?          uint32
|
|      +---ro dot1agCfmStackMepId?           cfm:Dot1agCfmMepIdOrZero
|
|      +---ro dot1agCfmStackMacAddress?       yang:mac-address

```

2.3. CFM Default MD

The Default MD is to control the MIP Half Function creation for VIDs that are not contained in the list VIDs associated with Maintenance Association.

```

+--rw dot1agCfmDefaultMd
|
|  +--rw dot1agCfmDefaultMdDefLevel?          cfm:Dot1agCfmMDLevel
|
|  +--rw dot1agCfmDefaultMdDefMhfCreation?    cfm:Dot1agCfmMhfCreation
|
|  +--rw dot1agCfmDefaultMdDefIdPermission?   cfm:Dot1agCfmIdPermission
|
|  +--rw dot1agCfmDefaultMdEntry* [dot1agCfmDefaultMdComponentId
dot1agCfmDefaultMdPrimaryVid]
|
|    +--rw dot1agCfmDefaultMdComponentId
cfm:Dot1agCfmPbbComponentIdentifier
|
|    +--rw dot1agCfmDefaultMdPrimaryVid       cfm:VlanId
|
|    +---ro dot1agCfmDefaultMdStatus?         boolean
|
|    +--rw dot1agCfmDefaultMdLevel?          cfm:Dot1agCfmMDLevelOrNone
|
|    +--rw dot1agCfmDefaultMdMhfCreation?    cfm:Dot1agCfmMhfCreation
|
|    +--rw dot1agCfmDefaultMdIdPermission?   cfm:Dot1agCfmIdPermission

```

2.4. CFM VLAN

It contains the association between VID and VLAN.

```

    +--rw dot1agCfmVlan
    |   +--rw dot1agCfmVlanEntry* [dot1agCfmVlanComponentId
dot1agCfmVlanVid]
    |       +--rw dot1agCfmVlanComponentId
cfm:Dot1agCfmPbbComponentIdentifier
    |       +--rw dot1agCfmVlanVid                cfm:VlanId
    |       +--rw dot1agCfmVlanPrimaryVid?        cfm:VlanId
    |       +--rw dot1agCfmVlanRowStatus?         cfm:RowStatus

```

2.5. CFM Config Error List

It contains the CFM configuration error for a VLAN ID on a port.

```

+--ro dot1agCfmConfigErrorList
|   +--ro dot1agCfmConfigErrorListEntry* [dot1agCfmConfigErrorListVid
dot1agCfmConfigErrorListIfIndex]
|       +--ro dot1agCfmConfigErrorListVid                cfm:VlanId
|       +--ro dot1agCfmConfigErrorListIfIndex            cfm:if-index
|       +--ro dot1agCfmConfigErrorListErrorType?
cfm:Dot1agCfmConfigErrors

```

2.6. CFM MD

Maintenance Domain container contains the Domain configuration information. The Maintenance Domain level which the monitor is at is also specified here.

```

+--rw dot1agCfmMd
|   +--ro dot1agCfmMdTableNextIndex?
cfm:Dot1afCfmIndexIntegerNextFree
|   +--rw dot1agCfmMdEntry* [dot1agCfmMdIndex]
|       +--rw dot1agCfmMdIndex                uint32
|       +--rw dot1agCfmMdFormat?
cfm:Dot1agCfmMaintDomainNameType
|       +--rw dot1agCfmMdName?
cfm:Dot1agCfmMaintDomainName
|       +--rw dot1agCfmMdMdLevel?                cfm:Dot1agCfmMDLevel
|       +--rw dot1agCfmMdMhfCreation?            cfm:Dot1agCfmMhfCreation
|       +--rw dot1agCfmMdMhfIdPermission?
cfm:Dot1agCfmIdPermission
|       +--ro dot1agCfmMdMaNextIndex?
cfm:Dot1afCfmIndexIntegerNextFree
|       +--rw dot1agCfmMdRowStatus?                cfm:RowStatus

```


2.7. CFM MA

The Maintenance Association contains the configuration related to the which flow the MA could monitor. For example, the VID of the flow and the CCM interval.

```

+--rw dot1agCfmMa
|
|  +--rw dot1agCfmMaNetEntry* [dot1agCfmMdIndex dot1agCfmMaIndex]
|
|   |  +--rw dot1agCfmMdIndex          ->
|   |  /dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
|
|   |  +--rw dot1agCfmMaIndex          uint32
|
|   |  +--rw dot1agCfmMaNetFormat?
|   |  cfm:Dot1agCfmMaintAssocNameType
|
|   |  +--rw dot1agCfmMaNetName?
|   |  cfm:Dot1agCfmMaintAssocName
|
|   |  +--rw dot1agCfmMaNetCcmInterval?  cfm:Dot1agCfmCcmInterval
|
|   |  +--rw dot1agCfmMaNetRowStatus?    cfm:RowStatus
|
|   +--rw dot1agCfmMaCompEntry* [dot1agCfmMaComponentId
|   dot1agCfmMdIndex dot1agCfmMaIndex]
|
|    |  +--rw dot1agCfmMdIndex          ->
|    |  /dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex
|
|    |  +--rw dot1agCfmMaIndex          ->
|    |  /dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex
|
|    |  +--rw dot1agCfmMaComponentId
|    |  cfm:Dot1agCfmPbbComponentIdentifier
|
|    |  +--rw dot1agCfmMaCompPrimaryVlanId?  cfm:VlanIdOrNone
|
|    |  +--rw dot1agCfmMaCompMhfCreation?
|    |  cfm:Dot1agCfmMhfCreation
|
|    |  +--rw dot1agCfmMaCompIdPermission?
|    |  cfm:Dot1agCfmIdPermission
|
|    |  +--ro dot1agCfmMaCompNumberOfVids?  uint32

```

```

    | | +--rw dot1agCfmMaCompRowStatus?      cfm:RowStatus

    | +--rw dot1agCfmMaMepListEntry* [dot1agCfmMdIndex
dot1agCfmMaIndex dot1agCfmMaMepListIdentifier]

    |     +--rw dot1agCfmMdIndex              ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex

    |     +--rw dot1agCfmMaIndex              ->
/dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex

    |     +--rw dot1agCfmMaMepListIdentifier  cfm:Dot1agCfmMepId

    |     +--rw dot1agCfmMaMepListRowStatus?  cfm:RowStatus

```

2.8. CFM MEP

The MEP container contains the configuration for a Maintenance point, for example, the direction, ID, VID and so on. It also contains the LTR related information and the database of MEP.

```

+--rw dot1agCfmMep

    +--rw dot1agCfmMepEntry* [dot1agCfmMdIndex dot1agCfmMaIndex
dot1agCfmMepIdentifier]

        | +--rw dot1agCfmMdIndex              ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex

        | +--rw dot1agCfmMaIndex              ->
/dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex

        | +--rw dot1agCfmMepIdentifier
cfm:Dot1agCfmMepId

        | +--rw dot1agCfmMepIfIndex?          cfm:if-
index-or-zero

        | +--rw dot1agCfmMepDirection?
cfm:Dot1agCfmMpDirection

        | +--rw dot1agCfmMepPrimaryVid?      uint32

        | +--rw dot1agCfmMepActive?          boolean

        | +--ro dot1agCfmMepFngState?
cfm:Dot1agCfmFngState

```

```

    | +---rw dot1agCfmMepCciEnabled?                boolean
    | +---rw dot1agCfmMepCcmLtmPriority?             uint32
    | +---ro dot1agCfmMepMacAddress?                yang:mac-
address
    | +---rw dot1agCfmMepLowPrDef?                  cfm:Dot1agCfmLowestAlarmPri
    | +---rw dot1agCfmMepFngAlarmTime?              cfm:TimeInterval
    | +---rw dot1agCfmMepFngResetTime?              cfm:TimeInterval
    | +---ro dot1agCfmMepHighestPrDefect?            cfm:Dot1agCfmHighestDefectPri
    | +---ro dot1agCfmMepDefects?                   cfm:Dot1agCfmMepDefects
    | +---ro dot1agCfmMepErrorCcmLastFailure?        binary
    | +---ro dot1agCfmMepXconCcmLastFailure?         binary
    | +---ro dot1agCfmMepCcmSequenceErrors?          yang:counter32
    | +---ro dot1agCfmMepCciSentCcms?                yang:counter32
    | +---ro dot1agCfmMepNextLbmTransId?              uint32
    | +---ro dot1agCfmMepLbrIn?                      yang:counter32
    | +---ro dot1agCfmMepLbrInOutOfOrder?            yang:counter32
    | +---ro dot1agCfmMepLbrBadMsdu?                 yang:counter32
    | +---ro dot1agCfmMepLtmNextSeqNumber?           uint32
    | +---ro dot1agCfmMepUnexpLtrIn?                 yang:counter32

```

```

    | +---ro dot1agCfmMepLbrOut?
yang:counter32

    | +---rw dot1agCfmMepTransmitLbmStatus?          boolean

    | +---rw dot1agCfmMepTransmitLbmDestMacAddress?  yang:mac-
address

    | +---rw dot1agCfmMepTransmitLbmDestMepId?
cfm:Dot1agCfmMepIdOrZero

    | +---rw dot1agCfmMepTransmitLbmDestIsMepId?      boolean

    | +---rw dot1agCfmMepTransmitLbmMessages?         int32

    | +---rw dot1agCfmMepTransmitLbmDataTlv?          binary

    | +---rw dot1agCfmMepTransmitLbmVlanPriority?      int32

    | +---rw dot1agCfmMepTransmitLbmVlanDropEnable?   boolean

    | +---ro dot1agCfmMepTransmitLbmResultOK?         boolean

    | +---ro dot1agCfmMepTransmitLbmSeqNumber?        uint32

    | +---ro dot1agCfmMepTransmitLtmStatus?           boolean

    | +---rw dot1agCfmMepTransmitLtmFlags?            bits

    | +---rw dot1agCfmMepTransmitLtmTargetMacAddress? yang:mac-
address

    | +---rw dot1agCfmMepTransmitLtmTargetMepId?
cfm:Dot1agCfmMepIdOrZero

    | +---rw dot1agCfmMepTransmitLtmTargetIsMepId?    boolean

    | +---rw dot1agCfmMepTransmitLtmTtl?              uint32

    | +---ro dot1agCfmMepTransmitLtmResult?           boolean

    | +---ro dot1agCfmMepTransmitLtmSeqNumber?        uint32

    | +---rw dot1agCfmMepTransmitLtmEgressIdentifier? binary

    | +---rw dot1agCfmMepRowStatus?
cfm:RowStatus

```

```

    +--rw dot1agCfmLtrEntry* [dot1agCfmMdIndex dot1agCfmMaIndex
dot1agCfmMepIdentifier dot1agCfmLtrSeqNumber
dot1agCfmLtrReceiveOrder]

      | +--rw dot1agCfmMdIndex ->
      /dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex

      | +--rw dot1agCfmMaIndex ->
      /dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex

      | +--rw dot1agCfmMepIdentifier ->
      /dot1agCfmMep/dot1agCfmMepEntry/dot1agCfmMepIdentifier

      | +--rw dot1agCfmLtrSeqNumber uint32

      | +--rw dot1agCfmLtrReceiveOrder uint32

      | +--ro dot1agCfmLtrTtl? uint32

      | +--ro dot1agCfmLtrForwarded? boolean

      | +--ro dot1agCfmLtrTerminalMep? boolean

      | +--ro dot1agCfmLtrLastEgressIdentifier? binary

      | +--ro dot1agCfmLtrNextEgressIdentifier? binary

      | +--ro dot1agCfmLtrRelay?
cfm:Dot1agCfmRelayActionFieldValue

      | +--ro dot1agCfmLtrChassisIdSubtype?
cfm:LldpChassisIdSubtype

      | +--ro dot1agCfmLtrChassisId?
cfm:LldpChassisId

      | +--ro dot1agCfmLtrManAddressDomain? cfm:TDomain

      | +--ro dot1agCfmLtrManAddress? cfm:TAddress

      | +--ro dot1agCfmLtrIngress?
cfm:Dot1agCfmIngressActionFieldValue

      | +--ro dot1agCfmLtrIngressMac? yang:mac-
address

```

```

    | +--ro dot1agCfmLtrIngressPortIdSubtype?
cfm:LldpPortIdSubtype

    | +--ro dot1agCfmLtrIngressPortId?                cfm:LldpPortId

    | +--ro dot1agCfmLtrEgress?
cfm:Dot1agCfmEgressActionFieldValue

    | +--ro dot1agCfmLtrEgressMac?                    yang:mac-
address

    | +--ro dot1agCfmLtrEgressPortIdSubtype?
cfm:LldpPortIdSubtype

    | +--ro dot1agCfmLtrEgressPortId?                cfm:LldpPortId

    | +--ro dot1agCfmLtrOrganizationSpecificTlv?    binary

    +--rw dot1agCfmMepDbEntry* [dot1agCfmMdIndex dot1agCfmMaIndex
dot1agCfmMepIdentifier dot1agCfmMepDbRMepIdentifier]

        +--rw dot1agCfmMdIndex                        ->
/dot1agCfmMd/dot1agCfmMdEntry/dot1agCfmMdIndex

        +--rw dot1agCfmMaIndex                        ->
/dot1agCfmMa/dot1agCfmMaNetEntry/dot1agCfmMaIndex

        +--rw dot1agCfmMepIdentifier                  ->
/dot1agCfmMep/dot1agCfmMepEntry/dot1agCfmMepIdentifier

        +--rw dot1agCfmMepDbRMepIdentifier            cfm:Dot1agCfmMepId

        +--ro dot1agCfmMepDbRMepState?
cfm:Dot1agCfmRemoteMepState

        +--ro dot1agCfmMepDbRMepFailedOkTime?        yang:timestamp

        +--ro dot1agCfmMepDbMacAddress?               yang:mac-address

        +--ro dot1agCfmMepDbRdi?                     boolean

        +--ro dot1agCfmMepDbPortStatusTlv?
cfm:Dot1agCfmPortStatus

        +--ro dot1agCfmMepDbInterfaceStatusTlv?
cfm:Dot1agCfmInterfaceStatus

```

```

    +---ro dot1agCfmMepDbChassisIdSubtype?
cfm:LldpChassisIdSubtype

    +---ro dot1agCfmMepDbChassisId?           cfm:LldpChassisId

    +---ro dot1agCfmMepDbManAddressDomain?    cfm:TDomain

    +---ro dot1agCfmMepDbManAddress?          cfm:TAddress

```

2.9. CFM FAULT ALARM NOTIFICATION

When there is fault alarm generated or cleared by the MEP fault notification generator state machine, the notification is sent to the management entity.

```
notification dot1agCfmFaultAlarm {
```

```
  description
```

```
    "A MEP has a persistent defect condition. A notification
    (fault alarm) is sent to the management entity with the OID
    of the MEP that has detected the fault.
```

Whenever a MEP has a persistent defect,
it may or may not generate a Fault Alarm to warn the system administrator of the problem, as controlled by the MEP Fault Notification Generator State Machine and associated Managed Objects. Only the highest-priority defect, as shown in Table 20-1, is reported in the Fault Alarm.

If a defect with a higher priority is raised after a Fault Alarm has been issued, another Fault Alarm is issued.

The management entity receiving the notification can identify the system from the network source address of the notification, and can identify the MEP reporting the defect by the indices in the OID of the dot1agCfmMepHighestPrDefect variable in the notification:

dot1agCfmMdIndex - Also the index of the MEP's Maintenance Domain table entry (dot1agCfmMdTable).

dot1agCfmMaIndex - Also an index (with the MD table index) of the MEP's Maintenance Association network table entry (dot1agCfmMaNetTable), and (with the MD table index and component ID) of the MEP's MA component table entry (dot1agCfmMaCompTable).

dot1agCfmMepIdentifier - MEP Identifier and final index into the MEP table (dot1agCfmMepTable).";

reference

"802.1ag clause 12.14.7.7";

container dot1agCfmFaultAlarm-dot1agCfmMepHighestPrDefect {
description

"The highest priority defect that has been present since the MEPS Fault Notification Generator State Machine was last in the FNG_RESET state";

```
leaf dot1agCfmMdIndex {  
    type leafref {  
        path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"  
            +":dot1agCfmMdIndex";  
    }  
    description  
        "Automagically generated leafref leaf.";  
}  
  
leaf dot1agCfmMaIndex {  
    type leafref {  
        path "/cfm:dot1agCfmMa/cfm:dot1agCfmMaNetEntry/cfm"  
            +":dot1agCfmMaIndex";  
    }  
    description  
        "Automagically generated leafref leaf.";  
}  
  
leaf dot1agCfmMepIdentifier {  
    type leafref {  
        path "/cfm:dot1agCfmMep/cfm:dot1agCfmMepEntry/cfm"  
            +":dot1agCfmMepIdentifier";
```

```
    }

    description
        "Automagically generated leafref leaf.";
    }

    leaf dot1agCfmMepHighestPrDefect {
        type cfm:Dot1agCfmHighestDefectPri;
        description
            "The highest priority defect that has been present
            since the MEPs Fault Notification Generator State
            Machine was last in the FNG_RESET state.";
        reference
            "802.1ag clause 12.14.7.1.3:n 20.33.9 and Table 21-1";
    }
}

}
```

3. CFM YANG Module

```
<CODE BEGINS> file ietf-cfm@2017-03-20.yang
module ietf-cfm {

    /*** NAMESPACE / PREFIX DEFINITION ***/

    namespace "urn:ietf:params:xml:ns:yang:ietf-cfm";
    prefix "cfm";

    /*** LINKAGE (IMPORTS / INCLUDES) ***/

    import ietf-interfaces { prefix "if"; }
    import ietf-yang-smiv2 { prefix "smi"; }
    import ietf-yang-types { prefix "yang"; }
```

```
/** META INFORMATION **/
```

```
organization
```

```
"IETF NETMOD (NETCONF Data Modeling Language) Working Group";
```

```
contact
```

```
"WG Web: <http://tools.ietf.org/wg/netmod/>
```

```
WG List: <mailto:netmod@ietf.org>
```

```
WG Chair: Thomas Nadeau
```

```
<mailto:tnadeau@lucidvision.com>
```

```
WG Chair: Juergen Schoenwaelder
```

```
<mailto:j.schoenwaelder@jacobs-university.de>
```

```
Editor: Kun Wang
```

```
<kun.s.wang@ericsson.com>
```

```
Alex Wang
```

```
<alex.g.wang@ericsson.com>
```

```
Chin Chen
```

```
<chin.chen@ericsson.com>
```

```
Hua Lv
```

```
<hua.lv@ericsson.com>";
```

```
description
```

```
"This module contains a collection of YANG definitions for  
Connectivity Fault Management.
```

```
Copyright (c) 2017 IETF Trust and the persons identified as  
authors of the code. All rights reserved.
```

```
Redistribution and use in source and binary forms, with or  
without modification, is permitted pursuant to, and subject  
to the license terms contained in, the Simplified BSD License  
set forth in Section 4.c of the IETF Trust's Legal Provisions  
Relating to IETF Documents
```

```
(http://trustee.ietf.org/license-info).";
```

```
revision "2017-03-29" {
```

```
description
```

```
"01 revision.";
```

```
reference
```

```
"RFC rfc6020";
```

```
}
```

```
/** TYPE DEFINITIONS **/
```



```
typedef if-index {
  type leafref {
    path "/if:interfaces-state/if:interface/if:if-index";
  }
  description
    "This type is used by data models that need to reference
    configured interfaces.";
}

typedef if-index-or-zero {
  type int32 {
    range "0..2147483647";
  }
  description
    "This textual convention 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 managed system. This extension permits the
    additional value of zero. the value zero is object-specific
    and must therefore be defined as part of the description of
    any object which uses this syntax. Examples of the usage of
    zero might include situations where interface was unknown,
    or when none or all interfaces need to be referenced.";
}

typedef VlanId {
  type int32 {
    range "1..4094";
  }
  description
    "The VLAN-ID that uniquely identifies a VLAN. This
    is the 12-bit VLAN-ID used in the VLAN Tag header.
    The range is defined by the REFERENCED specification.";
}

typedef VlanIdOrNone {
  type int32 {
    range "0..4094";
  }
  description
    "The VLAN-ID that uniquely identifies a specific VLAN,
    or no VLAN. The special value of zero is used to
    indicate that no VLAN-ID is present or used. This can
    be used in any situation where an object or a table entry
    must refer either to a specific VLAN, or to no VLAN.

    Note that a MIB object that is defined using this
```


TEXTUAL-CONVENTION should clarify the meaning of
'no VLAN' (i.e., the special value 0).";
}

```
typedef LldpChassisIdSubtype {  
  type enumeration {  
    enum chassisComponent      {  
      value 1;  
      description  
      "If the associated LldpChassisIdSubtype object has a  
      value of 'chassisComponent(1)', then the octet string  
      identifies a particular instance of the  
      entPhysicalAlias object (defined in IETF RFC 2737)  
      for a chassis component (i.e. an entPhysicalClass  
      value of 'chassis(3)').";  
    }  
    enum interfaceAlias        {  
      value 2;  
      description  
      "If the associated LldpChassisIdSubtype object has a  
      value of 'interfaceAlias(2)', then the octet string  
      identifies a particular instance of the ifAlias object  
      (defined in IETF RFC 2863) for an interface on the  
      containing chassis. If the particular ifAlias object  
      does not contain any values, another chassis  
      identifier type should be used.";  
    }  
    enum portComponent          {  
      value 3;  
      description  
      "If the associated LldpChassisIdSubtype object has a  
      value of 'portComponent(3)', then the octet string  
      identifies a particular instance of the  
      entPhysicalAlias object (defined in IETF RFC 2737)  
      for a port or backplane component within the containing  
      chassis.";  
    }  
    enum macAddress              {  
      value 4;  
      description  
      "If the associated LldpChassisIdSubtype object has a  
      value of 'macAddress(4)', then this string identifies  
      a particular unicast source address (encoded in network  
      byte order and IEEE 802.3 canonical bit order), of a  
      port on the containing chassis as defined in IEEE Std  
      802-2001.";  
    }  
  }  
}
```



```
enum networkAddress      {
    value 5;
    description
    "If the associated LldpChassisIdSubtype object has a
    value of 'networkAddress(5)', then this string
    identifies a particular network address, encoded in
    network byte order, associated with one or more ports
    on the containing chassis. The first octet contains
    the IANA Address Family Numbers enumeration value for
    the specific address type, and octets 2 through N
    contain the network address value in network byte
    order.";
}
enum interfaceName       {
    value 6;
    description
    "If the associated LldpChassisIdSubtype object has a
    value of 'interfaceName(6)', then the octet string
    identifies a particular instance of the ifName object
    (defined in IETF RFC 2863) for an interface on the
    containing chassis. If the particular ifName object
    does not contain any values, another chassis identifier
    type should be used.";
}
enum local                {
    value 7;
    description
    "If the associated LldpChassisIdSubtype object has a
    value of 'local(7)', then this string identifies a
    locally assigned Chassis ID.";
}
}
description
"This describes the format of a chassis identifier string.
Objects of this type are always used with an associated
LldpChassisIdSubtype object, which identifies the format of
the particular LldpChassisId object instance.";
}

typedef LldpChassisId {
    type int32 {
        range "1..255";
    }
    description
    "This describes the format of a chassis identifier string.
    Objects of this type are always used with an associated
    LldpChassisIdSubtype object, which identifies the format of
```


the particular LldpChassisId object instance.

If the associated LldpChassisIdSubtype object has a value of 'chassisComponent(1)', then the octet string identifies a particular instance of the entPhysicalAlias object (defined in IETF [RFC 2737](#)) for a chassis component (i.e., an entPhysicalClass value of 'chassis(3)').

If the associated LldpChassisIdSubtype object has a value of 'interfaceAlias(2)', then the octet string identifies a particular instance of the ifAlias object (defined in IETF [RFC 2863](#)) for an interface on the containing chassis. If the particular ifAlias object does not contain any values, another chassis identifier type should be used.

If the associated LldpChassisIdSubtype object has a value of 'portComponent(3)', then the octet string identifies a particular instance of the entPhysicalAlias object (defined in IETF [RFC 2737](#)) for a port or backplane component within the containing chassis.

If the associated LldpChassisIdSubtype object has a value of 'macAddress(4)', then this string identifies a particular unicast source address (encoded in network byte order and IEEE 802.3 canonical bit order), of a port on the containing chassis as defined in IEEE Std 802-2001.

If the associated LldpChassisIdSubtype object has a value of 'networkAddress(5)', then this string identifies a particular network address, encoded in network byte order, associated with one or more ports on the containing chassis. The first octet contains the IANA Address Family Numbers enumeration value for the specific address type, and octets 2 through N contain the network address value in network byte order.

If the associated LldpChassisIdSubtype object has a value of 'interfaceName(6)', then the octet string identifies a particular instance of the ifName object (defined in IETF [RFC 2863](#)) for an interface on the containing chassis. If the particular ifName object does not contain any values, another chassis identifier type should be used.

If the associated LldpChassisIdSubtype object has a value of 'local(7)', then this string identifies a locally assigned Chassis ID.";

}


```
typedef LldpPortIdSubtype {
  type enumeration {
    enum interfaceAlias      {
      value 1;
      description
      "a port identifier based on the ifAlias MIB object,
      defined in IETF RFC 2863.";
    }
    enum portComponent      {
      value 2;
      description
      "a port identifier based on the value of
      entPhysicalAlias (defined in IETF RFC 2737) for a port
      component (i.e., entPhysicalClass value of 'port(10)'),
      within the containing chassis.";
    }
    enum macAddress          {
      value 3;
      description
      "a port identifier based on a unicast source address
      (encoded in network byte order and IEEE 802.3 canonical
      bit order), which has been detected by the agent and
      associated with a particular port
      (IEEE Std 802-2001).";
    }
    enum networkAddress      {
      value 4;
      description
      "a port identifier based on a network address, detected
      by the agent and associated with a particular port.";
    }
    enum interfaceName       {
      value 5;
      description
      "a port identifier based on the ifName MIB object,
      defined in IETF RFC 2863.";
    }
    enum agentCircuitId      {
      value 6;
      description
      "a port identifier based on the agent-local identifier
      of the circuit (defined in RFC 3046), detected by the
      agent and associated with a particular port.";
    }
    enum local               {
      value 7;
      description
```



```
        "a port identifier based on a value locally assigned.";
    }
}
description
    "This describes the source of a particular type of port
    identifier used in the LLDP MIB.";
}
```

```
typedef LldpPortId {
    type int32 {
        range "1..255";
    }
    description
        "This describes the format of a port identifier string.
        Objects of this type are always used with an associated
        LldpPortIdSubtype object, which identifies the format of the
        particular LldpPortId object instance.
```

If the associated LldpPortIdSubtype object has a value of 'interfaceAlias(1)', then the octet string identifies a particular instance of the ifAlias object (defined in IETF [RFC 2863](#)). If the particular ifAlias object does not contain any values, another port identifier type should be used.

If the associated LldpPortIdSubtype object has a value of 'portComponent(2)', then the octet string identifies a particular instance of the entPhysicalAlias object (defined in IETF [RFC 2737](#)) for a port or backplane component.

If the associated LldpPortIdSubtype object has a value of 'macAddress(3)', then this string identifies a particular unicast source address (encoded in network byte order and IEEE 802.3 canonical bit order) associated with the port (IEEE Std 802-2001).

If the associated LldpPortIdSubtype object has a value of 'networkAddress(4)', then this string identifies a network address associated with the port. The first octet contains the IANA AddressFamilyNumbers enumeration value for the specific address type, and octets 2 through N contain the networkAddress address value in network byte order.

If the associated LldpPortIdSubtype object has a value of 'interfaceName(5)', then the octet string identifies a particular instance of the ifName object (defined in IETF [RFC 2863](#)). If the particular ifName object does not contain any values, another port identifier type should be used.

If the associated LldpPortIdSubtype object has a value of 'agentCircuitId(6)', then this string identifies a agent-local identifier of the circuit (defined in [RFC 3046](#)).

If the associated LldpPortIdSubtype object has a value of 'local(7)', then this string identifies a locally assigned port ID.";

}

```
typedef RowStatus {  
  type enumeration {  
    enum active {  
      value 1;  
      description  
        "The conceptual row is available for use by the managed  
        device."; }  
    enum notInService {  
      value 2;  
      description  
        "The conceptual row exists in the agent, but is  
        unavailable for use by the managed device;  
        'notInService' has no implication regarding the  
        internal consistency of the row, availability of  
        resources, or consistency with the current state of  
        the managed device."; }  
    enum notReady {  
      value 3;  
      description  
        "The conceptual row exists in the agent, but is missing  
        information necessary in order to be available for use  
        by the managed device (i.e., one or more required  
        columns in the conceptual row have not been  
        instantiated."; }  
    enum createAndGo {  
      value 4;  
      description  
        "It is supplied by a management station wishing to create  
        a new instance of a conceptual row and to have its  
        status automatically set to active, making it available  
        for use by the managed device."; }  
    enum createAndWait {  
      value 5;  
      description  
        "It is supplied by a management station wishing to create  
        a new instance of a conceptual row (but not make it  
        available for use by the managed device."; }
```



```
enum destroy      {
    value 6;
    description
        "It is supplied by a management station wishing to
        delete all of the instances associated with an
        existing conceptual row."; }
}
description
    "The RowStatus textual convention is used to manage the
    creation and deletion of conceptual rows, and is used as the
    value of the SYNTAX clause for the status column of a
    conceptual row.
    The status column has six defined values:

    Whereas five of the six values (all except `notReady') may
    be specified in a management protocol set operation, only
    three values will be returned in response to a management
    protocol retrieval operation: `notReady', `notInService' or
    `active'. That is, when queried, an existing conceptual row
    has only three states: it is either available for use by
    the managed device (the status column has value `active');
    it is not available for use by the managed device, though
    the agent has sufficient information to attempt to make it
    so (the status column has value `notInService'); or, it is
    not available for use by the managed device, and an attempt
    to make it so would fail because the agent has insufficient
    information (the state column has value `notReady').";
}

typedef TimeInterval {
    type int32 {
        range "0..2147483647";
    }
    description
        "A period of time, measured in units of 0.01 seconds.";
}

typedef TDomain {
    type yang:object-identifier;
    description
        "Denotes a kind of transport service.

        Some possible values, such as snmpUDPDomain, are defined in
        the SNMPv2-TM MIB module. Other possible values are defined
        in other MIB modules.";
    reference
        "The SNMPv2-TM MIB module is defined in RFC 1906.";
}
```



```
}

typedef TAddress {
  type binary {
    length "1..255";
  }
  description
    "Denotes a transport service address.

    A TAddress value is always interpreted within the context
    of a TDomain value. Thus, each definition of a TDomain
    value must be accompanied by a definition of a textual
    convention for use with that TDomain.";
}

typedef Dot1agCfmMaintDomainNameType {
  type enumeration {
    enum none {
      value 1;
      description
        "No format specified, usually because there is not (yet)
        a Maintenance Domain Name. In this case, a zero length
        OCTET STRING for the Domain Name field is acceptable.";
    }
    enum dnsLikeName {
      value 2;
      description
        "Domain Name like string, globally unique text string
        derived from a DNS name.";
    }
    enum macAddressAndUint {
      value 3;
      description
        "MAC address + 2-octet (unsigned) integer.";
    }
    enum charString {
      value 4;
      description
        "RFC2579 DisplayString, except that the character codes
        0-31 (decimal) are not used.";
    }
  }
  description
    "A value that represents a type (and thereby the format)
    of a Dot1agCfmMaintDomainName.

    To support future extensions, the
```


Dot1agCfmMaintDomainNameType textual convention SHOULD NOT be sub-typed in object type definitions. It MAY be sub-typed in compliance statements in order to require only a subset of these address types for a compliant implementation.

Implementations must ensure that Dot1agCfmMaintDomainNameType objects and any dependent objects (e.g., Dot1agCfmMaintDomainName objects) are consistent. An inconsistentValue error must be generated if an attempt to change an Dot1agCfmMaintDomainNameType object would, for example, lead to an undefined Dot1agCfmMaintDomainName value. In particular, Dot1agCfmMaintDomainNameType/Dot1agCfmMaintDomainName pairs must be changed together if the nameType changes.";

reference

"802.1ag clause 21.6.5, Table 21-19";

}

typedef Dot1agCfmMaintDomainName {

 type binary {
 length "1..43";
 }
}

description

"Denotes a generic Maintenance Domain Name.

A Dot1agCfmMaintDomainName value is interpreted within the context of a Dot1agCfmMaintDomainNameType value. Every usage of the Dot1agCfmMaintDomainName textual convention is required to specify the Dot1agCfmMaintDomainNameType object that provides the context. It is suggested that the Dot1agCfmMaintDomainNameType object be logically registered before the object(s) that use the Dot1agCfmMaintDomainName textual convention, if they appear in the same logical row.

The value of a Dot1agCfmMaintDomainName object must always be consistent with the value of the associated Dot1agCfmMaintDomainNameType object. Attempts to set an Dot1agCfmMaintDomainName object to a value inconsistent with the associated Dot1agCfmMaintDomainNameType must fail with an inconsistentValue error.

When this textual convention is used as the syntax of an index object, there may be issues with the limit of 128 sub-identifiers specified in SMIV2, IETF STD 58. In this case, the object definition MUST include a 'SIZE' clause

to limit the number of potential instance sub-identifiers; otherwise the applicable constraints MUST be stated in the appropriate conceptual row DESCRIPTION clauses, or in the surrounding documentation if there is no single DESCRIPTION clause that is appropriate.

A value of none(1) in the associated Dot1agCfmMaintDomainNameType object means that no Maintenance Domain name is present, and the contents of the Dot1agCfmMaintDomainName object are meaningless.

See the DESCRIPTION of the Dot1agCfmMaintAssocNameType TEXTUAL-CONVENTION for a discussion of the length limits on the Maintenance Domain name and Maintenance Association name.";

reference

"802.1ag clause 21.6.5";

}

typedef Dot1agCfmMaintAssocNameType {

 type enumeration {

 enum primaryVid {
 value 1;
 description
 "Primary VLAN ID";
 }

 enum charString {
 value 2;
 description
 "display string";
 }

 enum unsignedInt16 {
 value 3;
 description
 "2-octet integer/big endian";
 }

 enum rfc2865VpnId {
 value 4;
 description
 "[RFC 2685](#) VPN ID";
 }

 }

}

description

"A value that represents a type (and thereby the format) of a Dot1agCfmMaintAssocName. The value can be one of the following:

ieeeReserved(0) Reserved for definition by IEEE 802.1
recommend to not use zero unless
absolutely needed.

primaryVid(1) Primary VLAN ID.
12 bits represented in a 2-octet integer:
- 4 least significant bits of the first
byte contains the 4 most significant
bits of the 12 bits primary VID
- second byte contains the 8 least
significant bits of the primary VID

```

0 1 2 3 4 5 6 7 8
+---+---+---+---+
|0 0 0 0| (MSB) |
+---+---+---+---+
|  VID   LSB   |
+---+---+---+---+

```

charString(2) [RFC2579](#) DisplayString, except that the
character codes 0-31 (decimal) are not
used. (1..45) octets

unsignedInt16 (3) 2-octet integer/big endian

rfc2865VpnId(4) [RFC 2685](#) VPN ID
3 octet VPN authority Organizationally
Unique Identifier followed by 4 octet VPN
index identifying VPN according to the
OUI:

```

0 1 2 3 4 5 6 7 8
+---+---+---+---+
| VPN OUI (MSB) |
+---+---+---+---+
|   VPN OUI    |
+---+---+---+---+
| VPN OUI (LSB) |
+---+---+---+---+
|VPN Index (MSB)|
+---+---+---+---+
|  VPN Index   |
+---+---+---+---+
|  VPN Index   |
+---+---+---+---+
|VPN Index (LSB)|
+---+---+---+---+

```

ieeeReserved(xx) Reserved for definition by IEEE 802.1
xx values can be [5..31] and [64..255]

ituReserved(xx) Reserved for definition by ITU-T Y.1731
 xx values range from [32..63]

To support future extensions, the
Dot1agCfmMaintAssocNameType textual convention SHOULD NOT
be sub-typed in object type definitions. It MAY be
sub-typed in compliance statements in order to require
only a subset of these address types for a compliant
implementation.

Implementations must ensure that Dot1agCfmMaintAssocNameType
objects and any dependent objects (e.g.,
Dot1agCfmMaintAssocName objects) are consistent. An
inconsistentValue error must be generated if an attempt to
change an Dot1agCfmMaintAssocNameType object would, for
example, lead to an undefined Dot1agCfmMaintAssocName value.
In particular,
Dot1agCfmMaintAssocNameType/Dot1agCfmMaintAssocName pairs
must be changed together if the nameType changes.

The Maintenance Domain name and Maintenance Association
name, when put together into the CCM PDU, MUST total 48
octets or less. If the Dot1agCfmMaintDomainNameType
object contains none(1), then the Dot1agCfmMaintAssocName
object MUST be 45 octets or less in length. Otherwise,
the length of the Dot1agCfmMaintDomainName object plus the
length of the Dot1agCfmMaintAssocName object, added
together, MUST total less than or equal to 44 octets.";

reference

"802.1ag clause 21.6.5.4, Table 21-20";

}

typedef Dot1agCfmMaintAssocName {

 type binary {
 length "1..45";

 }

 description

 "Denotes a generic Maintenance Association Name. It is the
 part of the Maintenance Association Identifier which is
 unique within the Maintenance Domain Name and is appended
 to the Maintenance Domain Name to form the Maintenance
 Association Identifier (MAID)."

A Dot1agCfmMaintAssocName value is always interpreted within
the context of a Dot1agCfmMaintAssocNameType value. Every
usage of the Dot1agCfmMaintAssocName textual convention is
required to specify the Dot1agCfmMaintAssocNameType object

that provides the context. It is suggested that the Dot1agCfmMaintAssocNameType object be logically registered before the object(s) that use the Dot1agCfmMaintAssocName textual convention, if they appear in the same logical row.

The value of a Dot1agCfmMaintAssocName object must always be consistent with the value of the associated Dot1agCfmMaintAssocNameType object. Attempts to set an Dot1agCfmMaintAssocName object to a value inconsistent with the associated Dot1agCfmMaintAssocNameType must fail with an inconsistentValue error.

When this textual convention is used as the syntax of an index object, there may be issues with the limit of 128 sub-identifiers specified in SMIV2, IETF STD 58. In this case, the object definition MUST include a 'SIZE' clause to limit the number of potential instance sub-identifiers; otherwise the applicable constraints MUST be stated in the appropriate conceptual row DESCRIPTION clauses, or in the surrounding documentation if there is no single DESCRIPTION clause that is appropriate.";

reference

"802.1ag clauses 21.6.5.4, 21.6.5.5, 21.6.5.6";

}

typedef Dot1agCfmMDLevel {

type int32 {

range "0..7";

}

description

"Integer identifying the Maintenance Domain Level (MD Level). Higher numbers correspond to higher Maintenance Domains, those with the greatest physical reach, with the highest values for customers' CFM PDUs. Lower numbers correspond to lower Maintenance Domains, those with more limited physical reach, with the lowest values for CFM PDUs protecting single bridges or physical links.";

reference

"802.1ag clauses 18.3, 21.4.1";

smi:display-hint "d";

}

typedef Dot1agCfmMDLevelOrNone {

type int32 {

range "-1..7";

}

description


```
"Integer identifying the Maintenance Domain Level (MD Level).
Higher numbers correspond to higher Maintenance Domains,
those with the greatest physical reach, with the highest
values for customers' CFM packets. Lower numbers correspond
to lower Maintenance Domains, those with more limited
physical reach, with the lowest values for CFM PDUs
protecting single bridges or physical links.

The value (-1) is reserved to indicate that no MA Level has
been assigned.";
reference
  "802.1ag clauses 18.3, 12.14.3.1.3:c";
smi:display-hint "d";
}

typedef Dot1agCfmMpDirection {
  type enumeration {
    enum down {
      value 1;
      description
        "Sends Continuity Check Messages away from the MAC Relay
        Entity.";
    }
    enum up {
      value 2;
      description
        "Sends Continuity Check Messages towards the MAC Relay
        Entity.";
    }
  }
  description
    "Indicates the direction in which the Maintenance
    association (MEP or MIP) faces on the bridge port.";

  reference
    "802.1ag clauses 12.14.6.3.2:c";
}

typedef Dot1agCfmPortStatus {
  type enumeration {
    enum psNoPortStateTLV {
      value 0;
      description
        "Indicates either that no CCM has been received or
        that no port status TLV was present in the last
        CCM received.";
    }
    enum psBlocked {
```



```
        value 1;
        description
        "Ordinary data cannot pass freely through the port
         on which the remote MEP resides. Value of
         enableRmepDefect is equal to false.";
    }
    enum psUp {
        value 2;
        description
        "Ordinary data can pass freely through the port on which
         the remote MEP resides. Value of enableRmepDefect is
         equal to true.";
    }
}
description
    "An enumerated value from the Port Status TLV from the last
    CCM received from the last MEP. It indicates the ability
    of the Bridge Port on which the transmitting MEP resides
    to pass ordinary data, regardless of the status of the MAC
    (Table 21-10).

    NOTE: A 0 value is used for psNoPortStateTLV, so that
    additional code points can be added in a manner
    consistent with the Dot1agCfmInterfaceStatus textual
    convention.";
reference
    "802.1ag clause 12.14.7.6.3:f, 20.19.3 and 21.5.4";
}

typedef Dot1agCfmInterfaceStatus {
    type enumeration {
        enum isNoInterfaceStatusTLV {
            value 0;
            description
            "Indicates either that no CCM has been received or
             that no interface status TLV was present in the last
             CCM received.";
        }
        enum isUp {
            value 1;
            description
            "The interface is ready to pass packets.";
        }
        enum isDown {
            value 2;
            description
            "The interface cannot pass packets";
        }
    }
}
```



```
    }
    enum isTesting          {
        value 3;
        description
        "The interface is in some test mode.";
    }
    enum isUnknown          {
        value 4;
        description
        "The interface status cannot be determined for some
        reason.";
    }
    enum isDormant          {
        value 5;
        description
        "The interface is not in a state to pass packets but
        is in a pending state, waiting for some external
        event.";
    }
    enum isNotPresent       {
        value 6;
        description
        "Some component of the interface is missing";
    }
    enum isLowerLayerDown   {
        value 7;
        description
        "The interface is down due to state of the lower
        layer interfaces";
    }
}
description
    "An enumerated value from the Interface Status TLV from the
    last CCM received from the last MEP. It indicates the status
    of the Interface within which the MEP transmitting the CCM
    is configured, or the next lower Interface in the Interface
    Stack, if the MEP is not configured within an Interface.

    NOTE: A 0 value is used for isNoInterfaceStatusTLV, so that
    these code points can be kept consistent with new code
    points added to ifOperStatus in the IF-MIB.";
reference
    "802.1ag clause 12.14.7.6.3:g, 20.19.4 and 21.5.5";
}

typedef Dot1agCfmHighestDefectPri {
    type enumeration {
```



```
enum none      {
    value 0;
    description
        "no defects since FNG_RESET";
}
enum defRDICCM {
    value 1;
    description
        "DefRDICCM";
}
enum defMACstatus {
    value 2;
    description
        "DefMACstatus";
}
enum defRemoteCCM {
    value 3;
    description
        "DefRemoteCCM";
}
enum defErrorCCM {
    value 4;
    description
        "DefErrorCCM";
}
enum defXconCCM {
    value 5;
    description
        "DefXconCCM";
}
}
description
    "An enumerated value, equal to the contents of the variable
    highestDefect (20.33.9 and Table 20-1), indicating the
    highest-priority defect that has been present since the MEP
    Fault Notification Generator State Machine was last in the
    FNG_RESET state, either:

    none(0)          no defects since FNG_RESET
    defRDICCM(1)     DefRDICCM
    defMACstatus(2)   DefMACstatus
    defRemoteCCM(3)   DefRemoteCCM
    defErrorCCM(4)    DefErrorCCM
    defXconCCM(5)     DefXconCCM

    The value 0 is used for no defects so that additional higher
    priority values can be added, if needed, at a later time,
```



```
        and so that these values correspond with those in
        Dot1agCfmLowestAlarmPri.";
reference
    "802.1ag clause 20.1.2, 12.14.7.7.2:c and 20.33.9";
}

typedef Dot1agCfmLowestAlarmPri {
    type enumeration {
        enum allDef {
            value 1;
            description
                "DefRDICCM, DefMACstatus, DefRemoteCCM, DefErrorCCM
                and DefXconCCM";
        }
        enum macRemErrXcon {
            value 2;
            description
                "Only DefMACstatus, DefRemoteCCM, DefErrorCCM, and
                DefXconCCM (default)";
        }
        enum remErrXcon {
            value 3;
            description
                "Only DefRemoteCCM, DefErrorCCM, and DefXconCCM";
        }
        enum errXcon {
            value 4;
            description
                "Only DefErrorCCM and DefXconCCM";
        }
        enum xcon {
            value 5;
            description
                "Only DefXconCCM";
        }
        enum noXcon {
            value 6;
            description
                "No defects DefXcon or lower are to be reported";
        }
    }
}
description
    "An integer value specifying the lowest priority defect
    that is allowed to generate a Fault Alarm (20.9.5)";
reference
    "802.1ag clause 12.14.7.1.3:k and 20.9.5";
}
```



```
typedef Dot1agCfmMepId {
  type uint32 {
    range "1..8191";
  }
  description
    "Maintenance association End Point Identifier (MEPID):
     A small integer, unique over a given Maintenance
     Association, identifying a specific MEP.";
  reference
    "802.1ag clauses 3.18 and 19.2.1";
  smi:display-hint "d";
}

typedef Dot1agCfmMepIdOrZero {
  type uint32 {
    range "0..8191";
  }
  description
    "Maintenance association End Point Identifier (MEPID):
     A small integer, unique over a given Maintenance
     Association, identifying a specific MEP.

     The special value 0 is allowed to indicate special cases,
     for example that no MEPID is configured.

     Whenever an object is defined with this SYNTAX, then the
     DESCRIPTION clause of such an object MUST specify what the
     special value of 0 means.";
  reference
    "802.1ag clause 19.2.1";
  smi:display-hint "d";
}

typedef Dot1agCfmMhfCreation {
  type enumeration {
    enum defMHFnone {
      value 1;
      description
        "No MHFs can be created for this VID.";
    }
    enum defMHFdefault {
      value 2;
      description
        "MHFs can be created on this VID on any Bridge port
         through which this VID can pass.";
    }
  }
}
```



```
enum defMHFexplicit {
    value 3;
    description
        "MHFs can be created for this VID only on Bridge ports
        through which this VID can pass, and only if a MEP is
        created at some lower MD Level.";
}
enum defMHFdefer {
    value 4;
    description
        "The creation of MHFs is determined by the corresponding
        Maintenance Domain variable
        (dot1agCfmMaCompMhfcCreation).";
}
}
description
    "Indicates if the Management Entity can create MHFs.";
reference
    "802.1ag clause 12.14.5.1.3:c and 22.2.3";
}

typedef Dot1agCfmIdPermission {
    type enumeration {
        enum sendIdNone {
            value 1;
            description
                "The Sender ID TLV is not to be sent.";
        }
        enum sendIdChassis {
            value 2;
            description
                "The Chassis ID Length, Chassis ID Subtype, and Chassis
                ID fields of the Sender ID TLV are to be sent.";
        }
        enum sendIdManage {
            value 3;
            description
                "The Management Address Length and Management Address
                of the Sender ID TLV are to be sent.";
        }
        enum sendIdChassisManage {
            value 4;
            description
                "The Chassis ID Length, Chassis ID Subtype, Chassis ID,
                Management Address Length and Management Address
                fields are all to be sent.";
        }
    }
}
```



```
    enum sendIdDefer      {
        value 5;
        description
            "The contents of the Sender ID TLV are determined by
             the corresponding Maintenance Domain variable
             (dot1agCfmMaCompIdPermission).";
    }
}
description
    "Indicates what, if anything, is to be included in the Sender
     ID TLV transmitted in CCMs, LBMs, LTMs, and LTRs.";
reference
    "802.1ag clause 12.14.6.1.3:d and 21.5.3";
}

typedef Dot1agCfmCcmInterval {
    type enumeration {
        enum intervalInvalid {
            value 0;
            description
                "No CCMs are sent (disabled).";
        }
        enum interval300Hz {
            value 1;
            description
                "CCMs are sent every 3 1/3 milliseconds (300Hz).";
        }
        enum interval10ms {
            value 2;
            description
                "CCMs are sent every 10 milliseconds.";
        }
        enum interval100ms {
            value 3;
            description
                "CCMs are sent every 100 milliseconds.";
        }
        enum interval1s {
            value 4;
            description
                "CCMs are sent every 1 second.";
        }
        enum interval10s {
            value 5;
            description
                "CCMs are sent every 10 seconds.";
        }
    }
}
```



```
    enum interval1min    {
        value 6;
        description
            "CCMs are sent every minute.";
    }
    enum interval10min    {
        value 7;
        description
            "CCMs are sent every 10 minutes.";
    }
}
description
    "Indicates the interval at which CCMs are sent by a MEP.

    Note: enumerations start at zero to match the 'CCM Interval
    field' protocol field.";
reference
    "802.1ag clauses 12.14.6.1.3:e, 20.8.1 and 21.6.1.3";
}

typedef Dot1agCfmFngState {
    type enumeration {
        enum fngReset      {
            value 1;
            description
                "No defect has been present since the
                dot1agCfmMepFngResetTime timer expired, or since the
                state machine was last reset.";
        }
        enum fngDefect      {
            value 2;
            description
                "A defect is present, but not for a long enough time
                to be reported (dot1agCfmMepFngAlarmTime).";
        }
        enum fngReportDefect {
            value 3;
            description
                "A momentary state during which the defect is reported
                by sending a dot1agCfmFaultAlarm notification, if that
                action is enabled.";
        }
        enum fngDefectReported {
            value 4;
            description
                "A defect is present, and some defect has been
                reported.";
        }
    }
}
```



```
    }
    enum fngDefectClearing {
        value 5;
        description
            "No defect is present, but the dot1agCfmMepFngResetTime
            timer has not yet expired.";
    }
}
description
    "Indicates the diferent states of the MEP Fault Notification
    Generator State Machine.";
reference
    "802.1ag clause 12.14.7.1.3:f and 20.35";
}

typedef Dot1agCfmRelayActionFieldValue {
    type enumeration {
        enum rlyHit {
            value 1;
            description
                "The MP.s Mac address matches the LTM target MAC
                address.";
        }
        enum rlyFdb {
            value 2;
            description
                "The egress port is determined by filter database.";
        }
        enum rlyMpdb {
            value 3;
            description
                "The egress port is determined by the MIP CCM database.";
        }
    }
}
description
    "Possible values the Relay action field can take.";
reference
    "802.1ag clauses 12.14.7.5.3:g, 20.36.2.5, 21.9.5, and
    Table 21-27";
}

typedef Dot1agCfmIngressActionFieldValue {
    type enumeration {
        enum ingNoTlv {
            value 0;
            description
                "Ingress no TLV.";
        }
    }
}
```



```
    }
    enum ingOk      {
        value 1;
        description
            "The target data frame would be passed through the
            bridge.";
    }
    enum ingDown    {
        value 2;
        description
            "The bridge port.s MAC_operational parameter is false.";
    }
    enum ingBlocked {
        value 3;
        description
            "The target data from will not be forwarded due to the
            port is blocked.";
    }
    enum ingVid     {
        value 4;
        description
            "The port is not in the LTM.s VID member set, and the
            target data frame would be filtered at the ingress.";
    }
}
description
    "Possible values returned in the ingress action field.";
reference
    "802.1ag clauses 12.14.7.5.3:g, 20.36.2.6, 21.9.8.1, and
    Table 21-30";
}

typedef Dot1agCfmEgressActionFieldValue {
    type enumeration {
        enum egrNoTlv {
            value 0;
            description
                "Egress no TLV.";
        }
        enum egrOK     {
            value 1;
            description
                "The targeted data frame is forwarded.";
        }
        enum egrDown   {
            value 2;
            description
```



```
    "The egress port.s MAC_Operational parameter is false.";
  }
  enum egrBlocked {
    value 3;
    description
      "The data frame is not passed through the egress port
       due to the port is blocked.";
  }
  enum egrVid      {
    value 4;
    description
      "The port is not in the LTM.s VID member set, and the
       target data frame would be filtered at the ingress.";
  }
}
description
  "Possible values returned in the egress action field";
reference
  "802.1ag clauses 12.14.7.5.3:o, 20.36.2.10, 21.9.9.1, and
   Table 21-32";
}

typedef Dot1agCfmRemoteMepState {
  type enumeration {
    enum rMepIdle   {
      value 1;
      description
        "Momentary state during reset.";
    }
    enum rMepStart  {
      value 2;
      description
        "The timer has not expired since the state machine
         was reset, and no valid CCM has yet been received.";
    }
    enum rMepFailed {
      value 3;
      description
        "The timer has expired, both since the state machine
         was reset, and since a valid CCM was received.";
    }
    enum rMepOk     {
      value 4;
      description
        "The timer has not expired since a valid CCM was
         received.";
    }
  }
}
```



```
}
description
  "Operational state of the remote MEP state machine. This
  state machine monitors the reception of valid CCMS from a
  remote MEP with a specific MEPID. It uses a timer that
  expires in 3.5 times the length of time indicated by the
  dot1agCfmMaNetCcmInterval object.";
reference
  "802.1ag clauses 12.14.7.6.3:b, 20.22";
}

typedef Dot1agCfmIndexIntegerNextFree {
  type uint32 {
    range "0..4294967295";
  }
  description
    "An integer which may be used as a new Index in a table.

    The special value of 0 indicates that no more new entries
    can be created in the relevant table.

    When a MIB is used for configuration, an object with this
    SYNTAX always contains a legal value (if non-zero) for an
    index that is not currently used in the relevant table. The
    Command Generator (Network Management Application) reads
    this variable and uses the (non-zero) value read when
    creating a new row with an SNMP SET. When the SET is
    performed, the Command Responder (agent) must determine
    whether the value is indeed still unused; Two Network
    Management Applications may attempt to create a row
    (configuration entry) simultaneously and use the same value.
    If it is currently unused, the SET succeeds and the Command
    Responder (agent) changes the value of this object,
    according to an implementation-specific algorithm.
    If the value is in use, however, the SET fails.
    The Network Management Application must then re-read this
    variable to obtain a new usable value.

    An OBJECT-TYPE definition using this SYNTAX MUST specify the
    relevant table for which the object is providing this
    functionality.";
}

typedef Dot1agCfmMepDefects {
  type bits {
    bit bDefRDICCM    {
      position 0;
```



```
        description
        "A remote MEP is reported the RDI bit in its last CCM.";
    }
    bit bDefMACstatus {
        position 1;
        description
        "Either some remote MEP is reporting its Interface
        Status TLV as not isUp, or all remote MEPs are
        reporting a Port Status TLV that contains some value
        other than psUp.";
    }
    bit bDefRemoteCCM {
        position 2;
        description
        "The MEP is not receiving valid CCMs from at least
        one of the remote MEPs.";
    }
    bit bDefErrorCCM {
        position 3;
        description
        "The MEP has received at least one invalid CCM whose
        CCM Interval has not yet timed out.";
    }
    bit bDefXconCCM {
        position 4;
        description
        "The MEP has received at least one CCM from either
        another MAID or a lower MD Level whose CCM Interval
        has not yet timed out.";
    }
}
description
"A MEP can detect and report a number of defects, and
multiple defects can be present at the same time.";
reference
"802.1ag clauses 12.14.7.1.3:o, 12.14.7.1.3:p, 12.14.7.1.3:q,
12.14.7.1.3:r, and 12.14.7.1.3:s.";
}

typedef Dot1agCfmConfigErrors {
    type bits {
        bit cfmLeak {
            position 0;
            description
            "MA x is associated with a specific VID list, one or
            more of the VIDs in MA x can pass through the Bridge
            Port, no Down MEP is configured on any Bridge Port
```



```
        for MA x, and some other MA y, at a higher MD Level
        than MA x, and associated with at least one of the
        VID(s) also in MA x, does have a MEP configured on
        the Bridge Port."; }
    bit conflictingVids {
        position 1;
        description
        "MA x is associated with a specific VID list, an Up MEP
        is configured on MA x on the Bridge Port, and some
        other MA y, associated with at least one of the VID(s)
        also in MA x, also has an Up MEP configured on some
        Bridge Port."; }
    bit excessiveLevels {
        position 2;
        description
        "The number of different MD Levels at which MIPs are to
        be created on this port exceeds the Bridge's
        capabilities (see subclause 22.3)."; }
    bit overlappedLevels {
        position 3;
        description
        "A MEP is created for one VID at one MD Level, but a MEP
        is configured on another VID at that MD Level or
        higher, exceeding the Bridge's capabilities."; }
}
description
    "While making the MIP creation evaluation described in
    802.1ag clause 22.2.3, the management entity can encounter
    errors in the configuration.";

reference
    "802.1ag clause 12.14.4.1.3:b and clauses 22.2.3
    and 22.2.4";
}

typedef Dot1agCfmPbbComponentIdentifier {
    type uint32 {
        range "1..4294967295";
    }
}
description
    "A Provider Backbone Bridge (PBB) can comprise a number of
    components, each of which can be managed in a manner
    essentially equivalent to an 802.1Q bridge. In order to
    access these components easily, an index is used in a
    number of tables. If any two tables are indexed by
    Dot1agCfmPbbComponentIdentifier, then entries in those
    tables indexed by the same value of
```



```
        Dot1agCfmPbbComponentIdentifier correspond to the same
        component.";
    reference
        "802.1ag clause 17.5";
}

container dot1agCfmStack {
    config false;
    description
        "It enables the network administrator to discover the
        information about the Maintenance Points configured
        on a port.";

    list dot1agCfmStackEntry {
        key "dot1agCfmStackifIndex dot1agCfmStackVlanIdOrNone
            dot1agCfmStackMdLevel dot1agCfmStackDirection";
        description
            "The Stack table entry";

        leaf dot1agCfmStackifIndex {
            type cfm:if-index;
            config false;
            description
                "This object represents the Bridge Port or aggregated
                port on which MEPs or MHFs might be configured.

                Upon a restart of the system, the system SHALL, if
                necessary, change the value of this variable, and
                rearrange the dot1agCfmStackTable, so that it indexes
                the entry in the interface table with the same value
                of ifAlias that it indexed before the system restart.
                If no such entry exists, then the system SHALL delete
                all entries in the dot1agCfmStackTable with the
                interface index.";
            reference
                "802.1ag clause 12.14.2.1.2:a";
        }

        leaf dot1agCfmStackVlanIdOrNone {
            type cfm:VlanIdOrNone;
            config false;
            description
                "VLAN ID to which the MP is attached, or 0, if none.";
            reference
                "802.1ag clauses 12.14.2.1.2:d, 22.1.7";
        }
    }
}
```



```
leaf dot1agCfmStackMdLevel {
    type cfm:Dot1agCfmMDLevel;
    config false;
    description
        "MD Level of the Maintenance Point.";
    reference
        "802.1ag clause 12.14.2.1.2:b";
}

leaf dot1agCfmStackDirection {
    type cfm:Dot1agCfmMpDirection;
    config false;
    description
        "Direction in which the MP faces on the Bridge Port";
    reference
        "802.1ag clause 12.14.2.1.2:c";
}

leaf dot1agCfmStackMdIndex {
    type uint32;
    config false;
    description
        "The index of the Maintenance Domain in the
        dot1agCfmMdTable to which the MP is associated,
        or 0, if none.";
    reference
        "802.1ag clause 12.14.2.1.3:b";
}

leaf dot1agCfmStackMaIndex {
    type uint32;
    config false;
    description
        "The index of the MA in the dot1agCfmMaNetTable and
        dot1agCfmMaCompTable to which the MP is associated,
        or 0, if none.";
    reference
        "802.1ag clause 12.14.2.1.3:c";
}

leaf dot1agCfmStackMepId {
    type cfm:Dot1agCfmMepIdOrZero;
    config false;
    description
        "If an MEP is configured, the MEPID, else 0";
    reference
        "802.1ag clause 12.14.2.1.3:d";
}
```



```
    }

    leaf dot1agCfmStackMacAddress {
        type yang:mac-address;
        config false;
        description
            "MAC address of the MP.";
        reference
            "802.1ag clause 12.14.2.1.3:e";
    }
}

container dot1agCfmDefaultMd {
    description
        "Interface configuration parameters.";

    leaf dot1agCfmDefaultMdDefLevel {
        type cfm:Dot1agCfmMDLevel;
        description
            "A value indicating the MD Level at which MHFs are to be
            created, and Sender ID TLV transmission by those MHFs is
            to be controlled, for each dot1agCfmDefaultMdEntry whose
            dot1agCfmDefaultMdLevel object contains the value -1.

            After this initialization, this object needs to be
            persistent upon reboot or restart of a device.";
        reference
            "802.1ag clause 12.14.3.1.3:c, 12.14.3.2.2:b";
    }

    leaf dot1agCfmDefaultMdDefMhfCreation {
        type cfm:Dot1agCfmMhfCreation;
        description
            "A value indicating if the Management entity can create
            MHFs (MIP Half Function) for the VID, for each
            dot1agCfmDefaultMdEntry whose
            dot1agCfmDefaultMdMhfCreation
            object contains the value defMHFdefer. Since, in this
            variable, there is no encompassing Maintenance Domain,
            the value defMHFdefer is not allowed.

            After this initialization, this object needs to be
            persistent upon reboot or restart of a device.";
        reference
            "802.1ag clause 12.14.3.1.3:d";
    }
}
```



```
leaf dot1agCfmDefaultMdDefIdPermission {
  type cfm:Dot1agCfmIdPermission;
  description
    "Enumerated value indicating what, if anything, is to be
    included in the Sender ID TLV (21.5.3) transmitted by
    MHFs created by the Default Maintenance Domain, for each
    dot1agCfmDefaultMdEntry whose
    dot1agCfmDefaultMdIdPermission object contains the value
    sendIdDefer. Since, in this variable, there is no
    encompassing Maintenance Domain, the value sendIdDefer
    is not allowed.

    After this initialization, this object needs to be
    persistent upon reboot or restart of a device.";
  reference
    "802.1ag clause 12.14.3.1.3:e";
}

list dot1agCfmDefaultMdEntry {

  key "dot1agCfmDefaultMdComponentId
      dot1agCfmDefaultMdPrimaryVid";
  description
    "The Default MD Level table entry.";

  leaf dot1agCfmDefaultMdComponentId {
    type cfm:Dot1agCfmPbbComponentIdentifier;
    description
      "The bridge component within the system to which
      the information in this dot1agCfmDefaultMdEntry
      applies. If the system is not a Bridge, or if only
      one component is present in the Bridge, then this
      variable (index) must be equal to 1.";
    reference
      "802.1ag clause 17.5";
  }

  leaf dot1agCfmDefaultMdPrimaryVid {
    type cfm:VlanId;
    description
      "The Primary VID of the VLAN to which this entry's
      objects apply.";
  }

  leaf dot1agCfmDefaultMdStatus {
```



```
    type boolean;
    config false;
    description
        "State of this Default MD Level table entry. True if
        there is no entry in the Maintenance Association
        table defining an MA for the same VLAN ID and MD
        Level as this table's entry, and on which MA an Up
        MEP is defined, else false.";
    reference
        "802.1ag clause 12.14.3.1.3:b";
}

leaf dot1agCfmDefaultMdLevel {
    type cfm:Dot1agCfmMDLevelOrNone;
    description
        "A value indicating the MD Level at which MHFs are to
        be created, and Sender ID TLV transmission by those
        MHFs is to be controlled, for the VLAN to which this
        entry's objects apply. If this object has the value
        -1, the MD Level for MHF creation for this VLAN is
        controlled by dot1agCfmDefaultMdDefLevel.";
    reference
        "802.1ag clause 12.14.3.1.3:c, 12.14.3.2.2:b";
}

leaf dot1agCfmDefaultMdMhfCreation {
    type cfm:Dot1agCfmMhfCreation;
    description
        "A value indicating if the Management entity can
        create MHFs (MIP Half Function) for this VID at
        this MD Level. If this object has the value
        defMhfDefer, MHF creation for this VLAN
        is controlled by dot1agCfmDefaultMdDefMhfCreation.

        The value of this variable is meaningless if the
        values of dot1agCfmDefaultMdStatus is false.";
    reference
        "802.1ag clause 12.14.3.1.3:d";
}

leaf dot1agCfmDefaultMdIdPermission {
    type cfm:Dot1agCfmIdPermission;
    description
        "Enumerated value indicating what, if anything, is to
        be included in the Sender ID TLV (21.5.3)
        transmitted by MHFs created by the Default
        Maintenance Domain. If this object has the value
```


sendIdDefer, Sender ID TLV transmission for this VLAN is controlled by dot1agCfmDefaultMdDefIdPermission.

The value of this variable is meaningless if the values of dot1agCfmDefaultMdStatus is false.";

reference

"802.1ag clause 12.14.3.1.3:e";

```
}  
}  
}
```

```
container dot1agCfmVlan {  
  description
```

```
    "It contains the association between VID and VLAN.";
```

```
  list dot1agCfmVlanEntry {
```

```
    key "dot1agCfmVlanComponentId dot1agCfmVlanVid";  
    description
```

```
      "The VLAN table entry.";
```

```
    leaf dot1agCfmVlanComponentId {
```

```
      type cfm:Dot1agCfmPbbComponentIdentifier;
```

```
      description
```

```
        "The bridge component within the system to which the  
        information in this dot1agCfmVlanEntry applies.
```

```
        If the system is not a Bridge, or if only one  
        component is present in the Bridge, then
```

```
        this variable (index) must be equal to 1.";
```

```
      reference
```

```
        "802.1ag clause 17.5";
```

```
    }
```

```
    leaf dot1agCfmVlanVid {
```

```
      type cfm:VlanId;
```

```
      description
```

```
        "This is a VLAN ID belonging to a VLAN that is  
        associated with more than one VLAN ID, and this  
        is not the Primary VID of the VLAN.";
```

```
    }
```

```
    leaf dot1agCfmVlanPrimaryVid {
```

```
      type cfm:VlanId;
```

```
      description
```

```
        "This is the Primary VLAN ID of the VLAN with which  
        this entry's dot1agCfmVlanVid is associated. This
```



```
        value must not equal the value of dot1agCfmVlanVid.";
    }

    leaf dot1agCfmVlanRowStatus {
        type cfm:RowStatus;
        description
            "The status of the row.

            The writable columns in a row can not be changed if
            the row is active. All columns must have a valid
            value before a row can be activated.";
    }
}

container dot1agCfmConfigErrorList {
    config false;
    description
        "The CFM Configuration Error List provides a list of
        Interfaces and VIDs that are incorrectly configured.";

    list dot1agCfmConfigErrorListEntry {

        key "dot1agCfmConfigErrorListVid
            dot1agCfmConfigErrorListIfIndex";
        description
            "The Config Error List Table entry";

        leaf dot1agCfmConfigErrorListVid {
            type cfm:VlanId;
            description
                "The VLAN ID of the VLAN with interfaces in error.";
            reference
                "802.1ag Clause 12.14.4.1.2:a";
        }

        leaf dot1agCfmConfigErrorListIfIndex {
            type cfm:if-index;
            description
                "This object is the IfIndex of the interface.

                Upon a restart of the system, the system SHALL,
                if necessary, change the value of this variable
                so that it indexes the entry in the interface
                table with the same value of ifAlias that it indexed
                before the system restart. If no such entry exists,
                then the system SHALL delete any entries in
```



```
        dot1agCfmConfigErrorListTable indexed by that
        InterfaceIndex value.";
    reference
        "802.1ag clause 12.14.4.1.2:b";
}

leaf dot1agCfmConfigErrorListErrorType {
    type cfm:Dot1agCfmConfigErrors;
    description
        "A vector of Boolean error conditions from 22.2.4,
        any of which may be true:

        0) CFMleak;
        1) ConflictingVids;
        2) ExcessiveLevels;
        3) OverlappedLevels.";
    reference
        "802.1ag clause 12.14.4.1.3:b";
}
}

container dot1agCfmMd {
    description
        "A Maintenance Domain is described in 802.1ag (3.21) as the
        network or the part of the network for which faults in
        connectivity are to be managed. The boundary of a Maintenance
        Domain is defined by a set of DSAPs, each of which can become
        a point of connectivity to a service instance.";

    leaf dot1agCfmMdTableNextIndex {
        type cfm:Dot1afCfmIndexIntegerNextFree;
        config false;
        description
            "This object contains an unused value for dot1agCfmMdIndex
            in the dot1agCfmMdTable, or a zero to indicate that
            none exist.";
    }

    list dot1agCfmMdEntry {

        key "dot1agCfmMdIndex";
        description
            "The Maintenance Domain table entry. This entry is
            not lost upon reboot. It is backed up by stable
            storage.";
    }
}
```



```
leaf dot1agCfmMdIndex {
  type uint32 {
    range "1..4294967295";
  }
  description
    "The index to the Maintenance Domain table.

    dot1agCfmMdTableNextIndex needs to be inspected to
    find an available index for row-creation.

    Referential integrity is required, i.e., the index
    needs to be persistent upon a reboot or restart of
    a device. The index can never be reused for other
    Maintenance Domain. The index value should keep
    increasing up to the time that they wrap around.
    This is to facilitate access control based on OID."
}

leaf dot1agCfmMdFormat {
  type cfm:Dot1agCfmMaintDomainNameType;
  description
    "The type (and thereby format) of the Maintenance
    Domain Name.";
  reference
    "802.1ag clause 21.6.5.1";
}

leaf dot1agCfmMdName {
  type cfm:Dot1agCfmMaintDomainName;
  description
    "The Maintenance Domain name. The type/format of
    this object is determined by the value of the
    dot1agCfmMdNameType object.

    Each Maintenance Domain has unique name amongst
    all those used or available to a service provider
    or operator. It facilitates easy identification
    of administrative responsibility for each Maintenance
    Domain.

    Clause 3.23 defines a Maintenance Domain name as the
    identifier, unique over the domain for which CFM is to
    protect against accidental concatenation of Service
    Instances, of a particular Maintenance Domain."
  reference
    "802.1ag clauses 3.23, 12.14.5, and 21.6.5.3";
```



```
}

leaf dot1agCfmMdMdLevel {
    type cfm:Dot1agCfmMDLevel;
    description
        "The Maintenance Domain Level.";
    reference
        "802.1ag clause 12.14.5.1.3:b";
}

leaf dot1agCfmMdMhfCreation {
    type cfm:Dot1agCfmMhfCreation;
    description
        "Enumerated value indicating whether the management
        entity can create MHFs (MIP Half Function) for
        this Maintenance Domain. Since, in this variable,
        there is no encompassing Maintenance Domain,
        the value defMHFdefer is not allowed.";
    reference
        "802.1ag clause 12.14.5.1.3:c";
}

leaf dot1agCfmMdMhfIdPermission {
    type cfm:Dot1agCfmIdPermission;
    description
        "Enumerated value indicating what, if anything, is to
        be included in the Sender ID TLV (21.5.3) transmitted
        by MPs configured in this Maintenance Domain. Since,
        in this variable, there is no encompassing Maintenance
        Domain, the value sendIdDefer is not allowed.";
    reference
        "802.1ag clause 12.14.5.1.3:d";
}

leaf dot1agCfmMdMaNextIndex {
    type cfm:Dot1afCfmIndexIntegerNextFree;
    config false;
    description
        "Value to be used as the index of the MA table entries,
        both the dot1agCfmMaNetTable and the
        dot1agCfmMaCompTable, for this Maintenance Domain
        when the management entity wants to create a new row
        in those tables.";
}

leaf dot1agCfmMdRowStatus {
    type cfm:RowStatus;
```



```
        description
            "The status of the row.

            The writable columns in a row can not be changed if
            the row is active. All columns must have a valid
            value before a row can be activated.";
    }
}

container dot1agCfmMa {
    description
        "The Maintenance Association contains the VLAN ID that
        it wants to monitor.";

    list dot1agCfmMaNetEntry {

        key "dot1agCfmMdIndex dot1agCfmMaIndex";
        description
            "The MA table entry.";

        leaf dot1agCfmMdIndex {
            type leafref {
                path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"
                    +":dot1agCfmMdIndex";
            }
            description
                "Automagically generated leafref leaf.";
        }

        leaf dot1agCfmMaIndex {
            type uint32 {
                range "1..4294967295";
            }
            description
                "Index of the MA table dot1agCfmMdMaNextIndex needs to
                be inspected to find an available index for
                row-creation.";
        }

        leaf dot1agCfmMaNetFormat {
            type cfm:Dot1agCfmMaintAssocNameType;
            description
                "The type (and thereby format) of the Maintenance
                Association Name.";
            reference
                "802.1ag clauses 21.6.5.4";
        }
    }
}
```



```
    }

    leaf dot1agCfmMaNetName {
      type cfm:Dot1agCfmMaintAssocName;
      description
        "The Short Maintenance Association name. The
         type/format of this object is determined by the
         value of the dot1agCfmMaNetNameType object.
         This name must be unique within a maintenance
         domain.";
      reference
        "802.1ag clauses 21.6.5.6, and Table 21-20";
    }

    leaf dot1agCfmMaNetCcmInterval {
      type cfm:Dot1agCfmCcmInterval;
      description
        "Interval between CCM transmissions to be used by
         all MEPs in the MA.";
      reference
        "802.1ag clause 12.14.6.1.3:e";
    }

    leaf dot1agCfmMaNetRowStatus {
      type cfm:RowStatus;
      description
        "The status of the row.

         The writable columns in a row can not be changed
         if the row is active. All columns must have a valid
         value before a row can be activated.";
    }
  }
}

list dot1agCfmMaCompEntry {

  key "dot1agCfmMaComponentId dot1agCfmMdIndex
      dot1agCfmMaIndex";
  description
    "The MA table entry.";

  leaf dot1agCfmMdIndex {
    type leafref {
      path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"
        +":dot1agCfmMdIndex";
    }
    description

```



```
        "Automagically generated leafref leaf.";
    }
    leaf dot1agCfmMaIndex {
        type leafref {
            path "/cfm:dot1agCfmMa/cfm:dot1agCfmMaNetEntry/cfm"
                +":dot1agCfmMaIndex";
        }
        description
            "Automagically generated leafref leaf.";
    }

    leaf dot1agCfmMaComponentId {
        type cfm:Dot1agCfmPbbComponentIdentifier;
        description
            "The bridge component within the system to which
            the information in this dot1agCfmMaCompEntry applies.
            If the system is not a Bridge, or if only one
            component is present in the Bridge, then this
            variable (index) must be equal to 1.";
        reference
            "802.1ag clause 17.5";
    }

    leaf dot1agCfmMaCompPrimaryVlanId {
        type cfm:VlanIdOrNone;
        description
            "The Primary VLAN ID with which the Maintenance
            Association is associated, or 0 if the MA is not
            attached to any VID. If the MA is associated with
            more than one VID, the dot1agCfmVlanTable lists
            them.";
        reference
            "802.1ag clause 12.14.6.1.3:b";
    }

    leaf dot1agCfmMaCompMhfcCreation {
        type cfm:Dot1agCfmMhfcCreation;
        description
            "Indicates if the Management entity can create MHFs
            (MIP Half Function) for this MA.";
        reference
            "802.1ag clause 12.14.6.1.3:c";
    }

    leaf dot1agCfmMaCompIdPermission {
        type cfm:Dot1agCfmIdPermission;
        description
```



```
        "Enumerated value indicating what, if anything, is
        to be included in the Sender ID TLV (21.5.3)
        transmitted by MPs configured in this MA.";
    reference
        "802.1ag clause 12.14.6.1.3:d";
}

leaf dot1agCfmMaCompNumberOfVids {
    type uint32;
    config false;
    description
        "The number of VIDs associated with the MA.";
    reference
        "802.1ag clause 12.14.6.1.3:b";
}

leaf dot1agCfmMaCompRowStatus {
    type cfm:RowStatus;
    description
        "The status of the row.

        The writable columns in a row can not be changed if
        the row is active. All columns must have a valid
        value before a row can be activated.";
}
}

list dot1agCfmMaMepListEntry {

    key "dot1agCfmMdIndex dot1agCfmMaIndex
        dot1agCfmMaMepListIdentifier";
    description
        "The known MEPS table entry.";

    leaf dot1agCfmMdIndex {
        type leafref {
            path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"
                +":dot1agCfmMdIndex";
        }
        description
            "Automagically generated leafref leaf.";
    }
    leaf dot1agCfmMaIndex {
        type leafref {
            path "/cfm:dot1agCfmMa/cfm:dot1agCfmMaNetEntry/cfm"
                +":dot1agCfmMaIndex";
        }
    }
}
```



```
    }
    description
      "Automagically generated leafref leaf.";
  }

  leaf dot1agCfmMaMepListIdentifier {
    type cfm:Dot1agCfmMepId;
    description
      "MEPID";
    reference
      "802.1ag clause 12.14.6.1.3:g";
  }

  leaf dot1agCfmMaMepListRowStatus {
    type cfm:RowStatus;
    description
      "The status of the row. Read SNMPV2-TC (RFC1903)
      for an explanation of the possible values this
      object can take.";
  }
}

container dot1agCfmMep {
  description
    "The MEP container contains the configuration for a
    Maintenance point, for example, the direction, ID,
    VLAN ID and so on.";

  list dot1agCfmMepEntry {

    key "dot1agCfmMdIndex dot1agCfmMaIndex
        dot1agCfmMepIdentifier";
    description
      "The MEP table entry";

    leaf dot1agCfmMdIndex {
      type leafref {
        path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"
          +":dot1agCfmMdIndex";
      }
      description
        "Automagically generated leafref leaf.";
    }
    leaf dot1agCfmMaIndex {
      type leafref {
        path "/cfm:dot1agCfmMa/cfm:dot1agCfmMaNetEntry/cfm"
```



```
        +":dot1agCfmMaIndex";
    }
    description
        "Automagically generated leafref leaf.";
}

leaf dot1agCfmMepIdentifier {
    type cfm:Dot1agCfmMepId;
    description
        "Integer that is unique among all the MEPs in the
        same MA. Other definition is: a small integer,
        unique over a given Maintenance Association,
        identifying a specific Maintenance association
        End Point (3.18).

        MEP Identifier is also known as the MEPID.";
    reference
        "802.1ag clauses 3.18, 19.2 and 12.14.7";
}

leaf dot1agCfmMepIfIndex {
    type cfm:if-index-or-zero;
    description
        "This object is the interface index of the interface
        either a bridge port, or an aggregated IEEE 802.1
        link within a bridge port, to which the MEP is
        attached.

        Upon a restart of the system, the system SHALL, if
        necessary, change the value of this variable so that
        it indexes the entry in the interface table with the
        same value of ifAlias that it indexed before the
        system restart. If no such entry exists, then the
        system SHALL set this variable to 0.";
    reference
        "802.1ag clause 12.14.7.1.3:b";
}

leaf dot1agCfmMepDirection {
    type cfm:Dot1agCfmMpDirection;
    description
        "The direction in which the MEP faces on the Bridge
        port.";
    reference
        "802.1ag clauses 12.14.7.1.3:c and 19.2";
}
```



```
leaf dot1agCfmMepPrimaryVid {
  type uint32 {
    range "0..16777215";
  }
  description
    "An integer indicating the Primary VID of the MEP,
    always one of the VIDs assigned to the MEP's MA.
    The value 0 indicates that either the Primary VID
    is that of the MEP's MA, or that the MEP's MA is
    associated with no VID.";
  reference
    "802.1ag clauses 12.14.7.1.3:d";
}

leaf dot1agCfmMepActive {
  type boolean;
  description
    "Administrative state of the MEP

    A Boolean indicating the administrative state of
    the MEP.

    True indicates that the MEP is to function normally,
    and false that it is to cease functioning.";
  reference
    "802.1ag clauses 12.14.7.1.3:e and 20.9.1";
}

leaf dot1agCfmMepFngState {
  type cfm:Dot1agCfmFngState;
  config false;
  description
    "Current state of the MEP Fault Notification Generator
    State Machine.";
  reference
    "802.1ag clauses 12.14.7.1.3:f and 20.35";
}

leaf dot1agCfmMepCciEnabled {
  type boolean;
  description
    "If set to true, the MEP will generate CCM messages.";
  reference
    "802.1ag clauses 12.14.7.1.3:g and 20.10.1";
}

leaf dot1agCfmMepCcmLtmPriority {
```



```
    type uint32 {
        range "0..7";
    }
    description
        "The priority value for CCMs and LTMs transmitted by
        the MEP. Default Value is the highest priority value
        allowed to pass through the bridge port for any of
        this MEPs VIDs. The management entity can obtain the
        default value for this variable from the priority
        regeneration table by extracting the highest priority
        value in this table on this MEPs bridge port.
        (1 is lowest, then 2, then 0, then 3-7).";
    reference
        "802.1ag clause 12.14.7.1.3:h";
}

leaf dot1agCfmMepMacAddress {
    type yang:mac-address;
    config false;
    description
        "MAC address of the MEP.";
    reference
        "802.1ag clause 12.14.7.1.3:i and 19.4";
}

leaf dot1agCfmMepLowPrDef {
    type cfm:Dot1agCfmLowestAlarmPri;
    description
        "An integer value specifying the lowest priority
        defect that is allowed to generate fault alarm.";
    reference
        "802.1ag clause 12.14.7.1.3:k and 20.9.5 and Table
        20-1";
}

leaf dot1agCfmMepFngAlarmTime {
    type cfm:TimeInterval {
        range "250..1000";
    }
    description
        "The time that defects must be present before a Fault
        Alarm is issued (fngAlarmTime. 20.33.3)
        (default 2.5s).";
    reference
        "802.1ag clauses 12.14.7.1.3:l and 20.33.3";
}
```



```
leaf dot1agCfmMepFngResetTime {
  type cfm:TimeInterval {
    range "250..1000";
  }
  description
    "The time that defects must be absent before resetting
    a Fault Alarm (fngResetTime, 20.33.4) (default 10s).";
  reference
    "802.1ag clauses 12.14.7.1.3:m and 20.33.4";
}

leaf dot1agCfmMepHighestPrDefect {
  type cfm:Dot1agCfmHighestDefectPri;
  config false;
  description
    "The highest priority defect that has been present
    since the MEPs Fault Notification Generator State
    Machine was last in the FNG_RESET state.";
  reference
    "802.1ag clause 12.14.7.1.3:n 20.33.9 and Table 21-1";
}

leaf dot1agCfmMepDefects {
  type cfm:Dot1agCfmMepDefects;
  config false;
  description
    "A vector of Boolean error conditions from Table 20-1,
    any of which may be true:

    DefRDICCM(0)
    DefMACstatus(1)
    DefRemoteCCM(2)
    DefErrorCCM(3)
    DefXconCCM(4)";
  reference
    ".1ag clauses 12.14.7.1.3:o, 12.14.7.1.3:p,
    12.14.7.1.3:q, 12.14.7.1.3:r, 12.14.7.1.3:s,
    20.21.3, 20.23.3, 20.33.5, 20.33.6, 20.33.7.";
}

leaf dot1agCfmMepErrorCcmLastFailure {
  type binary {
    length "1..1522";
  }
  config false;
  description
    "The last-received CCM that triggered an DefErrorCCM
```



```
        fault.";
    reference
        "802.1ag clauses 12.14.7.1.3:t and 20.21.2";
}

leaf dot1agCfmMepXconCcmLastFailure {
    type binary {
        length "1..1522";
    }
    config false;
    description
        "The last-received CCM that triggered a DefXconCCM
        fault.";
    reference
        "802.1ag clauses 12.14.7.1.3:u and 20.23.2";
}

leaf dot1agCfmMepCcmSequenceErrors {
    type yang:counter32;
    config false;
    description
        "The total number of out-of-sequence CCMS received
        from all remote MEPs.";
    reference
        "802.1ag clauses 12.14.7.1.3:v and 20.16.12";
}

leaf dot1agCfmMepCciSentCcms {
    type yang:counter32;
    config false;
    description
        "Total number of Continuity Check messages
        transmitted.";
    reference
        "802.1ag clauses 12.14.7.1.3:w and 20.10.2";
}

leaf dot1agCfmMepNextLbmTransId {
    type uint32;
    config false;
    description
        "Next sequence number/transaction identifier to be
        sent in a Loopback message. This sequence number can
        be zero because it wraps around.";
    reference
        "802.1ag clauses 12.14.7.1.3:x and 20.28.2";
}
```



```
leaf dot1agCfmMepLbrIn {
    type yang:counter32;
    config false;
    description
        "Total number of valid, in-order Loopback Replies
        received.";
    reference
        "802.1ag clause 12.14.7.1.3:y and 20.31.1";
}

leaf dot1agCfmMepLbrInOutOfOrder {
    type yang:counter32;
    config false;
    description
        "The total number of valid, out-of-order Loopback
        Replies received.";
    reference
        "802.1ag clause 12.14.7.1.3:z and 20.31.1";
}

leaf dot1agCfmMepLbrBadMsdu {
    type yang:counter32;
    config false;
    description
        "The total number of LBRs received whose
        mac_service_data_unit did not match (except for
        the OpCode) that of the corresponding LBM (20.2.3).";
    reference
        "802.1ag clause 12.14.7.1.3:aa 20.2.3";
}

leaf dot1agCfmMepLtmNextSeqNumber {
    type uint32;
    config false;
    description
        "Next transaction identifier/sequence number to be
        sent in a Linktrace message. This sequence number
        can be zero because it wraps around.";
    reference
        "802.1ag clause 12.14.7.1.3:ab and 20.36.1";
}

leaf dot1agCfmMepUnexpltrIn {
    type yang:counter32;
    config false;
    description
```



```
        "The total number of unexpected LTRs received.";
    reference
        "802.1ag clause 12.14.7.1.3:ac 20.39.1";
}

leaf dot1agCfmMepLbrOut {
    type yang:counter32;
    config false;
    description
        "Total number of Loopback Replies transmitted.";
    reference
        "802.1ag clause 12.14.7.1.3:ad and 20.26.2";
}

leaf dot1agCfmMepTransmitLbmStatus {
    type boolean;
    description
        "A Boolean flag set to true by the bridge port to
        indicate that another LBM may be transmitted.";
}

leaf dot1agCfmMepTransmitLbmDestMacAddress {
    type yang:mac-address;
    description
        "The Target MAC Address Field to be transmitted:
        A unicast destination MAC address.
        This address will be used if the value of the column
        dot1agCfmMepTransmitLbmDestIsMepId is 'false'.";
    reference
        "802.1ag clause 12.14.7.3.2:b";
}

leaf dot1agCfmMepTransmitLbmDestMepId {
    type cfm:Dot1agCfmMepIdOrZero;
    description
        "The Maintenance association End Point Identifier of
        another MEP in the same Maintenance Association to
        which the LBM is to be sent.
        This address will be used if the value of the column
        dot1agCfmMepTransmitLbmDestIsMepId is 'true'.";
    reference
        "802.1ag clause 12.14.7.3.2:b";
}

leaf dot1agCfmMepTransmitLbmDestIsMepId {
    type boolean;
    description
```



```
        "True indicates that MEPID of the target MEP is used
        for Loopback transmission.
        False indicates that unicast destination MAC address
        of the target MEP is used for Loopback transmission.";
    reference
        "802.1ag clause 12.14.7.3.2:b";
}

leaf dot1agCfmMepTransmitLbmMessages {
    type int32 {
        range "1..1024";
    }
    description
        "The number of Loopback messages to be transmitted.";
    reference
        "802.1ag clause 12.14.7.3.2:c";
}

leaf dot1agCfmMepTransmitLbmDataTlv {
    type binary {
        length "0..1500";
    }
    description
        "An arbitrary amount of data to be included in the
        Data TLV, if the Data TLV is selected to be sent.";
    reference
        "802.1ag clause 12.14.7.3.2:d";
}

leaf dot1agCfmMepTransmitLbmVlanPriority {
    type int32 {
        range "0..7";
    }
    description
        "Priority. 3 bit value to be used in the VLAN tag,
        if present in the transmitted frame.

        The default value is CCM priority.";
    reference
        "802.1ag clause 12.14.7.3.2:e";
}

leaf dot1agCfmMepTransmitLbmVlanDropEnable {
    type boolean;
    description
        "Drop Enable bit value to be used in the VLAN tag,
        if present in the transmitted frame.
```



```
        For more information about VLAN Drop Enable,
        please check IEEE 802.1ad.";
    reference
        "802.1ag clause 12.14.7.3.2:e";
}

leaf dot1agCfmMepTransmitLbmResultOK {
    type boolean;
    config false;
    description
        "Indicates the result of the operation:

        - true      The Loopback Message(s) will be
                     (or has been) sent.
        - false     The Loopback Message(s) will not
                     be sent.";
    reference
        "802.1ag clause 12.14.7.3.3:a";
}

leaf dot1agCfmMepTransmitLbmSeqNumber {
    type uint32;
    config false;
    description
        "The Loopback Transaction Identifier
        (dot1agCfmMepNextLbmTransId) of the first LBM (to be)
        sent.
        The value returned is undefined if
        dot1agCfmMepTransmitLbmResultOK is false.";
    reference
        "802.1ag clause 12.14.7.3.3:a";
}

leaf dot1agCfmMepTransmitLtmStatus {
    type boolean;
    config false;
    description
        "A Boolean flag set to true by the bridge port to
        indicate that another LTM may be transmitted.
        Reset to false by the MEP Linktrace Initiator
        State Machine.";
}

leaf dot1agCfmMepTransmitLtmFlags {
    type bits {
        bit useFDBOnly {
```



```
        position 0;
        description
            "It is used for indicating if only bridge.s
            filter database is used for determining the
            egress port.";
    }
}
description
    "The flags field for LTMs transmitted by the MEP.";
reference
    "802.1ag clause 12.14.7.4.2:b and 20.37.1";
}

leaf dot1agCfmMepTransmitLtmTargetMacAddress {
    type yang:mac-address;
    description
        "The Target MAC Address Field to be transmitted:
        A unicast destination MAC address.
        This address will be used if the value of the column
        dot1agCfmMepTransmitLtmTargetIsMepId is 'false'.";
    reference
        "802.1ag clause 12.14.7.4.2:c";
}

leaf dot1agCfmMepTransmitLtmTargetMepId {
    type cfm:Dot1agCfmMepIdOrZero;
    description
        "An indication of the Target MAC Address Field to be
        transmitted:
        The Maintenance association End Point Identifier of
        another MEP in the same Maintenance Association
        This address will be used if the value of the column
        dot1agCfmMepTransmitLtmTargetIsMepId is 'true'.";
    reference
        "802.1ag clause 12.14.7.4.2:c";
}

leaf dot1agCfmMepTransmitLtmTargetIsMepId {
    type boolean;
    description
        "True indicates that MEPID of the target MEP is used
        for Linktrace transmission.
        False indicates that unicast destination MAC address
        of the target MEP is used for Loopback transmission.";
    reference
        "802.1ag clause 12.14.7.4.2:c";
}
```



```
leaf dot1agCfmMepTransmitLtmTtl {
  type uint32 {
    range "0..255";
  }
  description
    "The LTM TTL field. Default value, if not specified,
    is 64. The TTL field indicates the number of hops
    remaining to the LTM. Decrementd by 1 by each
    Linktrace Responder that handles the LTM. The
    value returned in the LTR is one less than that
    received in the LTM. If the LTM TTL is 0 or 1, the
    LTM is not forwarded to the next hop, and if 0, no
    LTR is generated.";
  reference
    "802.1ag clause 12.14.7.4.2:d and 21.8.4";
}

leaf dot1agCfmMepTransmitLtmResult {
  type boolean;
  config false;
  description
    "Indicates the result of the operation:

    - true    The Linktrace Message will be (or has been)
               sent.
    - false   The Linktrace Message will not be sent";
  reference
    "802.1ag clause 12.14.7.4.3:a";
}

leaf dot1agCfmMepTransmitLtmSeqNumber {
  type uint32;
  config false;
  description
    "The LTM Transaction Identifier
    (dot1agCfmMepLtmNextSeqNumber) of the LTM sent.
    The value returned is undefined if
    dot1agCfmMepTransmitLtmResult is false.";
  reference
    "802.1ag clause 12.14.7.4.3:a";
}

leaf dot1agCfmMepTransmitLtmEgressIdentifier {
  type binary {
    length "8";
  }
}
```


description

"Identifies the MEP Linktrace Initiator that is originating, or the Linktrace Responder that is forwarding, this LTM. The low-order six octets contain a 48-bit IEEE MAC address unique to the system in which the MEP Linktrace Initiator or Linktrace Responder resides. The high-order two octets contain a value sufficient to uniquely identify the MEP Linktrace Initiator or Linktrace Responder within that system.

For most Bridges, the address of any MAC attached to the Bridge will suffice for the low-order six octets, and 0 for the high-order octets. In some situations, e.g., if multiple virtual Bridges utilizing emulated LANs are implemented in a single physical system, the high-order two octets can be used to differentiate among the transmitting entities.

The value returned is undefined if
dot1agCfmMepTransmitLtmResult is false.";

reference

"802.1ag clause 12.14.7.4.3:b and 21.8.8";

}

leaf dot1agCfmMepRowStatus {

type cfm:RowStatus;

description

"The status of the row.

The writable columns in a row can not be changed if the row is active. All columns must have a valid value before a row can be activated.";

}

}

list dot1agCfmLtrEntry {

key "dot1agCfmMdIndex dot1agCfmMaIndex
dot1agCfmMepIdentifier dot1agCfmLtrSeqNumber
dot1agCfmLtrReceiveOrder";

description

"The Linktrace Reply table entry.";

leaf dot1agCfmMdIndex {

type leafref {


```
        path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"
        +":dot1agCfmMdIndex";
    }
    description
        "Automagically generated leafref leaf.";
}
leaf dot1agCfmMaIndex {
    type leafref {
        path "/cfm:dot1agCfmMa/cfm:dot1agCfmMaNetEntry/cfm"
        +":dot1agCfmMaIndex";
    }
    description
        "Automagically generated leafref leaf.";
}
leaf dot1agCfmMepIdentifier {
    type leafref {
        path "/cfm:dot1agCfmMep/cfm:dot1agCfmMepEntry/cfm"
        +":dot1agCfmMepIdentifier";
    }
    description
        "Automagically generated leafref leaf.";
}

leaf dot1agCfmLtrSeqNumber {
    type uint32 {
        range "0..4294967295";
    }
    description
        "Transaction identifier/Sequence number returned by
        a previous transmit linktrace message command,
        indicating which LTM's response is going to be
        returned.";
    reference
        "802.1ag clause 12.14.7.5.2:b";
}

leaf dot1agCfmLtrReceiveOrder {
    type uint32 {
        range "1..4294967295";
    }
    description
        "An index to distinguish among multiple LTRs with the
        same LTR. Transaction Identifier field value.
        dot1agCfmLtrReceiveOrder are assigned sequentially
        from 1, in the order that the Linktrace Initiator
        received the LTRs.";
    reference
```



```
        "802.1ag clause 12.14.7.5.2:c";
    }

    leaf dot1agCfmLtrTtl {
        type uint32 {
            range "0..255";
        }
        config false;
        description
            "TTL field value for a returned LTR.";
        reference
            "802.1ag clause 12.14.7.5 and 20.36.2.2";
    }

    leaf dot1agCfmLtrForwarded {
        type boolean;
        config false;
        description
            "Indicates if a LTM was forwarded by the responding
            MP, as returned in the 'FwdYes' flag of the flags
            field.";
        reference
            "802.1ag clauses 12.14.7.5.3:c and 20.36.2.1";
    }

    leaf dot1agCfmLtrTerminalMep {
        type boolean;
        config false;
        description
            "A boolean value stating whether the forwarded LTM
            reached a MEP enclosing its MA, as returned in the
            Terminal MEP flag of the Flags field.";
        reference
            "802.1ag clauses 12.14.7.5.3:d and 20.36.2.1";
    }

    leaf dot1agCfmLtrLastEgressIdentifier {
        type binary {
            length "8";
        }
        config false;
        description
            "An octet field holding the Last Egress Identifier
            returned in the LTR Egress Identifier TLV of the LTR.
            The Last Egress Identifier identifies the MEP
            Linktrace Initiator that originated, or the Linktrace
            Responder that forwarded, the LTM to which this LTR
```



```
        is the response. This is the same value as the
        Egress Identifier TLV of that LTM.";
    reference
        "802.1ag clauses 12.14.7.5.3:e and 20.36.2.3";
}

leaf dot1agCfmLtrNextEgressIdentifier {
    type binary {
        length "8";
    }
    config false;
    description
        "An octet field holding the Next Egress Identifier
        returned in the LTR Egress Identifier TLV of the LTR.
        The Next Egress Identifier Identifies the Linktrace
        Responder that transmitted this LTR, and can forward
        the LTM to the next hop. This is the same value as
        the Egress Identifier TLV of the forwarded LTM, if
        any. If the FwdYes bit of the Flags field is false,
        the contents of this field are undefined,
        i.e., any value can be transmitted, and the field
        is ignored by the receiver.";
    reference
        "802.1ag clauses 12.14.7.5.3:f and 20.36.2.4";
}

leaf dot1agCfmLtrRelay {
    type cfm:Dot1agCfmRelayActionFieldValue;
    config false;
    description
        "Value returned in the Relay Action field.";
    reference
        "802.1ag clauses 12.14.7.5.3:g and 20.36.2.5";
}

leaf dot1agCfmLtrChassisIdSubtype {
    type cfm:LldpChassisIdSubtype;
    config false;
    description
        "This object specifies the format of the Chassis ID
        returned in the Sender ID TLV of the LTR, if any.
        This value is meaningless if the
        dot1agCfmLtrChassisId has a length of 0.";
    reference
        "802.1ag clauses 12.14.7.5.3:h and 21.5.3.2";
}
```



```
leaf dot1agCfmLtrChassisId {
  type cfm:LldpChassisId;
  config false;
  description
    "The Chassis ID returned in the Sender ID TLV of the
    LTR, if any. The format of this object is determined
    by the value of the dot1agCfmLtrChassisIdSubtype
    object.";
  reference
    "802.1ag clauses 12.14.7.5.3:i and 21.5.3.3";
}

leaf dot1agCfmLtrManAddressDomain {
  type cfm:TDomain;
  config false;
  description
    "The TDomain that identifies the type and format of
    the related dot1agCfmMepDbManAddress object, used to
    access the SNMP agent of the system transmitting the
    LTR. Received in the LTR Sender ID TLV from that
    system.

    Typical values will be one of (not all inclusive)
    list:

        snmpUDPDomain          (from SNMPv2-TM, RFC3417)
        snmpIeee802Domain      (from SNMP-IEEE802-TM-MIB,
                               RFC4789)

    The value 'zeroDotZero' (from RFC2578) indicates
    'no management address was present in the LTR',
    in which case the related object
    dot1agCfmMepDbManAddress must have a zero-length
    OCTET STRING as a value.";
  reference
    "802.1ag clauses 12.14.7.5.3:j, 21.5.3.5, 21.9.6";
}

leaf dot1agCfmLtrManAddress {
  type cfm:TAddress;
  config false;
  description
    "The TAddress that can be used to access the SNMP
    agent of the system transmitting the CCM, received
    in the CCM Sender ID TLV from that system."
```



```
        If the related object dot1agCfmLtrManAddressDomain
        contains the value 'zeroDotZero', this object
        dot1agCfmLtrManAddress must have a zero-length
        OCTET STRING as a value.";
    reference
        "802.1ag clauses 12.14.7.5.3:j, 21.5.3.7, 21.9.6";
}

leaf dot1agCfmLtrIngress {
    type cfm:Dot1agCfmIngressActionFieldValue;
    config false;
    description
        "The value returned in the Ingress Action Field of
        the LTM. The value ingNoTlv(0) indicates that no
        Reply Ingress TLV was returned in the LTM.";
    reference
        "802.1ag clauses 12.14.7.5.3:k and 20.36.2.6";
}

leaf dot1agCfmLtrIngressMac {
    type yang:mac-address;
    config false;
    description
        "MAC address returned in the ingress MAC address field.
        If the dot1agCfmLtrIngress object contains the value
        ingNoTlv(0), then the contents of this object are
        meaningless.";
    reference
        "802.1ag clauses 12.14.7.5.3:l and 20.36.2.7";
}

leaf dot1agCfmLtrIngressPortIdSubtype {
    type cfm:LldpPortIdSubtype;
    config false;
    description
        "Format of the Ingress Port ID.
        If the dot1agCfmLtrIngress object contains the value
        ingNoTlv(0), then the contents of this object are
        meaningless.";
    reference
        "802.1ag clauses 12.14.7.5.3:m and 20.36.2.8";
}

leaf dot1agCfmLtrIngressPortId {
    type cfm:LldpPortId;
    config false;
    description
```



```
        "Ingress Port ID. The format of this object is
        determined by the value of the
        dot1agCfmLtrIngressPortIdSubtype object. If the
        dot1agCfmLtrIngress object contains the value
        ingNoTlv(0), then the contents of this object
        are meaningless.";
    reference
        "802.1ag clauses 12.14.7.5.3:n and 20.36.2.9";
}

leaf dot1agCfmLtrEgress {
    type cfm:Dot1agCfmEgressActionFieldValue;
    config false;
    description
        "The value returned in the Egress Action Field of the
        LTM. The value egrNoTlv(0) indicates that no Reply
        Egress TLV was returned in the LTM.";
    reference
        "802.1ag clauses 12.14.7.5.3:o and 20.36.2.10";
}

leaf dot1agCfmLtrEgressMac {
    type yang:mac-address;
    config false;
    description
        "MAC address returned in the egress MAC address field.
        If the dot1agCfmLtrEgress object contains the value
        egrNoTlv(0), then the contents of this object are
        meaningless.";
    reference
        "802.1ag clauses 12.14.7.5.3:p and 20.36.2.11";
}

leaf dot1agCfmLtrEgressPortIdSubtype {
    type cfm:LldpPortIdSubtype;
    config false;
    description
        "Format of the egress Port ID.
        If the dot1agCfmLtrEgress object contains the value
        egrNoTlv(0), then the contents of this object are
        meaningless.";
    reference
        "802.1ag clauses 12.14.7.5.3:q and 20.36.2.12";
}

leaf dot1agCfmLtrEgressPortId {
    type cfm:LldpPortId;
```



```
    config false;
    description
      "Egress Port ID. The format of this object is
       determined by the value of the
       dot1agCfmLtrEgressPortIdSubtype object.If the
       dot1agCfmLtrEgress object contains the value
       egrNoTlv(0), then the contents of this object are
       meaningless.";
    reference
      "802.1ag clauses 12.14.7.5.3:r and 20.36.2.13";
  }

  leaf dot1agCfmLtrOrganizationSpecificTlv {
    type binary {
      length "0 | 4..1500";
    }
    config false;
    description
      "All Organization specific TLVs returned in the LTR, if
       any. Includes all octets including and following
       the TLV Length field of each TLV, concatenated
       together.";
    reference
      "802.1ag clauses 12.14.7.5.3:s, 21.5.2";
  }
}

list dot1agCfmMepDbEntry {

  key "dot1agCfmMdIndex dot1agCfmMaIndex
    dot1agCfmMepIdentifier
    dot1agCfmMepDbRMepIdentifier";
  description
    "The MEP Database table entry.";

  leaf dot1agCfmMdIndex {
    type leafref {
      path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"
        +":dot1agCfmMdIndex";
    }
    description
      "Automagically generated leafref leaf.";
  }
  leaf dot1agCfmMaIndex {
    type leafref {
      path "/cfm:dot1agCfmMa/cfm:dot1agCfmMaNetEntry/cfm"
```



```
        +":dot1agCfmMaIndex";
    }
    description
        "Automagically generated leafref leaf.";
}
leaf dot1agCfmMepIdentifier {
    type leafref {
        path "/cfm:dot1agCfmMep/cfm:dot1agCfmMepEntry/cfm"
            +":dot1agCfmMepIdentifier";
    }
    description
        "Automagically generated leafref leaf.";
}

leaf dot1agCfmMepDbRMepIdentifier {
    type cfm:Dot1agCfmMepId;
    description
        "Maintenance association End Point Identifier of a
        remote MEP whose information from the MEP Database
        is to be returned.";
    reference
        "802.1ag clause 12.14.7.6.2:b";
}

leaf dot1agCfmMepDbRMepState {
    type cfm:Dot1agCfmRemoteMepState;
    config false;
    description
        "The operational state of the remote MEP IFF State
        machines.";
    reference
        "802.1ag clause 12.14.7.6.3:b and 20.22";
}

leaf dot1agCfmMepDbRMepFailedOkTime {
    type yang:timestamp;
    config false;
    description
        "The time (SysUpTime) at which the IFF Remote MEP
        state machine last entered either the RMEP_FAILED
        or RMEP_OK state.";
    reference
        "802.1ag clause 12.14.7.6.3:c";
}

leaf dot1agCfmMepDbMacAddress {
    type yang:mac-address;
```



```
    config false;
    description
      "The MAC address of the remote MEP.";
    reference
      "802.1ag clause 12.14.7.6.3:d and 20.19.7";
  }

  leaf dot1agCfmMepDbRdi {
    type boolean;
    config false;
    description
      "State of the RDI bit in the last received CCM
      (true for RDI=1), or false if none has been
      received.";
    reference
      "802.1ag clauses 12.14.7.6.3:e and 20.19.2";
  }

  leaf dot1agCfmMepDbPortStatusTlv {
    type cfm:Dot1agCfmPortStatus;
    config false;
    description
      "An enumerated value of the Port status TLV received
      in the last CCM from the remote MEP or the default
      value psNoPortStateTlv indicating either no CCM has
      been received, or that no port status TLV was
      received in the last CCM.";
    reference
      "802.1ag clause 12.14.7.6.3:f and 20.19.3";
  }

  leaf dot1agCfmMepDbInterfaceStatusTlv {
    type cfm:Dot1agCfmInterfaceStatus;
    config false;
    description
      "An enumerated value of the Interface status TLV
      received in the last CCM from the remote MEP or
      the default value isNoInterfaceStatus TLV indicating
      either no CCM has been received, or that no interface
      status TLV was received in the last CCM.";
    reference
      "802.1ag clause 12.14.7.6.3:g and 20.19.4";
  }

  leaf dot1agCfmMepDbChassisIdSubtype {
    type cfm:LldpChassisIdSubtype;
    config false;
```



```
    description
      "This object specifies the format of the Chassis ID
      received in the last CCM.";
    reference
      "802.1ag clauses 12.14.7.6.3:h and 21.5.3.2";
  }

  leaf dot1agCfmMepDbChassisId {
    type cfm:LldpChassisId;
    config false;
    description
      "The Chassis ID. The format of this object is
      determined by the value of the
      dot1agCfmLtrChassisIdSubtype object.";
    reference
      "802.1ag clauses 12.14.7.6.3:h and 21.5.3.3";
  }

  leaf dot1agCfmMepDbManAddressDomain {
    type cfm:TDomain;
    config false;
    description
      "The TDomain that identifies the type and format of
      the related dot1agCfmMepDbManAddress object, used to
      access the SNMP agent of the system transmitting the
      CCM. Received in the CCM Sender ID TLV from that
      system.

      Typical values will be one of (not all inclusive)
      list:

          snmpUDPDomain          (from SNMPv2-TM, RFC3417)
          snmpIeee802Domain      (from SNMP-IEEE802-TM-MIB,
                                RFC4789)

      The value 'zeroDotZero' (from RFC2578) indicates
      'no management address was present in the LTR',
      in which case the related object
      dot1agCfmMepDbManAddress must have a zero-length OCTET
      STRING as a value.";
    reference
      "802.1ag clauses 12.14.7.6.3:h, 21.5.3.5, 21.6.7";
  }

  leaf dot1agCfmMepDbManAddress {
    type cfm:TAddress;
```



```
    config false;
    description
      "The TAddress that can be used to access the SNMP
       agent of the system transmitting the CCM, received
       in the CCM Sender ID TLV from that system.

       If the related object dot1agCfmMepDbManAddressDomaini
       contains the value 'zeroDotZero', this object
       dot1agCfmMepDbManAddress must have a zero-length
       OCTET STRING as a value.";
    reference
      "802.1ag clauses 12.14.7.6.3:h, 21.5.3.7, 21.6.7";
  }
}
}
```

notification dot1agCfmFaultAlarm {

description

"A MEP has a persistent defect condition. A notification (fault alarm) is sent to the management entity with the OID of the MEP that has detected the fault.

Whenever a MEP has a persistent defect, it may or may not generate a Fault Alarm to warn the system administrator of the problem, as controlled by the MEP Fault Notification Generator State Machine and associated Managed Objects. Only the highest-priority defect, as shown in Table 20-1, is reported in the Fault Alarm.

If a defect with a higher priority is raised after a Fault Alarm has been issued, another Fault Alarm is issued.

The management entity receiving the notification can identify the system from the network source address of the notification, and can identify the MEP reporting the defect by the indices in the OID of the dot1agCfmMepHighestPrDefect variable in the notification:

dot1agCfmMdIndex - Also the index of the MEP's Maintenance Domain table entry (dot1agCfmMdTable).

dot1agCfmMaIndex - Also an index (with the MD table index) of the MEP's Maintenance Association network table entry (dot1agCfmMaNetTable), and (with the MD table index and component ID) of the MEP's MA component table entry


```

        (dot1agCfmMaCompTable).
    dot1agCfmMepIdentifier - MEP Identifier and final index
                           into the MEP table
                           (dot1agCfmMepTable).";
reference
    "802.1ag clause 12.14.7.7";

container dot1agCfmFaultAlarm-dot1agCfmMepHighestPrDefect {
    description
        "The highest priority defect that has been present since the
        MEPs Fault Notification Generator State Machine was last in
        the FNG_RESET state";

    leaf dot1agCfmMdIndex {
        type leafref {
            path "/cfm:dot1agCfmMd/cfm:dot1agCfmMdEntry/cfm"
                +":dot1agCfmMdIndex";
        }
        description
            "Automagically generated leafref leaf.";
    }
    leaf dot1agCfmMaIndex {
        type leafref {
            path "/cfm:dot1agCfmMa/cfm:dot1agCfmMaNetEntry/cfm"
                +":dot1agCfmMaIndex";
        }
        description
            "Automagically generated leafref leaf.";
    }
    leaf dot1agCfmMepIdentifier {
        type leafref {
            path "/cfm:dot1agCfmMep/cfm:dot1agCfmMepEntry/cfm"
                +":dot1agCfmMepIdentifier";
        }
        description
            "Automagically generated leafref leaf.";
    }
    leaf dot1agCfmMepHighestPrDefect {
        type cfm:Dot1agCfmHighestDefectPri;
        description
            "The highest priority defect that has been present
            since the MEPs Fault Notification Generator State
            Machine was last in the FNG_RESET state.";
        reference
            "802.1ag clause 12.14.7.1.3:n 20.33.9 and Table 21-1";
    }
}
```



```
    }  
  } /* end of module ietf-cfm */  
<CODE ENDS>
```

4. Security Considerations

The data model defined does not create any security implications.

5. IANA Considerations

This draft does not request any IANA action.

6. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC2578] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., Waldbusser, S., "Structure of Management Information Version 2 (SMIv2)", STD 58, [RFC 2578](#), April 1999.
- [RFC2579] McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, [RFC 2579](#), April 1999. Done.
- [RFC4789] Schoenwaelder, J., Jeffree, T., "Simple Network Management Protocol (SNMP) over IEEE 802 Networks", [RFC 4789](#), November 2006.
- [RFC3417] Presuhn, R, Case, J., McCloghrie, K., Rose, M., S. Waldbusser, "Transport Mappings for the Simple Network Management Protocol (SNMP)", [RFC 3417](#), December 2002.
- [RFC1903] Case, J., McCloghrie, K., Rose, M., S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", [RFC 1903](#), January 1996.

- [RFC2685] Fox, B., Bleeson, B. "Virtual Private Networks Identifier", [RFC 2685](#), September 1999.
- [RFC1906] Case, J., McCloghrie, K., Rose, M., S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", [RFC 1906](#), January 1996.
- [RFC2863] McCloghrie, K., Kastenholz, F., "The Interfaces Group MIB", [RFC 2863](#), June 2000.
- [RFC2737] McCloghrie, K., Bierman, A., "The Interfaces Group MIB", [RFC 2737](#), December 1999.
- [RFC3046] Patrick, M., " DHCP Relay Agent Information Option", [RFC 3046](#), January 2001.
- [IEEE802] "IEEE Standard for Local Area Networks: Overview and Architecture", IEEE Std. 802-2001.
- [IEEE802.1ag] "Virtual Bridged Local Area Networks - Amendment 5:Connectivity Fault Management", IEEE Std. June, 2007.

Authors' Addresses

Kun Wang
Ericsson
Ericsson Tower 2, No.5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China

Email kun.s.wang@ericsson.com

Alex Wang
Ericsson
Ericsson Tower 2, No.5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China

Email alex.g.wang@ericsson.com

Chin Chen
Ericsson
Ericsson Tower 2, No.5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China

Email chin.chen@ericsson.com

Hua Lv
Ericsson
Ericsson Tower 2, No.5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China

Email hua.lv@ericsson.com