Question(s):	**9**, 10, 12, 14	**Meeting, date:**	Beijing, September 5 – 9, 2011
Study Group:	15 **Working Party:** 3	**Intended type of document**(R-C-TD) :	WD45r14
Source:	Editors G.8121		
Title:	Draft new G.8121.2		

Contact:	Yuji Tochio Fujitsu Japan	Tel: +81-44-754-8829 Fax: +81-44-754-2741 Email: tochio@jp.fujitsu.com
Contact:	Huub van Helvoort Huawei Technologies Co., Ltd. P.R.China	Tel: +31 649248936 Fax: Email: hhelvoort@huawei.com

Please don’t change the structure of this table, just insert the necessary information.

Abstract

This document contains the consolidation of all WDs to new G.8121.2 as proposed in WD04.

Drafting status

Clause	Title (as proposed in WD04)	Input WDs, note, and updates
1	**Scope**	
2	**References**	
3	**Definitions**	
4	**Abbreviations**	WD24
5	**Conventions**	
6	**Supervision**	WD25→ updated to refer G.8121
7	**Information flow across reference points**	
8	**MPLS-TP processes**	
8.1	**G-ACh Process**	
8.2	**TC/Label processes**	
8.3	**Queuing process**	
8.4	**MPLS-TP-specific GFP-F processes**	
8.5	**Control Word (CW) processes**	updated to refer G.8121 from 8.1 to 8.5 and 8.7
8.6	**OAM related Processes used by Server adaptation functions**	
8.6.1	Selector Process	WD26 (Note:The Proposed structure differs from WD04 and seems to be based on current G.8121 (TD458/3))
8.6.2	Fault Management Insert Process	
8.7	**OAM related Processes used by adaptation functions**	
8.7.1	MCC/SCC Insert Process	WD26→WD45r1 mapping defined as:
8.7.2	MCC/SCC Extract Process	8.1.1 to 8.1
8.7.3	APS Insert Process	8.1.2 to 8.8.1
8.7.4	APS Extract Process	8.1.3 to 8.6.2
8.7.5	CSF Insert Process	8.1.4 to 8.8.2
8.7.6	CSF Extract Process	8.1.5 to new clause 8.8.x
8.8	**Pro-active and on-demand OAM related Processes**	8.1.6 remove (or FFS)
8.8.1	CC-CV-RDI Process	
8.8.2	On-demand CV Process	
8.8.3	Loss Measurement (LM) Process	
8.8.4	Delay Measurement (DM) Process	
8.8.5	One Way Delay Measurement (1DM) Process	
8.8.6	Test (TST) Process	
9	**MPLS-TP layer functions**	WD27 + some clauses updated to red G.8121
10	**MPLS-TP to Non-MPLS-TP client adaptation functions**	→ updated to refer G.8121
11	**Non-MPLS-TP Server to MPLS-TP adaptation functions**	→ updated to refer G.8121

Formatted: Font: (Asian) MS Mincho

Formatted: Highlight

Formatted: Font: 10 pt, Highlight

Formatted: Font: 10 pt, Highlight

Formatted: Font: 10 pt, Highlight

Formatted: Font: 10 pt, Highlight

Formatted: Font: 10 pt

Formatted: Font: (Asian) MS Mincho, (Asi
Japanese

G.8121.2/Y.1382.2

Characteristics of MPLS-TP equipment functional blocks supporting G.8113.2/Y.1373.2

Summary

<Mandatory material>

Keywords

<Optional>

1 Scope

This recommendation describes…

2 References

The following ITU-T Recommendations and other references contain provisions, which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

3 Definitions

<Check in the ITU-T Terms and definitions database on the public website whether the term is already defined in another Recommendation. It may be more consistent to refer to such a definition rather than redefine it>

3.1 Terms defined elsewhere:

This Recommendation uses the following terms defined elsewhere:

3.1.1 <Term 1> [Reference]: <optional quoted definition>

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 <Term 3>: <definition>

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

BFD	Bidirectional Forwarding Detection
CC	Continuity Check

CV	Continuity Verification
DLM	Direct Loss Measurement
DM	Delay Measurement
DSMap	Downstream Mapping
FEC	Forwarding Equivalence Class
GAL	Generic Associated Channel Label
G-ACh	Generic Associated Channel
ILM	Inferred Loss Measurement
LBI	Loopback Instruct
LDI	Local Down Indication
LKI	Lock Instruct
LKR	Lock Report
TC	Traffic Class
VCCV	Virtual Circuit Connectivity Verification

5 Conventions

6 Supervision

6.1 Defects

6.1.1 Summary of Entry/Exit conditions for defects

The defect Entry and Exit conditions are based on events. Occurrence or absence of specific events may raise or reset specific defects.

The events used by this recommendation are defined in Table 6-1/G.8121.

[Ed Note – check that all the events in G.8121 are used or no missing events]

7 Information flow across reference points

8 MPLS-TP processes

8.1 G-ACh Process

See the clause 8.1in [ITU-T G.8121]

8.1.1 OAM Insertion and Extraction Processes

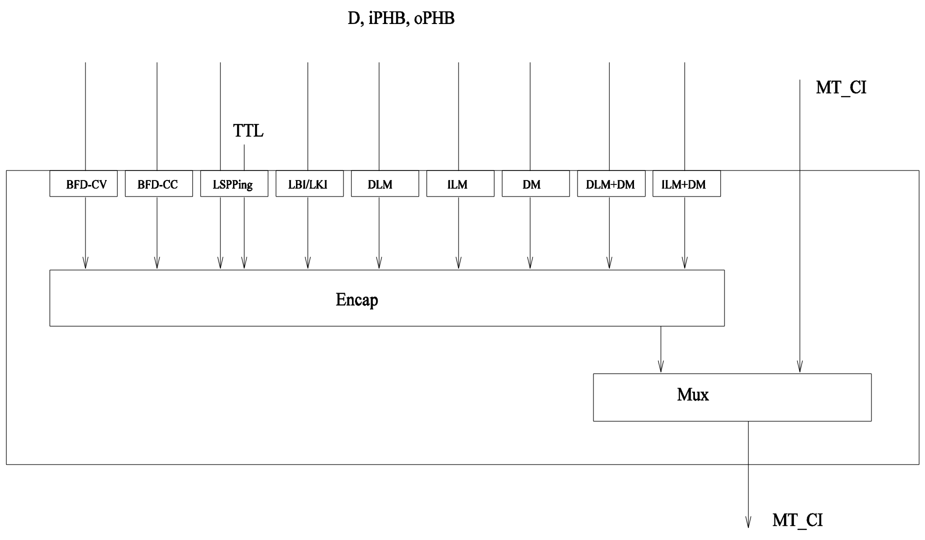
The OAM Insertion and Extraction processes insert OAM packets into the data stream, or extract OAM packets from it, respectively. OAM packets are identified by a GAL and G-ACh header as described in [RFC5586].

NOTE: In the case of PW, the GAL is not used (see RFC5586 Section 4.2 GAL Applicability and Usage). Encapsulation of OAM packets using IP/UDP, PW VCCV or other mechanisms are FFS.

There are OAM Insertion Processes in the MEP and MIP termination functions (MT_TT and MTDi_TT), and in the adaptation functions (MT/MT_A) – these differ in that the adaptation function version also handles selection of packets based on the MI_Admin_State.

The OAM Extraction Process only exists in the termination functions, but there are some differences in behaviour between the MEP and MIP as described below.

8.1.1.1 Termination Function OAM Insertion Process



The figure below shows the Termination Function OAM Insertion Process

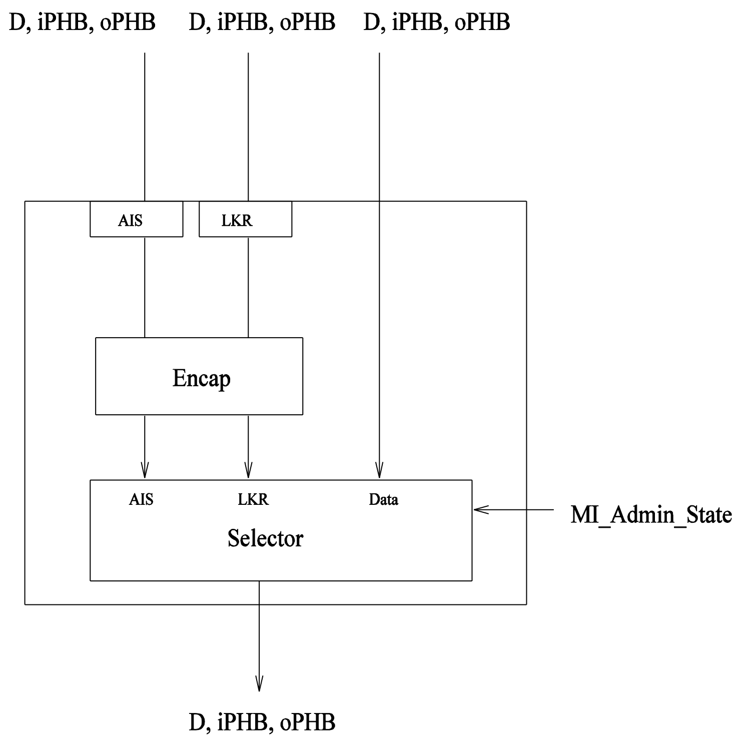
The Termination Function OAM insertion process encapsulates OAM packets and multiplexes received data packets with inserted OAM packets. The data packets are passed through unchanged, while the OAM packets are encapsulated as follows:

AG-ACh header is prepended to the OAM PDU, with the Channel Type field set to the appropriate value depending on the protocol port, as described in [IANA "PW Associated Channel Type" registry]. The function then further prepends a G-ACh Label (GAL) as described in [RFC5586]. If no TTL signal is received, the TTL field in the GAL is set to 255; otherwise it is set to the value in the TTL signal.

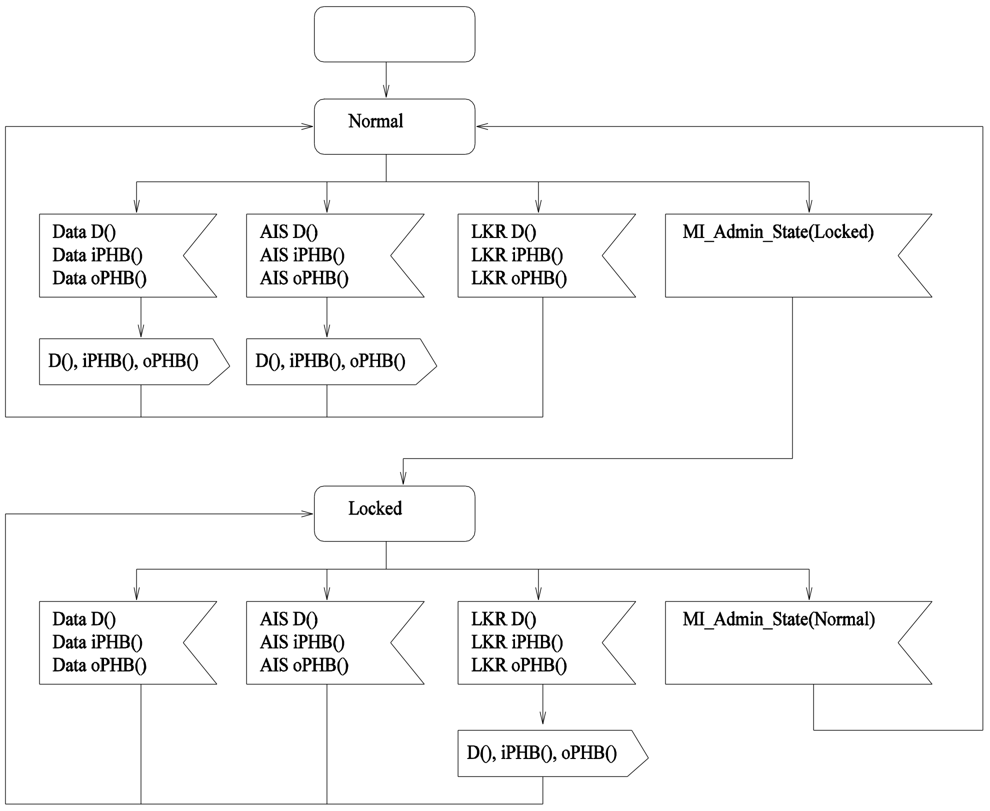
Encapsulation of OAM packets using IP/UDP, PW VCCV or other mechanisms are FFS.

8.1.1.2Adaptation Function OAM Insertion Process

The figure below shows the Adaptation Function OAM Insertion Process.

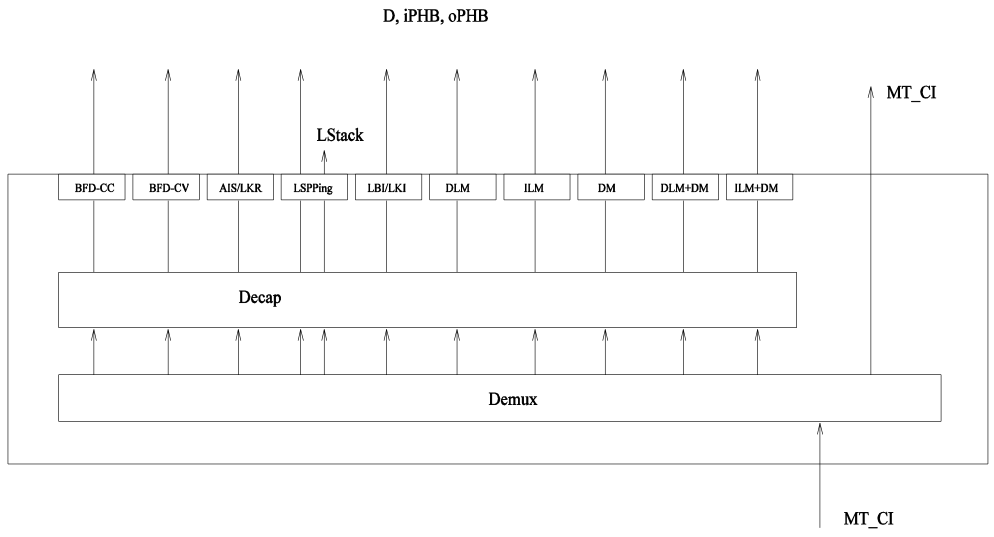


The Adaptation Function OAM Insertion process encapsulates OAM packets and inserts them into the data stream. The Encap block behaves as for the Termination Function OAM Insertion process described above. The behaviour of the Selector block is shown on the figure below – this passes normal traffic and AIS messages when MI_Admin_State is unlocked, and passes the LKR messages when it is locked.



8.1.1.3 OAM Extraction Processes

The figure below shows the OAM Extraction Processes:



The OAM extraction process extracts packets containing OAM information from the data stream. Other packets are passed through unmodified. The Demux stage identifies OAM packets, and differs between the MEP OAM Extraction process and the MIP OAM Extraction process. The Decap stage removes the GAL and G-ACh headers from the OAM PDU.

In the MEP OAM Extraction Process, the Demuxstage processes the packet according to the following pseudocode:

```
if (MT-label(D) == GAL) {
switch (ChannelType(G-ACh(D))) {
case <BFD-CC>: Extract BFD-CC traffic unit and forward to BFD-CC port
case <BFD-CV>: Extract BFD-CV traffic unit and forward to BFD-CV port
case <AIS/LKR>: Extract AIS/LKR traffic unit and forward to AIS/LKR port
case <LSP-ping>: Extract LSP-ping traffic unit and forward to LSP-ping port
case <LBI/LKI>: Extract LBI/LKI traffic unit and forward to LBI/LKI port
    case <DLM>: Extract DLM traffic unit and forward to DLM port
    case <ILM>: Extract ILM traffic unit and forward to ILM port
    case <DM>: Extract DM traffic unit and forward to DM port
    case <DLM+DM>: Extract DLM+DM traffic unit and forward to DLM+DM port
    case <ILM+DM>: Extract ILM+DM traffic unit and forward to ILM+DM port
    default: forward to data port
}
} else {
    forward to data port
}
```

In the MIP OAM Extraction Process, the Demux stage processes the packet according to the following pseudocode:

```
if (Ttl(MT-label(D)) == 0) {
    if (MT-label(D) == GAL) {
switch (ChannelType(G-ACh(D))) {
        case <LSP-ping>: Extract LSP-ping traffic unit and forward
 to LSP-ping port
   case <LBI/LKI>: Extract LBI/LKI traffic unit and forward to LBI/LKI port
   default: drop the packet
        }
    } else {
        drop the packet
    }
} else {
    forward to data port
}
```

Decapsulation and extraction of OAM packets using IP/UDP, PW VCCV or other mechanisms are FFS. Processing of performance-measurement packets at MIPs is also FFS.

8.2 TC/Label processes

See the clause 8.2in [ITU-T G.8121]

8.3 Queuing process

See the clause 8.3in [ITU-T G.8121]

8.4 MPLS-TP-specific GFP-F processes

See the clause 8.4in [ITU-T G.8121]

8.5 Control Word (CW) processes

See the clause 8.5in [ITU-T G.8121]

8.6 OAM related Processes used by Server adaptation functions

8.6.1 Selector Process

See the clause 8.6.1[ITU-T G.8121]

8.6.2 Fault Management Insert Process

[note: check if it is possible to align with G.8121 structure(splitting into AIS and LCK)]

The LKR generation, AIS Insertion and LKR/AIS Reception processes are described below.

8.7 OAM related Processes used by adaptation functions

8.7.1 MCC/SCC Mapping Insert and De-mapping Process

See the clause 8.7.1in [ITU-T G.8121]

8.7.2 APS Insert and ExtractProcess

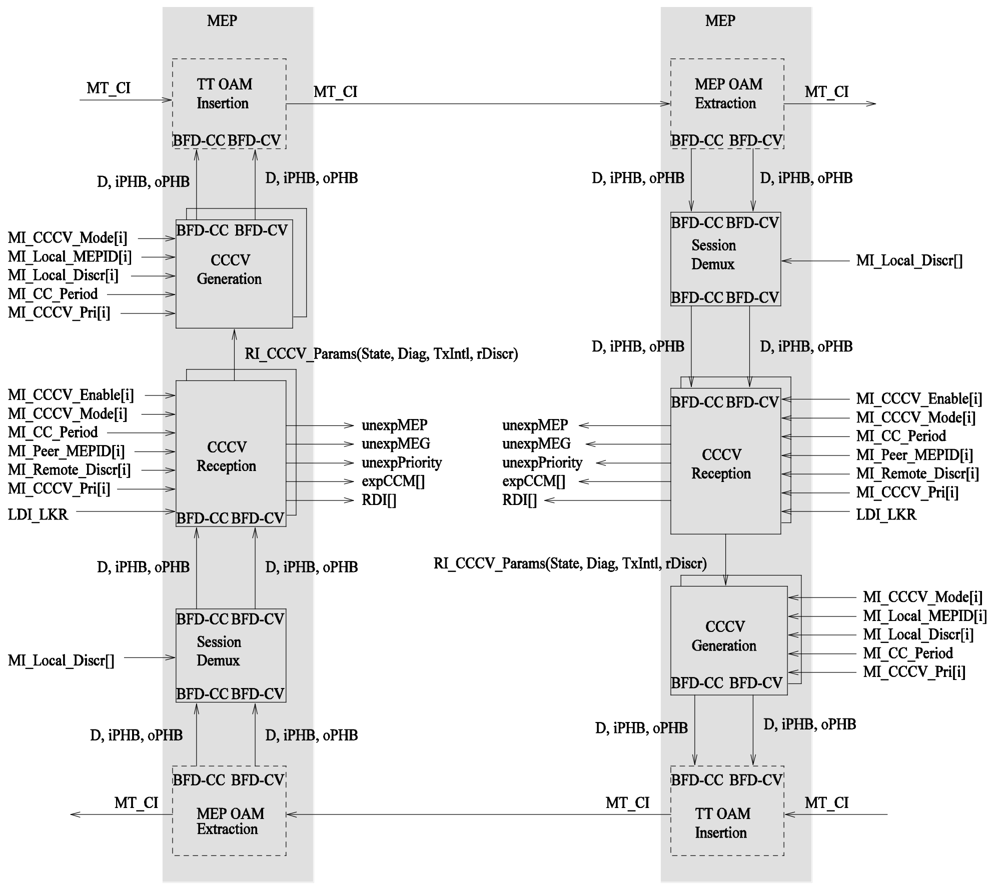
See the clause 8.7.2 in [ITU-T G.8121]

8.7.3 CSF Insert and Extract Process

See the clause 8.7.3 in [ITU-T G.8121]

8.8 Pro-active and on-demand OAM related Processes

8.8.1 CC/CV Processes



An overview of the CC/CV processes is shown in the figure below:

The CCCV reception process controls the operation of the CCCV protocol. It operates when MI_CCCV_Enable is TRUE, according to the value of MI_CCCV_Mode. MI_CCCV_Mode takes one of the following values:

Comment [DB1]: Diagram updated

- COORD – Co-ordinated mode; operate a single co-ordinated BFD session
- SRC – Independent Source; operate as the source MEP in an independent BFD session
- SINK – Independent Sink; operate as the sink MEP in an independent BFD session

Multiple instances of the CCCV reception process may be created for multiple BFD sessions; when operating in independent mode, it is expected that a pair of instances are created, one acting as the source and one as the sink.

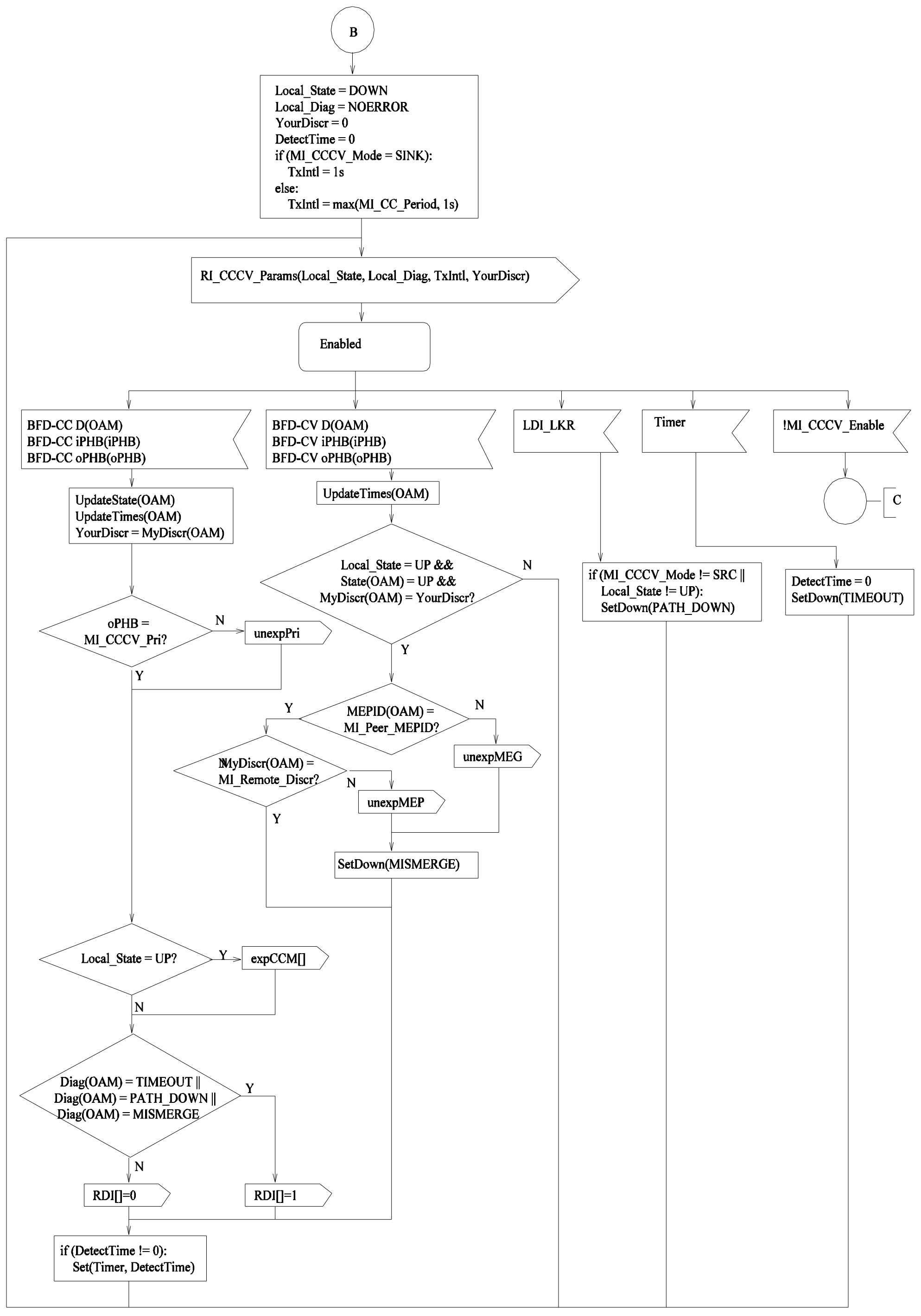
MI_CC_Period specifies the desired period between successive BFD-CC messages, and MI_Peer_MEPID specifies the MEP ID value to expect in received messages, in one of the formats described in [draft-ietf-mpls-tp-cc-cv-rdi].

The CCCV generation process sends periodic BFD-CC and BFD-CV messages, when MI_CCCV_Enable is TRUE. There is a separate instance of the process for each corresponding instance of the CCCV reception process. MI_Local_MEPID and MI_Local_Discr specify the local MEP ID and session discriminator values to send in the packets.

The Session Demux process demultiplexes received BFD-CC and BFD-CV messages to the correct instance of the CCCV reception process, based on the “Your discriminator” field in the received BFD-CC or BFD-CV packet.Demultiplexing of received packets where the “Your discriminator” field is 0 is FFS.

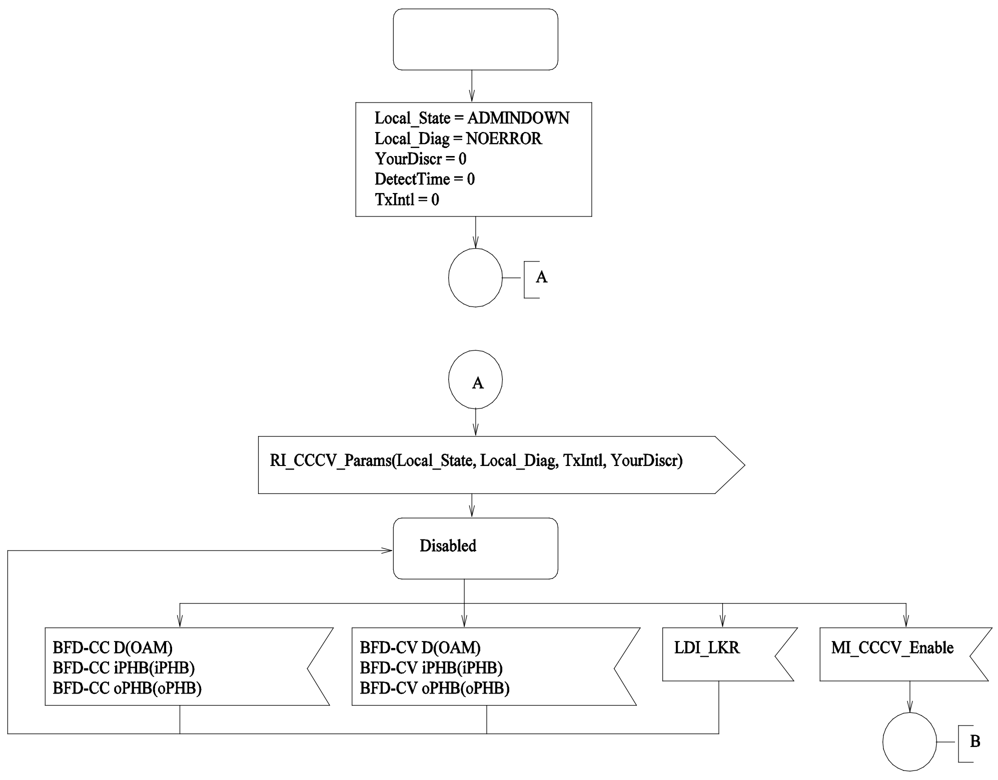
8.8.1.1 CCCV Reception Process

The CCCV Reception Process controls the operation of the BFD protocol, according to



MI_CCCV_Enable and MI_CCCV_Mode. Multiple instances of the CCCV Reception Process can be instantiated. Each one has a corresponding instance of the CCCV Generation Process; the contents and period for sending CCCV packets are controlled via the RI_CCCV_Params() signal.

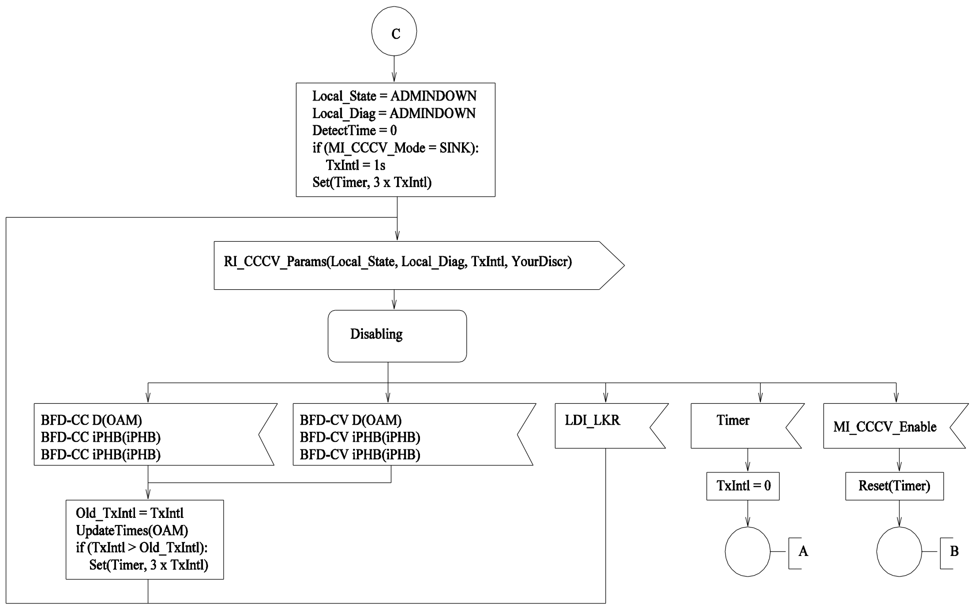
The CCCV Reception Process is described in the following figures. In Disabled state, all received BFD-CC and BFD-CV packets are discarded and no packets are sent. In Enabled state, received BFD-CC packets are processed, and received BFD-CV packets are processed when the BFD state machine is UP. BFD-CC and BFD-CV packets are sent, except if the process is operating in SINK mode. When MI_CCCV_Enabled is set to FALSE, the process moves to Disabling state so that the ADMIN_DOWN diagnostic code can be signalled to the peer MEP. The process stays in Disabling state for three times the transmit interval, before moving to Disabled state. In Disabling state, BFD-CC packets are sent, but received BFD-CC and BFD-CV packets are used only for updating the



timer.

Note: unexpPriority does not appear in draft-ietf-cc-cv-rdi, or in G.8121 (this event is FFS)

Comment [DB2]: Diagram updated.



The values of State and Diag correspond with those in [RFC5880] and [draft-ietf-mpls-tp-cc-cv-rdi].

The functions ‘SetDown’, ‘UpdateState’ and ‘UpdateTimes’ are described by the following pseudocode:

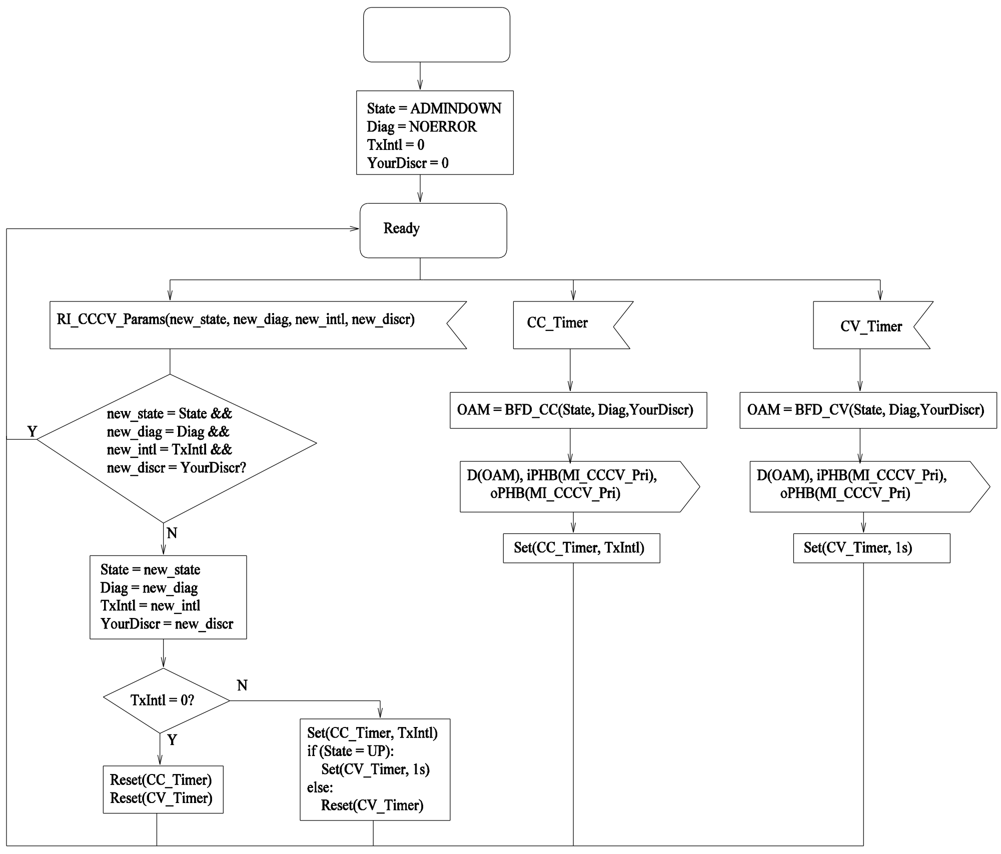
```
SetDown(new_diag) {
    if (Local_State != DOWN) {
Local_State = DOWN
        if (Local_Diag != PATH_DOWN || new_diag != TIMEOUT) {
Local_Diag = new_diag
        }
        if (MI_CCCV_Mode = SINK) {
TxIntl = 1s
        }
    }
}

UpdateState(OAM) {
    if (State(OAM) = ADMINDOWN) {
SetDown(NBR_DOWN)
} else {
        if (Local_State = DOWN) {
         if (State(OAM) = DOWN) {
Local_State = INIT
Local_Diag = NOERROR
} else if (State(OAM) = INIT ||
       (MI_CCCV_Mode = SINK && State(OAM) = UP)) {
Local_State = UP
Local_Diag = NOERROR
            }
} else if (Local_State = INIT) {
            if (State(OAM) = INIT || State(OAM) = UP) {
Local_State = UP
Local_Diag = NOERROR
            }
} else {
            // Local_State must be UP
            if (state(OAM) = DOWN &&MI_CCCV_Mode != SRC) {
SetDown(NBR_DOWN)
            }
        }
    }
}

UpdateTimes(OAM) {
    if (MI_CCCV_Mode = SRC) {
DetectTime = 0
} else {
DetectTime = 3 x max(MI_CC_Period, DesiredMinTxInterval(OAM))
    }
    if (MI_CCCV_Mode = SINK) {
  if (State(OAM) != LocalState) {
TxIntl = 1s
} else {
TxIntl = 0
        }
} else {
TxIntl = max(MI_CC_Period, RequiredMinRxInterval(OAM))
    }
}
```

Use of authentication for CC/CV is FFS.

Use of the BFD Poll/Final mechanism for changing the value of TxIntlis FFS.

8.8.1.2 CCCV Generation Process

The CCCV Generation Process is responsible for generating BFD-CC and BFD-CV packets, according to the parameters set by the corresponding CCCV Reception Process in the RI_CCCV_Params(state, diag, TX-interval,your-discriminator) signal. When the TX-interval is set to 0, no BFD-CC or BFD-CV packets are generated. Otherwise, BFD-CC packets are generated at the specified interval, and BFD-CV packets are generated if the state is up, at an interval of 1s.



The CCCV Generation Process is described in the following figure:

The BFD_CC function creates a BFD control packet according to the format described in [RFC5880]. The fields are filled in as follows:

- Vers: set to 1
- Diag: set to the value of Diag
- Sta: set to the value of State
- P, F, A, D, M flags: set to 0
- C flag: set appropriately dependent on the implementation
- Detect Mult: set to 3

- Length: set to 24
- My Discriminator: set to MI_Local_Discr
- Your Discriminator: set to YourDiscr
- Desired Min Tx Interval: set to 0 if MI_CCCV_Mode is SINK, otherwise set to MI_CC_Period
- Required Min Rx Interval: set to 0 if MI_CCCV_Mode is SRC, otherwise set to MI_CC_Period
- Required Min Echo Rx Interval: set to 0

No Authentication Section is added. Use of Authentication is FFS.

The BFD_CV function creates a BFD control packet in the same way as the BFD_CC function, and then appends a MEP Source ID TLV as described in [draft-ietf-mpls-tp-cc-cv-rdi], containing the value of MI_Local_MEPID.

8.8.1.3 Session Demux Process

The session demux process receives BFD-CC and BFD-CV packets from the MEP OAM Extraction process. It performs the following checks on the packet:

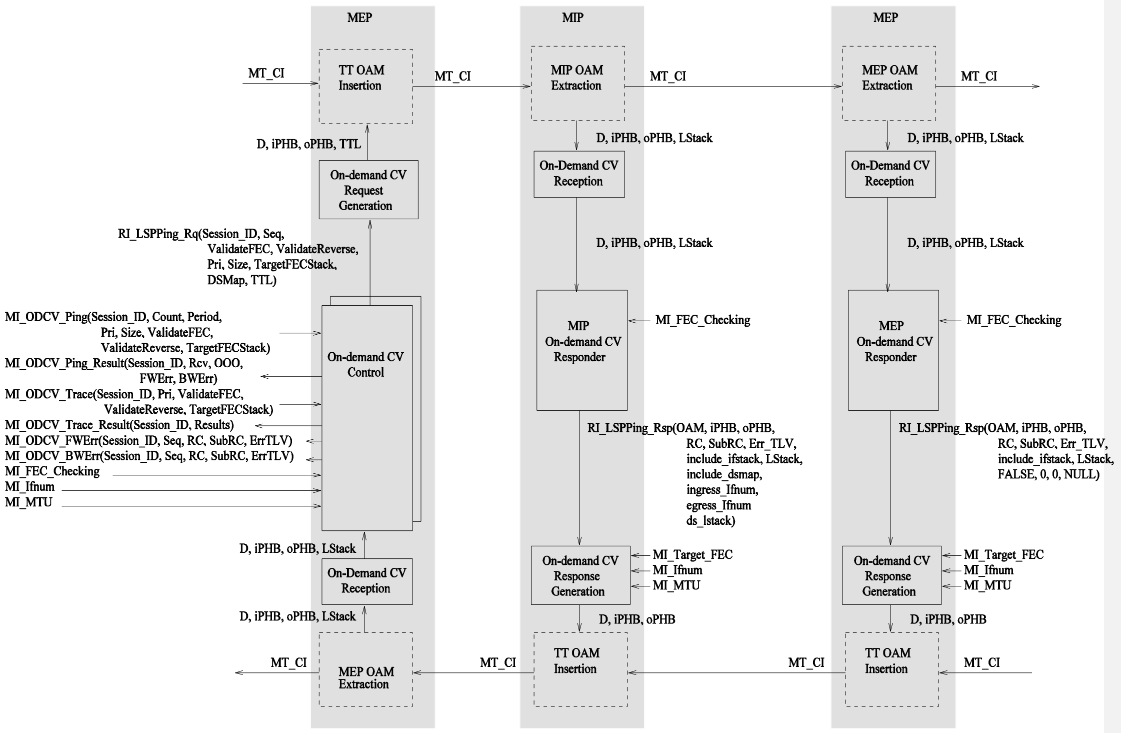
- If the version number is not 1, the packet is discarded
- If the length is less than 24, the packet is discarded
- If the Detect Mult field is 0, the packet is discarded
- If any of the P, F, A, D, or M flags are set, the packet is discarded
- If the My Discriminator field is 0, the packet is discarded
- If the Required Min Echo Rx Interval is not 0, the packet is discarded
- If the Your Discriminator field is 0 and the State is not DOWN or ADMINDOWN, the packet is discarded.
- If the Your Discriminator field is not 0 and no corresponding session can be found based on MI_Local_Discr[], the packet is discarded.

If the checks pass, the packet is passed to the instance of the CCCV Reception process whose MI_Local_Discr is equal to the Your Discriminator field. Packets received on the BFD-CC port from the OAM Extraction process are passed on to the BFD-CC port in the CCCV Reception process, and packets received on the BFD-CV port from the OAM Extraction process are passed on to the BFD-CV port in the CCCV Reception process.

Selection of the correct CCCV Reception process when the Your Discriminator field is 0 is FFS.

8.8.2 On-demand CV Processes

An overview of the On-demand CV Processes is shown on the figure below.



MEP
MIP
MEP
MT_CI
TT OAM Insertion
MT_CI
MIP OAM Extraction
MT_CI
MEP OAM Extraction
MT_CI
D, iPHB, oPHB, TTL
D, iPHB, oPHB, LStack
D, iPHB, oPHB, LStack
On-demand CV Request Generation
On-Demand CV Reception
On-Demand CV Reception
RI_LSPPing_Rq(Session_ID, Seq, ValidateFEC, ValidateReverse, Pri, Size, TargetFECStack, DSMap, TTL)
D, iPHB, oPHB, LStack
D, iPHB, oPHB, LStack
MI_ODCV_Ping(Session_ID, Count, Period, Pri, Size, ValidateFEC, ValidateReverse, TargetFECStack)
MI_ODCV_Ping_Result(Session_ID, Rcv, OOO, FWErr, BWErr)
MI_ODCV_Trace(Session_ID, Pri, ValidateFEC, ValidateReverse, TargetFECStack)
MI_ODCV_Trace_Result(Session_ID, Results)
MI_ODCV_FWErr(Session_ID, Seq, RC, SubRC, ErrTLV)
MI_ODCV_BWErr(Session_ID, Seq, RC, SubRC, ErrTLV)
MI_FEC_Checking
MI_Ifnum
MI_MTU
On-demand CV Control
MIP On-demand CV Responder
MI_FEC_Checking
MEP On-demand CV Responder
MI_FEC_Checking
RI_LSPPing_Rsp(OAM, iPHB, oPHB, RC, SubRC, Err_TLV, include_ifstack, LStack, include_dsmap, ingress_Ifnum, egress_Ifnum ds_lstack)
RI_LSPPing_Rsp(OAM, iPHB, oPHB, RC, SubRC, Err_TLV, include_ifstack, LStack, FALSE, 0, 0, NULL)
D, iPHB, oPHB, LStack
On-Demand CV Reception
On-demand CV Response Generation
MI_Target_FEC
MI_Ifnum
MI_MTU
On-demand CV Response Generation
MI_Target_FEC
MI_Ifnum
MI_MTU
D, iPHB, oPHB, LStack
D, iPHB, oPHB
D, iPHB, oPHB
MT_CI
MEP OAM Extraction
MT_CI
TT OAM Insertion
MT_CI
TT OAM Insertion
MT_CI

The On-demand CV protocol is controlled by the On-demand CV Control process. An on-demand session starts when the MI_ODCV_Ping() or MI_ODCV_Trace() signal is called. Multiple instances of the On-demand CV Control process can be used to run multiple on-demand CV sessions concurrently, provided each instance has a different session ID.

The On-demand CV Control process sends LSPPing Request packets via the On-Demand CV Request Generation Process, and receives LSPPing Responses via the On-Demand CV Reception process. Received responses may be checked for errors, if requested in the MI_ODCV_Ping() or MI_ODCV_Trace() signal.

The On-demand CV Control process reports errors in the forward direction via the MI_ODCV_FWErr() signal, and in the backward direction via the MI_ODCV_BWErr() signal. Results are reported via the MI_ODCV_Ping_Result() and MI_ODCV_Trace_Result() signals.

The MEP On-demand CV Responder and MIP On-demand CV Responder processes are responsible for checking received LSPPing Requests for errors, and sending responses via the On-demand CV Response Generation process.

The On-demand CV Request Generation and On-demand CV Response Generation processes generate LSPPing request and response packets in conformance with [RFC4379] and [draft-ietf-mpls-tp-on-demand-cv].

The MEP On-demand CV Responder, MIP On-demand CV Responder, and On-demand CV Control processes all perform similar steps to check received packets for errors. This checking uses the copy of the original label stack that is carried as part of the MT_CI. This common validation is described further below, followed by descriptions of each of the On-demand CV processes.

8.8.2.1 Common Validation

In the description below, label stacks and FEC stacks are denoted as arrays (Stack[]), where:

- Stack[1] is the bottom (innermost) label/FEC
- Stack[Count(Stack)] is the top (outermost) label/FEC
- Stack[0] is invalid

Count(Stack) returns the number of labels or FECs in the stack.

The validation is described by the following pseudocode. The values assigned to ‘rc’ are as described in [RFC4379].

```
ODCV_Validate (OAM, LStack_in[], FECStack[], MP_Type) {
rc = 0
sub_rc = 0
err_TLV = NULL
    done = FALSE
include_ifstack = FALSE
include_dsmap = FALSE
ldepth = 0
LStack = LStack_in
    if (malformed(OAM)) {
rc = 1
        done = TRUE
    } else if (OAM contains TLVs with types 4, 6, 8 or 10-32767) {
rc = 2
err_TLV = make_err_TLV(bad TLVs)
        done = TRUE
    } else {
        if (LStack[1] = GAL) {
remove_GAL_from_LStack()
        }
ldepth = count(LStack)
        while (!done &&ldepth> 0) {
            if (!label_known(LStack[ldepth]) {
rc = 11
sub_rc = ldepth
                done = TRUE
            }
ldepth--
        }
    }
    if (MP_Type = MEP) {
        if (!done) {
```

```
FECdepth = 1
            L = IMPLICIT_NULL
rc = 3
sub_rc = 1
            if (DSMAP(OAM) != NULL &&Ingress_Ifnum(DSMAP(OAM)) != 0) {
                if (DownstreamLabels(DSMAP(OAM)) != LStack) {
rc = 5
include_ifstack = TRUE
                    done = TRUE
                }
            }
        }
        while (!done) {
            (FECstatus, FECrc) = checkFEC(FECStack[FECdepth], L)
rc = FECrc
sub_rc = FECdepth
            if (FECstatus = 1) {
                done = TRUE
            } else {
FECdepth++
                if (FECdepth> count(FECStack)) {
                    done = TRUE
                }
            }
            if (!done) {
                if (FECstatus = 0) {
ldepth++
                    if (ldepth> count(LStack)) {
                        done = TRUE
                    } else {
                        L = LStack[ldepth]
                    }
                }
            }
        }
    } else {
        // MP_Type = MIP
        if (!done) {
rc = 8
sub_rc = 1
            if (DSMAP(OAM) != NULL) {
                if (Ingress_Ifnum(DSMAP(OAM)) = 0) {
rc = 6
include_ifstack = TRUE
                } else {
                    if (DownstreamLabels(DSMAP(OAM)) != LStack) {
rc = 5
include_ifstack = TRUE
                        done = TRUE
                    }
                }
            }
        }
        if (!done)
Egress_Ifnum = get_egress_interface()
            if (Egress_Ifnum = 0) {
rc = 9
                done = TRUE
            }
        }
        if (!done) {
            if (DSMAP(OAM) != NULL) {
include_dsmap = TRUE
            } else {
                done = TRUE
            }
        }
        if (!done) {
if (V(OAM) == 0 &&MI_FEC_Checking = 0) {
                done = TRUE
            }
        }
        if (!done) {
FECdepth = 0
            i = 1
            while (i > 0) {
FECdepth++
                if (DownstreamLabels(DSMAP(OAM))[FECdepth] != IMPLICIT_NULL) {
                   i--
                }
            }
            if (count(FECStack) >= FECdepth) {
```

```
(FECstatus, FECrc) = checkFEC(FECStack[FECdepth],
LStack[1])
                if (FECstatus = 2) {
rc = 10
                } else if (FECstatus = 1) {
rc = FECrc
sub_rc = FECdepth
                }
            }
        }
    }
    return(rc, sub_rc, err_TLV, include_ifstack, include_dsmap)
}
```

The utility functions used in the pseudocode above are described below:

- malformed(OAM) checks that the packet is in accordance with the format described in [RFC4379] and [draft-ietf-mpls-tp-on-demand-cv]. It also checks that:
    - If the packet is a request, it contains a Target FEC Stack TLV
    - If the packet is a reply and the R flag is set, it contains a Reverse Target FEC Stack TLV
    - The Target FEC Stack or Reverse Target FEC Stack TLVs contain only sub-types 'Static LSP', 'Static Pseudowire' and 'Nil FEC'. Use of other subtypes are FFS.
    - If the packet contains a Downstream Mapping TLV, the address type is 'Non-IP'. Use of other address types is FFS.
- make_err_TLV(TLVs) creates an 'Errored TLVs' TLV according to [RFC4379] and copies the bad TLVs into it.
- remove_GAL_from_LStack removes the GAL from the bottom of the label stack, so that LStack[1] now refers to the label that immediately preceded the GAL.
- label_known(Label) checks whether the Label value is known and can be processed.
- checkFEC(FEC, Label) implements the FEC checking procedure described in [RFC4379] section 4.4.1.
- get_egress_interface() returns MI_Ifnum if this is the egress interface, otherwise it uses forwarding information to find the egress interface and returns its interface number, or 0 if no egress interface was found or it is not MPLS-enabled.

8.8.2.2 On-demand CV Control process

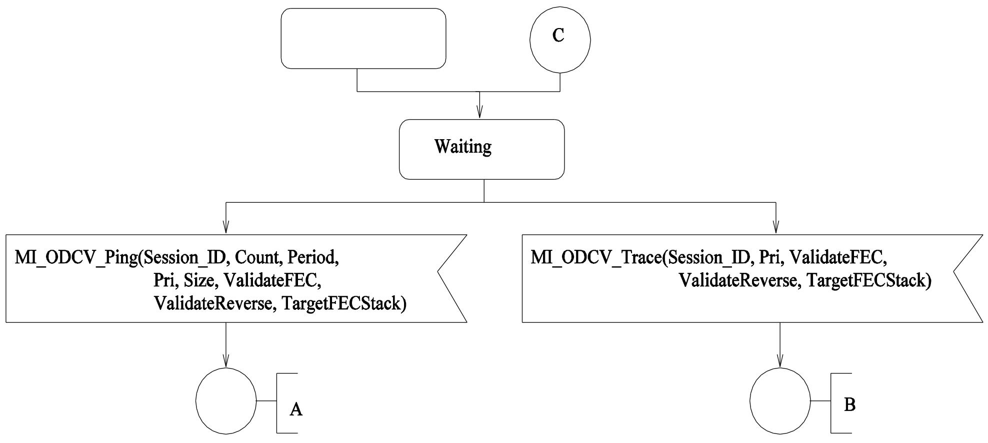
The On-demand CV Control process operates the LSPPing on-demand CV protocol. An LSPPing session is started by the MI_ODCV_Ping() or MI_ODCV_Trace() signals. In either case, a Session ID is supplied; multiple instances of the On-demand CV Control process can be created, provided each has a unique Session ID.

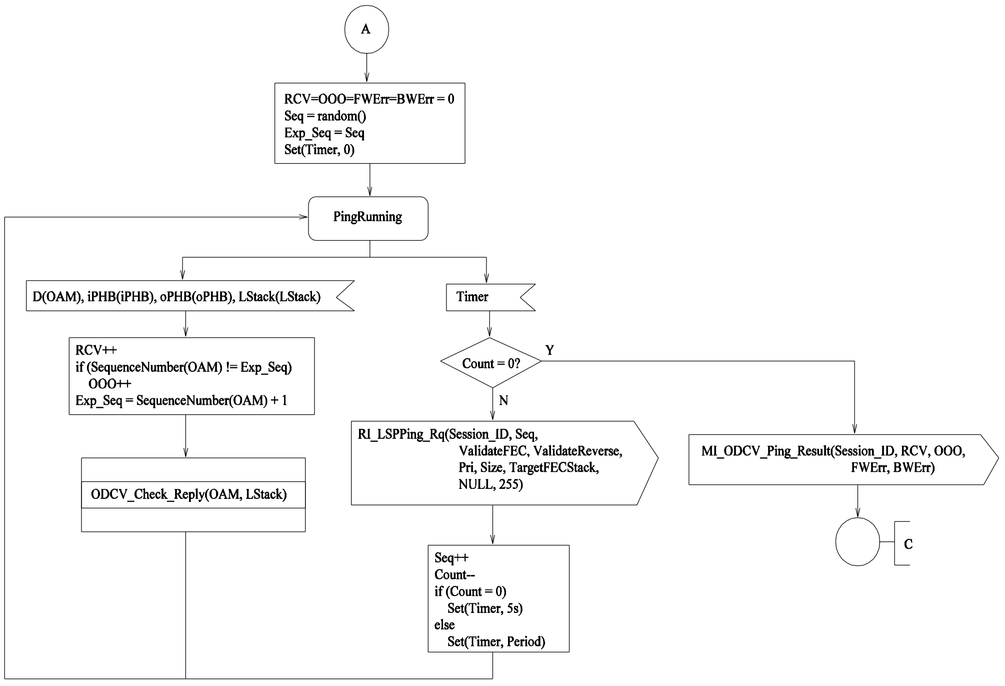
The Target FEC Stack to be checked by the peer device is specified in the MI_ODCV_Ping() or MI_ODCV_Trace() signal. Other mechanisms for deriving the Target FEC Stack, for example if dynamic signalling protocols are in use, are FFS. The Target FEC Stack passed in the

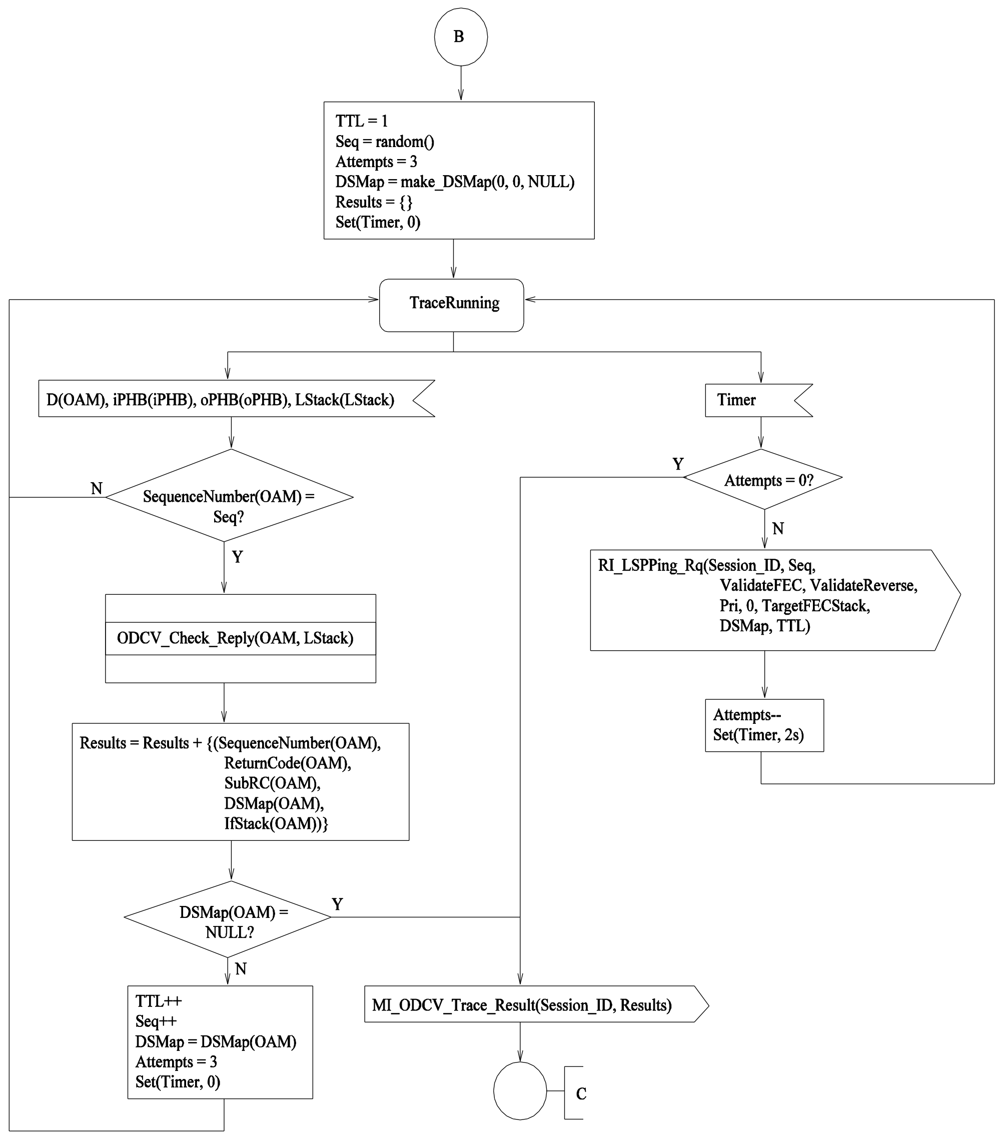
MI_ODCV_Ping() or MI_ODCV_Trace() signals must only contain FECs with subtypes 'Static LSP', 'Static Pseudowire' or 'Nil FEC'.

Results are reported by the On-demand CV Control process using the MI_ODCV_Ping_Result() or MI_ODCV_Trace_Result() signals when the session ends. In addition, any errors detected while the session is running are reported by using the MI_ODCV_FWErr() signal (for errors in the Control-to-Responder direction) or MI_ODCV_BWErr() signal (for errors on the Responder-to-Control direction). Note that errors in the Responder-to-Control direction are only detected if ValidateReverse is set to TRUE in the MI_ODCV_Ping() or MI_ODCV_Trace() signal.

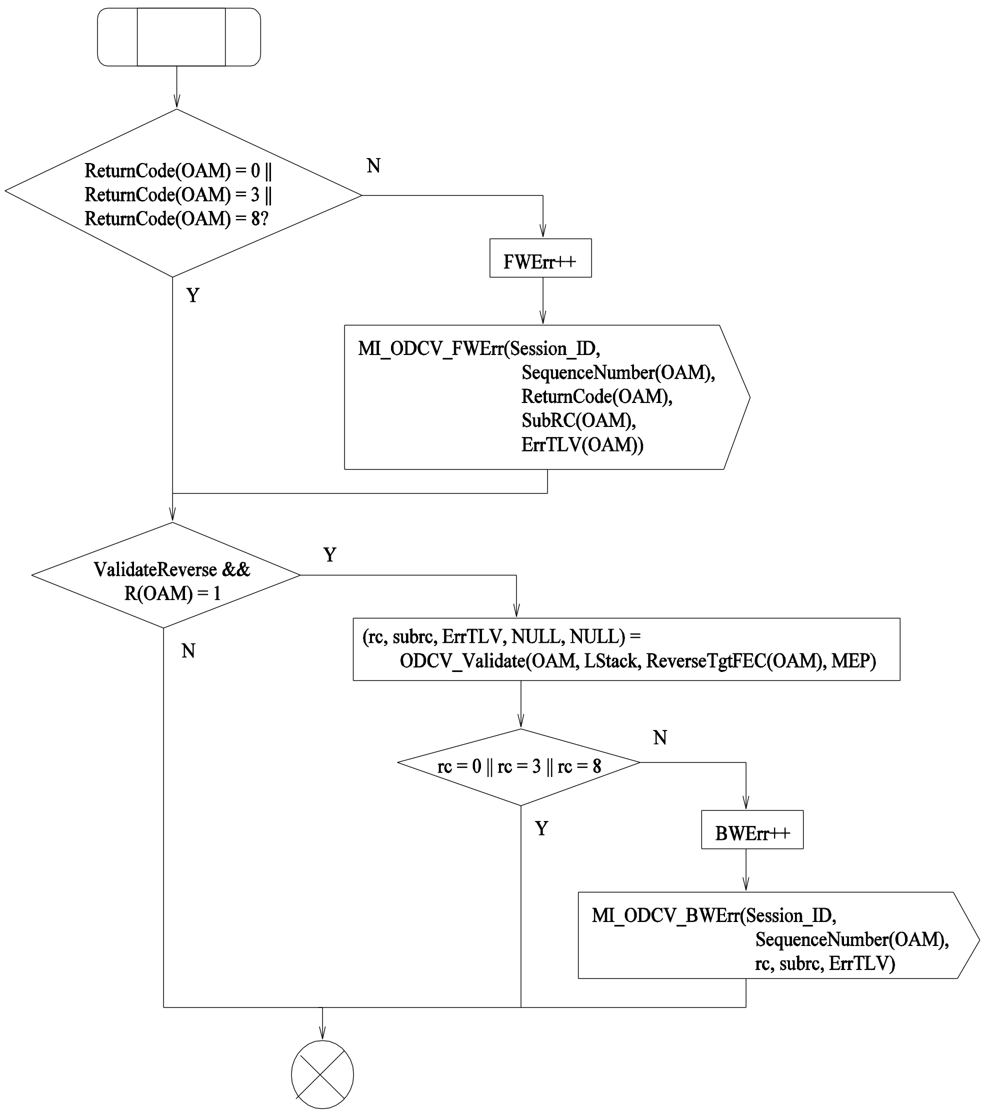
The behaviour of the On-demand CV Control process is shown in the figures below. In PingRunning state, the process sends LSPPing Requests periodically, and handles any received replies by counting them and checking for any errors. In TraceRunning state, an initial LSPPing Request is sent with TTL 1, so that it is intercepted by the first MIP (or MEP) reached. When a response is received, it is first checked for any errors. Then, if the response was from a MIP (ie it contains a DSMap TLV), the TTL is incremented and a new LSPPing request is sent. Incrementing the TTL ensures the request is intercepted by the next MIP (or MEP). If no response is received, 3 attempts are made to resend the request, before giving up and reporting any results collected so far.


C
Waiting
MI_ODCV_Ping(Session_ID, Count, Period, Pri, Size, ValidateFEC, ValidateReverse, TargetFECStack)
MI_ODCV_Trace(Session_ID, Pri, ValidateFEC, ValidateReverse, TargetFECStack)
A
B


A
RCV=OOO=FWErr=BWErr = 0
Seq = random()
Exp_Seq = Seq
Set(Timer, 0)
PingRunning
D(OAM), iPHB(iPHB), oPHB(oPHB), LStack(LStack)
Timer
RCV++
if (SequenceNumber(OAM) != Exp_Seq)
OOO++
Exp_Seq = SequenceNumber(OAM) + 1
Count = 0?
Y
N
ODCV_Check_Reply(OAM, LStack)
RI_LSPPing_Rq(Session_ID, Seq, ValidateFEC, ValidateReverse, Pri, Size, TargetFECStack, NULL, 255)
MI_ODCV_Ping_Result(Session_ID, RCV, OOO, FWErr, BWErr)
C
Seq++
Count--
if (Count = 0)
Set(Timer, 5s)
else
Set(Timer, Period)


B
TTL = 1
Seq = random()
Attempts = 3
DSMap = make_DSMap(0, 0, NULL)
Results = {}
Set(Timer, 0)
TraceRunning
D(OAM), iPHB(iPHB), oPHB(oPHB), LStack(LStack)
Timer
SequenceNumber(OAM) = Seq?
N
Y
Attempts = 0?
Y
N
RI_LSPPing_Rq(Session_ID, Seq, ValidateFEC, ValidateReverse, Pri, 0, TargetFECStack, DSMap, TTL)
ODCV_Check_Reply(OAM, LStack)
Attempts--
Set(Timer, 2s)
Results = Results + {(SequenceNumber(OAM), ReturnCode(OAM), SubRC(OAM), DSMap(OAM), IfStack(OAM))}
DSMap(OAM) = NULL?
Y
N
TTL++
Seq++
DSMap = DSMap(OAM)
Attempts = 3
Set(Timer, 0)
MI_ODCV_Trace_Result(Session_ID, Results)
C

Process ODCV_Check_Reply(OAM, LStack)



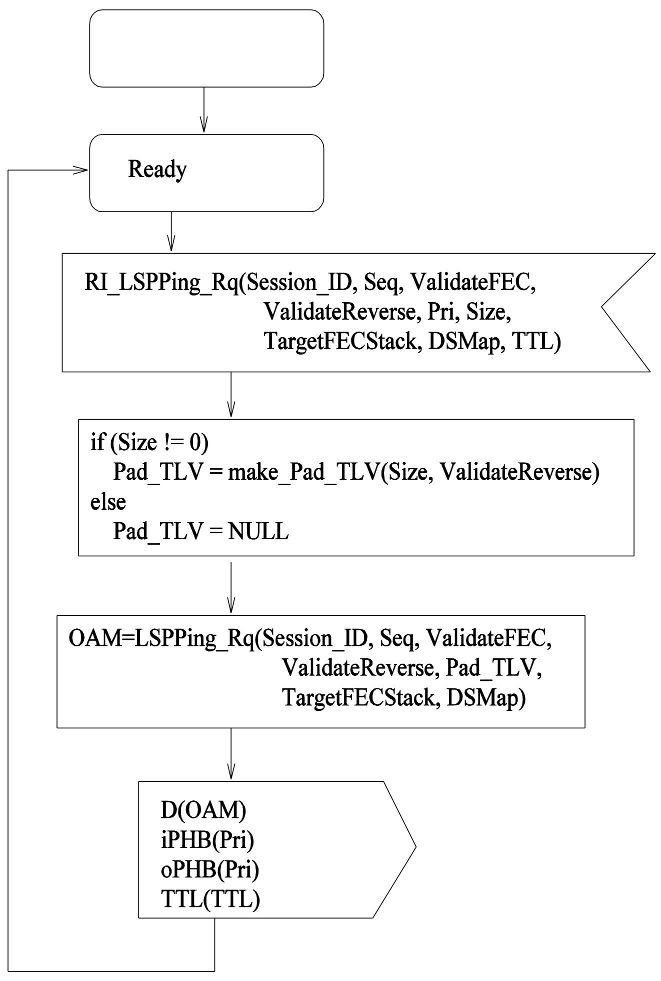
The make_DSMap(ingress_ifnum, egress_ifnum, ds_lstack) function creates a Downstream Mapping TLV according to [RFC4379] and [draft-ietf-mpls-tp-on-demand-cv]. The fields are filled in as follows:

- MTU: Set to MI_MTU
- Address Type: set to 5 (Non IP). Use of other address types is FFS.
- DS Flags: The I flag is set to 1, all other flags are set to 0.

- Ingress Ifnum: set to ingress_ifnum
- Egress Ifnum: set to egress_ifnum
- Multipath Type: set to 0 (no Multipath)
- Depth Limit: set to 0
- Multipath Length: set to 0
- Downstream Labels: derived from ds_lstack as described in [RFC4379]. The protocol is set to 1 (Static). Use of other values are FFS.

8.8.2.3 On-demand CV Request Generation process

The On-demand CV Request Generation process is shown in the following figure:


Ready
RI_LSPPing_Rq(Session_ID, Seq, ValidateFEC, ValidateReverse, Pri, Size, TargetFECStack, DSMap, TTL)
if (Size != 0)
Pad_TLV = make_Pad_TLV(Size, ValidateReverse)
else
Pad_TLV = NULL
OAM=LSPPing_Rq(Session_ID, Seq, ValidateFEC, ValidateReverse, Pad_TLV, TargetFECStack, DSMap)
D(OAM)
iPHB(Pri)
oPHB(Pri)
TTL(TTL)

The make_Pad_TLV(Size) function creates a Pad TLV in accordance with [RFC4379]. The Length field is set to Size. The first octet of the value field is set to 2 (Copy Pad TLV) if ValiadateReverse is FALSE, and 1 (Drop Pad TLV) if ValidateReverse is TRUE.

Note: Size is only non-zero in Ping mode, when no DSMap TLV is included. In this case, the responder will not add any additional TLVs (eg an interface and label stack TLV) to the reply unless the 'R' (ValidateReverse) flag is set, and so the Pad TLV can be safely copied into the reply.

The LSPPing_Rq function creates an LSPPing Echo Request packet in accordance with [RFC4379] and [draft-ietf-mpls-tp-on-demand-cv]. The fields are filled in as follows:

- Version Number: set to 1
- Global Flags: if ValidateFEC is TRUE, the V flag is set to 1; if ValidateReverse is TRUE, the R flag is set to 1; all other flags are set to 0.
- Message Type: set to MPLS Echo Request
- Reply Mode: set to 4 (reply via application control channel)

- Return Code: set to 0
- Return Subcode: set to 0
- Sender's Handle: set to the value of Session_ID
- Sequence Number: set to the value of Seq
- Timestamp Sent: set to LocalTime.
- Timestamp Received: set to 0.

The following TLVs are added:

- A Target FEC Stack TLV is added containing the contents of TargetFECStack.
- If Pad_TLV is not NULL, a Pad TLV is added containing the contents of Pad_TLV.
- If DSMap is not NULL, a Downstream Mapping TLV is added containing the contents of DSMap.

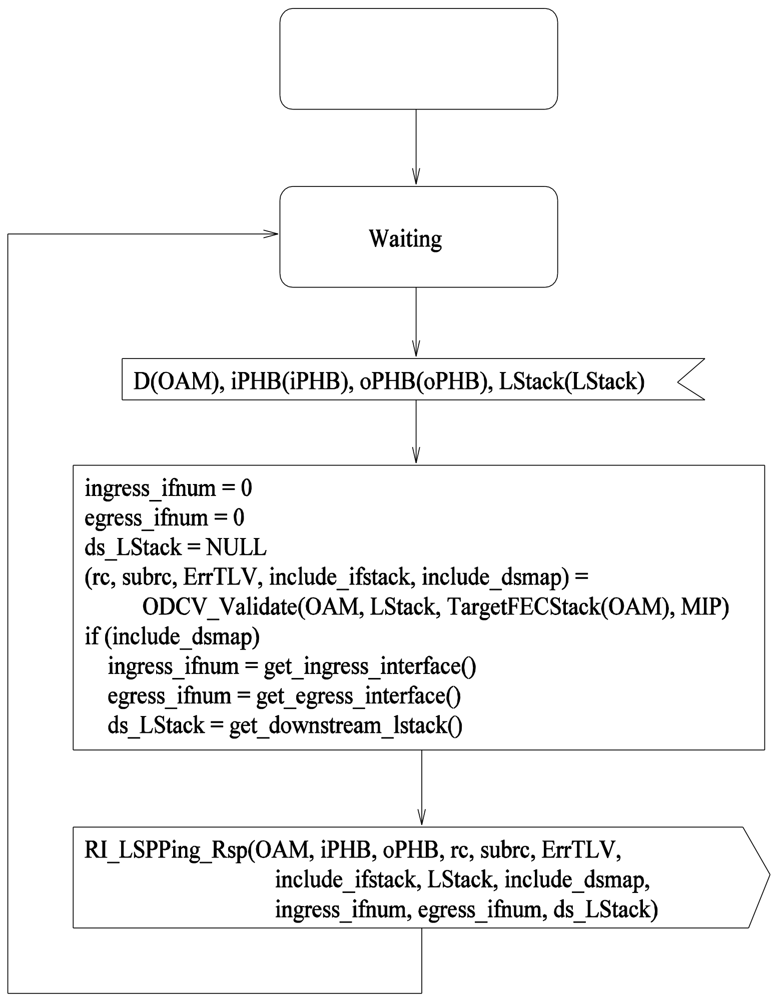
8.8.2.4 On-demand CV Reception process

The On-demand CV Reception process demultiplexes received LSPPing packets as follows:

- If the Message Type is MPLS Echo Request, the packet is passed to the MIP On-demand CV Responder or MEP On-demand CV Responder process as appropriate
- Otherwise, if this is a MIP the packet is discarded.
- If this is a MEP and the Message Type is MPLS Echo Reply, the On-demand CV Reception process passes the packet to the instance of the On-demand CV Control process whose Session ID is equal to the "Sender's handle" in the received packet. If there is no such instance of the On-demand CV Control process, the packet is discarded.

8.8.2.5 MIP On-demand CV Responder process

The MIP On-demand CV Responder process is described in the figure below.


Waiting
D(OAM), iPHB(iPHB), oPHB(oPHB), LStack(LStack)
ingress_ifnum = 0
egress_ifnum = 0
ds_LStack = NULL
(rc, subrc, ErrTLV, include_ifstack, include_dsmap) =
ODCV_Validate(OAM, LStack, TargetFECStack(OAM), MIP)
if (include_dsmap)
ingress_ifnum = get_ingress_interface()
egress_ifnum = get_egress_interface()
ds_LStack = get_downstream_lstack()
RI_LSPPing_Rsp(OAM, iPHB, oPHB, rc, subrc, ErrTLV,
include_ifstack, LStack, include_dsmap,
ingress_ifnum, egress_ifnum, ds_LStack)

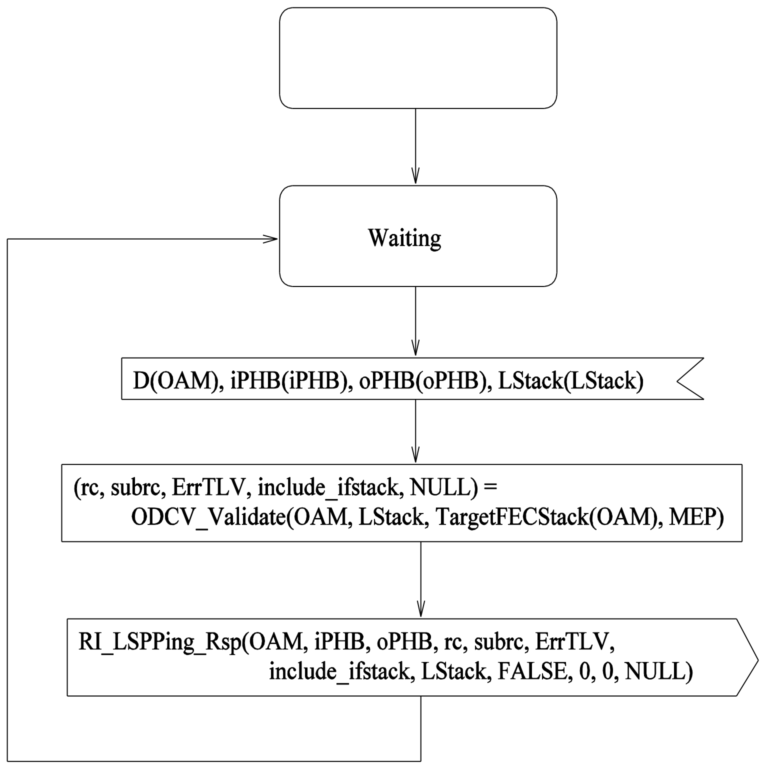
The get_egress_interface() function is described in section 8.1.4.1 above.

The get_ingress_interface() function returns MI_Ifnum if this is the ingress interface, otherwise it returns the interface number of the interface where the packet arrived.

The get_downsteam_lstack() returns the label stack that would be attached to the packet if it were to be forwarded out of the egress interface, derived as described in [RFC4379].

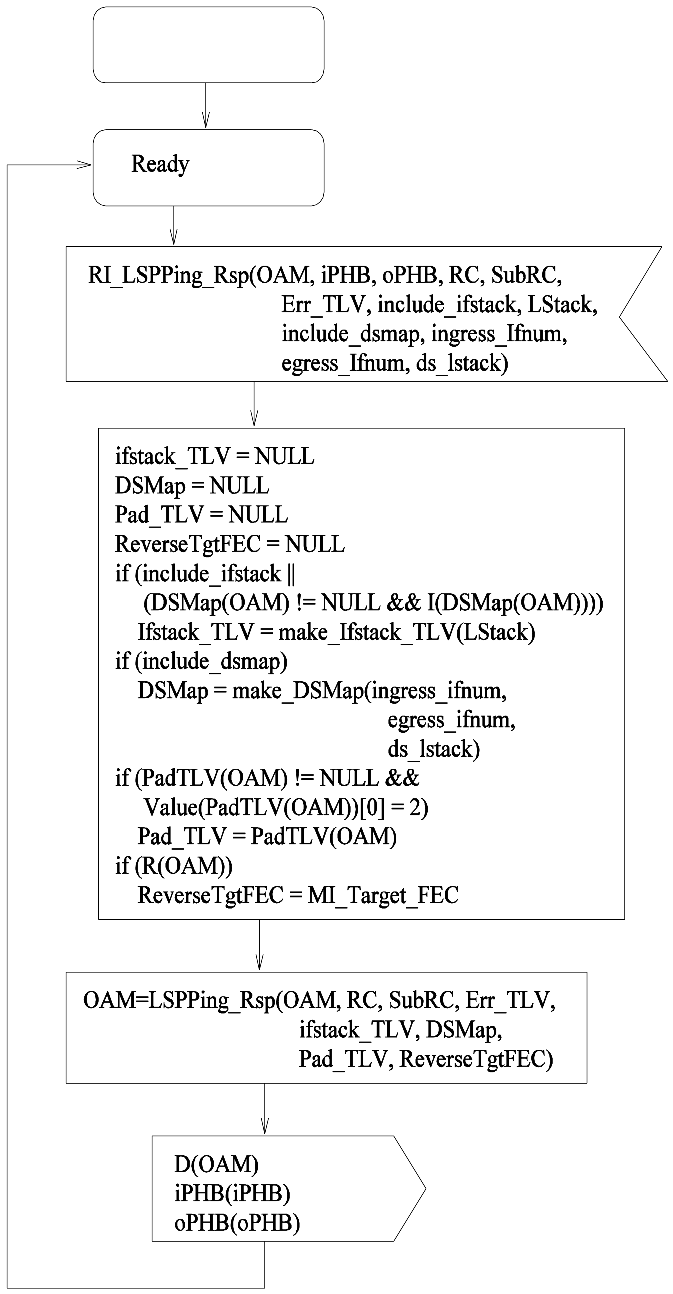
8.8.2.6 MEP On-demand CV Responder process

The MEP On-demand CV Responder process is described in the figure below.


Waiting
D(OAM), iPHB(iPHB), oPHB(oPHB), LStack(LStack)
(rc, subrc, ErrTLV, include_ifstack, NULL) = ODCV_Validate(OAM, LStack, TargetFECStack(OAM), MEP)
RI_LSPPing_Rsp(OAM, iPHB, oPHB, rc, subrc, ErrTLV, include_ifstack, LStack, FALSE, 0, 0, NULL)

8.8.2.7 On-demand CV Response Generation process

The On-demand CV Response Generation process is shown in the following figure:


Ready
RI_LSPPing_Rsp(OAM, iPHB, oPHB, RC, SubRC, Err_TLV, include_ifstack, LStack, include_dsmap, ingress_Ifnum, egress_Ifnum, ds_lstack)
ifstack_TLV = NULL
DSMap = NULL
Pad_TLV = NULL
ReverseTgtFEC = NULL
if (include_ifstack || (DSMap(OAM) != NULL && I(DSMap(OAM))))
Ifstack_TLV = make_Ifstack_TLV(LStack)
if (include_dsmap)
DSMap = make_DSMap(ingress_ifnum, egress_ifnum, ds_lstack)
if (PadTLV(OAM) != NULL && Value(PadTLV(OAM))[0] = 2)
Pad_TLV = PadTLV(OAM)
if (R(OAM))
ReverseTgtFEC = MI_Target_FEC
OAM=LSPPing_Rsp(OAM, RC, SubRC, Err_TLV, ifstack_TLV, DSMap, Pad_TLV, ReverseTgtFEC)
D(OAM)
iPHB(iPHB)
oPHB(oPHB)

The make_Ifstack_TLV(LStack) function creates an Interface and Label Stack TLV according to [RFC4379] and [draft-ietf-mpls-tp-on-demand-cv]. The fields are filled in as follows:

- Address Type: set to IPv4 Unnumbered
- IP Address: set to 0

- Interface: set to MI_Ifnum
- Label Stack: Copied from LStack.

Use of other values in the Interface and Label Stack TLV is FFS.

[Editor's note: this use of MI_Ifnum should align with [draft-ietf-mpls-tp-mib-management-overview]].

The make_DSMap(ingress_ifnum, egress_ifnum, ds_lstack) function is described in section 8.1.4.2 above.

The LSPPing_Rsp function creates an LSPPing Echo Reply packet in accordance with [RFC4379] and [draft-ietf-mpls-tp-on-demand-cv]. The fields are filled in as follows:

- Version Number: set to 1.
- Global Flags: copied from the received Echo Request.
- Message Type: set to MPLS Echo Reply.
- Reply Mode: set to 0 (do not reply).
- Return Code: set to RC.
- Return Subcode: set to SubRC.
- Sender's Handle: copied from the received Echo Request.
- Sequence Number: copied from the received Echo Request.
- Timestamp Sent: copied from the received Echo Request.
- Timestamp Received: set to LocalTime.

If reverse FEC checking was requested in the LSPPing request (ie, the R flag was set), a Reverse Target FEC Stack is created based on MI_Target_FEC. Other mechanisms for deriving the FEC stack, for example if dynamic signalling protocols are in use, are FFS.

The following TLVs are added:

- The TargetFECStack TLV is copied from the received packet.
- If Err_TLV is not NULL, an Errored TLVs TLV is added containing the contents of Err_TLV
- If Ifstack_TLV is not NULL, an Interface and Label Stack TLV is added containing the contents of Ifstack_TLV
- If DSMap is not NULL, a Downstream Mapping TLV is added containing the contents of DSMap.
- If Pad_TLV is not NULL, a Pad TLV is added containing the contents of Pad_TLV.
- If ReverseTgtFEC is not NULL, a Reverse-path Target FEC Stack TLV is added containing the contents of ReverseTgtFEC.

8.8.3 Performance Monitoring Processes

FFS

8.8.4 Lock/Loopback Instruct Processes

FFS

9 MPLS-TP layer functions

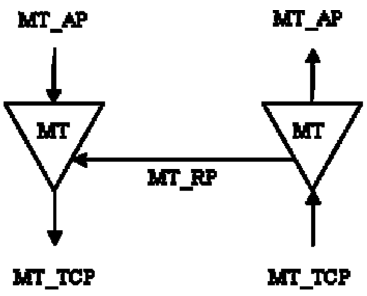
9.1 Connection Functions (MT_C)

Connection Functions are described in [G.8121]

9.2 Termination Functions

9.2.1 MPLS-TP Trail Termination function (MT_TT)

The bidirectional MPLS-TP Trail Termination (MT_TT) function terminates the MPLS-TP OAM to determine the status of the MPLS-TP (sub)layer trail. The MT_TT function is performed by a co-located pair of the MPLS-TP trail termination source (MT_TT_So) and sink (MT_TT_Sk) functions as shown in the figure below.

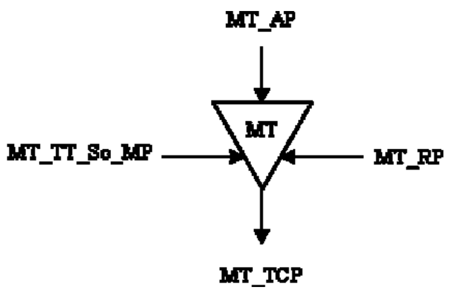


9.2.1.1 MPLS-TP Trail Termination Source function (MT_TT_So)

The MT_TT_So function determines and inserts the TTL value in the shim header TTL field and adds MPLS-TP OAM to the MT_AI signal at its MT_AP.

The information flow and processing of the MT_TT_So function is defined with reference to the figure below.

• Symbol:



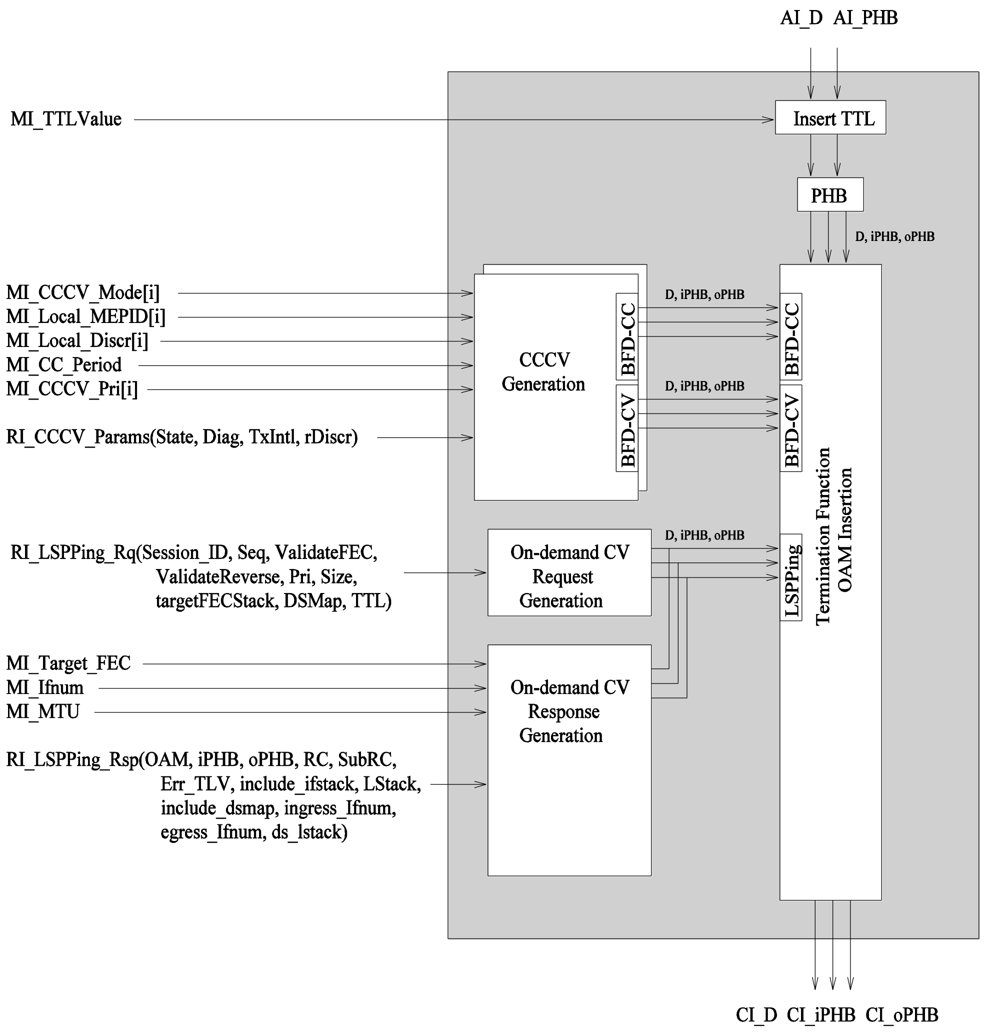
• Interfaces:

Table 9-*/G.8121.2/Y.1381.2 – MT_TT_So inputs and outputs

Input(s)	Output(s)
MT_AP: MT_AI_D MT_AI_PHB MT_AI_MCC MT_AI_SCC **MT_RP:** MT_RI_CCCV_Params MT_RI_LSPPing_Rq MT_RI_LSPPing_Rsp **MT_TT_So_MP:** MT_TT_So_MI_TTLValue MT_TT_So_MI_CCCV_Mode[] MT_TT_So_MI_Local_MEPID[] MT_TT_So_MI_Local_Discr[] MT_TT_So_MI_CC_Period MT_TT_So_MI_CCCV_Pri[] MT_TT_So_MI_Target_FEC MT_TT_So_MI_Ifnum MT_TT_So_MI_MTU	**MT_CP:** MT_CI_D MT_CI_oPHB MT_CI_iPHB

• Processes:

The processes associated with the MT_TT_So function are as depicted in the Figure below.



AI_D AI_PHB
MI_TTLValue
Insert TTL
PHB
D, iPHB, oPHB
MI_CCCV_Mode[i]
MI_Local_MEPID[i]
MI_Local_Discr[i]
MI_CC_Period
MI_CCCV_Pri[i]
RI_CCCV_Params(State, Diag, TxIntl, rDiscr)
CCCV Generation
BFD-CC
BFD-CV
D, iPHB, oPHB
D, iPHB, oPHB
BFD-CC
BFD-CV
Termination Function OAM Insertion
RI_LSPPing_Rq(Session_ID, Seq, ValidateFEC, ValidateReverse, Pri, Size, targetFECStack, DSMap, TTL)
On-demand CV Request Generation
D, iPHB, oPHB
LSPPing
MI_Target_FEC
MI_Ifnum
MI_MTU
On-demand CV Response Generation
RI_LSPPing_Rsp(OAM, iPHB, oPHB, RC, SubRC, Err_TLV, include_ifstack, LStack, include_dsmap, ingress_Ifnum, egress_Ifnum, ds_lstack)
CI_D CI_iPHB CI_oPHB

[Performance Monitoring process are FFS and are not depicted]

PHB: The AI_PHB signal is assigned to both the CI_iPHB and CI_oPHB signals at the MT_TCP reference point.

Insert TTL: The Time To Live value is inserted in the outer shim header's TTL field within the MT_AI traffic unit

Temination Function OAM Insertion: See 8.1.1.1

CCCV Generation Process: See 8.1.2.2

On-demand CV Request Generation: See8.1.4.3

On-demand CV Response Generation: See 8.1.4.7

• **Defects:**

None.

• **Consequent actions:**

None.

• **Defect correlations:**

None.

• **Performance monitoring:**

None.

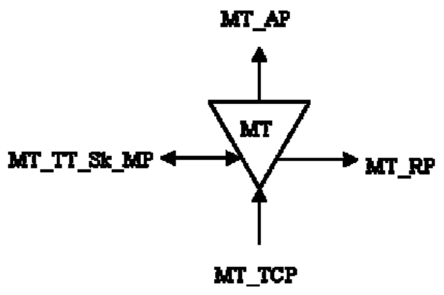
9.2.1.2 MPLS-TP Trail Termination Sink function (MT_TT_Sk)

The MT_TT_Sk function reports the state of the MPLS-TP Trail (Network Connection). It extracts MPLS-TP trail OAM from the MPLS-TP signal at its MT_TCP, detects defects, counts during 1-second periods errors and defects to feed Performance Monitoring when connected and forwards the defect information as backward indications to the companion MT_TT_So function.

Note – The MT_TT_Sk function extracts and processes one level of MPLS-TP OAM irrespective of the presence of more levels.

The information flow and processing of the MT_TT_Sk function is defined with reference to the Figure below.

• **Symbol:**



• **Interfaces:**

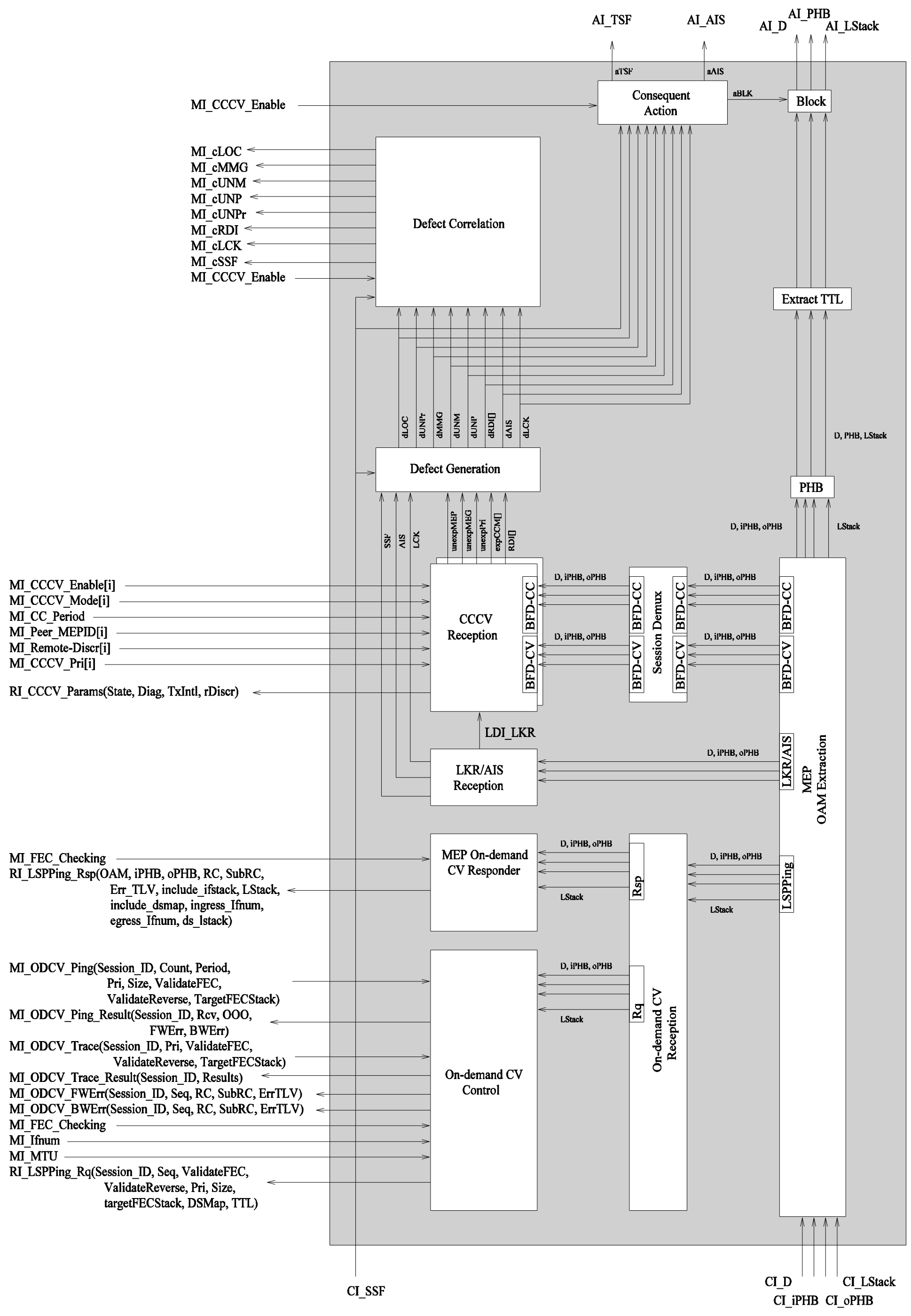
Table x/G.8121.2/Y.1381.2 – MT_TT_Sk inputs and outputs

Input(s)	Output(s)
MT_TCP: MT_CI_D MT_CI_iPHB MT_CI_oPHB MT_CI_SSF MT_CI_LStack **MT_TT_Sk_MP:** MT_TT_Sk_MI_CCCV_Enable[] MT_TT_Sk_MI_CCCV_Mode[]	**MT_AP:** MT_AI_D MT_AI_PHB MT_AI_TSF MT_AI_AIS MT_AI_MCC MT_AI_SCC MT_AI_LStack

Input(s)	Output(s)
MT_TT_Sk_MI_CC_Period MT_TT_Sk_MI_Peer_MEPID[] MT_TT_Sk_MI_Remote_Discr[] MT_TT_Sk_MI_CCCV_Pri[] MT_TT_Sk_MI_Local_Discr[] MT_TT_Sk_MI_ODCV_Ping MT_TT_Sk_MI_ODCV_Trace MT_TT_Sk_MI_FEC_Checking MT_TT_Sk_MI_Ifnum MT_TT_Sk_MI_MTU	**MT_RP:** MT_RI_CCCV_Params MT_RI_LSPPing_Rq MT_RI_LSPPing_Rsp **MT_TT_Sk_MP:** MT_TT_Sk_MI_cSSF MT_TT_Sk_MI_cLCK MT_TT_Sk_MI_cLOC[] MT_TT_Sk_MI_cMMG MT_TT_Sk_MI_cUNL MT_TT_Sk_MI_cUNM MT_TT_Sk_MI_cUNP MT_TT_Sk_MI_cUNPhb MT_TT_Sk_MI_cDEG MT_TT_Sk_MI_cRDI MT_TT_Sk_MI_pN_LF MT_TT_Sk_MI_pN_TF MT_TT_Sk_MI_pF_LF MT_TT_Sk_MI_pF_TF MT_TT_Sk_MI_pF_DS MT_TT_Sk_MI_pN_DS MT_TT_Sk_MI_ODCV_Ping_Result MT_TT_Sk_MI_ODCV_Trace_Result MT_TT_Sk_MI_ODCV_FWErr MT_TT_Sk_MI_ODCV_BWErr

• **Processes:**

The processes associated with the MT_TT_Sk function are as depicted in the figure Figure below.



AI_TSF
AI_AIS
AI_PHB
AI_D
AI_LStack
aTSF
aAIS
aBLK
Consequent Action
Block
MI_CCCV_Enable
MI_cLOC
MI_cMMG
MI_cUNM
MI_cUNP
MI_cUNPr
MI_cRDI
MI_cLCK
MI_cSSF
MI_CCCV_Enable
Defect Correlation
Extract TTL
dLOC
dUNPr
dMMG
dUNM
dUNP
dRDI[]
dAIS
dLCK
D, PHB, LStack
Defect Generation
PHB
SSF
AIS
LCK
unexpMEP
unexpMEG
unexpPeri
expCCM[]
RDI[]
D, iPHB, oPHB
LStack
MI_CCCV_Enable[i]
MI_CCCV_Mode[i]
MI_CC_Period
MI_Peer_MEPID[i]
MI_Remote-Discr[i]
MI_CCCV_Pri[i]
RI_CCCV_Params(State, Diag, TxIntl, rDiscr)
CCCV Reception
BFD-CC
BFD-CV
Session Demux
LDI_LKR
LKR/AIS Reception
LKR/AIS
MEP OAM Extraction
MI_FEC_Checking
RI_LSPPing_Rsp(OAM, iPHB, oPHB, RC, SubRC, Err_TLV, include_ifstack, LStack, include_dsmap, ingress_Ifnum, egress_Ifnum, ds_Istack)
MEP On-demand CV Responder
Rsp
LSPPing
MI_ODCV_Ping(Session_ID, Count, Period, Pri, Size, ValidateFEC, ValidateReverse, TargetFECStack)
MI_ODCV_Ping_Result(Session_ID, Rcv, OOO, FWErr, BWErr)
MI_ODCV_Trace(Session_ID, Pri, ValidateFEC, ValidateReverse, TargetFECStack)
MI_ODCV_Trace_Result(Session_ID, Results)
MI_ODCV_FWErr(Session_ID, Seq, RC, SubRC, ErrTLV)
MI_ODCV_BWErr(Session_ID, Seq, RC, SubRC, ErrTLV)
MI_FEC_Checking
MI_Ifnum
MI_MTU
RI_LSPPing_Rq(Session_ID, Seq, ValidateFEC, ValidateReverse, Pri, Size, targetFECStack, DSMap, TTL)
On-demand CV Control
Rq
On-demand CV Reception
CI_SSF
CI_D
CI_iPHB
CI_oPHB
CI_LStack

[Performance Monitoring process are FFS and are not depicted]

Note: unexpPriority (dUNPr) does not appear in draft-ietf-cc-cv-rdi, or in G.8121 (this event is FFS)

PHB: The CI_oPHB signal is assigned to the AI_PHB signal at the reference point MT_AP.

Note that the CI_iPHB signal is not used by any of the processes in the function.

Extract TTL: The Time To Live value is extracted from the outer shim header's TTL field within the MT_CI traffic unit

Block: When the aBlock consequent action is asserted, this process drops all traffic units arriving at its input.

MEP OAM Extraction: see 8.1.1.3

Session Demux: see 8.1.2.3

CCCV Reception: see 8.1.2.1

LKR/AIS Reception: see 8.1.3.3

On-demand CV Reception: see 8.1.4.4

On-demand CV Control: see 8.1.4.2

MEP On-demand CV Responder: see 8.1.4.6

Performance Counter Process: This process is for further study.

Defect Generation: This process raises and clears the defects as defined in clause 6.1.

[Note: the defects, consequent actions, performance monitoring etc are common and can be described in G.8121 and referred to by G.8121.1 and G.8121.2]

• Defects:

See [G.8121]

• Consequent actions:

See [G.8121]

• Defect correlations:

See [G.8121]

• Performance monitoring:

See [G.8121]

9.3 Adaptation Functions

9.3.1 MPLS-TP to MPLS-TP Adaptation function (MT/MT_A)

This atomic functions are defined in clause 9.3.1 in G.8121. They use the OAM protocol specific AIS insertion process and LCK generation process as defined in clause 8.6.2.

~~9.3.1~~

Formatted: Font: (Asian) MS Mincho, (Asi
Japanese

Formatted: Normal

9.4 Diagnostic Functions

This clause describes Termination Functions and Adaptation Functions relating to OAM.

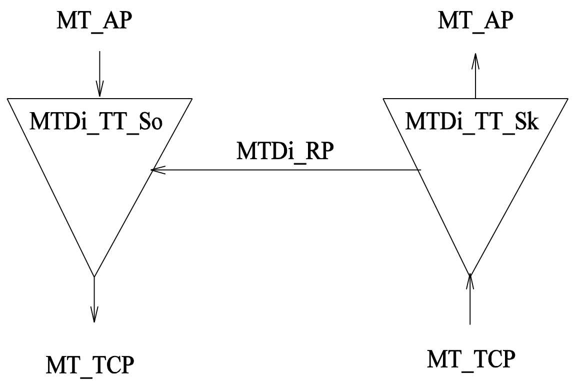
9.4.1 Diagnostic Functions for MEPs

Diagnostic functions for MEPs are includes in the MPLS-TP Trail Termination function (see 9.2.1) and MPLS-TP to MPLS-TP Adaptation function (see 9.3.1).

9.4.2 Diagnostic Functions for MIPs

9.4.2.1 MPLS-TP MIP Diagnostic Trail Termination function (MTDi_TT)

The bidirectional MPLS-TP MIP DiagnosticTrail Termination (MTDi_TT) function is performed by a co-located pair of the MPLS-TP trail termination source (MTDi_TT_So) and sink (MTDi_TT_Sk)



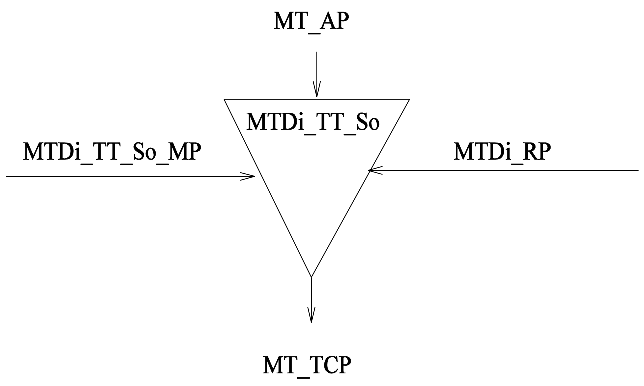
functions as shown in the figure below.

9.4.2.1.1 MPLS-TP MIP Diagnostic Trail Termination Source function (MTDi_TT_So)

The MTDi_TT_So function adds MPLS-TP OAM to the MT_AI signal at its MT_AP.

The information flow and processing of the MTDi_TT_So function is defined with reference to the figure below.

• Symbol:



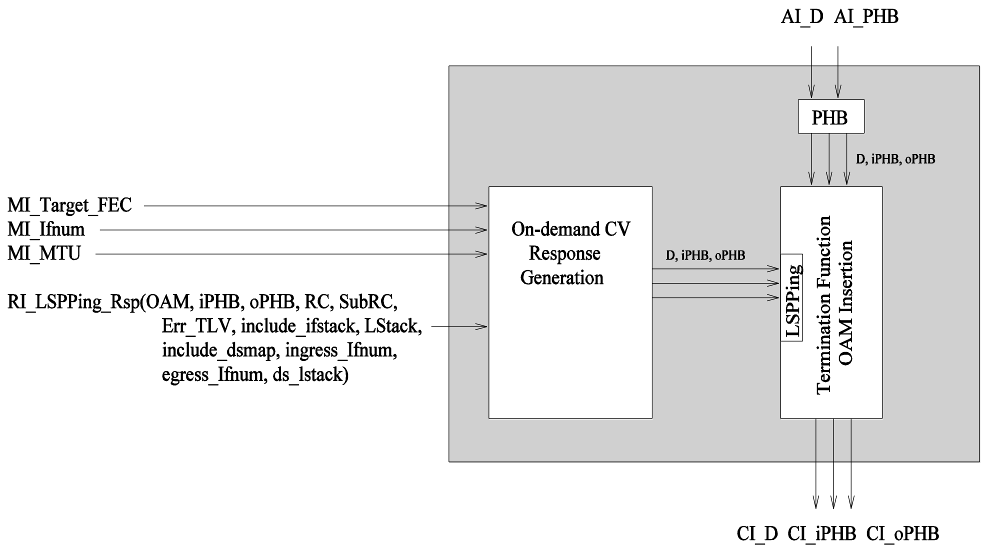
• Interfaces:

Table 9-*/G.8121.2/Y.1381.2 – MTDi_TT_So inputs and outputs

Input(s)	Output(s)
MT_AP: MT_AI_D MT_AI_PHB **MTDi_RP:** MTDi_RI_LSPPing_Rsp **MTDi_TT_So_MP:** MTDi_TT_So_MI_Target_FEC MTDi_TT_So_MI_Ifnum MTDi_TT_So_MI_MTU	**MT_CP:** MT_CI_D MT_CI_oPHB MT_CI_iPHB

• Processes:

The processes associated with the MTDi_TT_So function are as depicted in the Figure below.


AI_D AI_PHB
PHB
D, iPHB, oPHB
MI_Target_FEC
MI_Ifnum
MI_MTU
RI_LSPPing_Rsp(OAM, iPHB, oPHB, RC, SubRC, Err_TLV, include_ifstack, LStack, include_dsmap, ingress_Ifnum, egress_Ifnum, ds_lstack)
On-demand CV Response Generation
D, iPHB, oPHB
LSPPing
Termination Function OAM Insertion
CI_D CI_iPHB CI_oPHB

PHB: The AI_PHB signal is assigned to both the CI_iPHB and CI_oPHB signals at the MT_TCP reference point.

Insert TTL: The Time To Live value is inserted in the outer shim header's TTL field within the MT_AI traffic unit

Temination Function OAM Insertion: See 8.1.1.1

On-demand CV Response Generation: See 8.1.4.7

• Defects:

None.

• Consequent actions:

None.

• Defect correlations:

None.

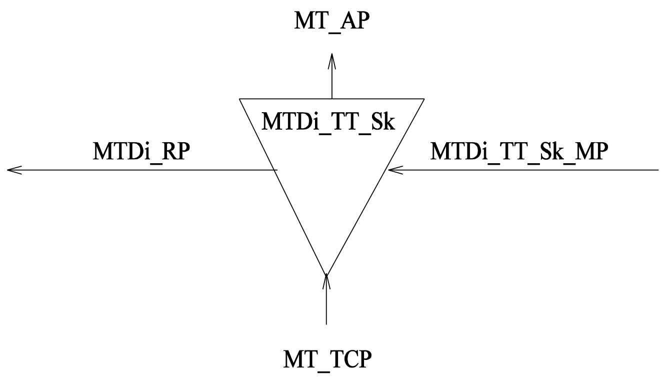
• Performance monitoring:

None.

9.4.2.1.2 MPLS-TP MIP Diagnostic Trail Termination Sink function (MTDi_TT_Sk)

The information flow and processing of the MTDi_TT_Sk function is defined with reference to the Figure below.

• Symbol:



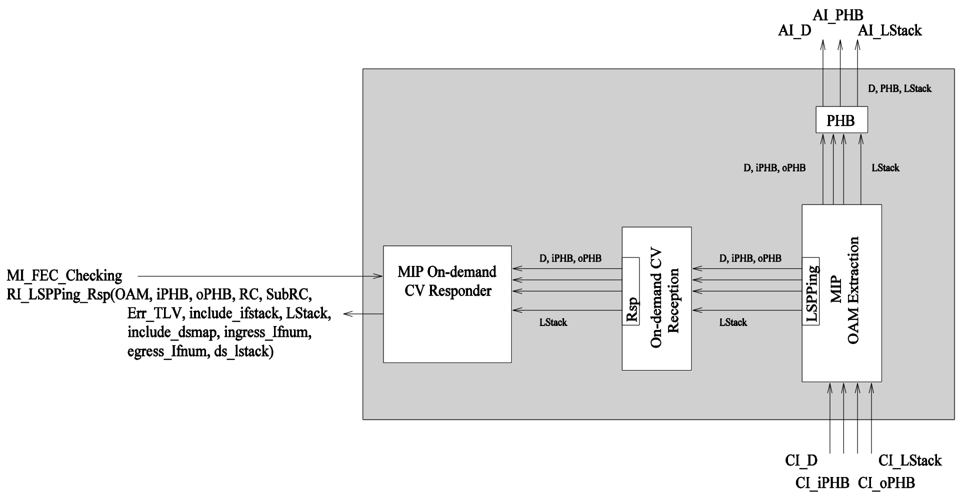
• **Interfaces:**

Table x/G.8121.2/Y.1381.2 – MTDi_TT_Sk inputs and outputs

Input(s)	Output(s)
MT_TCP: MT_CI_D MT_CI_iPHB MT_CI_oPHB MT_CI_LStack **MTDi_TT_Sk_MP:** MTDi_TT_Sk_MI_FEC_Checking	**MT_AP:** MT_AI_D MT_AI_PHB MT_AI_LStack **MTDi_RP:** MTDi_RI_LSPPing_Rsp

• **Processes:**

The processes associated with the MTDi_TT_Sk function are as depicted in the figure below.



PHB: The CI_oPHB signal is assigned to the AI_PHB signal at the reference point MT_AP. Note that the CI_iPHB signal is not used by any of the processes in the function.

MIP OAM Extraction: see 8.1.1.3

On-demand CV Reception: see 8.1.4.4

MIP On-demand CV Responder: see 8.1.4.5

• **Defects:**

None

• **Consequent actions:**

None

• **Defect correlations:**

None

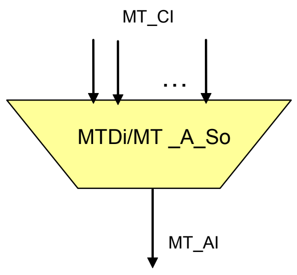
• **Performance monitoring:**

None

9.4.2.2 MPLS-TP MIP Diagnostic Adaptation function (MTDi/MT_A)

9.4.2.2.1 MPLS-TP to MPLS-TP adaptation source function (MTDi/MT_A_So)

This function maps client MT_CI traffic units into server MT_AI traffic units.



• **Interfaces:**

Table x/G.8121.2/Y.1381.2 – MTDi/MT_A_So interfaces

Inputs	Outputs
Each MT_CP: MT_CI_Data MT_CI_iPHB MT_CI_oPHB	**MT_AP:** MT_AI_Data MT_AI_PHB

• **Processes:**

The MTDi/MT_A_So function maps MT_CI_Data to MT_AI_Data and MT_CI_oPHB to MT_AI_PHB.

• **Defects:**

None.

• **Consequent actions:**

None.

• **Defect correlations:**

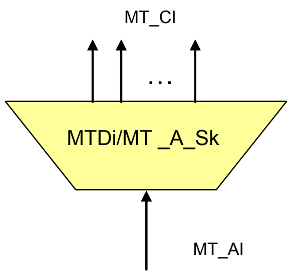
None.

• **Performance monitoring:**

None.

9.4.2.2.2 MPLS-TP to MPLS-TP adaptation sink function (MTDi/MT_A_Sk)

This function retrieves client MT_CI traffic units from server MT_AI traffic units.



• Interfaces:

Table x/G.8121.2/Y.1381.2 – MTDi/MT_A_Sk interfaces

Inputs	Outputs
MT_AP: MT_AI_Data MT_AI_PHB MT_AI_LStack	**Each MT_CP:** MT_CI_Data MT_CI_iPHB MT_CI_oPHB MT_CI_LStack

• Processes:

The MTDi/MT_A_Sk function maps MT_AI_Data to MT_CI_Data, MT_AI_PHB to MT_CI_oPHB and MT_CI_iPHB, and MT_AI_LStack to MT_CT_LStack.

• Defects:

None.

• Consequent actions:

None.

• Defect correlations:

None.

• Performance monitoring:

None.

10 MPLS-TP to Non-MPLS-TP client adaptation functions

This atomic functions are defined in clause 10 in G.8121.

11 Non-MPLS-TP Server to MPLS-TP adaptation functions

These atomic functions are defined in clause 11 in G.8121. They use the OAM protocol specific AIS insertion process and LCK generation process as defined in clause 8.6.2 and 8.6.3.