

MPLS Working Group  
Internet Draft  
Document: [draft-ietf-mpls-ldp-ft-04.txt](#)  
Expiration Date: January 2003

Adrian Farrel  
Movaz Networks, Inc.

Paul Brittain  
Data Connection Ltd.

Philip Matthews  
Hyperchip

Eric Gray  
Celox Networks

Toby Smith  
Laurel Networks

Andrew G. Malis  
Jack Shaio  
Vivace Networks

July 2002

## **Fault Tolerance for the Label Distribution Protocol (LDP)**

[draft-ietf-mpls-ldp-ft-04.txt](#)

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#) [[RFC2026](#)].

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>.

NOTE: The new TLV type numbers, bit values for flags specified in this draft, and new LDP status code values are preliminary suggested values and have yet to be approved by IANA or the MPLS WG. See the section "IANA

Considerations" for further details.

Farrel, et al.

[Page 1]

## Abstract

MPLS systems will be used in core networks where system downtime must be kept to an absolute minimum. Many MPLS LSRs may, therefore, exploit Fault Tolerant (FT) hardware or software to provide high availability of the core networks.

The details of how FT is achieved for the various components of an FT LSR, including LDP, the switching hardware and TCP, are implementation specific. This document identifies issues in the LDP specification [[RFC3036](#)] that make it difficult to implement an FT LSR using the current LDP protocols, and proposes enhancements to the LDP specification to ease such FT LSR implementations.

The issues and extensions described here are equally applicable to CR-LDP [[RFC3212](#)].

## Contents

1. Conventions and Terminology used in this document	3
2. Introduction	4
2.1. Fault Tolerance for MPLS	5
2.2. Issues with LDP and CR-LDP	5
3. Overview of LDP FT Enhancements	7
3.1. Establishing an FT LDP Session	8
3.1.1 Interoperation with Non-FT LSRs	8
3.2. TCP Connection Failure	9
3.2.1 Detecting TCP Connection Failures	9
3.2.2 LDP Processing after Connection Failure	9
3.3. Data Forwarding During TCP Connection Failure	10
3.4. FT LDP Session Reconnection	10
3.5. Operations on FT Labels	11
3.6. Check-Pointing	12
3.6.1 Graceful Termination	13
3.7. Label Space Depletion and Replenishment	13
4. FT Operations	14
4.1. FT LDP Messages	14
4.1.1 Sequence Numbered FT Label Messages	14
4.1.2 FT Address Messages	15
4.1.3 Label Resources Available Notifications	15
4.2. FT Operation ACKs	17
4.3. Preservation of FT State	17
4.4. FT Procedure After TCP Failure	19
4.4.1 FT LDP Operations During TCP Failure	20
4.5. FT Procedure After TCP Re-connection	21
4.5.1 Re-Issuing FT Messages	22

4.5.2 Interaction with CR-LDP LSP Modification	22
5. Checkpointing Procedures	23
5.1. Checkpointing with the Keepalive Message	23
5.2. Quiesce and Keepalive	24

6. Changes to Existing Messages	24
6.1. LDP Initialization Message	24
6.2. LDP Keepalive Messages	25
6.3. All Other LDP Session Messages	25
7. New Fields and Values	26
7.1. Status Codes	26
7.2. FT Session TLV	27
7.3. FT Protection TLV	30
7.4. FT ACK TLV	32
7.5. FT Cork TLV	34
8. Example Use	35
8.1. Session Failure and Recovery - FT Procedures	36
8.2. Use of Check-Pointing With FT Procedures	38
8.3. Temporary Shutdown With FT Procedures	40
8.4. Temporary Shutdown With FT Procedures and Check-Pointing	42
8.5. Checkpointing Without FT Procedures	44
8.6. Graceful Shutdown With Checkpointing But No FT Procedures	46
9. Security Considerations	47
10. Implementation Notes	49
10.1. FT Recovery Support on Non-FT LSRs	49
10.2. ACK generation logic	49
10.2.1 Ack Generation Logic When Using Check-Pointing	49
11. Acknowledgments	50
12. Intellectual Property Consideration	50
13. Full Copyright Statement	51
14. IANA Considerations	51
14.1. New TLVs	52
14.2. New Status Codes	52
15. Authors' Addresses	53
16. References	53
16.1. Normative References	53
16.2. Informative References	53

## **1. Conventions and Terminology used in this document**

Definitions of key words and terms applicable to LDP and CR-LDP are inherited from [[RFC3212](#)] and [[RFC3036](#)].

The term "FT Label" is introduced in this document to indicate a label for which some fault tolerant operation is used. A "non-FT Label" is not fault tolerant and is handled as specified in [[RFC3212](#)] and [[RFC3036](#)].

The term "Sequence Numbered FT Label" is used to indicate an FT label which is secured using the sequence number in the FT Protection TLV described in this document.

The term "Checkpointable FT Label" is used to indicate an

FT label which is secured by using the checkpointing techniques described in this document.

The extensions to LDP specified in this document are collectively referred to as the "LDP FT enhancements".

Within the context of this draft, "Checkpointing" refers to a process of messages exchanges that confirm receipt and processing (or secure storage) of specific protocol messages.

In the examples quoted, the following notation is used:

Ln : An LSP. For example L1.

Pn : An LDP peer. For example P1.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC-2119](#) [[RFC2119](#)].

## **2. Introduction**

High Availability (HA) is typically claimed by equipment vendors when their hardware achieves availability levels of at least 99.999% (five 9s). To implement this, the equipment must be capable of recovering from local hardware and software failures through a process known as fault tolerance (FT).

The usual approach to FT involves provisioning backup copies of hardware and/or software. When a primary copy fails, processing is switched to the backup copy. This process, called failover, should result in minimal disruption to the Data Plane.

In an FT system, backup resources are sometimes provisioned on a one-to-one basis (1:1), sometimes as many-to-one (1:n), and occasionally as many-to-many (m:n). Whatever backup provisioning is made, the system must switch to the backup automatically on failure of the primary, and the software and hardware state in the backup must be set to replicate the state in the primary at the point of failure.





## **2.1. Fault Tolerance for MPLS**

MPLS will be used in core networks where system downtime must be kept to an absolute minimum. Many MPLS LSRs may, therefore, exploit FT hardware or software to provide high availability of core networks.

In order to provide HA, an MPLS system needs to be able to survive a variety of faults with minimal disruption to the Data Plane, including the following fault types:

- failure/hot-swap of a physical connection between LSRs
- failure/hot-swap of the switching fabric in the LSR
- failure of the TCP or LDP stack in an LSR
- software upgrade to the TCP or LDP stacks.

The first two examples of faults listed above are confined to the Data Plane. Such faults can be handled by providing redundancy in the Data Plane which is transparent to LDP operating in the Control Plane. The last two example types of fault require action in the Control Plane to recover from the fault without disrupting traffic in the Data Plane. This is possible because many recent router architectures separate the Control and Data Planes such that forwarding can continue unaffected by recovery action in the Control Plane.

## **2.2. Issues with LDP and CR-LDP**

LDP and CR-LDP use TCP to provide reliable connections between LSRs over which to exchange protocol messages to distribute labels and to set up LSPs. A pair of LSRs that have such a connection are referred to as LDP peers.

TCP enables LDP and CR-LDP to assume reliable transfer of protocol messages. This means that some of the messages do not need to be acknowledged (for example, Label Release).

LDP and CR-LDP are defined such that if the TCP connection fails, the LSR should immediately tear down the LSPs associated with the session between the LDP peers, and release any labels and resources assigned to those LSPs.

It is notoriously hard to provide a Fault Tolerant

implementation of TCP. To do so might involve making copies of all data sent and received. This is an issue familiar to implementers of other TCP applications such as BGP.

During failover affecting the TCP or LDP stacks, therefore, the TCP connection may be lost. Recovery from this position is made worse by the fact that LDP or CR-LDP control messages may have been lost during the connection failure. Since these messages are unconfirmed, it is possible that LSP or label state information will be lost.

This draft describes a solution which involves

- negotiation between LDP peers of the intent to support extensions to LDP that facilitate recovery from failover without loss of LSPs
- selection of FT survival on a per LSP/label basis
- acknowledgement of LDP messages to ensure that a full handshake is performed on those messages either frequently (such as per message) or less frequently as in checkpointing
- solicitation of up-to-date acknowledgement (checkpointing) of previous LDP messages to ensure the current state is flushed to disk/NVRAM, with an additional option that allows an LDP partner to request that state is flushed in both directions if graceful shutdown is required.
- re-issuing lost messages after failover to ensure that LSP/label state is correctly recovered after reconnection of the LDP session.

Other objectives of this draft are to

- offer back-compatibility with LSRs that do not implement these proposals
- preserve existing protocol rules described in [[RFC3212](#)] and [[RFC3036](#)] for handling unexpected duplicate messages and for processing unexpected messages referring to unknown LSPs/labels
- integrate with the LSP modification function described in [[RFC3214](#)]
- avoid full state refresh solutions (such as those present in RSVP: see [[RFC2205](#)], [[RFC2961](#)], [[RFC3209](#)] and [LDP-RESTART]) whether they be full-time, or limited to post-failover recovery.



Note that this draft concentrates on the preservation of label state for labels exchanged between a pair of adjacent LSRs when the TCP connection between those LSRs is lost. This is a requirement for Fault Tolerant operation of LSPs, but a full implementation of end-to-end protection for LSPs requires that this is combined with other techniques that are outside the scope of this draft.

In particular, this draft does not attempt to describe how to modify the routing of an LSP or the resources allocated to a label or LSP, which is covered by [\[RFC3214\]](#). This draft also does not address how to provide automatic layer 2/3 protection switching for a label or LSP, which is a separate area for study.

This specification does not preclude an implementation from attempting (or require it to attempt) to use the FT behavior described here to recover from a preemptive failure of a connection on a non-FT system due to, for example, a partial system crash. Note, however, that there are potential issues too numerous to list here - not least the likelihood that the same crash will immediately occur when processing the restored data.

### **3. Overview of LDP FT Enhancements**

The LDP FT enhancements consist of the following main elements, which are described in more detail in the sections that follow.

- The presence of an FT Session TLV on the LDP Initialization message indicates that an LSR supports some form of protection or recovery from session failure. A flag bit within this TLV (the S bit) indicates that the LSR supports the LDP FT enhancements on this session. Another flag (the C bit) indicates that the checkpointing procedures are to be used.
- An FT Reconnect Flag in the FT Session TLV indicates whether an LSR has preserved FT Label state across a failure of the TCP connection.
- An FT Reconnection Timeout, exchanged on the LDP Initialization message, that indicates the maximum time peer LSRs will preserve FT Label state after a failure of the TCP connection.
- An FT Protection TLV used to identify operations that

affect LDP labels. All LDP messages carrying the FT Protection TLV need to be secured (e.g. to NVRAM) and ACKed to the sending LDP peer in order that the state for Sequence Numbered FT Labels can be correctly recovered after LDP session reconnection.

Note that the implementation within an FT system is left open by this draft. An implementation could choose to secure entire messages relating to Sequence Numbered FT Labels, or it could secure only the relevant state information.

- Address advertisement may also be secured by use of the FT Protection TLV. This enables recovery after LDP session reconnection without the need to re-advertise what may be a very large number of addresses.
- The FT Protection TLV may also be used on the Keepalive message to flush acknowledgement of all previous FT operations. This enables a check-point for future recovery, either in mid-session or prior to graceful shutdown of an LDP session. This procedure may also be used to checkpoint all (that is both FT and non-FT) operations for future recovery.

### **3.1. Establishing an FT LDP Session**

In order that the extensions to LDP [[RFC3036](#)] and CR-LDP [[RFC3212](#)] described in this draft can be used successfully on an LDP session between a pair of LDP peers, they MUST negotiate that the LDP FT enhancements are to be used on the LDP session.

This is done on the LDP Initialization message exchange using a new FT Session TLV. Presence of this TLV indicates that the peer wants to support some form of protection or recovery processing. The S bit within this TLV indicates that the peer wants to support the LDP FT enhancements on this LDP session. The C bit indicates that the peer wants to support the checkpointing functions described in this draft. The S and C bits may be set independently.

The relevant LDP FT enhancements MUST be supported on an LDP session if both LDP peers include an FT Session TLV on the LDP Initialization message and have the same setting of the S or C bit.

If either LDP Peer does not include the FT Session TLV LDP Initialization message or if there is no match of S and C bits between the peers, the LDP FT enhancements MUST NOT be used during this LDP session. Use of LDP FT enhancements by a sending LDP peer MUST be interpreted by the receiving LDP peer as a serious protocol error

causing the session to be terminated.

An LSR MAY present different FT/non-FT behavior on different TCP connections, even if those connections are successive instantiations of the LDP session between the same LDP peers.



### **3.1.1 Interoperation with Non-FT LSRs**

The FT Session TLV on the LDP Initialization message carries the U-bit. If an LSR does not support any protection or recovery mechanisms, it will ignore this TLV. Since such partners also do not include the FT Session TLV, all LDP sessions to such LSRs will not use the LDP FT enhancements.

The rest of this draft assumes that the LDP sessions under discussion are between LSRs that do support the LDP FT enhancements, except where explicitly stated otherwise.

## **3.2. TCP Connection Failure**

### **3.2.1 Detecting TCP Connection Failures**

TCP connection failures may be detected and reported to the LDP component in a variety of ways. These should all be treated in the same way by the LDP component.

- Indication from the management component that a TCP connection or underlying resource is no longer active.
- Notification from a hardware management component of an interface failure.
- Sockets keepalive timeout.
- Sockets send failure.
- New (incoming) Socket opened.
- LDP protocol timeout.

### **3.2.2 LDP Processing after Connection Failure**

If the LDP FT enhancements are not in use on an LDP session, the action of the LDP peers on failure of the TCP connection is as specified in [[RFC3212](#)] and [[RFC3036](#)].

All state information and resources associated with non-FT Labels MUST be released on the failure of the TCP connection, including deprogramming the non-FT Label from the switching hardware. This is equivalent to the behavior specified in [[RFC3036](#)].



If the LDP FT enhancements are in use on an LDP session, both LDP peers SHOULD preserve state information and resources associated with FT Labels exchanged on the LDP session. Both LDP peers SHOULD use a timer to release the preserved state information and resources associated with FT-labels if the TCP connection is not restored within a reasonable period. The behavior when this timer expires is equivalent to the LDP session failure behavior described in [[RFC3036](#)].

The FT Reconnection Timeout each LDP peer intends to apply to the LDP session is carried in the FT Session TLV on the LDP Initialization messages. Both LDP peers MUST use the value that corresponds to the lesser timeout interval of the two proposed timeout values from the LDP Initialization exchange, where a value of zero is treated as positive infinity.

### **[3.3.](#) Data Forwarding During TCP Connection Failure**

An LSR that implements the LDP FT enhancements SHOULD preserve the programming of the switching hardware across a failover. This ensures that data forwarding is unaffected by the state of the TCP connection between LSRs.

It is an integral part of FT failover processing in some hardware configurations that some data packets might be lost. If data loss is not acceptable to the applications using the MPLS network, the LDP FT enhancements described in this draft SHOULD NOT be used.

### **[3.4.](#) FT LDP Session Reconnection**

When a new TCP connection is established, the LDP peers MUST exchange LDP Initialization messages. When a new TCP connection is established after failure, the LDP peers MUST re-exchange LDP Initialization messages.

If an LDP peer includes the FT Session TLV with the S bit set in the LDP Initialization message for the new instantiation of the LDP session, it MUST also set the FT Reconnect Flag according to whether it has been able to preserve label state. The FT Reconnect Flag is carried in the FT Session TLV.

If an LDP peer has preserved all state information for previous instantiations of the LDP session, then it

SHOULD set the FT Reconnect Flag to 1 in the FT Session TLV. Otherwise, it MUST set the FT Reconnect Flag to 0.

If either LDP peer sets the FT Reconnect Flag to 0, or omits the FT Session TLV, both LDP peers MUST release any state information and resources associated with the previous instantiation of the LDP session between the same LDP peers, including FT Label state and Addresses. This ensures that network resources are not permanently lost by one LSR if its LDP peer is forced to undergo a cold start.

If an LDP peer changes any session parameters (for example, the label space bounds) from the previous instantiation the nature of any preserved labels may have changed. In particular, previously allocated labels may now be out of range. For this reason, session reconnection MUST use the same parameters as were in use on the session before the failure. If an LDP peer notices that the parameters have been changed by the other peer it SHOULD send a Notification message with the 'FT Session parameters changed' status code.

If both LDP peers set the FT Reconnect Flag to 1, both LDP peers MUST use the procedures indicated in this draft to complete any label operations on Sequence Numbered FT Labels that were interrupted by the LDP session failure.

If an LDP peer receives an LDP Initialization message with the FT Reconnect Flag set before it sends its own Initialization message, but has retained no information about the previous version of the session, it MUST respond with an Initialization message with the FT Reconnect Flag clear. If an LDP peer receives an LDP Initialization message with the FT Reconnect Flag set in response to an Initialization message that it has sent with the FT Reconnect Flag clear it MUST act as if no state was retained by either peer on the session.

### **3.5. Operations on FT Labels**

Label operations on Sequence Numbered FT Labels are made Fault Tolerant by providing acknowledgement of all LDP messages that affect Sequence Numbered FT Labels. Acknowledgements are achieved by means of sequence numbers on these LDP messages.

The message exchanges used to achieve acknowledgement of label operations and the procedures used to complete interrupted label operations are detailed in the section "FT Operations".

Using these acknowledgements and procedures, it is not necessary for LDP peers to perform a complete re-synchronization of state for all Sequence Numbered FT Labels, either on re-connection of the LDP session between the LDP peers or on a timed basis.

### **3.6. Check-Pointing**

Check-pointing is a useful feature that allows nodes to reduce the amount of processing that they need to do to acknowledge LDP messages. The C bit in the FT Session TLV is used to indicate that checkpointing is supported.

Under the normal operation on Sequence Numbered FT Labels, acknowledgments may be deferred during normal processing and only sent periodically. Check-pointing may be used to flush acknowledgement from a peer by including a sequence number on a Keepalive message requesting acknowledgement of that message and all previous messages. In this case, all Sequence Numbered FT Labels are Checkpointable FT Labels.

If the S bit is not agreed upon, checkpointing may still be used. In this case it is used to acknowledge all messages exchanged between the peers, and all labels are Checkpointable FT Labels.

This offers an approach where acknowledgements need not be sent to every message or even frequently, but are only sent as check-points in response to requests carried on Keepalive messages. Such an approach may be considered optimal in systems that do not show a high degree of change over time (such as targeted LDP session or CR-LDP systems) and that are prepared to risk loss of state for the most recent LDP exchanges. More dynamic systems (such as LDP discovery sessions) are more likely to want to acknowledge state changes more frequently so that the maximum amount of state can be preserved over a failure.

Note that an important consideration of this draft is that nodes acknowledging messages on a one-for-one basis, nodes deferring acknowledgements, and nodes relying on check-pointing should all interoperate seamlessly and without protocol negotiation beyond session initialization.

Further discussion of this feature is provided in the section "FT Operations".





### **3.6.1 Graceful Termination**

A feature that builds on check-pointing is graceful termination.

In some cases, such as controlled failover or software upgrade, it is possible for a node to know in advance that it is going to terminate its session with a peer.

In these cases the node that intends terminating the session can flush acknowledgement using a check-point request as described above. The sender SHOULD not send further label or address-related messages after requesting shutdown check-pointing in order to preserve the integrity of its saved state.

This, however, only provides for acknowledgement in one direction, and the node that is terminating also requires to know that it has secured all state sent by its peer. This is achieved by a three-way hand shake of the check-point which is requested by an additional TLV (the Cork TLV) in the Keepalive message.

Further discussion of this feature is provided in the section "FT Operations".

### **3.7. Label Space Depletion and Replenishment**

When an LDP peer is unable to satisfy a Label Request message because it has no more available labels, it sends a Notification message carrying the status code 'No label resources'. This warns the requesting LDP peer that subsequent Label Request messages are also likely to fail for the same reason. This message does not need to be acknowledged for FT purposes since Label Request messages sent after session recovery will receive the same response. However, the LDP peer that receives a 'No label resources' Notification stops sending Label Request messages until it receives a 'Label resources available' Notification message. Since this unsolicited Notification might get lost during session failure, it may be protected using the procedures described in this draft.

An alternative approach allows that an implementation may always assume that labels are available when a session is re-established. In this case, it is possible that it may throw away the 'No label resources' information from the previous incarnation of the session and may send a batch

of LDP messages on session re-establishment that will fail and that it could have known would fail.

Note that the sender of a 'Label resources available' Notification message may choose whether to add a sequence number requesting acknowledgement. Conversely, the receiver of 'Label resources available' Notification message may choose to acknowledge the message without actually saving any state.

This is an implementation choice made possible by making the FT parameters on the Notification message optional. Implementations will interoperate fully if they take opposite approaches, but additional LDP messages may be sent unnecessarily on session recovery.

#### **4. FT Operations**

Once an FT LDP session has been established, using the S bit in the FT Session TLV on the Session Initialization message as described in the section "Establishing an FT LDP Session", both LDP peers MUST apply the procedures described in this section for FT LDP message exchanges.

If the LDP session has been negotiated to not use the LDP FT enhancements, these procedures MUST NOT be used.

##### **4.1. FT LDP Messages**

###### **4.1.1 Sequence Numbered FT Label Messages**

A label is identified as being a Sequence Numbered FT Label if the initial Label Request or Label Mapping message relating to that label carries the FT Protection TLV.

It is a valid implementation option to flag all labels as Sequence Numbered FT Labels. Indeed this may be a preferred option for implementations wishing to use Keepalive messages carrying the FT Protection TLV to achieve periodic saves of the complete label forwarding state.

If a label is a Sequence Numbered FT Label, all LDP messages affecting that label MUST carry the FT Protection TLV in order that the state of the label can be recovered after a failure of the LDP session.

A valid option is for no labels to be Sequence Numbered FT Labels. In this case checkpointing using the Keepalive message applies to all messages exchanged on

the session.

Farrel, et al.

[Page 14]

#### **4.1.1.1 Scope of FT Labels**

The scope of the FT/non-FT status of a label is limited to the LDP message exchanges between a pair of LDP peers.

In Ordered Control, when the message is forwarded downstream or upstream, the TLV may be present or absent according to the requirements of the LSR sending the message.

If a platform-wide label space is used for FT Labels, an FT Label value MUST NOT be reused until all LDP FT peers to which the label was passed have acknowledged the withdrawal of the FT Label, either by an explicit LABEL WITHDRAW/LABEL RELEASE exchange or implicitly if the LDP session is reconnected after failure but without the FT Reconnect Flag set. In the event that a session is not re-established within the Reconnection Timeout, a label MAY become available for re-use if it is not still in use on some other session.

#### **4.1.2 FT Address Messages**

If an LDP session uses the LDP FT enhancements, both LDP peers MUST secure Address and Address Withdraw messages using FT Operation ACKs, as described below. This avoids any ambiguity over whether an Address is still valid after the LDP session is reconnected.

If an LSR determines that an Address message that it sent on a previous instantiation of a recovered LDP session is no longer valid, it MUST explicitly issue an Address Withdraw for that address when the session is reconnected.

If the FT Reconnect Flag is not set by both LDP peers on reconnection of an LDP session (i.e. state has not been preserved), both LDP peers MUST consider all Addresses to have been withdrawn. The LDP peers SHOULD issue new Address messages for all their valid addresses as specified in [[RFC3036](#)].

#### **4.1.3 Label Resources Available Notifications**

In LDP, it is possible that a downstream LSR may not have labels available to respond to a Label Request. In this case, as specified in [RFC3036](#), the downstream LSR must respond with a Notification - No Label Resources message.

The upstream LSR then suspends asking for new labels until it receives a Notification - Label Resources Available message from the downstream LSR.

When the FT extensions are used on a session, implementations may choose whether to secure the label resource state of their peer or not. This choice impacts the number of LDP messages that will be incorrectly routed to a peer with depleted resources on session re-establishment, but does not otherwise impact interoperability.

For full preservation of state:

- The downstream LSR must preserve the label availability state across a failover so that it remembers to send Notification - Label Resources Available when the resources become available.
- The upstream LSR must recall the label availability state across failover so that it can optimize not sending Label Requests when it recovers.
- The downstream LSR must use sequence numbers on Notification - Label Resources Available so that it can check that LSR A has received the message and clear its secured state, or resend the message if LSR A recovers without having received it.

However, the following options also exist:

- The downstream LSR may choose to not include a sequence number on Notification - Label Resources Available. This means that on session re-establishment it does not know what its peer thinks the LSR's resource state is, because the Notification may or may not have been delivered. Such an implementation MUST begin recovered sessions by sending an additional Notification - Label Resources Available to reset its peer.
- The upstream node may choose not to secure information about its peer's resource state. It would acknowledge a Notification - Label Resources Available, but would not save the information. Such an implementation MUST assume that its peer's resource state has been reset to Label Resources Available when the session is re-established.

If the FT Reconnect Flag is not set by both LDP peers on reconnection of an LDP session (i.e. state has not been preserved), both LDP peers MUST consider the label availability state to have been reset as if the session had been set up for the first time.





#### **4.2. FT Operation ACKs**

Handshaking of FT LDP messages is achieved by use of ACKs. Correlation between the original message and the ACK is by means of the FT Sequence Number contained in the FT Protection TLV, and passed back in the FT ACK TLV. The FT ACK TLV may be carried on any LDP message that is sent on the TCP connection between LDP peers.

An LDP peer maintains a separate FT sequence number for each LDP session it participates in. The FT Sequence number is incremented by one for each FT LDP message (i.e. containing the FT Protection TLV) issued by this LSR on the FT LDP session with which the FT sequence number is associated.

When an LDP peer receives a message containing the FT Protection TLV, it MUST take steps to secure this message (or the state information derived from processing the message). Once the message is secured, it MUST be ACKed. However, there is no requirement on the LSR to send this ACK immediately.

ACKs may be accumulated to reduce the message flow between LDP peers. For example, if an LSR received FT LDP messages with sequence numbers 1, 2, 3, 4, it could send a single ACK with sequence number 4 to ACK receipt and securing of all these messages. There is no protocol reason why the number of ACKs accumulated or the time for which an ACK is deferred should not be allowed to become relatively large.

ACKs MUST NOT be sent out of sequence, as this is incompatible with the use of accumulated ACKs. Duplicate ACKs (that is two successive messages that acknowledge the same sequence number) are acceptable.

If an LDP peer discovers that its sequence number space for a specific session is full of un-acknowledged sequence numbers (because its partner on the session has not acknowledged them in a timely way) it cannot allocate a new sequence number for any further FT LDP message. It SHOULD send a Notification message with the status code "FT Seq Numbers Exhausted".

#### **4.3. Preservation of FT State**

If the LDP FT enhancements are in use on an LDP session, each LDP peer SHOULD NOT release the state information

and resources associated with FT Labels exchanged on that LDP session when the TCP connection fails. This is contrary to [[RFC3212](#)] and [[RFC3036](#)], but allows label operations on FT Labels to be completed after re-connection of the TCP connection.

Both LDP peers on an LDP session that is using the LDP FT enhancements SHOULD preserve the state information and resources they hold for that LDP session as described below.

- An upstream LDP peer SHOULD release the resources (in particular bandwidth) associated with a Sequence Numbered FT Label when it initiates a Label Release or Label Abort message for the label. The upstream LDP peer MUST preserve state information for the Sequence Numbered FT Label, even if it releases the resources associated with the label, as it may need to reissue the label operation if the TCP connection is interrupted.
- An upstream LDP peer MUST release the state information and resources associated with a Sequence Numbered FT Label when it receives an acknowledgement to a Label Release or Label Abort message that it sent for the label, or when it sends a Label Release message in response to a Label Withdraw message received from the downstream LDP peer.
- A downstream LDP peer SHOULD NOT release the resources associated with a Sequence Numbered FT Label when it sends a Label Withdraw message for the label as it has not yet received confirmation that the upstream LDP peer has ceased to send data using the label. The downstream LDP peer MUST NOT release the state information it holds for the label as it may yet have to reissue the label operation if the TCP connection is interrupted.
- A downstream LDP peer MUST release the resources and state information associated with a Sequence Numbered FT Label when it receives an acknowledgement to a Label Withdraw message for the label.
- When the FT Reconnection Timeout expires, an LSR SHOULD release all state information and resources from previous instantiations of the (permanently) failed LDP session.
- Either LDP peer MAY elect to release state information based on its internal knowledge of the loss of integrity of the state information or an inability to pend (or queue) LDP operations (as described in [section 4.4.1](#)) during a TCP failure. That is, the peer is not required to wait for the duration of the FT Reconnection Timeout before releasing state; the timeout provides an upper limit on the persistence of state. However, in the event that a peer releases state before the expiration of the Reconnection Timeout it MUST NOT re-use any label that was in use on the

session until the Reconnection Timeout has expired.

- When an LSR receives a Status TLV with the E-bit set in the status code, which causes it to close the TCP connection, the LSR MUST release all state information and resources associated with the session. This behavior is mandated because it is impossible for the LSR to predict the precise state and future behavior of the partner LSR that set the E-bit without knowledge of the implementation of that partner LSR.

Note that the "Temporary Shutdown" status code does not have the E-bit set, and MAY be used during maintenance or upgrade operations to indicate that the LSR intends to preserve state across a closure and re-establishment of the TCP session.

- If an LSR determines that it must release state for any single FT Label during a failure of the TCP connection on which that label was exchanged, it MUST release all state for all labels on the LDP session.

The release of state information and resources associated with non-FT labels is as described in [[RFC3212](#)] and [[RFC3036](#)].

Note that a Label Release and the acknowledgement to a Label Withdraw may be received by a downstream LSR in any order. The downstream LSR MAY release its resources on receipt of the first message and MUST release its resources on receipt of the second message.

#### **4.4. FT Procedure After TCP Failure**

When an LSR discovers or is notified of a TCP connection failure it SHOULD start an FT Reconnection Timer to allow a period for re-connection of the TCP connection between the LDP peers.

The RECOMMENDED default value for this timer is 5 seconds. During this time, failure must be detected and reported, new hardware may need to be activated, software state must be audited, and a new TCP session must be set up.

Once the TCP connection between LDP peers has failed, the active LSR SHOULD attempt to re-establish the TCP connection. The mechanisms, timers and retry counts to re-establish the TCP connection are an implementation choice. It is RECOMMENDED that any attempt to re-establish the connection take account of the failover

processing necessary on the peer LSR, the nature of the network between the LDP peers, and the FT Reconnection Timeout chosen on the previous instantiation of the TCP connection (if any).

If the TCP connection cannot be re-established within the FT Reconnection Timeout period, the LSR detecting this timeout SHOULD release all state preserved for the failed LDP session. If the TCP connection is subsequently re-established (for example, after a further Hello exchange to set up a new LDP session), the LSR MUST set the FT Reconnect Flag to 0 if it released the preserved state information on this timeout event.

If the TCP connection is successfully re-established within the FT Reconnection Timeout, both peers MUST re-issue LDP operations that were interrupted by (that is, un-acknowledged as a result of) the TCP connection failure. This procedure is described in section "FT Procedure After TCP Re-connection".

The Hold Timer for an FT LDP Session (see [\[RFC3036\]](#) [section 2.5.5](#)) SHOULD be ignored while the FT Reconnection Timer is running. The hold timer SHOULD be restarted when the TCP connection is re-established.

#### **4.4.1 FT LDP Operations During TCP Failure**

When the LDP FT enhancements are in use for an LDP session, it is possible that an LSR may determine that it needs to send an LDP message to an LDP peer but that the TCP connection to that peer is currently down. These label operations affect the state of FT Labels preserved for the failed TCP connection, so it is important that the state changes are passed to the LDP peer when the TCP connection is restored.

If an LSR determines that it needs to issue a new FT LDP operation to an LDP peer to which the TCP connection is currently failed, it MUST pend the operation (e.g. on a queue) and complete that operation with the LDP peer when the TCP connection is restored, unless the label operation is overridden by a subsequent additional operation during the TCP connection failure (see section "FT Procedure After TCP Re-connection").

If, during TCP Failure, an LSR determines that it cannot pend an operation which it cannot simply fail (for example, a Label Withdraw, Release or Abort operation), it MUST NOT attempt to re-establish the previous LDP session. The LSR MUST behave as if the Reconnection Timer expired and release all state information with respect to the LDP peer. An LSR may be unable (or

unwilling) to pend operations; for instance, if a major



routing transition occurred while TCP was inoperable between LDP peers it might result in excessively large numbers of FT LDP Operations. An LSR that releases state before the expiration of the Reconnection Timeout MUST NOT re-use any label that was in use on the session until the Reconnection Timeout has expired.

In ordered operation, received FT LDP operations that cannot be correctly forwarded because of a TCP connection failure MAY be processed immediately (provided sufficient state is kept to forward the label operation) or pended for processing when the onward TCP connection is restored and the operation can be correctly forwarded upstream or downstream. Operations on existing FT Labels SHOULD NOT be failed during TCP session failure.

It is RECOMMENDED that Label Request operations for new FT Labels are not pended awaiting the re-establishment of TCP connection that is awaiting recovery at the time the LSR determines that it needs to issue the Label Request message. Instead, such Label Request operations SHOULD be failed and, if necessary, a notification message containing the "No LDP Session" status code sent upstream.

Label Requests for new non-FT Labels MUST be rejected during TCP connection failure, as specified in [[RFC3212](#)] and [[RFC3036](#)].

#### **4.5. FT Procedure After TCP Re-connection**

The FT operation handshaking described above means that all state changes for Sequence Numbered FT Labels and Address messages are confirmed or reproducible at each LSR.

If the TCP connection between LDP peers fails but is re-connected within the FT Reconnection Timeout, and both LSRs have indicated they will be re-establishing the previous LDP session, both LDP peers on the connection MUST complete any label operations for Sequence Numbered FT Labels that were interrupted by the failure and re-connection of the TCP connection.

The procedures for FT Reconnection Timeout MAY have been invoked as a result of either LDP peer being unable (or unwilling) to pend operations which occurred during the TCP Failure (as described in [section 4.4.1](#)).

If, for any reason, an LSR has been unable to pend operations

with respect to an LDP peer, as described in [section 4.4.1](#), the LSR MUST set the FT Reconnect Flag to 0 on re-connection to that LDP peer indicating that no FT state has been preserved.

Label operations are completed using the procedure described below.

#### **4.5.1 Re-Issuing FT Messages**

On restoration of the TCP connection between LDP peers, any LDP messages for Sequence Numbered FT Labels that were lost because of the TCP connection failure are re-issued. The LDP peer that receives a re-issued message processes the message as if received for the first time.

"Net-zero" combinations of messages need not be re-issued after re-establishment of the TCP connection between LDP peers. This leads to the following rules for re-issuing messages that are not ACKed by the LDP peer on the LDP Initialization message exchange after re-connection of the TCP session.

- A Label Request message MUST be re-issued unless a Label Abort would be re-issued for the same Sequence Numbered FT Label.
- A Label Mapping message MUST be re-issued unless a Label Withdraw message would be re-issued for the same Sequence Numbered FT Label.
- All other messages on the LDP session that carried the FT Protection TLV MUST be re-issued if an acknowledgement had not previously been received.

Any FT Label operations that were pended (see [section 4.4.1](#)) during the TCP connection failure MUST also be issued on re-establishment of the LDP session, except where they form part of a "net-zero" combination of messages according to the above rules.

The determination of "net-zero" FT Label operations according to the above rules MAY be performed on pended messages prior to the re-establishment of the TCP connection in order to optimize the use of queue resources. Messages that were sent to the LDP peer before the TCP connection failure, or pended messages that are paired with them, MUST NOT be subject to such optimization until an FT ACK TLV is received from the LDP peer. This ACK allows the LSR to identify which messages were received by the LDP peer prior to the TCP connection failure.

#### **4.5.2 Interaction with CR-LDP LSP Modification**

Re-issuing LDP messages for FT operation is orthogonal to the use of duplicate messages marked with the Modify ActFlg, as specified in [[RFC3214](#)]. Each time an LSR uses

the modification procedure for an FT LSP to issue a new Label Request message, the FT label operation procedures MUST be separately applied to the new Label Request message to the same degree that they were applied to the original Label Request.

## **5. Checkpointing Procedures**

Checkpointing can be selected independently from the FT procedures described above by using the C bit in the FT Session TLV on the Session Initialization message. Note, however, that checkpointing is an integral part of the FT procedures. Setting the S and the C bit will achieve the same function as setting just the S bit.

If the C bit is set, but the S bit is not set, no label is a Sequence Numbered FT Label. Instead, all labels are Checkpointable FT Labels. Checkpointing is used to synchronize all labels exchanges. No message apart from the checkpoint request and acknowledgement carries an active sequence number. (Note that the Session Initialization message may carry a sequence number to confirm that the checkpoint is still in place).

It is an implementation matter to decide the ordering of received messages and checkpoint requests to ensure that checkpoint acknowledgements are secured.

If the S and C bits are both set, or only the S bit is set, checkpointing applies only to Sequence Numbered FT Labels and to address messages.

The set of all messages that are checkpointed in this way is called the Checkpointable Messages.

### **5.1. Checkpointing with the Keepalive Message**

If an LSR receives a FT Protection TLV on a Keepalive message, this is a request to flush the acknowledgements for all previously received Checkpointable Messages on the session.

As soon as the LSR has completed securing the Checkpointable Messages (or state changes consequent on those messages) received before the Keepalive, it MUST send an acknowledgement to the sequence number of the Keepalive message.

In the case where the FT procedures are in use and acknowledgements have been stored up, this may be immediately on receipt of the Keepalive.

An example message flow showing this use of the Keepalive message to perform a periodic check-point of state is shown in [section 8](#).

An example message flow showing the use of checkpointing without the FT procedures is shown in [section 8](#).

## 5.2. Quiesce and Keepalive

If the Keepalive Message also contains the FT Cork TLV, this indicates that the peer LSR wishes to quiesce the session prior to a graceful restart.

It is RECOMMENDED that on receiving a Keepalive with the FT CORK TLV, an LSR should cease to send any further label or address related messages on the session until it has been disconnected and reconnected, other than any messages generated while processing and securing any previously unacknowledged messages received from the peer requesting the quiesce. It should also attempt to complete this processing and return a Keepalive with the FT ACK TLV as soon as possible in order to allow the session to be quiesced.

An example message flow showing this use of the FT Cork TLV to achieve three-way handshake of state synchronization between two LDP peers is given in [section 8](#).

## 6. Changes to Existing Messages

### 6.1. LDP Initialization Message

The LDP FT enhancements add the following optional parameters to a LDP Initialization message

Optional Parameter	Length	Value
FT Session TLV	4	See below
FT ACK TLV	4	See below

The encoding for these TLVs is found in Section "New Fields and Values".

#### FT Session

If present, specifies the FT behavior of the LDP session.

#### FT ACK TLV

If present, specifies the last FT message that the sending LDP peer was able to secure prior to the failure of the previous instantiation of the LDP session. This TLV is only present if the FT Reconnect flag is set in the FT Session TLV, in which case this TLV MUST be present.





## 6.2. LDP Keepalive Messages

The LDP FT enhancements add the following optional parameters to a LDP Keepalive message

Optional Parameter	Length	Value
FT Protection TLV	4	See below
FT Cork TLV	0	See below
FT ACK TLV	4	See below

The encoding for these TLVs is found in Section "New Fields and Values".

### FT Protection

If present, specifies FT Sequence Number for the LDP message. When present on a Keepalive message, this indicates a solicited flush of the acknowledgements to all previous LDP messages containing sequence numbers and issued by the sender of the Keepalive on the same session.

### FT Cork

Indicates that the remote LSR wishes to quiesce the LDP session. See [section 5](#) for the recommended action in such cases.

### FT ACK TLV

If present, specifies the most recent FT message that the sending LDP peer has been able to secure.

## 6.3. All Other LDP Session Messages

The LDP FT enhancements add the following optional parameters to all other message types that flow on an LDP session after the LDP Initialization message

Optional Parameter	Length	Value
FT Protection TLV	4	See below
FT ACK TLV	4	See below

The encoding for these TLVs is found in the section "New Fields and Values".

### FT Protection

If present, specifies FT Sequence Number

for the LDP message.

FT ACK

If present, identifies the most recent FT  
LDP message ACKed by the sending LDP peer.

Farrel, et al.

[Page 25]

## 7. New Fields and Values

### 7.1. Status Codes

The following new status codes are defined to indicate various conditions specific to the LDP FT enhancements. These status codes are carried in the Status TLV of a Notification message.

The "E" column is the required setting of the Status Code E-bit; the "Status Data" column is the value of the 30-bit Status Data field in the Status Code TLV.

Note that the setting of the Status Code F-bit is at the discretion of the LSR originating the Status TLV. However, it is RECOMMENDED that the F-bit is not set on Notification messages containing status codes except "No LDP Session" because the duplication of messages SHOULD be restricted to being a per-hop behavior.

Status Code	E	Status Data
No LDP Session	0	0x000000xx
Zero FT seqnum	1	0x000000xx
Unexpected TLV / Session Not FT	1	0x000000xx
Unexpected TLV / Label Not FT	1	0x000000xx
Missing FT Protection TLV	1	0x000000xx
FT ACK sequence error	1	0x000000xx
Temporary Shutdown	0	0x000000xx
FT Seq Numbers Exhausted	1	0x000000xx
FT Session parameters / changed	1	0x000000xx
Unexpected FT Cork TLV	1	0x000000xx

The Temporary Shutdown status code SHOULD be used in place of the Shutdown status code (which has the E-bit set) if the LSR that is shutting down wishes to inform its LDP peer that it expects to be able to preserve FT Label state and to return to service before the FT Reconnection Timer expires.



## 7.2. FT Session TLV

LDP peers can negotiate whether the LDP session between them supports FT extensions by using a new OPTIONAL parameter, the FT Session TLV, on LDP Initialization Messages.

The FT Session TLV is encoded as follows.

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|1|0| FT Session TLV (0x0503) | Length (= 12) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   FT Flags   |   Reserved   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   FT Reconnect Timeout (in milliseconds)   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Recovery Time (in milliseconds)   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

### FT Flags

FT Flags: A 16 bit field that indicates various attributes the FT support on this LDP session. This fields is formatted as follows:

```

      0                               1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|R|   Reserved   |S|A|C|L|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

R: FT Reconnect Flag.

Set to 1 if the sending LSR has preserved state and resources for all FT-labels since the previous LDP session between the same LDP peers, and set to 0 otherwise. See the section "FT LDP Session Reconnection" for details of how this flag is used.

If the FT Reconnect Flag is set, the sending LSR MUST include an FT ACK TLV on the LDP Initialization message.



S: Save State Flag.

Set to 1 if the use of the FT Protection TLV is supported on messages other than the KeepAlive message used for checkpointing (see the C bit). I.e., the S bit indicates that some label on the session may be a Sequence Numbered FT Label.

A: All-Label Protection Required

Set to 1 if all labels on the session MUST be treated as Sequence Numbered FT Labels. This removes from a node the option of treating some labels as FT Labels and some labels as non-FT Labels.

Passing this information may be considered helpful to a peer since it may allow it to make optimizations in its processing.

The A bit only has meaning if the S bit is set.

C: Checkpointing Flag.

Set to 1 to indicate that the checkpointing procedures in this draft are in use.

If the S bit is also set to 1 then the C bit indicates that checkpointing is applied only to Sequence Numbered FT Labels.

If the S bit is set to 0 (zero) then the C bit indicates that checkpointing applies to all labels - all labels are Checkpointable FT Labels.

L: Learn From Network Flag.

Set to 1 if the Fault Recovery procedures of [[LDP-RESTART](#)] are to be used to re-learn state from the network.

It is not valid for all of the S, C and L bits to be zero.



It is not valid for both the L and either the S or C bits to be set to 1.

All other bits in this field are currently reserved and SHOULD be set to zero on transmission and ignored on receipt.

The following table summarizes the settings of these bits.

S	A	C	L	Comments
=====				
0	x	0	0	Invalid
0	x	0	1	See [ <a href="#">LDP-RESTART</a> ]
0	x	1	0	Checkpointing of all labels
0	x	1	1	Invalid
1	0	0	0	Full FT on selected labels
1	1	0	0	Full FT on all labels
1	x	0	1	Invalid
1	x	1	0	Same as (S=1,A=x,C=0,L=0)
1	x	1	1	Invalid.

#### FT Reconnection Timeout

If the S bit or C bit in the FT Flags field is set this indicates the period of time the sending LSR will preserve state and resources for FT Labels exchanged on the previous instantiation of an FT LDP session that has currently failed. The timeout is encoded as a 32-bit unsigned integer number of milliseconds.

A value of zero in this field means that the sending LSR will preserve state and resources indefinitely.

See [section 4.4](#) for details of how this field is used.

If the L bit is set to 1 in the FT Flags field, the meaning of this field is defined in [[LDP-RESTART](#)].

#### Recovery Time

The Recovery Time only has meaning if the L bit is set in the FT Flags. The meaning is defined in [[LDP-RESTART](#)].



### **7.3. FT Protection TLV**

LDP peers use the FT Protection TLV to indicate that an LDP message contains an FT label operation.

The FT Protection TLV MUST NOT be used in messages flowing on an LDP session that does not support the LDP FT enhancements. Its presence in such messages SHALL be treated as a protocol error by the receiving LDP peer which SHOULD send a Notification message with the 'Unexpected TLV Session Not FT' status code. LSRs that do not recognize this TLV SHOULD respond with a Notification message with the 'Unknown TLV' status code.

The FT Protection TLV MAY be carried on an LDP message transported on the LDP session after the initial exchange of LDP Initialization messages. In particular, this TLV MAY optionally be present on the following messages:

- Label Request Messages in downstream on-demand distribution mode
- Label Mapping messages in downstream unsolicited mode
- Keepalive messages used to request flushing of acknowledgement of all previous messages that contained this TLV.

If a label is to be a Sequence Numbered FT Label, then the Protection TLV MUST be present:

- on the Label Request message in DoD mode
- on the Label Mapping message in DU mode
- on all subsequent messages concerning this label.

Here 'subsequent messages concerning this label' means any message whose Label TLV specifies this label or whose Label Request Message ID TLV specifies the initial Label Request message.

If a label is not to be a Sequence Numbered FT Label, then the Protection TLV MUST NOT be present on any of these messages that relate to the label. The presence of the FT TLV on a message relating to a non-FT Label SHALL be treated as a protocol error by the receiving LDP peer which SHOULD send a notification message with the 'Unexpected TLV Label Not FT' status code.



This field MUST be reset to 0x00000001 if either LDP peer does not set the FT

Reconnect Flag on re-establishment of the  
TCP connection.

See the section "FT Operation Acks" for  
details of how this field is used.

Farrel, et al.

[Page 31]

The special use of 0x00000000 is discussed in the section "FT ACK TLV" below.

If an LSR receives an FT Protection TLV on a session that does not support the FT LDP enhancements, it SHOULD send a Notification message to its LDP peer containing the "Unexpected TLV, Session Not FT" status code. LSRs that do not recognize this TLV SHOULD respond with a Notification message with the 'Unknown TLV' status code.

If an LSR receives an FT Protection TLV on an operation affecting a label that it believes is a non-FT Label, it SHOULD send a Notification message to its LDP peer containing the "Unexpected TLV, Label Not FT" status code.

If an LSR receives a message without the FT Protection TLV affecting a label that it believes is a Sequence Numbered FT Label, it SHOULD send a Notification message to its LDP peer containing the "Missing FT Protection TLV" status code.

If an LSR receives an FT Protection TLV containing a zero FT Sequence Number, it SHOULD send a Notification message to its LDP peer containing the "Zero FT Seqnum" status code.

#### 7.4. FT ACK TLV

LDP peers use the FT ACK TLV to acknowledge FT Label operations.

The FT ACK TLV MUST NOT be used in messages flowing on an LDP session that does not support the LDP FT enhancements. Its presence on such messages SHALL be treated as a protocol error by the receiving LDP peer.

The FT ACK TLV MAY be present on any LDP message exchanged on an LDP session after the initial LDP Initialization messages. It is RECOMMENDED that the FT ACK TLV is included on all FT Keepalive messages in order to ensure that the LDP peers do not build up a large backlog of unacknowledged state information.

The FT ACK TLV is encoded as follows.

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-

[Page 32]



## FT ACK Sequence Number

The sequence number for the most recent FT label message that the sending LDP peer has received from the receiving LDP peer and secured against failure of the LDP session. It is not necessary for the sending peer to have fully processed the message before ACKing it. For example, an LSR MAY ACK a Label Request message as soon as it has securely recorded the message, without waiting until it can send the Label Mapping message in response.

ACKs are cumulative. Receipt of an LDP message containing an FT ACK TLV with an FT ACK Sequence Number of 12 is treated as the acknowledgement of all messages from 1 to 12 inclusive (assuming the LDP session started with a sequence number of 1).

This field MUST be set to 0 if the LSR sending the FT ACK TLV has not received any FT label operations on this LDP session. This would apply to LDP sessions to new LDP peers or after an LSR determines that it must drop all state for a failed TCP connection.

See the section "FT Operation Acks" for details of how this field is used.

If an LSR receives a message affecting a label that it believes is a Sequence Numbered FT Label and that message does not contain the FT Protection TLV, it SHOULD send a Notification message to its LDP peer containing the "Missing FT Protection TLV" status code.

If an LSR receives an FT ACK TLV that contains an FT ACK Sequence Number that is less than the previously received FT ACK Sequence Number (remembering to take account of wrapping), it SHOULD send a Notification message to its LDP peer containing the "FT ACK Sequence Error" status code.



### 7.5. FT Cork TLV

LDP peers use the FT Cork TLV on FT Keepalive messages to indicate that they wish to quiesce the LDP session prior to a controlled shutdown and restart, for example during control-plane software upgrade.

The FT Cork TLV is encoded as follows.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|0|0|  FT Cork (0x0505)          |      Length (= 0)          |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

On receipt of a Keepalive message with the FT Cork TLV and the FT Protection TLV, an LSR SHOULD perform the following actions.

- Process and secure any messages from the peer LSR that have sequence numbers less than (accounting for wrap) that contained in the FT Protection TLV on the Keepalive message.
- Send a Keepalive message back to the peer containing the FT Cork TLV and the FT ACK TLV specifying the FT ACK sequence number equal to that in the original Keepalive message (i.e. ACKing all messages up to that point).
- If this LSR has not yet received an FT ACK to all the messages it has sent containing the FT Protection TLV, then also include an FT Protection TLV on the Keepalive sent to the peer LSR. This tells the remote peer that the local LSR has saved state prior to quiesce but is still awaiting confirmation that the remote peer has saved state.
- Cease sending any further state changing messages on this LDP session until it has been disconnected and recovered.

On receipt of a Keepalive message with the FT Cork TLV and an FT ACK TLV that acknowledges the previously sent Keepalive that carried the FT Cork TLV, an LSR knows that quiesce is complete. If the received Keepalive also carries the FT Protection TLV, the LSR must respond with a further Keepalive to complete the 3-way handshake. It SHOULD now send a "Temporary Shutdown" Notification message, disconnect the TCP session and perform whatever control plane actions required this session shutdown.

An example such 3-way handshake for controlled shutdown is given in [section 8](#).

If an LSR receives a message that should not carry the FT Cork TLV, or if the FT Cork TLV is used on a Keepalive message without one of the FT Protection or FT ACK TLVs present, , it SHOULD send a Notification message to its LDP peer containing the "Unexpected FR Cork TLV" status code.

## **8. Example Use**

Consider two LDP peers, P1 and P2, implementing LDP over a TCP connection that connects them, and the message flow shown below.

The parameters shown on each message shown below are as follows:

message (label, senders FT sequence number, FT ACK number)

A "-" for FT ACK number means that the FT ACK TLV is not included on that message. "n/a" means that the parameter in question is not applicable to that type of message.

In the diagrams below, time flows from top to bottom. The relative position of each message shows when it is transmitted. See the notes for a description of when each message is received, secured for FT or processed.



**8.1. Session Failure and Recovery - FT Procedures**

notes	P1	P2
=====	==	==
(1)	Label Request(L1,27,-)	
	----->	
	Label Request(L2,28,-)	
	----->	
(2)	Label Request(L3,93,27)	
	<-----	
(3)		Label Request(L1,123,-)
		----->
		Label Request(L2,124,-)
		----->
(4)		Label Mapping(L1,57,-)
		<-----
	Label Mapping(L1,94,28)	
	<-----	
(5)		Label Mapping(L2,58,-)
		<-----
	Label Mapping(L2,95,-)	
	<-----	
(6)	Address(n/a,29,-)	
	----->	
(7)	Label Request(L4,30,-)	
	----->	
(8)	Keepalive(n/a,-,94)	
	----->	
(9)	Label Abort(L3,96,-)	
	<-----	
(10)	===== TCP Session lost =====	
	:	
(11)	:	Label Withdraw(L1,59,-)
	:	<-----
	:	
(12)	=== TCP Session restored ===	
	LDP Init(n/a,n/a,94)	
	----->	
	LDP Init(n/a,n/a,29)	
	<-----	
(13)	Label Request(L4,30,-)	
	----->	
(14)	Label Mapping(L2,95,-)	
	<-----	
	Label Abort(L3,96,30)	
	<-----	
(15)	Label Withdraw(L1,97,-)	
	<-----	





Notes:

=====

- (1) Assume that the LDP session has already been initialized. P1 issues 2 new Label Requests using the next sequence numbers.
- (2) P2 issues a third Label request to P1. At the time of sending this request, P2 has secured the receipt of the label request for L1 from P1, so it includes an ACK for that message.
- (3) P2 Processes the Label Requests for L1 and L2 and forwards them downstream. Details of downstream processing are not shown in the diagram above.
- (4) P2 receives a Label Mapping from downstream for L1, which it forwards to P1. It includes an ACK to the Label Request for L2, as that message has now been secured and processed.
- (5) P2 receives the Label Mapping for L2, which it forwards to P1. This time it does not include an ACK as it has not received any further messages from P1.
- (6) Meanwhile, P1 sends a new Address Message to P2.
- (7) P1 also sends a fourth Label Request to P2
- (8) P1 sends a Keepalive message to P2, on which it includes an ACK for the Label Mapping for L1, which is the latest message P1 has received and secured at the time the Keepalive is sent.
- (9) P2 issues a Label Abort for L3.
- (10) At this point, the TCP session goes down.
- (11) While the TCP session is down, P2 receives a Label Withdraw Message for L1, which it queues.
- (12) The TCP session is reconnected and P1 and P2 exchange LDP Initialization messages on the recovered session, which include ACKS for the last message each peer received and secured prior to the failure.
- (13) From the LDP Init exchange, P1 determines that it needs to re-issue the Label request for L4.
- (14) Similarly, P2 determines that it needs to re-issue the Label Mapping for L2 and the Label Abort.
- (15) P2 issues the queued Label Withdraw to P1.



**8.2. Use of Check-Pointing With FT Procedures**

notes	P1	P2
=====	==	==
(1)	Label Request(L1,27,-) ----->	
	Label Request(L2,28,-) ----->	
(2)	Label Request(L3,93,-) <-----	
(3)		Label Request(L1,123,-) ----->
		Label Request(L2,124,-) ----->
(4)		Label Mapping(L1,57,-) <-----
	Label Mapping(L1,94,-) <-----	
(5)		Label Mapping(L2,58,-) <-----
	Label Mapping(L2,95,-) <-----	
(6)	Address(n/a,29,-) ----->	
(7)	Label Request(L4,30,-) ----->	
(8)	Keepalive(n/a,31,-) ----->	
(9)	Keepalive(n/a,-,31) <-----	
(10)		Keepalive(n/a,59,124) <-----
(11)		Keepalive(n/a,-,59) ----->



Notes:

=====

Notes (1) through (7) are as in the previous example except note that no acknowledgements are piggy-backed on reverse direction messages. This means that at note (8) there are deferred acknowledgements in both directions on both links.

- (8) P1 wishes to synchronize state with P2. It sends a Keepalive message containing a FT Protection TLV with sequence number 31. Since it is not interested in P2's perception of the state that it has stored, it does not include an FT ACK TLV.
- (9) P2 responds at once with a Keepalive acknowledging the sequence number on the received Keepalive. This tells P1 that P2 has preserved all state/messages previously received on this session.
- (10) P3 wishes to synchronize state with P2. It sends a Keepalive message containing a FT Protection TLV with sequence number 59. P3 also takes this opportunity to get up to date with its acknowledgements to P2 by including an FT ACK TLV acknowledging up to sequence number 124.
- (11) P2 responds at once with a Keepalive acknowledging the sequence number on the received Keepalive.



**8.3. Temporary Shutdown With FT Procedures**

notes	P1	P2
=====	==	==
(1)	Label Request(L1,27,-)	
	----->	
	Label Request(L2,28,-)	
	----->	
(2)	Label Request(L3,93,27)	
	<-----	
(3)		Label Request(L1,123,-)
		----->
		Label Request(L2,124,-)
		----->
(4)		Label Mapping(L1,57,-)
		<-----
	Label Mapping(L1,94,28)	
	<-----	
(5)		Label Mapping(L2,58,-)
		<-----
	Label Mapping(L2,95,-)	
	<-----	
(6)	Address(n/a,29,-)	
	----->	
(7)	Label Request(L4,30,-)	
	----->	
(8)	Keepalive(n/a,-,94)	
	----->	
(9)	Label Abort(L3,96,-)	
	<-----	
(10)	Notification(Temporary shutdown)	
	----->	
	===== TCP Session shutdown =====	
	:	
(11)	:	Label Withdraw(L1,59,-)
	:	<-----
	:	
	===== TCP Session restored =====	
(12)	LDP Init(n/a,n/a,94)	
	----->	
	LDP Init(n/a,n/a,29)	
	<-----	
(13)	Label Request(L4,30,-)	
	----->	
(14)	Label Mapping(L2,95,-)	
	<-----	
	Label Abort(L3,96,30)	
	<-----	
(15)	Label Withdraw(L1,97,-)	

<-----



Notes:

=====

Notes are as in the previous example except as follows.

- (10) P1 needs to upgrade the software or hardware that it is running. It issues a Notification message to terminate the LDP session, but sets the status code as 'Temporary shutdown' to inform P2 that this is not a fatal error, and P2 should maintain FT state. The TCP connection may also fail during the period that the LDP session is down (in which case it will need to be re-established), but it is also possible that the TCP connection will be preserved.



**8.4. Temporary Shutdown With FT Procedures and Check-Pointing**

```

notes      P1      P2
=====
(1)      Label Request(L1,27,-)
          ----->
          Label Request(L2,28,-)
          ----->
(2)      Label Request(L3,93,-)
          <-----
          Label Request(L1,123,-)
          ----->
          Label Request(L2,124,-)
          ----->
          Label Mapping(L1,57,-)
          <-----
(3)      Label Mapping(L1,94,-)
          <-----
          Label Mapping(L2,58,-)
          <-----
          Label Mapping(L2,95,-)
          <-----
(4)      Address(n/a,29,-)
          ----->
(5)      Label Request(L4,30,-)
          ----->
(6)      Keepalive(n/a,31,95) * with FT Cork TLV *
          ----->
(7)      Label Abort(L3,96,-)
          <-----
(8)      Keepalive(n/a,97,31) *with FT Cork TLV *
          <-----
(9)      Keepalive(n/a,-,97) *with FT Cork TLV *
          ----->
(10)     Notification(Temporary shutdown)
          ----->
          ===== TCP Session shutdown =====
          :
          :      Label Withdraw(L1,59,-)
          :      <-----
          :
          ===== TCP Session restored =====
(11)     LDP Init(n/a,n/a,96)
          ----->
          LDP Init(n/a,n/a,31)
          <-----
          Label Withdraw(L1,97,-)
          <-----

```



Notes:

=====

This example operates much as the previous one. However, at (1), (2), (3), (4) and (5) no acknowledgements are made.

At (6), P1 determines that graceful shutdown is required and sends a Keepalive acknowledging all previously received messages and itself containing a FT Protection TLV number and the FT Cork TLV.

The Label abort at (7) crosses with this Keepalive, so at (8) P2 sends a Keepalive that acknowledges all messages received so far, but also including the FT Protection and FT Cork TLVs to indicate that there are still messages outstanding to be acknowledged.

P1 is then able to complete the 3-way handshake at (9) and close the TCP session at (10).

Upon recovery at (11) there are no messages to be re-sent because the Keepalives flushed the acknowledgements. The only messages sent after recovery is the Label Withdraw that was pended during the TCP session failure.



**8.5. Checkpointing Without FT Procedures**

notes	P1	P2
=====	==	==
(1)	Label Request(L1)	
	----->	
(2)	Label Request(L2)	
	<-----	
		Label Request(L1)
		----->
		Label Mapping(L1)
		<-----
(3)	Label Mapping(L1)	
	<-----	
(4)	Keepalive(n/a,12,-)	
	----->	
(5)	Label Request(L3)	
	----->	
(6)	Keepalive(n/a,-,12)	
	<-----	
		Label Request(L3)
		----->
		Label Mapping(L3)
		<-----
(7)	Label Mapping(L3)	
	<-----	
	===== TCP Session failure =====	
	:	
	:	
	:	
	===== TCP Session restored =====	
(8)	LDP Init(n/a,n/a,23)	
	----->	
	LDP Init(n/a,n/a,12)	
	<-----	
(9)	Label Request(L3)	
	----->	
		Label Request(L3)
		----->
		Label Mapping(L3)
		<-----
(10)	Label Mapping(L3)	
	<-----	
(11)	Label Request(L2)	
	<-----	





Notes:

=====

- (1), (2) and (3) show label distribution without FT sequence numbers.
- (4) A checkpoint request from P1. It carries the sequence number of the checkpoint request.
- (5) P1 immediately starts a new label distribution request.
- (6) P2 confirms that it has secured all previous transactions.
- (7) The subsequent (un-acknowledged) label distribution completes.
- (8) The session fails and is restarted. Initialization messages confirm the sequence numbers of the secured checkpoints.
- (9) P1 recommences the unacknowledged label distribution request.
- (10) P2 recommences an unacknowledged label distribution request.



**8.6. Graceful Shutdown With Checkpointing But No FT Procedures**

notes	P1	P2
=====	==	==
(1)	Label Request(L1)	
	----->	
(2)	Label Request(L2)	
	<-----	
		Label Request(L1)
		----->
		Label Mapping(L1)
		<-----
(3)	Label Mapping(L1)	
	<-----	
(4)	Keepalive(n/a,12,23) * With Cork TLV	
	----->	
(5)	:	
	:	
	:	
(6)	Keepalive(n/a,24,12) * With Cork TLV	
	<-----	
(7)	Keepalive(n/a,-,24) * With Cork TLV	
	----->	
(8)	Notification(Temporary shutdown)	
	----->	
	===== TCP Session failure =====	
	:	
	:	
	:	
	===== TCP Session restored =====	
(9)	LDP Init(n/a,n/a,24)	
	----->	
	LDP Init(n/a,n/a,12)	
	<-----	
(10)	Label Request(L3)	
	----->	
		Label Request(L3)
		----->
		Label Mapping(L3)
		<-----
(11)	Label Mapping(L3)	
	<-----	
(12)	Label Mapping(L2)	
	----->	



Notes:

=====

- (1), (2) and (3) show label distribution without FT sequence numbers.
- (4) A checkpoint request from P1. It carries the sequence number of the checkpoint request and a Cork TLV.
- (5) P1 has sent a Cork TLV so quieces.
- (6) P2 confirms the checkpoint and continues the three-way handshake by including a Cork TLV itself.
- (7) P1 completes the three-way handshake. All operations have now been checkpointed and the session is quiesced.
- (8) The session is gracefully shut down.
- (9) The session recovers and the peers exchange the sequence numbers of the last secured checkpoints.
- (10) P1 starts a new label distribution request.
- (11) P1 continues processing a previously received label distribution request.

## **9. Security Considerations**

The LDP FT enhancements inherit similar security considerations to those discussed in [[RFC3212](#)] and [[RFC3036](#)].

The LDP FT enhancements allow the re-establishment of a TCP connection between LDP peers without a full re-exchange of the attributes of established labels, which renders LSRs that implement the extensions specified in this draft vulnerable to additional denial-of-service attacks as follows:

- An intruder may impersonate an LDP peer in order to force a failure and reconnection of the TCP connection, but where the intruder does not set the FT Reconnect Flag on re-connection. This forces all FT labels to be released.
- Similarly, an intruder could set the FT Reconnect Flag on re-establishment of the TCP session without preserving the state and resources for FT labels.
- An intruder could intercept the traffic between LDP peers

and override the setting of the FT Label Flag to be set to 0  
for all labels.

All of these attacks may be countered by use of an authentication scheme between LDP peers, such as the MD5-based scheme outlined in [[RFC3036](#)].

Alternative authentication schemes for LDP peers are outside the scope of this draft, but could be deployed to provide enhanced security to implementations of LDP, CR-LDP and the LDP FT enhancements.

As with LDP and CR-LDP, a security issue may exist if an LDP implementation continues to use labels after expiration of the session that first caused them to be used. This may arise if the upstream LSR detects the session failure after the downstream LSR has released and re-used the label. The problem is most obvious with the platform-wide label space and could result in mis-routing of data to other than intended destinations and it is conceivable that these behaviors may be deliberately exploited to either obtain services without authorization or to deny services to others.

In this draft, the validity of the session may be extended by the FT Reconnection Timeout, and the session may be re-established in this period. After the expiry of the Reconnection Timeout the session must be considered to have failed and the same security issue applies as described above.

However, the downstream LSR may declare the session as failed before the expiration of its Reconnection Timeout. This increases the period during which the downstream LSR might reallocate the label while the upstream LSR continues to transmit data using the old usage of the label. To reduce this issue, this draft requires that labels are not re-used until the Reconnection Timeout has expired.

A further issue might apply if labels were re-used prior to the expiration of the FT Reconnection Timeout, but this is forbidden by this draft.





## **10. Implementation Notes**

### **10.1. FT Recovery Support on Non-FT LSRs**

In order to take full advantage of the FT capabilities of LSRs in the network, it may be that an LSR that does not itself contain the ability to recover from local hardware or software faults still needs to support the LDP FT enhancements described in this draft.

Consider an LSR, P1, that is an LDP peer of a fully Fault Tolerant LSR, P2. If P2 experiences a fault in the hardware or software that serves an LDP session between P1 and P2, it may fail the TCP connection between the peers. When the connection is recovered, the LSPs/labels between P1 and P2 can only be recovered if both LSRs were applying the FT recovery procedures to the LDP session.

### **10.2. ACK generation logic**

FT ACKs SHOULD be returned to the sending LSR as soon as is practicable in order to avoid building up a large quantity of unacknowledged state changes at the LSR. However, immediate one-for-one acknowledgements would waste bandwidth unnecessarily.

A possible implementation strategy for sending ACKs to FT LDP messages is as follows:

- An LSR secures received messages in order and tracks the sequence number of the most recently secured message, Sr.
- On each LDP KeepAlive that the LSR sends, it attaches an FT ACK TLV listing Sr
- Optionally, the LSR may attach an FT ACK TLV to any other LDP message sent between Keepalive messages if, for example, Sr has increased by more than a threshold value since the last ACK sent.

This implementation combines the bandwidth benefits of accumulating ACKs while still providing timely ACKs.

#### **10.2.1 Ack Generation Logic When Using Check-Pointing**

If check-pointing is in use, the LSRs need not be concerned to send ACKs in such a timely manner.

Check-points are solicitations for acknowledgement conveyed as a sequence number in an FT Protection TLV on a Keepalive message. Such check-point requests could be issued on a timer, after a significant amount of change, or before controlled shutdown of a session.

The use of check-pointing may considerably simplify an implementation since it does not need to track the sequence numbers of all received LDP messages. It must, however, still ensure that all received messages (or the consequent state changes) are secured before acknowledging the sequence number on the Keepalive.

This approach may be considered optimal in systems that do not show a high degree of change over time (such as targeted LDP session or CR-LDP systems) and that are prepared to risk loss of state for the most recent LDP exchanges. More dynamic systems (such as LDP discovery sessions) are more likely to want to acknowledge state changes more frequently so that the maximum amount of state can be preserved over a failure.

## **11. Acknowledgments**

The work in this draft is based on the LDP and CR-LDP ideas expressed by the authors of [[RFC3212](#)] and [].

The ACK scheme used in this draft was inspired by the proposal by David Ward and John Scudder for restarting BGP sessions now included in [[BGP-RESTART](#)].

The authors would also like to acknowledge the careful review and comments of Nick Weeds, Piers Finlayson, Tim Harrison, Duncan Archer, Peter Ashwood-Smith, Bob Thomas, S.Manikantan, Adam Sheppard, Alan Davey and Iftekhar Hussain.

## **12. Intellectual Property Consideration**

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information, consult the online list of claimed rights.



### **13. Full Copyright Statement**

Copyright (c) The Internet Society (2000, 2001, 2002). All Rights Reserved. This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

### **14. IANA Considerations**

This draft requires the use of a number of new TLVs and status codes from the number spaces within the LDP protocol. This section explains the logic used by the authors to choose the most appropriate number space for each new entity, and is intended to assist in the determination of any final values assigned by IANA or the MPLS WG in the event that the MPLS WG chooses to advance this draft on the standards track.

This section will be removed when the TLV and status code values have been agreed with IANA.



### 14.1. New TLVs

The FT Protection TLV carries attributes that affect a single label exchanged between LDP peers. It is taken from the 0x02xx range for TLVs that is used in [RFC3036] by other TLVs carrying label attributes. The next available value in this range is 0x0203.

The FT Session TLV carries attributes that affect the entire LDP session between LDP peers. It is taken from the 0x05xx range for TLVs that is used in [RFC3036] by other TLVs carrying session-wide attributes. The next available value in this range is 0x0503.

The FT Protection TLV may ACK many label operations at once if cumulative ACKS are used. It is taken from the 0x05xx range for TLVs that is used in [RFC3036] by other TLVs carrying session-wide attributes. The next available value in this range is 0x0504.

The FT Cork TLV carries attributes that apply to all labels exchanged between LDP peers. It is taken from the 0x05xx range for TLVs that is used in [RFC3036] by other TLVs carrying label attributes. The next available value in this range is 0x0505.

In summary:

FT Protection TLV	0x0203
FT Session TLV	0x0503
FT Ack TLV	0x0504
FT Cork TLV	0x0505

### 14.2. New Status Codes

LDP status codes are not sub-divided into specific ranges for different types of error. Hence, the numeric status code values are selected as the next available.

[Section 7.1](#) lists the new status codes required by this draft and gives interpretative information. The new codes are as follows.

Status Code	E	Status Data
No LDP Session	0	0x0000001A
Zero FT seqnum	1	0x0000001B
Unexpected TLV / Session Not FT	1	0x0000001C
Unexpected TLV / Label Not FT	1	0x0000001D
Missing FT Protection TLV	1	0x0000001E
FT ACK sequence error	1	0x0000001F

Temporary Shutdown	0	0x00000020
FT Seq Numbers Exhausted	1	0x00000021
FT Session parameters / changed	1	0x00000022
Unexpected FT Cork TLV	1	0x00000023

Farrel, et al.

[Page 52]



## **15. Authors' Addresses**

Adrian Farrel (editor)  
Movaz Networks, Inc.  
7926 Jones Branch Drive, Suite 615  
McLean, VA 22102  
Phone: +1 703-847-1867  
Email: [afarrel@movaz.com](mailto:afarrel@movaz.com)

Paul Brittain  
Data Connection Ltd.  
Windsor House, Pepper Street,  
Chester, Cheshire  
CH1 1DF, UK  
Phone: +44-(0)20-8366-1177  
Email: [pjb@dataconnection.com](mailto:pjb@dataconnection.com)

Philip Matthews  
Hyperchip  
1800 Rene-Levesque Blvd W  
Montreal, Quebec H3H 2H2  
Canada  
Phone: +1 514-906-4965  
Email: [pmatthews@hyperchip.com](mailto:pmatthews@hyperchip.com)

Eric Gray  
Celox Networks, Inc.  
2 Park Central Drive,  
Southborough, MA 01772  
Phone: +1 508-305-7214  
Email: [egray@celoxnetworks.com](mailto:egray@celoxnetworks.com)

Jack Shaio  
Vivace Networks  
2730 Orchard Parkway  
San Jose, CA 95134  
Phone: +1 408 432 7623  
Email: [jack.shaio@vivacenetworks.com](mailto:jack.shaio@vivacenetworks.com)

Toby Smith  
Laurel Networks, Inc.  
1300 Omega Drive  
Pittsburgh, PA 15205  
Email: [tob@laurelnetworks.com](mailto:tob@laurelnetworks.com)

Andrew G. Malis  
Vivace Networks  
2730 Orchard Parkway  
San Jose, CA 95134  
Phone: +1 408 383 7223  
[andy.malis@vivacenetworks.com](mailto:andy.malis@vivacenetworks.com)

## **16. References**

### **16.1. Normative References**

- [RFC3036] Andersson, L., et. al., LDP Specification, [RFC 3036](#), January 2001.
- [RFC3212] Jamoussi, B., et. al., Constraint-Based LSP Setup using LDP, [RFC 3212](#), January 2002.

### **16.2. Informative References**

- [RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", [BCP 9](#), [RFC 2026](#), October 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.

[RFC2205] Braden, R., et al., Resource ReSerVation Protocol (RSVP) -- Version 1, Functional Specification, [RFC 2205](#), September 1997.

Farrel, et al.

[Page 53]

- [RFC2961] Berger, L., et al., RSVP Refresh Reduction Extensions, [RFC 2961](#), April 2001.
- [RFC3209] Awduche, D., et al., Extensions to RSVP for LSP Tunnels, [RFC 3209](#), December 2001.
- [RFC3214] Ash, G., et al., LSP Modification Using CR-LDP, [RFC 3214](#), January 2001.
- [BGP-RESTART] Sangli, S., et al., Graceful Restart Mechanism for BGP, [draft-ietf-idr-restart-05.txt](#), June 2002 (work in progress).
- [LDP-RESTART] Leelanivas, M., et al., Graceful Restart Mechanism for LDP, [draft-ietf-ldp-restart-02.txt](#), June 2002, work in progress.

