

# **Integrating Control-Plane and Data-Plane Software Components:**

API specifications for IPv4, IPv6,  
and MPLS Functionality



## Table of Contents

<b>Introduction</b> .....	<b>3</b>
<b>IPv4 Unicast Forward Table Interface</b> .....	<b>3</b>
<i>IPv4 Unicast Data Structures</i> .....	4
<b>IPv4 Address Resolution Protocol (ARP) Table Interface</b> .....	<b>5</b>
<b>IPv4 Multicast Forwarding Table Interface</b> .....	<b>6</b>
<b>IPv6 Unicast, Multicast, and ARP</b> .....	<b>7</b>
<b>MPLS and MPLS-VPN Interface</b> .....	<b>7</b>
<i>ILM/FTN Interface (LDP control-plane to MPLS Forwarder)</i> .....	7
<i>VRF Label Interface (BGP-VPN to MPLS Forwarder)</i> .....	8
<i>Traffic Engineering Label Interface (RSVP-TE and CR-LDP to MPLS Forwarder)</i> .....	9
<b>Conclusion</b> .....	<b>9</b>

## Introduction

Equipment vendors are increasingly looking for off-the-shelf hardware and software components to quickly deliver access, edge, and core routing and switching solutions to market. Many of these components exist, but integrating them into a total solution is still a difficult and custom process. With the range of network processors coming to market, there is still a question of how these processors are integrated with control plane software, such as the ZebOS™ Advanced Routing Suite provided by IP Infusion.

The [Network Processing Forum](#) (NP Forum) was recently established to solve some of these problems. Specifically, the Software Working Group is tasked with developing standards and protocols that integrate control plane routing and switching software with network processing components. As a member of the NP Forum, IP Infusion is helping to develop and promote these standards. Together with a number of partners, IP Infusion has submitted a draft proposal to the NP Forum for an IPv4 Unicast Software API. We intend to assist in developing and promoting standard API's for IPv6, MPLS, and other popular protocols, through the NP Forum.

To support these efforts, IP Infusion has implemented our draft proposal for the IPv4 Unicast Software API and will track and change our implementation to fully support the final standard. This paper discusses our current implementation and describes how control and data plane components can be integrated into a total routing solution. We will also briefly discuss the integration of IPv4 ARP and multicast; IPv6 Unicast, ARP, and Multicast; and MPLS VPN and Traffic Engineering functionality into a network processing environment. For more details on these interface specifications, contact IP Infusion directly or register on our web site to download the appropriate interface documents.

## IPv4 Unicast Forward Table Interface

IP Infusion has designed and implemented a control-plane to data-plane API we refer to as the NPapi. The NPapi has a number of components to provide for IPv4, IPv6, and MPLS control-plane to data-plane interface specifications. We have submitted our NPapi to the NP Forum Software Working Group as a draft standard for the IPv4 Unicast forward table interface. Other vendors have submitted their own versions as well. The Software Task Group is working together to forge a common NP software API from all of these submissions. IP Infusion is committed to fully support and promote whatever standard comes out of the process. In the meantime, we are working with a number of vendors to show interoperability using our NPapi. Diagram 1 below shows the interaction between the ZebOS routing software, the operating system or RTOS, and the network processor.

Essentially, the NPapi for IPv4 unicast defines a set of functions to manage forwarding table updates. The required data structures and functions for each are outlined below.

## IPv4 Unicast Data Structures

IPv4 uses a quad-byte address type. A simple definition would look like the following:

```
typedef quadbyte npfIPv4Address_t;
```

To support the unicast forwarding table, a forwarding table data structure is needed. This would start by defining the unicast next hop entry as follows:

```
typedef struct npfIPv4UCNextHopEntry_s {  
    npfInterfaceHandle_t    egressInterface;  
    npfIPv4Address_t        nextHop;  
} npfIPv4UCNextHopEntry_t;
```

*npfInterfaceHandle\_t egressInterface is the handle for the outbound interface, to which this next hop router is connected or the network/host is directly connected.*

*npfIPv4Address\_t nextHop is the address for the next hop router if the entry does not describe a connected network. This value should be set to all zeros (0's) if it describes a directly connected network.*

Now we can define the forwarding table entry data structure.

```
typedef struct npfIPv4UCForwardTableEntry_s {  
    npfIPv4Address_t        address;  
    npfIPv4Address_t        mask;  
    unsigned int            nextHopCount;  
    npfIPv4UCNextHopEntry_t nextHop [nextHopCount];  
} npfIPv4UCForwardTableEntry_t;
```

*npfIPv4Address\_t address is the network or machine address for this entry.*

*npfIPv4Address\_t mask is the subnet mask indicating which bits are significant in this entry.*

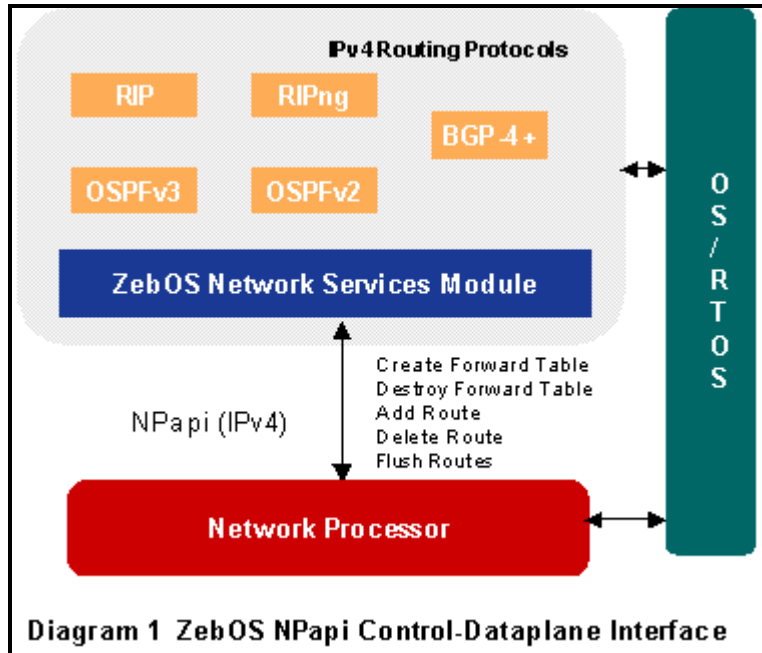
*unsigned int nextHopCount is the number of nextHopEntry's which follow.*

*npfIPv4UCNextHopEntry\_t nextHop[nextHopCount] is the next hop entries. If more than one next hop entry is present, they are treated as equal cost.*

Now that the data structures are defined, we can define the unicast forwarding table update calls. Each of these calls returns a value in *npresult* format (for more detailed information on this, please see our web site ([www.ipinfusion.com](http://www.ipinfusion.com)) to download the more detailed API documents).

- *npfIPv4UCForwardTableCreate (npfIPv4UCForwardTableHandle\_t) - Creates a forwarding table. Returns a handle.*
- *npfIPv4UCForwardTableDestroy (npfIPv4UCForwardTableHandle\_t) – Destroys a forwarding table.*

- *npfUCForwardTableEntryAdd (npfIPv4UCForwardTableHandle\_t, npfIPv4UCForwardTableEntry\*)*– Adds network prefix to the forward table with the list of next hop addresses)
- *npfIPv4UCForwardTableEntryDel (npfIPv4UCForwardTableHandle\_t, npfIPv4UCForwardTableEntry\*)* – Deletes network prefix from the forward table. We are considering adding a capability to delete specific next hop addresses as well, but this hasn't been added yet.
- *npfIPv4UCForwardTableFlush (npfIPv4UCForwardTableHandle\_t)* – This function flushes the forward table without having to destroy and then re-create.



These functions provide a simple mechanism for control-plane to data-plane interaction. Note that providing a forwarding table handle allows multiple forwarding tables to be supported. This is especially important for virtual routing functionality. IP Infusion is working with a number of network processor vendors to implement these functions.

## IPv4 Address Resolution Protocol (ARP) Table Interface

The functions outlined below manage address resolution information on those interfaces, which require some form of address or channel resolution. Again, the data structure for the ARP table needs to be defined.

```
typedef byte[40] npfIPv4LowerAddress_t;

typedef struct npfIPv4ARPTableEntry_s {
    npfIPv4Address_t      address;
    npfIPv4LowerAddress_t lowerAddress;
    quadByte             TTL;
} npfIPv4ARPTableEntry_t;
```

*npfIPv4Address\_t address is the protocol address for this entry.*

*npfIPv4LowerAddress\_t lowerAddress is the 'next level down' address in the case of tunneling, etc...*

*quadByte is the number of seconds of disuses before this entry expires. 0xFFFFFFFF means never expire. Forwarding planes, which do not support automatic expiration of entries, may ignore this value, but must clear their referenced flag on setting an entry. When reading, this is nonzero if this entry has been referenced since the last time it was read or updated.*

With the ARP table data structure defined, we can now define the ARP Table control to data plane interface.

- *npfIPv4UCARPTableCreate (npfIPv4UCARPTableHandle\_t) - Creates an ARP table. Returns a handle.*
- *npfIPv4UCARPTableDestroy (npfIPv4UCARPTableHandle\_t) – Destroys an ARP table.*
- *npfUCARPTableEntryAdd (npfIPv4UCARPTableHandle\_t, npfIPv4UCARPTableEntry\*)– Adds an entry to the ARP table. If a matching entry already exists, then this completely replaces the previous entry with the new data.*
- *npfIPv4UCARPTableEntryDelCond (npfIPv4UCARPTableHandle\_t, npfIPv4UCARPTableEntry\*) – If there is a matching entry in the table, which has not been referenced by a frame since the last time this request was made against the entry, this completely deletes the entry, and returns success.*
- *npfIPv4UCARPTableFlush (npfIPv4UCARPTableHandle\_t) – This function flushes the ARP table without having to destroy and then re-create.*

IPv4 Address Resolution Protocol (ARP) updates use the standard operating system features rather than a specific control-data plane API.

Network equipment manufacturers or network processor vendors that would like to get more information or access to the source code files for the actual API, please contact IP Infusion.

## **IPv4 Multicast Forwarding Table Interface**

We will provide a brief overview of the important attributes of the IPv4 multicast forwarding table interface. For more details on our specification, please contact [IP Infusion](#) directly to access detailed specification documents.

An IPv4 multicast forwarding table entry contains a handle for the outbound interface. At least one multicast client should be connected to the network on this interface, whether this is a downstream router or an actual end client.

The forwarding table data structure itself is composed of the following entries:

- The multicast target group address
- The time-to-live
- The number of source address entries provided
- The number of next hop entries provided
- The source entries
- The next hop entries

The IPv4 multicast interfaces that IP Infusion has defined are similar to the unicast functions. These include the following functions:

- Creating the multicast forwarding table
- Flushing the multicast forwarding table
- Adding an entry to the multicast forwarding table
- Deleting an entry from the multicast forwarding table
- Conditionally removing an entry from the multicast forwarding table

## IPv6 Unicast, Multicast, and ARP

The control-plane/data-plane API for IPv6 will look similar to the one defined above for IPv4 unicast. The difference is that both ARP and multicast are mandatory features in IPv6, whereas they are optional in IPv4. For this, additional tables will need to be defined, and interface functions will be developed to create, destroy, add, delete, and flush the unicast, multicast, and ARP tables.

For more detailed information on the IPv6 unicast, ARP, and multicast interfaces, contact [IP Infusion](#) directly to get more detailed specifications.

## MPLS and MPLS-VPN Interface

In addition, IP Infusion has defined a set of interface specifications for control-plane/data-plane MPLS forwarder interaction.

Essentially, three types of interface specifications have been defined:

- ILM/FTN Interface (LDP control plane to MPLS forwarder)
- VRF Label Interface (BGP-VPN to MPLS forwarder)
- TE Label Interface (RSVP-TE and CR-LDP to MPLS forwarder)

### ILM/FTN Interface (LDP control-plane to MPLS Forwarder)

The ILM is the Incoming Label Mapping, while the FTN is the FEC (Forward Equivalency Class) to NHLFE (Next Hop Label Forwarding Entry). The ILM/FTN interface defines how labels are distributed and managed from the control-plane to data-plane. IP Infusion has already implemented this API in the ZebOS Advanced

Routing Suite and will submit this as a draft MPLS API to the NP Forum at the appropriate time. The basic functions provided by this interface are the following:

- `ipi_mpls_init_all_handles` (Protocol) – Create the FTN and ILM tables for “ALL”, “LDP”, or “BGP”
- `ipi_mpls_close_all_handles` (Protocol) – Destroys the FTN and ILM tables for “ALL”, “LDP”, or “BGP”
- `ipi_mpls_ilm_entry_add` (Protocol, In\_label, Out\_label, Index\_in, Index\_out, FEC\_IP, FEC\_len, IP\_NH, Egress, Opcode) – Adds or replaces ILM entry for “ALL”, “LDP”, or “BGP”
- `ipi_mpls_ilm_entry_del` (Protocol, In\_label, Index) – Deletes the specific ILM entry
- `ipi_mpls_ftn_entry_add` (ID, Protocol, Out\_label, FEC\_IP, FEC\_len, IP\_NH, Index, Opcode) – Adds or replaces FTN entry for “ALL”, “LDP”, or “BGP”.
- `ipi_mpls_ftn_entry_del` (ID, Protocol, Index) – Deletes the specific ILM entry
- `ipi_mpls_if_init` (index, label\_space) – Enables MPLS forwarding on the specified interface (label\_space).
- `ipi_mpls_if_end` (index) - Disables MPLS forwarding on the specified interface.
- `ipi_mpls_clean_fib_for` (protocol) – This function cleans up all the ILM/FTN table entries that were populated by the specified protocol. The protocol’s supported include: All, LDP, BGP

Together these functions specify an interface to allow LDP control-plane interaction with an MPLS forwarder.

### VRF Label Interface (BGP-VPN to MPLS Forwarder)

The VRF (Virtual Routing Forwarding) Label Interface defines the control-plane to data-plane functions that are required to support MPLS BGP-VPN capabilities. MPLS-VPN uses both LDP and VPN extensions to BGP. The BGP-VPN extensions allow BGP to create and manage multiple routing tables, known as VRFs; each of which corresponds to a VPN interface or set of interfaces. Maintaining separate VRFs per VPN ensures that route information is not propagated between VPNs, which guarantees the security of every VPN implementation. The VRF Label Interface is the set of functions required to communicate between the control-plane BGP-VPN software and IP Infusion’s MPLS Forwarder. Consequently, these functions would need to be enabled in a third party MPLS forwarder to support IP Infusion’s BGP- VPN extensions.

The VRF Label Interface functions that are defined are:

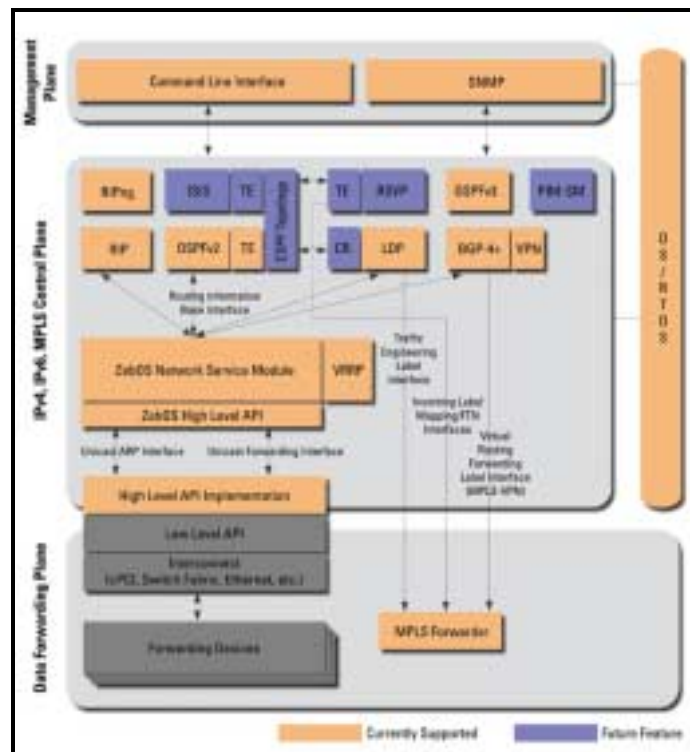
- `ipi_mpls_vrf_init` (identifier) – This function binds the specified interface to the appropriate VRF table (based on the identifier). If this VRF table does not exist, a fresh one is created. For each packet received by this interface, the forwarder will first try and find a match for the specified FEC in the VRF table, and then find a match in the global FTN table (if required).
- `ipi_mpls_vrf_end` (identifier) – This function unbinds the specified interface from the VRF table that is was previously bound to.

- ipi\_mpls\_clean\_vrf\_for (protocol) – This function cleans up all the VRF tables for entries that were populated by the specified protocol. The only protocol supported is for BGP.
- ipi\_mpls\_ip\_update\_vrf (index, label\_space, identifier) – This function updates the VRF which the specified interface is bound to. All FTN entries for this interface should subsequently be written to the global FTN table.

In conjunction with the ILM/FTN interface functions, the VRF functions specify a set of control-data plane specifications to enable an MPLS forwarder in the data plane to communicate with LDP and BGP-VPN functions in a control processor. IP Infusion supports these interface specifications in our current product and fully intends to submit these (or some modification therein) to the NP Forum. We also intend to modify these specifications to support the final standard as it is developed.

### Traffic Engineering Label Interface (RSVP-TE and CR-LDP to MPLS Forwarder)

Currently, IP Infusion is developing the Traffic Engineering Label Interface specification at this time. We will publish this specification when the product is released. Like our ILM/FTN and VRF interface specifications, we intend to submit these to the NP Forum at the appropriate time.



## Conclusion

The diagram above shows the entire set of control-plane to data-plane interfaces that we have defined. These include the following specifications:

- IPv4 Unicast, ARP, Multicast
- IPv6 Unicast, ARP, Multicast
- MPLS LDP via ILM/FTN interface
- MPLS BGP-VPN via VRF interface
- MPLS RSVP-TE/CR-LDP via TE interface

These interface specifications assist equipment manufacturers in quickly bringing solutions to market. We have gone through rigorous testing procedures to validate all of our supported interface specifications, both from a conformance, interoperability, and performance standpoint.

This document outlines details of some of these interfaces specifications. For more details on these entire interface specifications, please contact IP Infusion directly, or go to our web site ([www.ipinfusion.com](http://www.ipinfusion.com)) where you can register to download more detailed documentation.

IP Infusion is a member of the NP Forum and is working to standardize IPv4 control-plane to data-plane interface specifications. We intend to draft and support control-plane to data-plane interfaces with our strategic partners and other NP Forum members for additional IPv4, IPv6, MPLS, and other software modules. We believe that supporting standards for these specifications will further assist equipment manufacturers in developing more complex and practical network solutions.



IP Infusion Inc.  
111 W. St. John Street  
Suite 910  
San Jose, CA 95113  
Tel: 408.794.1500  
Fax: 408.278.0521  
[info@ipinfusion.com](mailto:info@ipinfusion.com)  
[www.ipinfusion.com](http://www.ipinfusion.com)