Weave Device Local Addressing and Routing Behavior

2020/03/02
Revision 11

Introduction

During the course of its operation, a Weave device will need to configure its local network interfaces to facilitate communication with other devices in the HAN, and with a supporting cloud service. This configuration will include the assignment of various IP addresses. Similarly, depending on its role in the network and the state of its network interfaces, a Weave device will need to configure one or more network routes to direct the flow of traffic into and out of its interfaces. Some of this addressing and routing information will be derived from the network environment in which the device operates–for example, local IP configuration supplied via DHCP from a home gateway. Other configuration information, in particular its Weave address and routing information, will be derived from the role of the device itself, its hardware configuration, and its mode of operation at a given point in time.

This document describes the expected behavior of a Weave device with respect to how it configures its network interfaces and network routing information in various situations. It identifies the inputs (both static and dynamic) to the configuration logic as well as the fundamental operations that are relied upon to effect the necessary configuration. Finally the document gives a number of example configurations for different categories of devices in typical situations.

Background

One of the features of Weave is the ability for a collection of Weave-enabled devices to self-organize into a logical network known as a *Weave Fabric Network*. A Weave Fabric Network is a logical IPv6 inter-network that overlays onto the underlying physical networks in the user's home (typically a home WiFi network and a low-power Thread network). Weave Fabric Networks are formed automatically by the participating devices themselves, without specific action by an end-user or network administrator.

Weave Fabric Networks provide a stable and consistent addressing architecture that makes it easier to build robust IoT applications. Chief among its benefits is direct mapping between a logical id for a device and its network routable address. This eliminates the need for (and dependence on) a separate address resolution infrastructure (e.g. DNS). When the appropriate types of devices are present in the home, a Weave Fabric Networks facilitate application-level communication between:

- Weave devices residing on the same network (WiFi or Thread)
- Weave devices residing on different networks (e.g. WiFi to Thread)
- Weave devices and private, externally-hosted cloud services
- Weave devices and mobile applications, either inside or outside the home

This document describes the self-organizing behavior (in terms of address and route formation) necessary for a device to participate in a Weave Fabric Network.

While support for Weave Fabric Networks is inherent in Weave, its use by applications running on those devices is optional. The Weave application protocol is itself agnostic to any particular underlying network architecture. Thus devices with interfaces that can speak IPv4 natively, or that have access to publicly routable global IPv6 addresses (GUAs) are free to use those technologies to conduct their conversations, either as an alternative to or in parallel with the use of a private Weave Fabric Network. Accordingly, this document also identifies the points at which Weave devices employ more traditional address and route acquisition mechanisms such as IPv4 DHCP and IPv6 SLAAC.

Static Configuration Options

The following values represent statically determined configuration options that affect the behavior of the device with respect to local addressing and routing. These are expected to be build-time choices that derive from the nature of the device itself.

SupportsWiFi -- True if the device has a WiFi station network interface

SupportsThread -- True if the device has a Thread network interface

SupportsThreadRouting -- True if the device can serve as a Thread router. Only meaningful if SupportsThread == true.

SupportsLegacyNetwork -- True if the device supports the legacy ConnectIP network. Only meaningful if SupportsThread == true.

SupportsCellular -- True if the device has a cellular data interface.

SupportsWeaveTunnel -- True if the device supports a Weave tunnel to the service

SupportsBorderRouting -- True if the device supports being a Weave border router. Only meaningful if SupportsWiFi == true.

EnableFabricDefaultRouting -- True if the device has a policy for routing its outbound external traffic for unknown subnets within the fabric. Only meaningful when all of the following options are true:

- SupportsThread
- SupportsWeaveTunnel
- SupportsThreadRouting
- SupportsBorderRouting

EnableBackupRoutingOverThread -- True if the device has a policy for allowing its outbound Service or other subnet traffic to be routed over the local Thread network to a

connected border router when its Weave tunnel to the Service is down. Only meaningful when all of the following options are true:

- SupportsThread
- SupportsWeaveTunnel
- SupportsThreadRouting
- SupportsBorderRouting

Device State Inputs

The following values represent runtime state variables that are used in determining the device's local addressing and routing configuration.

IsFabricMember -- True if the device is currently a member of a fabric.

FabricId -- The id of the fabric to which the device belongs. Only meaningful if IsFabricMember == true.

IsThreadRouter -- True if the device is currently serving as a Thread router.

IsWeaveBorderRouter -- True if the device is currently serving as a Weave border router.

WiFiConnected -- True if the device's WiFi station interface is currently connected and available for communication.

ThreadConnected -- True if the device's Thread interface is currently connected and available for communication.

CellularConnected -- True if the device's cellular data interface is currently connected and available for communication.

TunnelInterfaceEnabled-- True if the device's tunnel interface is currently enabled. Note that the state of the tunnel interface is independent of the state of the device's tunnel connection to the service.

TunnelState -- State value indicating the availability of the device's network tunnel to the service. Possible values:

NoTunnel -- Tunnel is currently disconnected
Normal -- Tunnel is connected over WiFi
NormalAndBackup -- Tunnel is connected over WiFi and backup link (e.g. cellular)
BackupOnly -- Tunnel is connected over backup link (e.g. cellular)

Network Configuration Functions

The following abstract functions are used in later pseudocode to describe the fundamental actions performed during the process of configuring a Weave device's local addresses and routes. In general the corresponding functionality is expected to be supplied by the device's underlying network services platform.

Note that the particular interfaces depicted here are meant for descriptive purposes only; they are not meant to constrain any actual implementation.

AssignHostAddress(Interface, Address, OnLinkPrefixLength)

- Assign an IPv6 address to the specified interface on host OS. The OnLinkPrefixLength argument describes which addresses related to the assigned address are considered 'on-link' (i.e. directly accessible) via the interface.

RemoveHostAddress(Interface, Address, OnLinkPrefixLength)

- Remove an IPv6 address (if present) from specified interface on host OS.

AddHostRoute(Prefix, PrefixLength, DestInterface, Priority)

- Add a routing entry to the host OS routing table. If the specified Prefix is all zeros (::) and the PrefixLength is 0, a default route is added to the host routing table.

RemoveHostRoute(Prefix, PrefixLength, DestInterface)

- Remove a routing entry from the host OS routing table. If the specified Prefix is all zeros (::) and the PrefixLength is 0, the default route (if any) is removed from the host routing table.

AssignThreadAddress(Address)

- Configure the Thread stack to assign a ULA address to the Thread interface. The prefix length is assumed to be /64.

RemoveThreadAddress(Address)

- Configure the Thread stack to remove the specified ULA address from the Thread interface.

ConfigureThreadAddressAdvertisment(Prefix)

- Configure the Thread stack to advertise a ULA prefix for address assignment by other devices on the PAN. The prefix length is assumed to be /64.

RemoveThreadAddressAdvertisment(Prefix)

- Configure the Thread stack to stop advertising a ULA prefix for address assignment by other devices on the PAN.

AddThreadRoute(Prefix, PrefixLen, Priority)

- Configure the Thread stack to advertise the local device as default router for a specified IPv6 prefix.

RemoveThreadRoute(Prefix, PrefixLen)

- Configure the Thread stack to stop advertising local device as default router for a specified IPv6 prefix.

UpdateThreadRoutePriority(Prefix, PrefixLen, Priority)

- Configure the Thread stack to change the priority of a previously advertised route.

AssignLegacyThreadAddress(Address)

- Configure the Thread stack to assign a ULA address to the legacy Thread interface. The prefix length is assumed to be /64.

RemoveLegacyThreadAddress(Address)

- Configure Thread stack to remove a ULA address from the legacy Thread interface.

AcquireIPv4ConfigDHCP(Interface)

- For the specified interface, request and apply an IPv4 address and associated configuration (default router, DNS server, etc.) from a local IPv4 DHCP server.

AcquireIPv6GlobalConfig(Interface)

- For the specified interface, request and apply one or more IPv6 global addresses and associated configuration (default router, DNS server, etc.) from a local server using either IPv6 SLAAC or DHCP.

RemoveAddressesAndConfiguration(Interface)

- Remove all IPv4 and IPv6 addresses and related network configuration (routes, DNS servers, etc.) associated with the specified interface.

AcquireIPv4ConfigCellular(Interface)

- For the specified cellular interface, request and apply a provider-supplied IPv4 address and associated configuration (default router, DNS server, etc.) using an appropriate method. This may be accomplished via DHCP or by some other provider-specific means.

AcquireIPv6ConfigCellular(Interface)

- For the specified cellular interface, request and apply a provider-supplied IPv6 global address and associated configuration (default router, DNS server, etc.) using an appropriate method. This may be accomplished via SLAAC, DHCPv6 or some other provider-specific means.

Well-Known Values

In the course of configuring a device's addresses and routes, the following well-known values are used.

WeaveWiFiULA -- Weave WiFi ULA generated from the combination of the FabricId, the WiFi subnet id and the device node id.

WeaveThreadULA -- Weave Thread ULA generated from the combination of the FabricId, the Thread subnet id and the device node id.

WeaveLegacyULA -- Weave Legacy Thread ULA generated from the combination of the FabricId, the legacy Thread subnet id and the device node id.

WeaveFabricPrefix -- IPv6 ULA prefix of the local Weave fabric. This prefix is generated from the Weave FabricId and has a prefix length of /48.

WeaveServicePrefix -- IPv6 ULA prefix for the Service subnet. This prefix is generated from a combination of the Weave FabricId and the Service SubnetId and has a prefix length of /64.

DeviceNodeId -- The Weave node id for the local device.

WiFiInterface -- The device's WiFi station interface.

ThreadInterface -- The device's Thread interface.

LegacyInterface -- The device's Legacy Thread interface.

Address and Route Configuration Behavior

The section describes the behavior of a Weave device with respect to when and how it configures its local network addresses and routes. The behaviors are described as a sequence of actions (in pseudo-code) that take place upon the occurrence of a particular network-related event, e.g. a network interface being enabled, or a network tunnel connection being established.

Note that some of the actions described here may in fact be realized by features inherent to the device's underlying network stack; for example, the step of acquiring an IPv4 address by querying a local DHCP server. Such actions are included for completeness, so that a full picture of device addressing and routing behavior can be seen. This is not meant to imply a requirement to re-implement these features at a Weave level.

WiFi State Change Events

The following actions are taken when the state of the device's WiFi station interface changes.

WiFi Station Interface Up

On WiFi station connected and station interface up (i.e. WiFiConnected: false => true):

```
// If the device is a member of a fabric...
if (IsFabricMember) {
    // Assign Weave WiFi ULA to host WiFi interface with /64 prefix.
    AssignHostAddress(Interface=WiFiInterface, Address=WeaveWiFiULA, OnLinkPrefixLength=64)
}

// Request IPv4 network configuration via DHCP, if available.
AcquireIPv4ConfigDHCP(Interface=WiFiInterface)

// Request IPv6 global address configuration, if available.
AcquireIPv6GlobalConfig(Interface=WiFiInterface)
```

WiFi Station Interface Down

On WiFi station disconnected / station interface down (i.e. WiFiConnected: true => false):

```
// Remove all assigned addresses and associated network configuration
RemoveAddressesAndConfiguration(Interface=WiFiInterface)
```

Thread State Change Events

The following actions are taken when the state of the device's Thread interface changes.

Thread Interface Up

On Thread PAN join complete or Thread network connected and interface up (i.e. ThreadConnected: false => true):

```
// If the device is a member of a fabric...
if (IsFabricMember) {

    // Configure Thread stack to assign Weave Thread ULA to device.
    AssignThreadAddress(Address=WeaveThreadULA)

    // Assign Thread ULA to host's Thread interface with /64 prefix length.
    AssignHostAddress(Interface=ThreadInterface, Address=WeaveThreadULA, OnLinkPrefixLength=64)
```

```
// If the device is acting as a Thread router...
if (SupportsThreadRouting && IsThreadRouter) {

    // Configure Thread stack to advertise Weave Thread prefix to other devices on PAN
    ConfigureThreadAddressAdvertisment(Prefix=WeaveThreadPrefix)

    // If the device is a border router and has an established Weave tunnel to the service
    if (SupportsBorderRouting && IsWeaveBorderRouter && SupportsWeaveTunnel &&
      TunnelStatus != NoTunnel) {

        // Compute a route priority for Weave fabric route:
        if (TunnelAvailability == NormalAndBackup || TunnelAvailability == Normal) {
            SelectedPriority = Medium
        } else { // i.e. TunnelAvailability == BackupOnly
            SelectedPriority = Low
        }

        // Configure Thread stack to advertise local device as default router for Weave fabric prefix
        AddThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, Priority=SelectedPriority)
    }
}

// If the device supports WiFi or cellular, add a service-subnet specific /64 or a fabric-default /48 route,
// with Low route priority, to the host routing table pointing to Thread interface provided backup routing
// is enabled.
if (SupportsWiFi || SupportsCellular) {
   // Device supports WiFi or Cellular...
   if (EnableBackupRoutingOverThread) {
      // Backup routing over the Thread network is enabled...
      if (EnableFabricDefaultRouting) {
         // Add fabric-default /48 route
         AddHostRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, DestInterface=ThreadInterface,
             Priority=Low)
      } else {
         // Add subnet-specific /64 route
         AddHostRoute(Prefix=WeaveServicePrefix, PrefixLength=64, DestInterface=ThreadInterface,
            Priority=Low)
      } // !EnableFabricDefaultRouting
   } // EnableBackupRoutingOverThread
}

// Otherwise, since the device only supports one interface, add a default route...
else {
    AddHostRoute(Prefix=::, PrefixLength=0, DestInterface=ThreadInterface, Priority=Low)
}

// If the device supports the legacy Thread network...
if (SupportsLegacyThreadNetwork) {
```

```
            // Configure Thread stack to assign Legacy Thread ULA to device.
            AssignLegacyThreadAddress(Address=WeaveLegacyULA);

            // Assign Legacy Thread ULA to host's legacy Thread interface with /64 prefix length.
            AssignHostAddress(Interface=LegacyThreadInterface, Address=WeaveLegacyULA,
                OnLinkPrefixLen=64);
        }
    }
```

Thread Interface Down

On Thread PAN leave or Thread network disconnected (i.e. ThreadConnected: true => false):

```
    // If the device is a member of a fabric...
    if (IsFabricMember) {

        // Configure Thread stack to remove the Weave Thread ULA.
        RemoveThreadAddress(Address=WeaveThreadULA)

        // Remove Thread ULA to host's Thread interface with /64 prefix length.
        RemoveHostAddress(Interface=ThreadInterface, Address=WeaveThreadULA, OnLinkPrefixLength=64)

        // If the device is acting as a Thread router...
        if (SupportsThreadRouting && IsThreadRouter) {

            // Configure Thread stack to stop advertising the Weave Thread prefix for address assignment.
            RemoveThreadAddressAdvertisment(Prefix=WeaveThreadPrefix)

            // If the device is a border router and has an established Weave tunnel to the service
            if (SupportsBorderRouting && IsWeaveBorderRouter && SupportsWeaveTunnel &&
              TunnelStatus != NoTunnel) {

                // Configure Thread stack to stop advertising the local device as a default router for Weave
                // fabric prefix
                RemoveThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48)
            }
        }

        // Remove Weave fabric-default /48 or a service-subnet specific /64 route pointing to Thread interface
        // from the host routing table.
        if (SupportsWiFi || SupportsCellular) {
            // Device supports WiFi or Cellular...
            if (EnableBackupRoutingOverThread) {
                // Backup routing over the Thread network is enabled...
                If (EnableFabricDefaultRouting) {
                    // Remove fabric-default /48 route
```

```
                RemoveHostRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, DestInterface=ThreadInterface,
                    Priority=Low)
            } else {
                // Remove subnet-specific /64 route
                RemoveHostRoute(Prefix=WeaveServicePrefix, PrefixLength=64, DestInterface=ThreadInterface,
                    Priority=Low)
            } // !EnableFabricDefaultRouting
        } // EnableBackupRoutingOverThread
    }
    else {
        RemoveHostRoute(Prefix=::, PrefixLength=0, DestInterface=ThreadInterface, Priority=Low)
    }

    // If the device supports the legacy Thread network...
    if (SupportsLegacyThreadNetwork) {

        // Configure Thread stack to remove the Legacy Thread ULA.
        RemoveLegacyThreadAddress(Address=WeaveLegacyULA);

        // Remove the Legacy Thread ULA from the host's legacy Thread interface.
        RemoveHostAddress(Interface=LegacyThreadInterface, Address=WeaveLegacyULA,
            OnLinkPrefixLen=64);
    }
}
```

Thread Routing Enabled

On Thread routing enabled (IsThreadRouter: false => true):

```
// If the device is a member of a fabric...
if (IsFabricMember) {

    // Configure Thread stack to advertise Weave Thread prefix to other devices on PAN
    ConfigureThreadAddressAdvertisment(Prefix=WeaveThreadPrefix)

    // If the device is a border router and has an established Weave tunnel to the service
    if (SupportsBorderRouting && IsWeaveBorderRouter && SupportsWeaveTunnel &&
      TunnelStatus != NoTunnel) {

        // Compute a route priority for Weave fabric route:
        if (TunnelAvailability == NormalAndBackup || TunnelAvailability == Normal) {
            SelectedPriority = Medium
        } else { // i.e. TunnelAvailability == BackupOnly
            SelectedPriority = Low
        }

        // Configure Thread stack to advertise local device as default router for Weave fabric prefix
        AddThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, Priority=SelectedPriority)
```

```
    }
}
```

Thread Routing Disabled

On Thread routing disabled (IsThreadRouter: true => false):

```
// If the device is a member of a fabric...
if (IsFabricMember) {

    // Configure Thread stack to stop advertising the Weave Thread prefix for address assignment.
    RemoveThreadAddressAdvertisment(Prefix=WeaveThreadPrefix)

    // If the device is a border router and has an established Weave tunnel to the service
    if (SupportsBorderRouting && IsWeaveBorderRouter && SupportsWeaveTunnel &&
      TunnelStatus != NoTunnel) {

        // Configure Thread stack to stop advertising the local device as a default router for
        // Weave fabric prefix
        RemoveThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48)
    }
}
```

Cellular State Change Events

The following actions are taken when the state of the device's cellular data interface changes.

Cellular Interface Up

On cellular data connected and interface up (i.e. CellularConnected: false => true):

```
// Request IPv4 network configuration via DHCP, if available.
AcquireIPv4ConfigDHCP(Interface=CellularInterface)

// Request IPv6 global address configuration, if available.
AcquireIPv6GlobalConfig(Interface=CellularInterface)
```

Cellular Interface Down

On cellular data disconnected / interface down (i.e. CellularConnected: true => false):

```
// Remove all assigned addresses and associated network configuration
RemoveAddressesAndConfiguration(Interface=CellularInterface)
```

Fabric State Change Events

The following actions are taken when the device joins or leaves a Weave fabric.

Fabric Join

On fabric join (i.e. IsFabricMember: false => true):

```
// If the device is connected to a WiFi network...
if (SupportsWiFi && WiFiConnected) {
    // Assign Weave WiFi ULA to host WiFi interface with /64 prefix.
    AssignHostAddress(Interface=WiFiInterface, Address=WeaveWiFiULA, OnLinkPrefixLength=64)
}

// If the device is connected to a Thread network...
if (SupportsThread && ThreadConnected) {

    // Configure Thread stack to assign Weave Thread ULA to device.
    AssignThreadAddress(Address=WeaveThreadULA)

    // Assign Thread ULA to host's Thread interface with /64 prefix length.
    AssignHostAddress(Interface=ThreadInterface, Address=WeaveThreadULA,
            OnLinkPrefixLength=64)

    // If the device is acting as a Thread router...
    if (SupportsThreadRouting && IsThreadRouter) {
        // Configure Thread stack to advertise Weave Thread prefix to other devices on PAN
        ConfigureThreadAddressAdvertisment(Prefix=WeaveThreadPrefix)
    }

    // If the device supports WiFi or cellular, add a service-subnet specific /64 or a fabric-default /48 route,
    // with Low route priority, to the host routing table pointing to Thread interface provided backup routing
    // is enabled.
    if (SupportsWiFi || SupportsCellular) {
       // Device supports WiFi or Cellular...
       if (EnableBackupRoutingOverThread) {
          // Backup routing over the Thread network is enabled...
          If (EnableFabricDefaultRouting) {
             // Add fabric-default /48 route
             AddHostRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, DestInterface=ThreadInterface,
                 Priority=Low)
          } else {
             // Add subnet-specific /64 route
             AddHostRoute(Prefix=WeaveServicePrefix, PrefixLength=64, DestInterface=ThreadInterface,
                Priority=Low)
          } // !EnableFabricDefaultRouting
```

```
            } // EnableBackupRoutingOverThread

        }
        else {
            AddHostRoute(Prefix=::, PrefixLength=0, DestInterface=ThreadInterface, Priority=Low)
        }

        // If the device supports the legacy Thread network...
        if (SupportsLegacyThreadNetwork) {

            // Configure Thread stack to assign Legacy Thread ULA to device.
            AssignLegacyThreadAddress(Address=WeaveLegacyULA);

            // Assign Legacy Thread ULA to host's legacy Thread interface with /64 prefix length.
            AssignHostAddress(Interface=LegacyThreadInterface, Address=WeaveLegacyULA,
                OnLinkPrefixLen=64);
        }
    }

    // If the device supports Weave tunneling...
    if (SupportsWeaveTunnel) {

        // NOTE: The following actions ensure the selection of an appropriate Weave ULA (WiFi or Thread) as
        // the source address for packets originating on the local node but destined for addresses reachable
        // via the tunnel.  Without this, the default IPv6 source address selection algorithm might choose an
        // inappropriate source address, making it impossible for the destination node to respond.

        // If the device supports WiFi, assign the Weave WiFi ULA to host tunnel interface with /128 prefix.
        if (SupportsWiFi) {
            AssignHostAddress(Interface=TunnelInterface, Address=WeaveWiFiULA,
                OnLinkPrefixLength=128)
        }

        // Otherwise, if the device supports Thread, assign the Weave Thread ULA to the tunnel interface
        // with a /128 prefix.
        // NOTE: This case is only meaningful for non-WiFi devices that have some other interface
        // over which a tunnel can be formed, e.g. a Thread+Cellular device.
        else if (SupportsThread) {
            AssignHostAddress(Interface=TunnelInterface, Address=WeaveThreadULA,
                OnLinkPrefixLength=128)
        }
    }
```

Fabric Leave

On fabric leave (i.e. IsFabricMember: true => false):

```
    // If the device is connected to a WiFi network, remove Weave WiFi ULA from host WiFi interface.
```

```
if (SupportsWiFi && WiFiConnected) {
    RemoveHostAddress(Interface=WiFiInterface, Address=WeaveWiFiULA, OnLinkPrefixLength=64)
}

// If the device is connected to a Thread network...
if (SupportsThread && ThreadConnected) {

    // Configure Thread stack to remove Weave Thread ULA.
    RemoveThreadAddress(Address=WeaveThreadULA)

    // Remove Thread ULA from host's Thread interface.
    RemoveHostAddress(Interface=ThreadInterface, Address=WeaveThreadULA, OnLinkPrefixLength=64)

    // If the device was acting as a Thread router, configure Thread stack to stop advertising
    // Weave Thread prefix to other devices on PAN.
    if (SupportsThreadRouting && IsThreadRouter) {
        RemoveThreadAddressAdvertisment(Prefix=WeaveThreadPrefix)
    }

    // Remove Weave fabric-default /48 or a service-subnet specific /64 route pointing to Thread interface
    // from the host routing table.
    if (SupportsWiFi || SupportsCellular) {
       // Device supports WiFi or Cellular...
       if (EnableBackupRoutingOverThread) {
          // Backup routing over the Thread network is enabled...
          If (EnableFabricDefaultRouting) {
             // Remove fabric-default /48 route
             RemoveHostRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, DestInterface=ThreadInterface,
                 Priority=Low)
          } else {
             // Remove subnet-specific /64 route
             RemoveHostRoute(Prefix=WeaveServicePrefix, PrefixLength=64, DestInterface=ThreadInterface,
                Priority=Low)
          } // !EnableFabricDefaultRouting
       } // EnableBackupRoutingOverThread

    }
    else {
        RemoveHostRoute(Prefix=::, PrefixLength=0, DestInterface=ThreadInterface, Priority=Low)
    }

    // If the device supports the legacy Thread network...
    if (SupportsLegacyThreadNetwork) {

        // Configure Thread stack to remove the Legacy Thread ULA.
        RemoveLegacyThreadAddress(Address=WeaveLegacyULA);

        // Remove the Legacy Thread ULA from the host's legacy Thread interface.
```

```
            RemoveHostAddress(Interface=LegacyThreadInterface, Address=WeaveLegacyULA,
                OnLinkPrefixLen=64);
        }
    }

    // If the device supports Weave tunneling, and the tunnel interface is up...
    if (SupportsWeaveTunnel && TunnelInterfaceEnabled) {

        // If the device supports WiFi, remove the Weave WiFi ULA from the host tunnel interface.
        if (SupportsWiFi) {
            RemoveHostAddress(Interface=TunnelInterface, Address=WeaveWiFiULA, OnLinkPrefixLength=128)
        }

        // Otherwise, if the device supports Thread, remove the Weave thread ULA from the host
        // tunnel interface.
        else if (SupportsThread) {
            RemoveHostAddress(Interface=TunnelInterface, Address=WeaveThreadULA,
                OnLinkPrefixLength=128)
        }
    }
```

Tunnel State Change Events

The following actions are taken when the device's network tunnel to the Nest service changes state.

Tunnel Interface Up

On tunnel interface enabled (i.e. TunnelInterfaceEnabled: false => true):

```
    // If the device is a member of a fabric...
    if (IsFabricMember) {

        // NOTE: The following actions ensure the selection of an appropriate Weave ULA (WiFi or Thread) as
        // the source address for packets that originate on the local node but are destined for addresses reachable
        // via the tunnel.  Without this, the default IPv6 source address selection algorithm might choose an
        // inappropriate source address, making it impossible for the destination node to respond.
        //
        // The logic here prioritizes use of the Thread address (when available) over the WiFi address in order to
        // improve connectivity in cases where the WiFi network is unavailable and another path exists to the
        // fabric.  For example, consider a device that has both Thread and WiFi interfaces, and exists on the
        // same network as another device with a cellular interface that is acting as a border router.  If the
        // WiFi network fails, entities outside of the home network (e.g. the service) can still reach the first
        // device by addressing packets to its Thread address, causing them to route down the cellular tunnel
        // and across the Thread network.  Therefore, it is beneficial for the first device to always select its
        // Thread address as the source address for packets it sends out of the network, such that external
        // nodes will choose that address when responding.
```

```
        // If the device supports Thread, assign the Weave Thread ULA to the tunnel interface
        // with a /128 prefix.
        if (SupportsThread) {
            AssignHostAddress(Interface=TunnelInterface, Address=WeaveThreadULA,
                OnLinkPrefixLength=128)
        }

        // Otherwise, if the device supports WiFi, assign the Weave WiFi ULA to host tunnel interface
        // with /128 prefix.
        // NOTE: This case is only meaningful for non-WiFi devices that have some other interface
        // over which a tunnel can be formed, e.g. a Thread+Cellular device.
        else if (SupportsWiFi) {
            AssignHostAddress(Interface=TunnelInterface, Address=WeaveWiFiULA,
                OnLinkPrefixLength=128)
        }
    }
```

Tunnel Interface Down

On tunnel interface disabled (i.e. TunnelInterfaceEnabled: true => false):

```
    // If the device is a member of a fabric...
    if (IsFabricMember) {

        // If the device supports WiFi, remove the Weave WiFi ULA from the host tunnel interface.
        if (SupportsWiFi) {
            RemoveHostAddress(Interface=TunnelInterface, Address=WeaveWiFiULA, OnLinkPrefixLength=128)
        }

        // Otherwise, if the device supports Thread, remove the Weave thread ULA from the host
        // tunnel interface.
        else if (SupportsThread) {
            RemoveHostAddress(Interface=TunnelInterface, Address=WeaveThreadULA,
                OnLinkPrefixLength=128)
        }
    }
```

Tunnel Established

On service tunnel established (i.e. TunnelState: NoTunnel => Normal|NormalAndBackup|BackupOnly):

```
    // Add Weave fabric-default /48 route or service-subnet specific /64 route to host routing table pointing
    // to Tunnel interface with Medium route priority based on whether fabric-default routing is enabled or not.
    If (EnableFabricDefaultRouting) {
       // Add a fabric-default /48 route
```

```
    AddHostRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, DestInterface=TunnelInterface,
    Priority=Medium)
  }  else {
    // Add Service subnet-specific /64 route
    AddHostRoute(Prefix=WeaveServicePrefix, PrefixLength=64, DestInterface=TunnelInterface,
    Priority=Medium)
  }

  // If the device is acting as a border router...
  if (SupportsBorderRouting && IsWeaveBorderRouter) {

      // If the device is acting as a Thread router...
      if (SupportsThread && SupportsThreadRouting && ThreadConnected && IsThreadRouter) {

          // Compute a route priority for Weave fabric route in Thread:
          if (TunnelAvailability == NormalAndBackup || TunnelAvailability == Normal) {
              SelectedPriority = Medium
          } else { // i.e. TunnelAvailability == BackupOnly
              SelectedPriority = Low
          }

          // Configure Thread stack to advertise local device as default router for Weave fabric prefix
          AddThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, Priority=SelectedPriority)
      }
  }
```

Tunnel Disconnected

On tunnel disconnected (i.e. TunnelState: Normal|NormalAndBackup|BackupOnly => NoTunnel):

```
  // Remove Weave fabric-default /48 route or service-subnet specific /64 to host routing table pointing to
  // Tunnel interface with Medium route priority.
  If (EnableFabricDefaultRouting) {
    // Remove fabric-default /48 route
    RemoveHostRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, DestInterface=TunnelInterface,
    Priority=Medium)
  } else {
    // Remove Service subnet-specific /64 route
    RemoveHostRoute(Prefix=WeaveServiceFabricPrefix, PrefixLength=64, DestInterface=TunnelInterface,
    Priority=Medium)
  }

  // If the device is acting as a border router...
  if (SupportsBorderRouting && IsWeaveBorderRouter) {

      // If the device is acting as a Thread router...
      if (SupportsThread && SupportsThreadRouting && ThreadConnected && IsThreadRouter) {
```

```
            // Configure Thread stack to stop advertising local device as default router for Weave fabric prefix.
            RemoveThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48)
        }
    }
```

Tunnel Mode Change

On tunnel mode change (i.e. TunnelState: Normal|NormalAndBackup|BackupOnly => Normal|NormalAndBackup|BackupOnly):

```
    // If the device is acting as a border router…
    if (SupportsBorderRouting && IsWeaveBorderRouter) {

        // If the device is acting as a Thread router…
        if (SupportsThread && SupportsThreadRouting && ThreadConnected && IsThreadRouter) {

            // Compute a route priority for Weave fabric route:
            if (TunnelAvailability == NormalAndBackup || TunnelAvailability == Normal) {
                SelectedPriority = Medium
            } else { // i.e. TunnelAvailability == BackupOnly
                SelectedPriority = Low
            }

            // Configure Thread stack to update the priority of the route for the Weave fabric prefix.
            UpdateThreadRoutePriority(Prefix=WeaveFabricPrefix, PrefixLength=48, Priority=SelectedPriority)
        }
    }
```

Border Router State Changes

The following actions are taken when the device transitions to or from a role as a Weave border router.

Border Routing Enabled

On border routing enabled (IsWeaveBorderRouter: false => true):

```
    // If the device is acting as a Thread router…
    if (SupportsThread && SupportsThreadRouting && ThreadConnected && IsThreadRouter) {

        // If the device has an established Weave tunnel to the service
        if (TunnelStatus != NoTunnel) {

            // Compute a route priority for Weave fabric route:
            if (TunnelAvailability == NormalAndBackup || TunnelAvailability == Normal) {
                SelectedPriority = Medium
```

```
        } else { // i.e. TunnelAvailability == BackupOnly
            SelectedPriority = Low
        }

        // Configure Thread stack to advertise local device as default router for Weave fabric prefix
        AddThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48, Priority=SelectedPriority)
    }
}
```

Border Routing Disabled

On border routing disabled (IsWeaveBorderRouter: true => false):

```
// If the device is acting as a Thread router...
if (SupportsThread && SupportsThreadRouting && ThreadConnected && IsThreadRouter) {

    // If the device has an established Weave tunnel to the service
    if (TunnelStatus != NoTunnel) {

        // Configure Thread stack to stop advertising local device as default router for Weave fabric prefix.
        RemoveThreadRoute(Prefix=WeaveFabricPrefix, PrefixLength=48)
    }
}
```

Example Device Configurations

The following sections show a set of possible network address and route configurations for different categories of devices in typical situations.

WiFi+Thread Device, Thread Routing, No Tunnel Support, No Border Router Support

Host Stack Configuration

- Addresses:
  - WiFi:
    - Weave WiFi Fabric Address /64
    - DHCP acquired IPv4 address and subnet
  - Thread: Weave Thread Fabric Address /64
  - Legacy: Weave Legacy Thread Fabric Address /64
- Local Routes:
  - Weave Fabric Prefix /48 -> Thread Interface, low priority

Thread Stack Configuration

- Address: Weave Thread Fabric Address /64
- Address Advertisement: Weave Thread Fabric Address /64
- Route Advertisements: None
- Legacy Thread Address: Weave Legacy Thread Fabric Address /64

Thread-only Device, No Thread Routing

Host Stack Configuration

- Addresses:
  - Thread: Weave Thread Fabric Address /64
  - Legacy: Weave Legacy Thread Fabric Address /64 (if supported)
- Local Routes:
  - Weave Fabric Prefix /48 -> Thread Interface, low priority

Thread Stack Configuration

- Address: Weave Thread Fabric Address /64
- Address Advertisement: None
- Route Advertisements: None
- Legacy Thread Address: Weave Legacy Thread Fabric Address /64 (if supported)

WiFi+Thread Device, Thread Routing, Tunnel Support, Border Router Support

Host Stack Configuration

- Addresses:
  - WiFi:
    - Weave WiFi Fabric Address /64
    - DHCP acquired IPv4 address and subnet
  - Thread: Weave Thread Fabric Address /64
  - Legacy: Weave Legacy Thread Fabric Address /64 (if supported)
  - Tunnel: Weave WiFi Fabric Address /128
- Local Routes:
  - Weave Fabric Prefix /48 -> Thread Interface, low priority
  - Weave Fabric Prefix /48 -> Tunnel Interface, normal priority (when tunnel available)

Thread Stack Configuration

- Address: Weave Thread Fabric Address /64
- Address Advertisement: Weave Thread Fabric Address /64
- Route Advertisements:
  - Weave Fabric Prefix /48 -> (local node) (when tunnel available, priority depends on mode of tunnel)
- Legacy Thread Address: Weave Legacy Thread Fabric Address /64 (if supported)

WiFi+Thread Device, Tunnel Support, No Thread Routing, No Border Router Support

Host Stack Configuration

- Addresses:
  - WiFi:

        - Weave WiFi Fabric Address /64
        - DHCP acquired IPv4 address and subnet
    - Thread: Weave Thread Fabric Address /64
    - Legacy: Weave Legacy Thread Fabric Address /64 (if supported)
- Local Routes:
    - Weave Fabric Prefix /48 -> Thread Interface, low priority
    - Weave Fabric Prefix /48 -> Tunnel Interface, normal priority (when tunnel available)

Thread Stack Configuration

- Address: Weave Thread Fabric Address /64
- Address Advertisement: None
- Route Advertisements: None
- Legacy Thread Address: Weave Legacy Thread Fabric Address /64 (if supported)

WiFi-only Device, Tunnel Support

Host Stack Configuration

- Addresses:
    - WiFi: Weave WiFi Fabric Address /64
    - Tunnel: Weave WiFi Fabric Address /128
- Local Routes:
    - Weave Fabric Prefix /48 -> Tunnel Interface, normal priority (when tunnel available)

Revision History

Revision	Date	Change Description
1	2015/10/21	Initial revision.
2	2015/10/23	Some small clarifying edits.
3	2015/10/26	Added clarifying text regarding assignment of /128 address to tunnel interface. Added behavior description for cellular interface (for device's so equipped). Various other minor edits and clean-up.
4	2015/10/28	Modified pseudo-code to add the Weave Thread ULA to the tunnel interface with a /128 prefix length in the case where the device supports Thread and Weave Tunneling but not

		WiFi (e.g. a theoretical Thread+Cellular device).
5	2015/11/09	Added new input state variable TunnelInterfaceEnabled and association state change behaviors. Remove pseudo code that added/removed the Weave ULA address for the tunnel interface when the service tunnel state changed.
6	2015/11/11	Updated pseudo-code to add a default route to the host routing table, instead of a /48 Weave fabric route, in the case where a device only supports Thread. This allows thread-only devices to implement a simplified routing architecture that only supports /64 interfaces routes plus a default route.
7	2016/09/29	Modified handling of Tunnel Interface Up event so that the behavior prioritizes assigning the Thread address to the tunnel interface, if available, versus the WiFi address. Expanded the note in that section explaining why this is desirable.
8	2017/04/19	Modified the priority selection algorithm for the Weave default subnet route (/48) for the Thread network such that the maximum priority is Medium, rather than High.
9	2019/03/23	Minor editorial changes and clarifications.
10	2020/02/28	Added background section. Make minor clarifications to WiFi and Cellular behaviors.
11	2020/03/02	Rewrote the background section based on feedback. Adjusted fonts.