**Sirius – Service Tunnel and Private Link Features**

Contents

[Introduction 2](#_Toc76047562)

[Additional Resources 2](#_Toc76047563)

[Requirements 2](#_Toc76047564)

[Service Tunnel Feature 3](#_Toc76047565)

[Summary 3](#_Toc76047566)

[Packet Transformation 4](#_Toc76047567)

[Routing 5](#_Toc76047568)

[Private Link 6](#_Toc76047569)

[Summary 6](#_Toc76047570)

[Packet Transformation 7](#_Toc76047571)

[Cosmos Scenario 10](#_Toc76047572)

# Introduction

This document provides an overview to “Service Tunnel – Service Endpoint” and “Private Link” features for Sirius product perspective - a Software Defined Networking (SDN) Appliance - and describes the traffic pattern between feature enabled customer workloads and service providers.

# Additional Resources

[Virtual network integration of Azure services for network isolation | Microsoft Docs](https://docs.microsoft.com/en-us/azure/virtual-network/vnet-integration-for-azure-services)

[Azure virtual network service endpoints | Microsoft Docs](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-service-endpoints-overview)

[Azure service tags overview | Microsoft Docs](https://docs.microsoft.com/en-us/azure/virtual-network/service-tags-overview)

[What is Azure Private Link? | Microsoft Docs](https://docs.microsoft.com/en-us/azure/private-link/private-link-overview?toc=/azure/virtual-network/toc.json)

# Requirements

SmartNICs of Sirius product are required to support:

* Transformed packet structure of Service Tunnel and Private Link features.
* Fast path upon SLB MUXs’ redirection.

# Service Tunnel Feature

## Summary

Service providers offer customers shared services such as Storage, SQL. Services have multIPle Virtual IP Addresses (VIP) per Azure region. These VIPs - i.e. 50.12.11.8 - are internet routable and publicly accessible. Customers require to create Access Control Lists (ACL) and allow internet access to be able to access the shared services. Customers use Fully Qualified Domain Names (FQDN) for their services – i.e. account1.blob.storage.net – and Service Tunnel feature prevents internet access to these services.

A preferred way of defining ACLs is to set them in service provider side and only allow access from a specific virtual network (vnet). This functionality requires us to detect the specific vnet prefix that initiates the traffic and understand if the access is coming from internet or vnet. Service Tunnel feature provides this capability by encoding “region id”, “vnet id”, “subnet id” via packet transformation.

Microsoft publishes the IP ranges for each service regularly and the changes in the address spaces require customers to update their Network Security Groups (NSG) with the new ACLs. Service Tags prevent these manual changes and allows customers to tag their vnets for specific services. Microsoft manages and automatically updates the prefixes of a given service tag. Identifying the IP address of the service is necessary for service tag support and Service Tunnel feature establishes this capability via packet transformation.

Timeline

Description automatically generated

*Sample Diagram*

## Packet Transformation

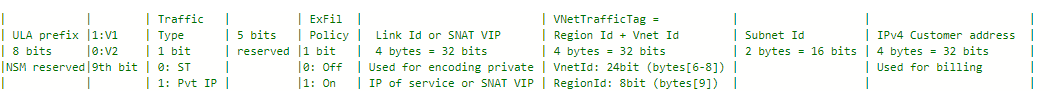
Service tunnel feature utilizes 128 bits IPv6 packet size and performs packet transformation by encoding vnet identity in the IPv6 packet. This transformation is not mapping based, and it is calculated dynamically. Right 32 bits of the IPv6 packet is used for the IPv4 address, while left 96 bits of the IPv6 packet is used for encoding. Packet transformation enables customers to use IPv6 address prefixes in their ACLs and avoid making additional rule updates.

Timeline

Description automatically generated with medium confidence

*Sample Packet Transformation*

Azure have implemented rules to monitor destination prefixes. When a customer Virtual Machine (VM) with service tunnel feature enabled sends an IPv4 packet and hits the rule, packet transformation will be performed and IPv4 addresses will be transformed to IPv6. “Region Id” and “Vnet Id” are encoded as “VNetTrafficTag” on the specific bits of the IPv6 packet and “Subnet Id” is used to differentiate shared services. “IPv4 Customer address” is used for VM identity. Packet transformation is performed both for source and destination addresses.



*Transformed Packet Breakdown*

Rules’ condition is based on set of prefixes. There is a specific rule and a defined source & destination prefix for individual shared services.

* ST Source V2 Prefix:  fd00::​/8

ST S​ource V1 Prefix:   fde4:8dba::/32 [Old prefix]

* ST Destination V1/V2 Prefix = 2603:10e1:100:2::/64

Packet transformation allows customers to identify the specific prefix that packet is coming from and set ACLs in the service side.

## Routing

After packet transformation, packet will be encapsulated into VXLAN. Destination VIP - last 32 bits of transformed IPv6 packet – will be extracted and encapsulated into outer header. Source IP address of the outer encap is the source network address translated (NAT) VIP of the specific VM. Packets will go through Software Load Balancer (SLB) MultIPlexers (MUX) and eventually traffic will transmit among VIPs after fast path kicks in.

Once the traffic is initiated by the VM, packet will have source and destination IP addresses as VIPs. Packet will hit SLB MUX and SLB MUX will send the VM an “ICMP Redirect” request. Source IP address will remain the same and destination IP address will be updated to physical address (PA) in the outer header. Service’s reply will go through a similar redirection and the packet will be transmitted via fast path with source IP addresses as VIPs and destination IP addresses as PAs.

Fast path prevents overloading SLB MUXs. SLB MUXs are used for connection establishment (TCP SYN, SYN-ACK, ACK packets), and fast path will kick in after the redirection.

Timeline

Description automatically generated with low confidence

Chart, line chart

Description automatically generated

*Routing Diagram*

# Private Link

## Summary

Service tunnel feature encodes vnet identity in the packet and enables customers to define their ACLs in the destination service provider side. However, customers using this feature still outreaches traffic towards public VIP and they still need to configure ACLs to allow outbound access to INTERNET. Customers have capabilities to define ACL rules for the specific vnet prefixes – instead of allowing all outbound access to internet – but this approach still raises a security concern for customers due to public facing IP addresses. Customers’ requirement is to block outbound traffic to INTERNET and only allow access within their VNET. Although some shared services such as SQL MI provides functionality to deploy a SQL VM within customer VNET, this functionality is not available for other shared services like Big SQL, CosmosDB. It is not possible to inject these shared services as a VM in customers’ vnet. Private link feature addresses this problem.

Private Link feature is an extension to Service Tunnel feature and enables customers to access public facing shared services via their private IP addresses within their vnet. As an example, a customer using 10.0.0.0/8 vnet space will be able to allocate 10.0.0.7 IP address as an endpoint and all the traffic sent to this endpoint will be routed to the shared service. This feature gives customers capability to deny all outbound traffic to INTERNET.

As seen in the sample diagram below, a service can have an internal or public facing load balancer but expose it as a linkable private service. Customers may discover these services that will allocate a private IP address within their vnet and link the service to private link. When packets are sent to this private address, they will be routed to the service. Service will no longer use VIPs, instead they will use the CA (Customer Address) in their vnet space.

Timeline

Description automatically generated

*Sample Diagram*

Customers don’t want to expose entire service even via private IP address and they require granularity to define individual service accounts as endpoints. Customers need to define multIPle endpoints to support this scenario and service provider needs to understand which link the request is coming from. As an example, while 10.0.0.7 endpoint is assigned to Account 1 in a Storage Service, and 10.0.0.8 endpoint is assigned to Account 2, 10.0.0.7 endpoint should not be able to access Account 2.

## Packet Transformation

Packet transformation is performed by encoding the “Link Id” in the bits of IPv6 packet. For the First Parties, we use ST (Internet Stream Protocol) as a technology. Services listen for ST packet and parse the link ID from the packet.

Diagram

Description automatically generated

*Transformed Packet Breakdown*

We are plumbing the rules per destination IP addresses. Initially, specific /32 udrs (User Defined Routing) were defined for each endpoint - i.e., three separate routing rules for 10.0.0.7, 10.0.0.8, 10.0.0.9 endpoints -, but due to /32 route taking precedence over lower prefixes, this functionality has changed to mapping. Upon mapping lookup, if the IP address maps to a regular private IP address within the vnet space, packet will be encapsulated to VXLAN. If the IP address maps to a private link IP address, action will be to perform service tunnel transposition as an ST packet.

In the example diagram below, customer will send the traffic from 10.0.0.5 to 10.0.0.6 and we will get the VIP of the shared service via lookup. We will transform IPv4 packet to another IPv4 packet by updating the destination IP address of the VIP. Then, we perform the packet transformation and encode the link id in the additional bits of the IPv6 packet.

A picture containing timeline

Description automatically generated

*Sample Packet Transformation*

For implementation purposes, Azure uses two LPM groups with duplicate rules. The difference among these groups is whether the source IP address is a vnet or private link address and they will be assigned based on the IP mapping lookup.

Diagram

Description automatically generated

*Sample Customer Setup*

## Cosmos Scenario

Cosmos uses a slightly different architecture, and it has a service that acts like a GW (Gateway). Once customers initiate a connection to Cosmos DB, they will connect to a specific database that is served from a specific port. GW will dynamically redirect customer traffic to the specific VIPs that will correlate to certain databases. Cosmos has multiple VIPs for each database, but customers will have a single endpoint. Therefore, mapping table will include destination port in addition to support this scenario.

Graphical user interface, text, application

Description automatically generated

*Sample Cosmos Architecture*

Table

Description automatically generated

*Sample Mapping Table*