GLOBALPLATFORM®
THE STANDARD FOR SECURE DIGITAL SERVICES AND DEVICES

GlobalPlatform Technology
Virtual Primary Platform – Network Protocol
Version 1.0.1.21

Public Review

February 2021

Document Reference: GPC_SPE_140 (formerly GPC_FST_140)

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

Figures

Tables

1 Introduction

1.1 Audience

This document is intended to specify, for application makers and system architects, the VPP Network Protocol (VNP), a protocol for supporting a network of logical data circuits called Pipes for interfacing with applications.

The lower-layer protocols that carry VNP Packets are out of scope of this document.

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://globalplatform.org/specifications/ip-disclaimers/. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

Table 1-1: Normative References

Standard / Specification	Description	Ref
GPC_SPE_134	GlobalPlatform Technology Open Firmware Loader for Tamper Resistant Element v2.0	[OFL]
GPC_SPE_141	GlobalPlatform Technology Virtual Primary Platform – OFL VNP Extension v2.0	[VOFL]
GPC_SPE_142	GlobalPlatform Technology Virtual Primary Platform – Concepts and Interfaces v2.0	[VCI]
GPC_SPE_143	GlobalPlatform Technology Virtual Primary Platform – VPP Firmware Format v2.0	[VFF]
ETSI TS 102 622	Smart Cards; UICC – Contactless Front-end (CLF) Interface; Host Controller Interface (HCI) Technical specification.	[HCI]
RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC 2119]
RFC 2141	URN Syntax	[RFC 2141]
RFC 4122	A Universally Unique IDentifier (UUID) URN Namespace	[RFC 4122]
OSI	ISO/IEC 7498-1:1994 and ITU-T X.200 (1994) "Open Systems Interconnection – Model and Notation – Basic Reference Model: The Basic Model"	[OSI]

1.4 Terminology and Definitions

ETSI specifications use the term “clause” rather than the term “section”.

Terms starting with a Capital letter are defined in Table 1-2 or described in a section of this document or both.

Table 1-2: Terminology and Definitions

Term	Definition
Active Host	Host able to process VNP Packets.
Application	Instance of an executable module after it has been installed.
Application Gate	A Gate attached to an Application that is consuming a Service.
Application Layer	According to [OSI], the protocol layer above the Presentation Layer. Protocol layer interacting directly with an Application or a Service.
Arbiter	The entity that manages Hosts Arbitration.
Credit	Parameter used in the data flow control on a pipe.
Data Link Layer	According to [OSI], the protocol layer between the Physical Layer and the Network Layer. It transports Data Link service data units up to the MTU of the Data Link Layer.
Execution Environment	An environment that hosts and executes software. This could be an REE, with hardware hosting Android, Linux, Windows, an RTOS, or other software; it could be a Secure Element or a TEE.
Frame	Protocol data unit used by the Data Link Layer.
Gate	An entry point of the Pipe towards a Service/Application that is operated inside a Host.
HCP Packet	Protocol data unit of the Transport Layer as defined in section 4.4.1.
Host	Group of Applications or Services sharing the same overall lifecycle; a logical entity that operates one or more Gate(s).
Host Alias	Short alias of a Host Identifier. The coding of the Host Alias is implementation dependent and allows a fast routing of VNP Packets to Hosts.
Host Domain	Group of Hosts sharing the same Host Domain Identifier.
Hosts Arbitration	The process of selecting the active Host(s) within a Host Domain.
Maximum Transmission Unit (MTU)	The size of the largest network layer protocol data unit that can be communicated in a single network transaction.
Mobile Broadband Modem	A hardware unit, discrete or an integrated subsystem that provides cellular network connectivity.
Namespace	Groups of identifiers around a particular functionality that are structured to avoid name collisions.
Network Layer	According to [OSI], the protocol layer above the Data Link Layer. This protocol layer conveys HCP Packets.
Pipe	A logical data circuit, conveying messages between two Gates using mechanisms of the Transport Layer and Network Layer as defined in this specification.

Term	Definition
Pipe Session	A pair of Dynamic unidirectional Pipes between two Gates. This allows the Gates to exchange messages in both directions as long as these Pipes exist.
Presentation Layer	According to [OSI], the protocol layer above the Session Layer.
Registry	Configuration data related to a Gate, stored as a collection of parameter – value pairs.
Regular Execution Environment (REE)	An Execution Environment comprising at least one Regular OS and all other components of the device (SoCs, other discrete components, firmware, and software) which execute, host, and support the Regular OS (excluding any Secure Components included in the device). From the viewpoint of a Secure Component, everything in the REE is considered untrusted, though from the Regular OS point of view there may be internal trust structures. (Formerly referred to as a *Rich Execution Environment (REE)*.) Contrast *Trusted Execution Environment (TEE)*.
Service	Provide function(s) consumed by an Application. If the Host runs within a VPP Application as defined in [VCI], a Service is embedded in an HLOS Application or an application of an Execution Environment within a HLOS Application.
Service Gate	A Gate attached to an Application that is implementing a Service.
Session Layer	According to [OSI], the protocol layer between the Presentation Layer and the Transport Layer. It controls the dialogues (connections) between two Gates. It establishes, manages, and terminates the connections between the source Host and destination Host.
Stream	Continuous flow of sequenced data chunks without duplication.
Tamper Resistant Element (TRE)	As defined in [VCI].
Transport Layer	According to [OSI], the protocol layer above the Network Layer. It controls the reliability of a given Pipe through segmentation of message and reassembly of message segments and potentially flow control mechanisms. The protocol data unit of the Transport Layer is the HCP Packet.
Transport Layer Credit-based Data Flow Control	Mechanism in the Transport Layer in which a gate acknowledges and indicates to the sending host its capability to receive data linked with a pipe session using a credit scheme that prevents the sender from sending more data than indicated by the receiver.
Transport Layer Data Acknowledgement	Mechanism in the Transport Layer in which a gate acknowledges to the sending host data received using a pipe session.
TRE Host	Any Host executing in a TRE.
TRE Host Domain	The group of TRE Hosts.

Term	Definition
Trusted Execution Environment (TEE)	An Execution Environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets against a set of defined threats which include general software attacks as well as some hardware attacks, and defines rigid safeguards as to data and functions that a program can access. There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly. Contrast *Regular Execution Environment (REE)*.
VNP Packet	Protocol data unit of the Network Layer as described in section 4.3.1.
VPP Application	As defined in [VCI].
VPP Firmware	Firmware compliant with the VPP Firmware Format as defined in [VFF].
VPP Firmware Family	As defined in [VFF].
VPP Network Layer	A network layer involving at least two Hosts and a particular Router supporting this specification.
VPP Network Protocol (VNP)	Network protocol of the VPP ecosystem as defined in this document.

1.5 Abbreviations and Notations

Selected abbreviations and notations used in this document are included in Table 1-3.

Table 1-3: Abbreviations

Abbreviation	Meaning
CB	Chaining Bit
HCI	Host Controller Interface as defined in [HCI]
HCP	Host Controller Protocol as defined in [HCI] clause 5.1
MBM	Mobile Broadband Modem
MTU	Maximum Transmission Unit
NA	Not Applicable
NCE	Network Controller Environment
NCH	Network Controller Host
NID	Namespace Identifier
NSS	Namespace Specific String
OEM	Original Equipment manufacturer
REE	Regular Execution Environment
RFU	Reserved for Future Use
RO	Read-Only
RW	Read/Write

Abbreviation	Meaning
SDO	Standards Development Organization (e.g. ETSI, ISO…)
SoC	System on Chip
TEE	Trusted Execution Environment
TRE	Tamper Resistant Element
URN	Uniform Resource Name
UUID	Universally Unique IDentifier as defined in [RFC 4122]
VNP	VPP Network Protocol
VPP	Virtual Primary Platform as defined in [VCI]
WO	Write-Only

The notations defined in Table 1-4 are used in the document.

Table 1-4: Notations

Notation	Meaning
$GATE_{XX}$	A Gate with a G_{ID} value equal to XX
G_{ID}	Gate Identifier as defined in section 3.2.2
P_{ID}	Pipe Identifier as defined in section 3.10
$PIPE_{ADM}$	Administration Pipe equivalent to $PIPE_{01}$
$PIPE_{LINK}$	Administration Pipe equivalent to $PIPE_{00}$
$PIPE_{XX}$	A Pipe with a P_{ID} value equal to XX

For the purposes of the present document, the following coding conventions apply:

- All lengths are presented in bytes, unless otherwise stated. Each byte is represented by bits b8 to b1, where b8 is the most significant bit and b1 is the least significant bit. In each representation, the leftmost bit is the most significant bit.
- Hexadecimal values are specified between single quotes, e.g. '1F'.
- All bytes specified as RFU shall be set to '00' and all bits specified as RFU shall be set to 0.
- Names starting with a capital letter have either a definition, an abbreviation, or a section defining the term.

1.6 Revision History

GlobalPlatform technical documents numbered *n*.0 are major releases. Those numbered *n*.1, *n*.2, etc., are minor releases where changes typically introduce supplementary items that do not impact backward compatibility or interoperability of the specifications. Those numbered *n.n*.1, *n.n*.2, etc., are maintenance releases that incorporate errata and precisions; all non-trivial changes are indicated, often with revision marks.

Table 1-5: Revision History

Date	Version	Description
March 2018	1.0	Public Release (with document reference GPC_FST_140)
October 2018	1.0.1	Maintenance Release
October 2019	1.0.1.7	Committee Review
December 2019	1.0.1.8	Member Review
February 2021	1.0.1.21	Public Review
TBD	2.0	Public Release (with document reference GPC_SPE_140)

2 Requirements

The VPP Network Protocol was designed under the global (GL) requirements described in Table 2-1.

Table 2-1: VNP Requirements

Ref	Requirement	Rationale
GL1	A Host may create or delete a Pipe Session to another Host during its whole life cycle. This should be performed with a minimum of messages. (Expecting 1 message from each Host.)	During its whole life cycle, the Host may create a Pipe Session and then accept a communication request for a given Gate from another Host. In addition, a Host may close a Pipe Session in order to prevent unsolicited communication.
GL2	Any Host shall be notified when its Pipe Session has been created.	Notifications are more efficient than the Host having to poll or query the destination Host before sending data.
GL3	Any Host shall be notified when its Pipe Session has been deleted.	A notification informs the Host that no more messages shall be sent over a Pipe.
GL4	Data flow control shall be possible per Pipe without preventing a stream protocol.	Each Pipe has its own data flow control even if the link layer is common and sharing a flow control. The Data Link layer may transport multiple VNP Packets related to the same Pipe or not and may appear as a stream for VNP Packets.
GL5	A Pipe may support a stream protocol.	In this mode, the message does not end. For this reason, VNP shall use another Pipe for notifications, reliability, and data flow control on the Pipe.
GL6	A Host shall have means for filtering Hosts that are allowed to communicate with it.	Whitelist of Hosts for reduced attack surface and the risk of DoS attacks.
GL7	Pipes shall keep the data conveyed unaltered.	The data sent through a Pipe shall remain unchanged.
GL8	Any data traffic on a Pipe to a Host may initiate the connection of the Host, for a Host Domain that supports a limited number of Active Hosts (e.g. TRE Host Domain).	Data traffic on a Pipe to a Host may be used to initiate the connection of a Host, eliminating the use of an explicit request for connecting a Host.
GL9	The Host shall provide generic means allowing another Host in the network to discover its identity and its Services.	Allow discovery of Hosts and Service capabilities.
GL10	The Host shall provide a generic Application for testing.	A loopback mechanism is essential for functional test suites (e.g. ETSI TS 102 695-1 / ETSI TS 102 695-3).
GL11	A Host shall have a means to notify another Host when its whitelist of Host has been updated.	Support for a Host to discover changes in the network of Hosts and Pipes.

Ref	Requirement	Rationale
GL12	The routing solution should apply a best effort mechanism to prevent identity spoofing of Hosts and domains.	Best effort for Host identification and domain in the limit of the surrounding security capability of the ecosystem.
GL13	A Host shall provide means for performing data flow control per Pipe.	Each Pipe shall independently manage its data flow control.
GL14	The Gate shall provide means for retrieving the size of the received Application data per Pipe.	It shall not be necessary to embed the content length in the Application data.
GL15	The Data Link Layer shall provide a reliable channel (i.e. error-free delivery, delivery failure detection, and in sequence).	The Network Layer shall ensure that all VNP Packets are conveyed properly without error and in sequence. In case of failure a notification shall be returned.
GL16	The Data Link Layer and/or the Physical Layer protocol shall provide a flow control which is independent of the flow control of the Network Layer protocol.	The Data Link Layer multiplexes the data flows of multiple Pipes. Then the data flow control of the Data Link Layer is independent of the Network Layer. How it is managed either by the Data Link Layer or the Physical Layer is not specified in this requirement.
GL17	The network protocol shall support at least 126 Pipes per Host.	Support of the capability of existing legacy (e.g. [HCI]) without adding a limitation.
GL18	The network protocol shall support at least 255 Hosts.	Support of the capability of existing legacy (e.g. [HCI]).
GL19	The identification of the Hosts and Gates shall be dynamic and shall not involve a registrar.	Remove the need to maintain a central repository/registrar of well-known Hosts.
GL20	The Data Link Layer protocol is shared between all Pipes.	Possibility for an implementation to share the transport of VNP Packets belonging to different Pipes.
GL21	All data units linked to an existing Pipe Session shall be flushed when the Pipe Session is deleted.	The reliability is guaranteed only during an existing Pipe Session. No store and forward on the data circuit for conveying the data units between Gates is required.
GL22	The VNP control and management need to have at least an equivalent level of security when compared to the TRE / TRE Host Domain.	The registration of the TRE Host has to be performed via the VPP Firmware Management Service interface and has an equivalent or greater security.

Editor's note: GL22 is subject to further discussion regarding its applicability on the embedded TRE.

3 VNP

3.1 Overview

A valid network is a collection of at least two Hosts that are logically connected together and physically connected via the Router to which the Host has the Static Pipe $PIPE_{00}$ (see Figure 3-1). VNP defines the interface between Hosts. More specifically, the network has three levels:

- a collection of Gates that shall exchange messages;
- a messaging mechanism that may exchange commands, responses, and events; and
- a routing principle for VNP Packets.

3.2 Topology

3.2.1 Main Principles

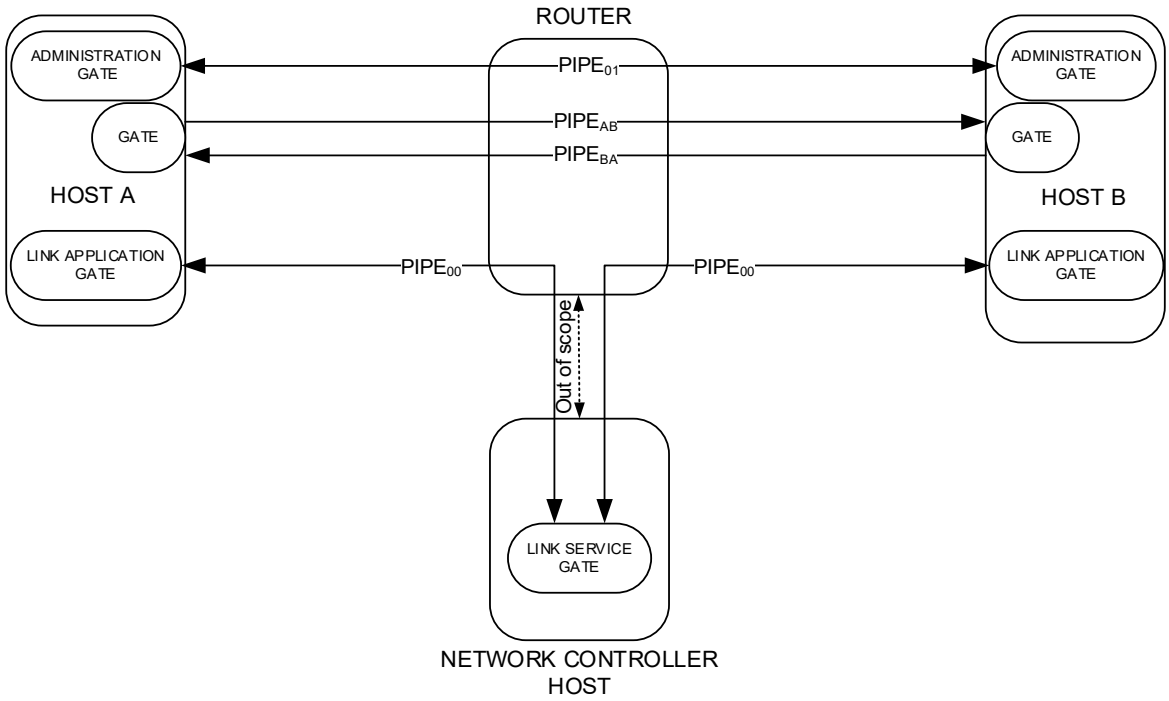
The VNP Network as described in this document consists of:

- Host Domains – Grouping Hosts (see Figure 3-4) according to functional perimeters which are dependent on a given implementation or defined within this specification (i.e. the TRE Host Domain).
- Hosts – Grouping Applications or Services.
- A Router to route VNP Packets between Hosts/Host Domains. At least one Host shall be in the TRE Host Domain. For example, routing VNP Packets between a Host in the TRE Host Domain and a Host in another Host Domain or for example, routing VNP Packets between TRE Host Domain and a non-TRE Host Domain.
- A Network Controller Host to control the Router.

Note: Transport of VNP Packets (e.g. between Host Domains, Hosts, non-VNP Routers, dispatchers...) outside the TRE Host Domain is out of scope of this document.

Applications and Services communicate over the network using Gates as entry point to this VNP network. Gates are connected by Pipes conveying messages. The Router is not a Host and is in charge of routing VNP Packets from a Host to another Host. The logical topology is a mesh network of Pipes. The Router appears transparent for any Pipe.

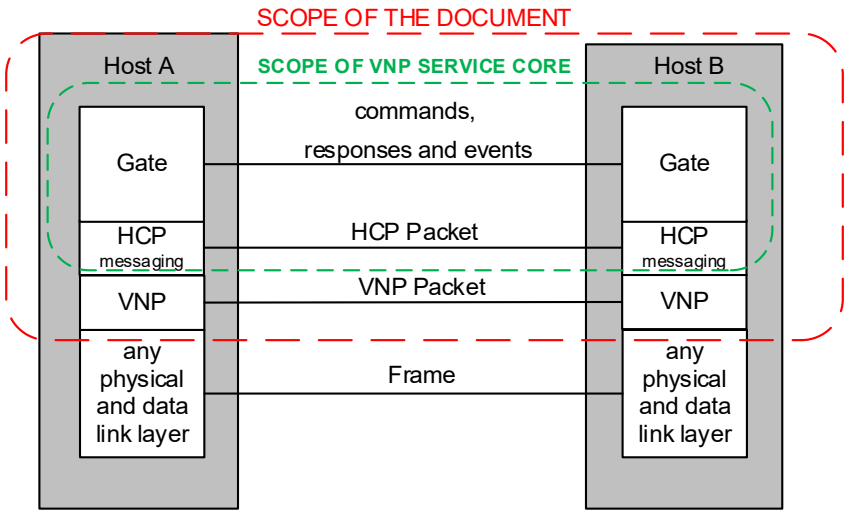
Figure 3-1: The Network



The communication always takes place between Gates. As illustrated in Figure 3-1, a Pipe may support either a bidirectional transfer of messages (i.e. $PIPE_{00}$ and $PIPE_{01}$) or a unidirectional transfer of messages (i.e. $PIPE_{AB}$ and $PIPE_{BA}$).

Figure 3-2 illustrates the protocol stack of an exemplary network. HCI commands, responses, and events in the figure are transported in the HCP messages.

Figure 3-2: Network Stack



Note: For clarity, only one Gate per Host is shown. In reality the Host also has Gates that connect via Pipes to other Gates and the Router (transparent at Pipe level). For simplicity, the Pipe between Gates appears bidirectional for illustrating any type of Pipes (static and dynamic).

3.2.2 Actors inside the TRE and Their Relationships

In the scope of this document, VNP is always used with a TRE Host Domain. In such a setup, several other hosts may be connected to the network. As this document defines parameters also for such hosts, this section introduces, as an example, a setup of a VNP network involving a TRE.

The following specific hosts are shown in Figure 3-3:

- The TRE Firmware Loader Agent host inside the TEE Host Domain, providing the user interface for the TRE Firmware Loader
- The TRE Firmware Loader host, responsible for loading a VPP Application into the TRE
- The VPP Application host
- An REE App host, which connects to the VPP application

Entities involved in the transmission of VNP Packets to their destination/source outside of the TRE Host domain are not represented.

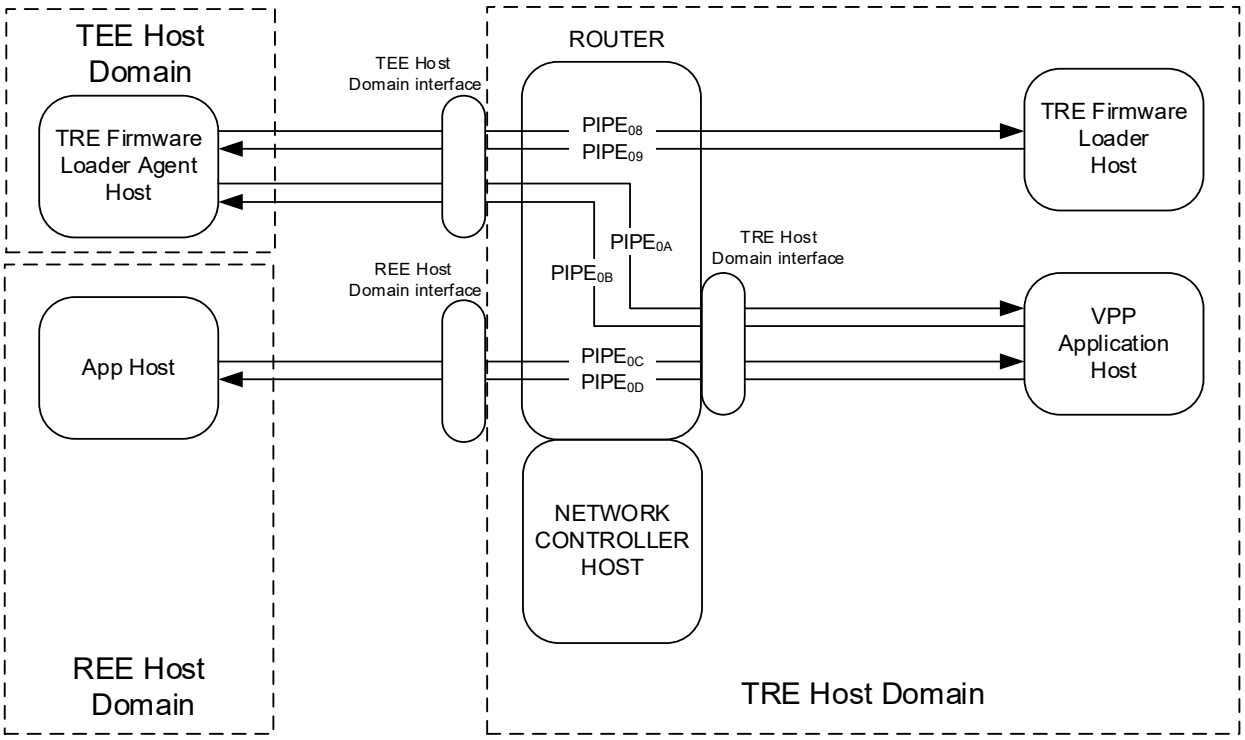
Other entities, e.g. MBM, are not shown in the example below for the sake of simplicity.

Note: While the section mentions VPP Application, an implementation may also decide to use the VNP without the specific requirements of VCI ([VCI]).

If the device implements the Open Firmware Loader ([OFL]), the TRE Firmware Loader is implemented by the OFL and the TRE Firmware Loader Agent by the OFL Agent.

Administrative and Link Service Pipes as well as the gates of the hosts are not shown in the diagram, and pipes identifiers below are provided as examples.

Figure 3-3: The TRE Network



$PIPE_{08}$ and $PIPE_{09}$ are used for loading a VPP Application.

$PIPE_{0A}$ and $PIPE_{0B}$ are used for initializing a VPP Application.

$PIPE_{0C}$ and $PIPE_{0D}$ are used for the operational communication between the App and the VPP Application.

3.2.3 VNP Router Interfaces and Delivery Mechanisms

The VNP Packet exchange between Hosts in the TRE Host Domain and the VNP Packets entering or leaving the TRE Host Domain are carried out by the Router. These VNP Packets are delivered to the destination using mechanisms that are not described in this document (e.g. REE inter process communication, physical channels, multiplexing, logical channels).

These mechanisms shall act transparently to the VNP Packet transmission as defined in section 4. However, beyond these provisions, their behavior and interaction are out of scope of this document.

Router stands only for the Router inside the TRE Host Domain. Therefore, router mechanisms defined in section 3.4 only apply to the Router in the TRE Host Domain.

3.3 Identifiers

3.3.1 Overview

Identifiers are used for uniquely addressing Hosts, Gates, and Host Domains. An Identifier is a UUID as defined in [RFC 4122] and is generated from names.

Identifiers as UUID version 5 from URN using the syntax defined in [RFC 2141]:

- Host Domain Identifier:
  - := NID is [domain.name] and NSS is the Host Domain name assigned by the OEM device maker.
  - := NID is [SDO domain.name] and NSS is the Host Domain name assigned by the SDO.
- Gate Identifier := from Gate URN as defined in Table 3-3 and Table 3-4
- VPP Firmware Family Identifier := NID is [OEM domain.name] or NID is [SDO domain.name] and NSS is VPP Firmware Family name.

Identifiers generated by the Network Controller Host:

- Host Identifier := a UUID version 4

Identifiers which may use either a UUID version 3 or version 5:

- Use Case Identifier := NID is [OEM domain.name] or [SDO domain.name] and NSS is the Use Case name, e.g. “keyless entry service”, as used in section 5.7.5.

In considering the following definitions:

- [domain.name] := The Fully Qualified Domain Name (FQDN) of the company defining the Host (e.g. BananaTel.com)
- [SDO domain.name] := The FQDN of a standards development organization or an industry forum (e.g. globalplatform.org)

3.3.2 Predefined Host Identifiers

Table 3-1 defines the generic Host Identifiers.

Table 3-1: Host Identifiers

Host	Pre-Generated Identifier
Wildcard Host	FFFFFFFF - FFFF - FFFF - FFFF - FFFFFFFFFFFF

3.3.3 Predefined Host Domain Identifiers

Table 3-2 defines the generic Host Domain Identifiers.

Table 3-2: Host Domain Identifiers

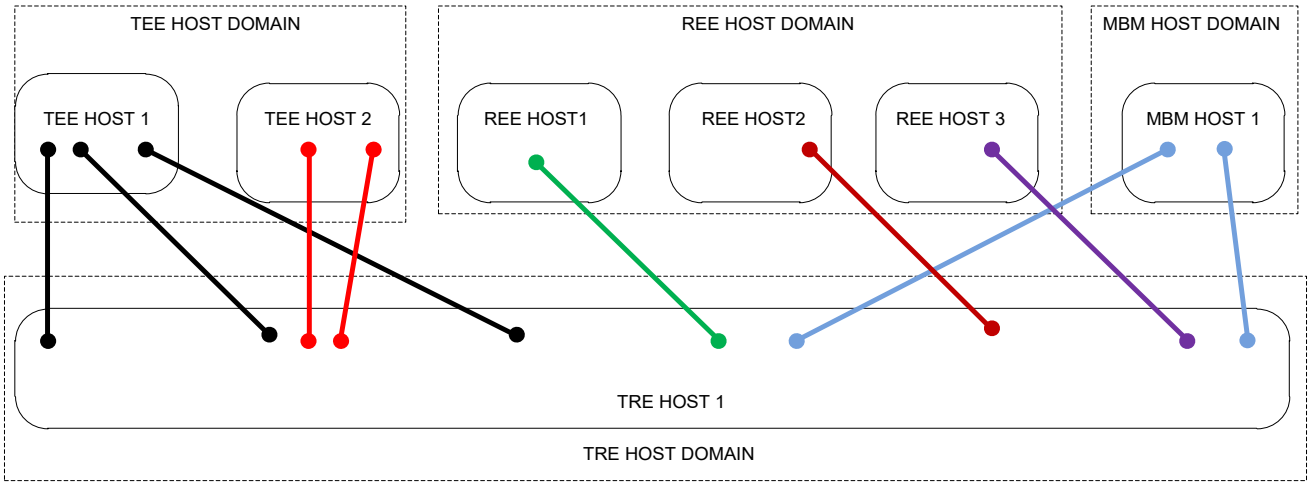
NSS	NID	URN	Pre-Generated Identifier
TRE	globalplatform.org	urn:globalplatform.org:TRE	DCDC79DF-29FA-5E57-B1CF-2026DDDB8C47
REE		urn:globalplatform.org:REE	E7A14FE4-378D-51AC-85C8-05F6504A7C91
TEE		urn:globalplatform.org:TEE	F69AAE7B-A8C2-57BD-BAEF-99A4534A20D4
MBM		urn:globalplatform.org:MBM	6798BF4D-92C0-569C-BC72-D552B8757A04

The Host Domain Identifier of FFFFFFFF - FFFF - FFFF - FFFF - FFFFFFFFFFFF shall be reserved for wildcard Host Domain Identifier, used by Host Domain whitelist.

Figure 3-4 illustrates an example of a network supporting the following Host Domain assignments:

- TRE Host Domain for the group of Hosts in the TRE
- TEE Host Domain for the group of Hosts in the Trusted Execution Environment
- MBM Host Domain for the group of Hosts running on the Mobile Broadband Modem
- REE Host Domain for the group of Hosts in the Rich Execution Environment

Figure 3-4: Example of Network Involving Multiple Host Domains



3.4 Router

3.4.1 Overview

The Router is in charge of forwarding VNP Packets from one Host to another.

The Router routes VNP Packets based on the Host Aliases found in each VNP Packet.

The Router shall be able to route VNP Packets from/to a Host within the TRE Host Domain.

The Router may route VNP Packets between Hosts in any Host Domain.

The routing of VNP Packets between Hosts shall be error-free and the order of the received/sent VNP Packets shall be respected.

The Router shall verify the Host Alias of the source Host Identifier (i.e. SH_{AL}) of each VNP Packet against the identification of its originator Host.

All statements in this section apply within the limits set out in GL12 in Table 2-1.

3.4.2 Host and Host Domain Whitelists

The Network Controller Host shall manage two separate whitelists per Host, to filter access to the Host. One whitelist is based on Host Identifiers and the other list is based on Host Domain Identifiers.

The Router shall only route VNP Packets from one Host to another if at least one of the following conditions is fulfilled:

- The source Host Identifier is in the Host whitelist of the destination Host or the whitelist contains a wildcard for that Host Identifier.
- The source Host Domain Identifier of the source Host is in the Host Domain whitelist of the destination Host or the whitelist contains a wildcard for that Host Domain Identifier.

The Router shall not forward any VNP Packet from Host A to Host B if Host A is not in the whitelist of Host B.

Note: A TRE Host has its Host Domain whitelist automatically appended with the TRE Firmware Loader Agent Host Domain by the Network Host Controller as defined in section 5.3.2.6. Therefore, a TRE Host cannot have an empty Host Domain whitelist.

The Network Controller Host Identifier is implicitly in the Host Whitelist of any Host.

3.5 Host

A Host contains Gates. The Gates interface with either Applications (consuming a function) or Services (providing a function). A Host is identified by its Host Identifier (see section 3.2.2) and supports the mandatory core Gates in Table 3-3.

For Hosts belonging to the TRE Host Domain, the following statements shall apply:

- The registration of a Host with the Network Controller Host is done with its VPP Firmware Family Identifier as defined in [VFF] by the TRE Firmware Loader as defined in [VCI]. This happens at the time of enabling the VPP Firmware.
- The deregistration of a Host from the Network Controller Host is done with its VPP Firmware Family Identifier as defined in [VFF] by the TRE Firmware Loader as defined in [VCI]. This happens at the time of disabling the VPP Firmware.
- There is a single Host per VPP Application as defined in [VCI].
- The interface between the TRE and the Network Controller Host for performing this registration and the entities controlling this process are implementation dependent and out of scope of this specification.

Registration and deregistration of Hosts not belonging to the TRE Host Domain are performed using the commands LINK_REGISTER_HOST (see section 5.3.2.3) and LINK_DEREGISTER_HOST (see section 5.3.2.4).

3.6 Network Controller Host

The Network Controller Host provides functionalities to:

- Register each Host in the Host network for Host discovery.
- Assign and manage the uniqueness of the aliases of the registered Hosts.
- Manage two whitelists for each Host, of other Hosts and other Host Domains that are permitted to send VNP Packets to it.
- Notify all registered Hosts, if requested, about the registration of a new Host.
- Notify all registered Hosts, if requested, about a Host deregistration.
- Report errors in routing to Hosts.

The Host Alias of the Network Controller Host is 0.

The interfaces between the TRE and the Network Controller Host for performing this management and the entities controlling this process are out of scope of this specification.

3.7 Host Alias

To reduce the data overhead of the VNP Packet, a Host shall have an alias of its Host Identifier. The Host Alias is dynamically assigned by the Network Controller Host at registration time. This mechanism applies for Hosts outside the TRE Host Domain as defined in section 5.3.2.3. The coding of Host Alias is implementation dependent.

Unregistered Host shall use (SH_{AL}) Host Alias '01'.

The Host Alias is persistent while the Host is registered with the Network Controller Host.

Note: Lifetime of registration is implementation dependent; e.g. it may be related (or not) to the power cycle of the device implementing the host.

3.8 Hosts Arbitration

3.8.1 Overview

Hosts Arbitration is the process for selecting, within a Host Domain, the Active Hosts able to receive incoming VNP Packets.

Hosts Arbitration requires an Arbiter, as an abstract entity able to make the selection. Hosts Arbitration is required when the number of Hosts having incoming VNP Packets exceeds the limit of Active Hosts.

The Arbiter decides which Host should become or remain active, and which Host may no longer be active.

The communication between the Network Controller Host and the Arbiter is implementation dependent and out of scope of this specification. The Arbiter interacts with the control of the lifecycle VPP Application session (see [VCI] section 6.3) and is out of scope of this specification.

3.8.2 Host Domain Priority

Each Host Domain has a fixed priority.

The Network Controller Host publishes the Host Domain priority which is accessible to any Host as defined in section 5.3.2.2.

Depending on whether the Host sends or receives VNP Packets (i.e. source or destination Host), its priority is dynamic or static.

A source Host sending a VNP Packet to a destination Host inherits the static priority of the source Host's Host Domain.

A destination Host, having a pending incoming VNP Packet from a source Host, has a dynamic Host Domain priority. The dynamic Host Domain priority is equal to the highest static Host Domain priority of its source Host(s).

On Host Domains that requires Hosts Arbitration (e.g. TRE host domain), an Active Host may be pre-empted by a non-Active Host within the same Host Domain if:

- The non-Active Host has a higher dynamic Host Domain priority than the Active Host

  Or

- The Active Host has declared no more activity by sending the EVT_LINK_END_OF_OPERATION as defined in section 5.3.4.2.

The Hosts Arbitration shall favor a round-robin principle when multiple Hosts compete within the same Host Domain and with the same dynamic Host Domain priority.

The Host that gains by the Hosts Arbitration shall be an Active Host. Incoming VNP Packets for non-Active Host(s) may be dropped. As a consequence, a non-Active Host may appear as not reachable.

3.9 Gates

A Gate provides a single entry point for a Pipe toward a Service or to an Application running in a Host.

VNP Core Services enable the communication between Service Gates and Application Gates, e.g. for the discovery for Service Gates or creation of Pipe Sessions.

A Gate is identified by its Identifier, namely a UUID, computed from the Gate URN as defined in section 3.3. The coding of the Gate URN, as defined in [RFC 2141], ensures the uniqueness of a Gate Identifier for the same Service name belonging to Hosts within different Host Domains.

The Core Gate URN are listed in Table 3-3.

Table 3-3: Core Gates URN

Gate	URN	Identifiers
Identity Gate	urn:globalplatform.org:TRE:HCI:identity_ gate_V10	416B66AC-A134-5082-8160-FA1BA497F917
Loopback Gate	urn:globalplatform.org:TRE:HCI:loopback_ gate_V10	1CE3D0F5-3B55-5470-B6F1-168352F27440

Table 3-4 predefines Gate URN for addressing generic purpose Services using Presentation and Application layers defined in section 5. The Gates use the protocol accommodations defined in Annex B.

Table 3-4: Pre-defined Generic Gates URN

Gate	NID	NSS
Any Gate from OEM	[OEM domain.name]	[Host Domain Name]:[Presentation Layer name]:[service name]
Any Standard Gate from a SDO	[SDO domain.name]	[Host Domain Name]:[Presentation Layer name]:[service name]
ISO 7816 Gate	globalplatform.org	[Host Domain Name]:ISO7816:[service name]
IPv4 Gate	globalplatform.org	[Host Domain Name]:IPV4:[service name]
IPv6 Gate	globalplatform.org	[Host Domain Name]:IPV6:[service name]
HCI Gate	globalplatform.org	[Host Domain Name]:HCI:[service name]

Where:

- [OEM domain.name] is the Fully Qualified Domain Name of the company issuing the Host.
- [SDO domain.name] is the Fully Qualified Domain Name of a standards developing organization or a forum (e.g. globalplatform.org).
- [service name] is the name of the specific Service (e.g. “loopback_V10”).
- [Host Domain Name] is the name of the Host Domain defined by the OEM or a SDO (e.g. TRE).

The example below details how the URN of the ASN Gate for interfacing the file system service within the TRE host domain is built as defined in Table 3-4:

urn:globalplatform.org:TRE:ASN-1:File_System_V10

The computed URN above (the UUID) becomes the ASN Gate Identifier according to section 3.3:

64AB623F-271C-5438-93BB-E3F9A9875193

A Host may provide predefined or proprietary Gates.

The Gate providing a Service shall be in the GATE_LIST registry entry of the Identity Service Gate of its Host (see section 5.7.5).

3.10 Pipes

3.10.1 Overview

The Pipe Identifier, P_{ID}, is a 7-bit unsigned integer. The value of P_{ID} is used as inter-Host routing information.

A Dynamic Pipe is unidirectional.[1] It allows a Gate of the receiving Host to receive messages from a Gate of the sending Host to which it has been bound. The Gates on the receiving and sending Host have the same Gate Identifier.

Messages exchanged between two Gates may be conveyed on different Dynamic Pipes.[1]

The Host initiating the Dynamic Pipe creation is the receiving Host of this Pipe and assigns the identifier of the Pipe by means of the EVT_ADM_BIND management command (see section 5.5.3.2).

For Static Pipes, the Pipe Identifiers are predefined with values as defined in Table 3-5.

A Static Pipe is always bidirectional, while Dynamic Pipes are used in pairs in the context of Pipe Sessions.

Table 3-5: Pipe Identifiers

P_{ID}	Pipe ending at:	Pipe type
'00'	The Network Controller Host Link Service/Application Gate	Static
'01'	A Host Administration Gate	Static
'02' to '07'	RFU	RFU
'08' to '7E'	any Gate	Dynamic
'7F'	Reserved for the notification of the rejection of a Pipe Session creation	Static

[1] This definition deviates from the definition in [HCI].

3.10.2 Pipe Session

A Pipe Session consist of a pair of Dynamic Pipes. A Pipe Session is required for a bidirectional communication between two Gates. The Pipe Identifiers may be different for both Pipes.[1]

The Pipe Session is created after two successful pipe binding operations as defined in section 5.5.3.2.

Any VNP Packet on a Pipe shall be ignored unless a Pipe Session (i.e. pair of unidirectional Dynamic Pipes) has been successfully created using that Pipe.

The Pipe Session may be deleted as defined in section 5.5.3.3 or in the situations outlined in section 3.10.3 (e.g. application termination, power loss).

When a Pipe Session is deleted, then it is no longer possible to transfer messages using this Pipe Session and all HCP Packets associated with this Pipe Session will be discarded.

There can be one or more Pipe Sessions between two Gates with the same Gate Identifier of two Hosts.

A Pipe Session can be created between $GATE_A$ in Host A and $GATE_B$ in Host B as follows:

- Host A allocates $PIPE_A$ for $GATE_A$ (implementation dependent).
- Host A sends EVT_ADM_BIND to Host B with Gate Identifier of $GATE_A$ and the Pipe Identifier of $PIPE_A$.
- Host B allocates $PIPE_B$ for $GATE_B$ (implementation dependent).
- Host B sends EVT_ADM_BIND to Host A with Gate Identifier of $GATE_B$ and the Pipe Identifier of $PIPE_B$; $GATE_A$ and $GATE_B$ have the same Gate Identifier.

Once a Pipe Session is created, then:

- $GATE_A$ may receive VNP Packets from $GATE_B$ over $PIPE_A$.
- $GATE_B$ may receive VNP Packets from $GATE_A$ over $PIPE_B$.

The Pipe Identifiers bound with the Gates $GATE_A$ and $GATE_B$ may be different.

If the Transport Layer Data Acknowledgement mechanism is used, then it shall be implemented for both Pipes belonging to a Pipe Session, as defined in section 5.5.3.5.

If the Transport Layer Credit-based Data Flow Control mechanism is used, then the Transport Layer Data Acknowledgement mechanism shall be used and both mechanisms shall be implemented for both Pipes belonging to a Pipe Session, as defined in sections 5.5.3.4 and 5.5.3.5.

3.10.3 Global Pipe Session Deletion

All Pipe Sessions between two Hosts are deleted and all related Pipe Identifiers are unbound if at least one of the following events is received by one of these two Hosts:

- EVT_LINK_HOST_DEREGISTERED as defined in section 5.4.4.5
- EVT_LINK_HOST_NOT_REACHABLE as defined in section 5.4.4.3
- EVT_LINK_HOST_ACCESS_DENIED as defined in section 5.4.4.2
- EVT_ADM_ALL_PIPE_SESSIONS_CLOSED as defined in section 5.5.3.8

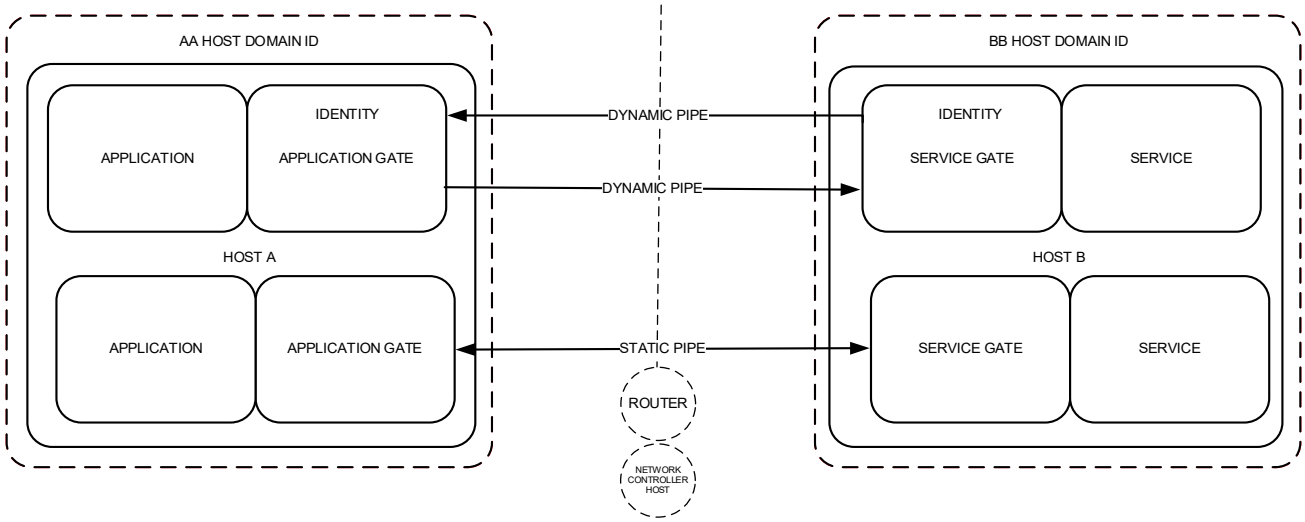
Subsequent to the above global Pipe Session deletion or a Host registration, the registered Host A may send to the registered Host B the following events:

- EVT_ADM_HOST_PING as defined in section 5.5.3.7
- EVT_ADM_GATE_LIST_UPDATED as defined in section 5.5.3.6
- EVT_ADM_CREDIT as defined in section 5.5.3.4
- EVT_ADM_BIND as defined in section 5.5.3.2

3.11 A VNP Network

Figure 3-5 illustrates an example of a VNP Network as defined in this document.

Figure 3-5: Example VNP Network



The following rules apply to Pipes:

- A Pipe conveys messages between two Gates located in two different Hosts.
- A Dynamic Pipe conveys messages between two Gates with the same Gate Identifier.
- A Static Pipe connects administrative gates. The only administrative gates in the current version of the specification are the Administration Application/Service Gates and the Link Application/Service Gates.

The Gate interfaces with an Application or a Service or both (e.g. Administration Gate which is Application and Service implemented in itself). A Service provides a function and an Application consumes the function.

Service Gates using Dynamic Pipes shall be listed in the GATE_LIST registry entry of the Identity Service Gate defined in section 5.7.5. Service Gates that use Static Pipes shall not be listed in the GATE_LIST registry entry.

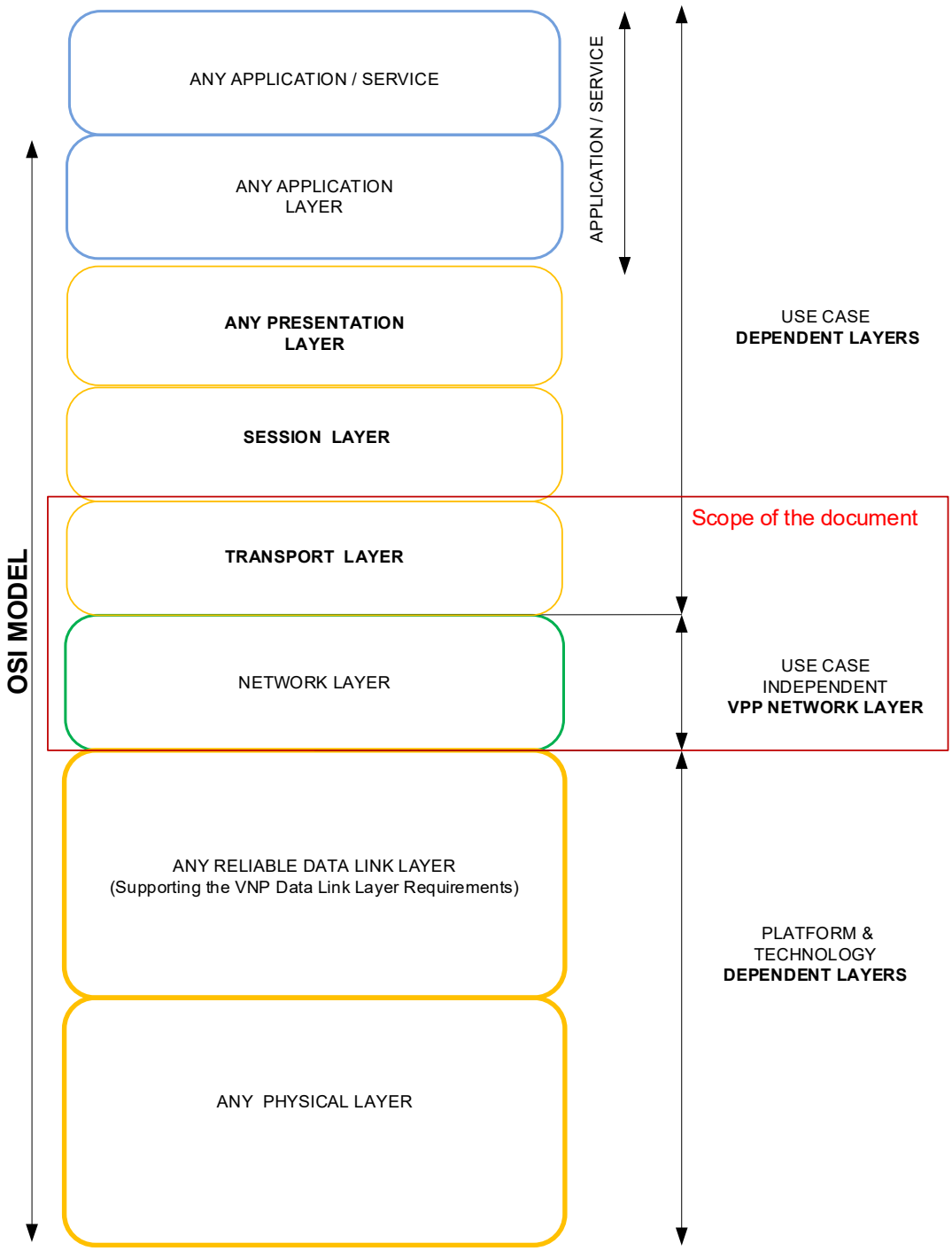
A VNP Network as defined in this document has a single Network Controller Host and a single Router.

4 Protocol Layers

4.1 Overview

Figure 4-1 illustrates the protocol stack supporting the VNP.

Figure 4-1: Protocol Stack



4.2 Data Link Layer

For proper operation, the VNP network layer requires underlying Data Link layers which shall support the following requirements:

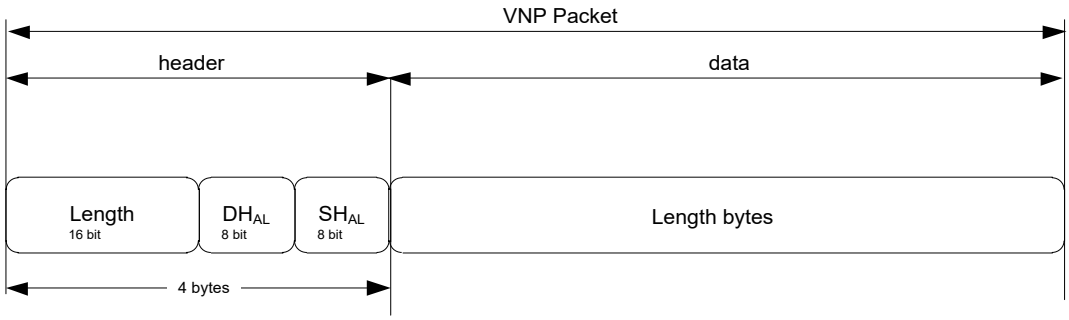
- Transfer VNP Packets between a Host and the Router.
- Be error-free and respect the order of the received/sent VNP Packets without duplication of VNP Packets.
- Provide its own data flow control.
- Be balanced (no master/slave behavior).
- Be able to convey VNP Packets of the upper layer (PDUs of the upper layer protocol) up to a maximum size specific to the Data Link layer (MTU). The MTU shall be the one referenced in Table 5-17 (VNP_MTU).
- Report the size of each received VNP Packet to its upper layer.
- Report the maximum supported VNP Packet length (i.e. MTU).

4.3 Network Layer

4.3.1 VNP Packet

Using the Data Link Layer, Hosts exchange VNP Packets. The format of a VNP Packet structure is shown in Figure 4-2.

Figure 4-2: VNP Packet Structure



Where:

- Length is the length of the data. The Length is encoded as big endian.

The routing of VNP Packets between two Hosts uses the following information:

- DH_{AL} is the alias of the destination Host Identifier and its Host Domain Identifier, defined in section 3.3.
- SH_{AL} is the alias of the source Host Identifier and its Host Domain Identifier, defined in section 3.3.

The Network Layer is considered reliable due to the support of the Data Link Layer requirements:

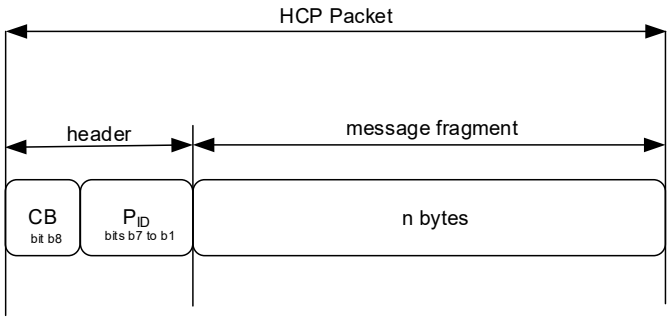
- VNP Packets are conveyed error-free, without duplication and in sequence.
- VNP Packet delivery failure is reported by the Network Controller Host to the other Host.

4.4 Transport Layer

4.4.1 HCP Packet

The message fragments resulting from the fragmentation process described in section 4.4.2 are encapsulated in HCP Packets as defined in [HCI] clause 5.1. The HCP Packet is the protocol data unit of the Transport Layer. The structure is illustrated in Figure 4-3.

Figure 4-3: HCP Packet Structure



The delivery of HCP Packets within a destination Host is performed using the Pipe Identifiers P_{ID} (7 bit).

Pipes and Pipe Sessions are based on the Transport Layer.

The Pipe Identifier (P_{ID}) is defined in Table 3-5 for Static Pipes and in section 5.5.3.2 for Dynamic Pipes.

4.4.2 Message Fragmentation and Reassembly

The service data unit of the Transport Layer is the message. It has an arbitrary length and will be fragmented and reassembled as described in this section.

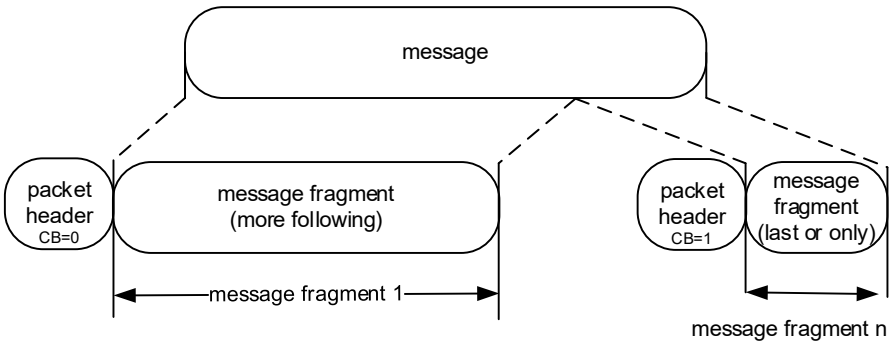
Due to the overhead of the VNP Network Layer (4 bytes) and the Transport Layer (1 byte), messages that are larger than the service data unit of the Data Link Layer must be fragmented as defined in [HCI] clause 5.3 and illustrated in Figure 4-2. This results in a maximum un-fragmented message size of MTU of the Data Link Layer minus 5 bytes.

The support of fragmentation and reassembly of the Transport Layer is optional in an implementation, if all messages exchanged have a size lower than MTU minus 5 bytes.

Fragmentation and reassembly of message fragments are performed based on the chaining bit (CB) in the HCP Packet header as defined in [HCI] clause 5.3 and illustrated in Figure 4-2.

If fragmentation and reassembly of the Transport Layer is not used, then CB shall be set to 1 and the size of the message shall not exceed the MTU minus 5 bytes.

Figure 4-4: Fragmentation and Reassembly



4.5 Session Layer

The Session Layer and the layers above are out of scope of this document.

5 VNP Core Services

5.1 Overview

All Hosts shall implement all the Gates defined in section 5 with the following exceptions:

- The Network Controller Host shall not implement the Link Application Gate.
- Only the Network Controller Host shall implement the Link Service Gate.

The Gates defined in Table 3-3 shall be implemented.

The core Gates provide the minimal services for:

- Setting up of Pipe(s) between two Gates of two Hosts allowing the Router to perform routing of VNP Packets
- Testing communication to a Host
- Discovering the identity and the capability of a Host

All Gates defined in section 5 shall use the HCI Presentation Layer defined in [HCI] clause 5.2 by mapping an HCP message as defined in [HCI] clause 5.2 on a VNP Packet as defined in section 4.3.1. The HCP message structure defines a data representation as type, instruction, parameters where the parameters are mapped on a structure of data and its semantics which is type and instruction dependent.

For instructions the following rules apply:

- An event is defined within the scope of the Gate which accepts the events.
- A command is defined within the scope of the Gate which accepts the command.
- A response is defined within the scope of its associated command.

A Gate shall not send a command on the Pipe of a Pipe Session when it is waiting for a response to a previous command from the Pipe of that Pipe Session; however, the Gate is able to send events on the Pipe of that Pipe Session. It is assumed that the receiving Gate is able to receive an event while processing a command.

Note: This is a deviation from the restrictions described in [HCI] where events shall not be sent when waiting for a response.

Gates of the VNP Core Service using Static Pipes do not require the Transport Layer Data Acknowledgement and Transport Layer Credit-based Data Flow Control as defined in sections 5.5.3.4 and 5.5.3.5.

All Gates of the VNP Core Service using Dynamic Pipes require the Transport Layer Data Acknowledgement and Transport Layer Credit-based Data Flow Control as defined in sections 5.5.3.4 and 5.5.3.5.

The endianness used in the VNP Core Service is big endian.

The VNP version as defined in section 5.5.4 supported by a Host during the communication with another Host shall be the smallest VNP_VERSION of both Hosts but shall not be lower than their respective MIN_VNP_VERSION.

For clarity, sections 5 and 6 refer to Host A or Host B to prevent any ambiguity about the actors involved in a dialogue between two Hosts.

5.2 Common Core Services

5.2.1 Overview

The following sections describe features used by all Gates of the VNP Core Services.

5.2.2 Commands

5.2.2.1 Overview

Table 5-1 lists all commands that will be described in the following sections.

There are two types of commands:

- Common Core commands that are applicable to all VNP Core Service Gates (described below); and
- VNP Core Service specific commands needed for the management of the network (described in section 5.3.2).

Table 5-1: Common Core Commands

Gate	Command	Description	Instruction
All Core Gates	ANY_SET_PARAMETER	The command to set a parameter in a registry	'01'
	ANY_GET_PARAMETER	The command to get a parameter from a registry	'02'
	NA	RFU	'03' to '0F'
	Gate specific commands	The command is defined in the scope of the Gate	'10' to '3F'

For every command the following information is provided:

- A description of the command
- A list of parameters for the command
- The response on successful completion of the command (with optional parameters)

The mapping of possible error codes is given in section 5.2.3.

5.2.2.2 ANY_SET_PARAMETER

The command ANY_SET_PARAMETER is used to write a parameter value in the registry of a Gate.

The command parameters are as follows:

Table 5-2: ANY_SET_PARAMETER Command Parameters

Description	Length
Identifier of the entry in the registry	1
Value of the parameter; the length and content of the parameter depend on the registry entry	N_1

When successful, the Host shall respond with ANY_OK without any parameters.

5.2.2.3 ANY_GET_PARAMETER

The command ANY_GET_PARAMETER is used to read a parameter value from a Gate registry entry.

The command parameters are as follows:

Table 5-3: ANY_GET_PARAMETER Command Parameter

Description	Length
Index of the entry in the registry	1

When successful, the Host shall respond with ANY_OK with parameter as follows:

Table 5-4: ANY_GET_PARAMETER Response Parameter

Description	Length
Value of the parameter; the length and contents of the parameter depend on the Gate	N_2

5.2.3 Responses

For the commands specified in section 5.2.2, Table 5-5 defines the possible response codes. Table 5-6 defines the possible response code for each command. Unless stated elsewhere, these responses carry no additional parameters.

Table 5-5: Common Core Responses

Value	Response	Description
'00'	ANY_OK	Command completed successfully (with optional parameters).
'01'	NA	RFU
'02'	ANY_E_CMD_PAR_UNKNOWN	The format of the command parameters is wrong.
'03'	ANY_E_NOK	Command was rejected and/or not completed.
'04'	NA	RFU
'05'	ANY_E_REG_PAR_UNKNOWN	The registry parameter identifier is either unknown to the registry or an optional registry parameter is not implemented.
'06'	ANY_E_CMD_NOT_SUPPORTED	The command is not supported by the Gate.
'07'	NA	RFU
'08'	ANY_E_REG_ACCESS_DENIED	Permission denied while writing a value to a registry.
'09' to '3F'	NA	RFU

Table 5-6: Reponse/Command Matrix

Error Code / Command	ANY_OK	ANY_E_CMD_PAR_UNKNOWN	ANY_E_NOK	ANY_E_REG_PAR_UNKNOWN	ANY_E_REG_ACCESS_DENIED
ANY_SET_PARAMETER	•	•	•	•	•
ANY_GET_PARAMETER	•	•	•	•	•

5.2.4 Registry

Each Gate has a registry. Its entries contain definitions related to the Gate. The following rules apply:

- Registry entries are identified by identifiers consisting of one byte.
- These identifiers are unique within the scope of a particular Gate (e.g. Table 5-32: Administration Service Gate Registry).
- A new instance of the registry is created on each end of the Pipe(s) that connects to the Gate. Upon Pipe Session creation, all registry entries shall be set to their default values, by the entity managing the Gate and its registry.
- A Host is responsible for managing its associated registries. When a Pipe Session is deleted, the registry instance associated with these Pipe(s) is deleted.

The value of a registry entry may be stored or retrieved. Registry entries vary by type of Gate.

5.3 Link Service Management and Link Service Gate

5.3.1 Overview

The Link Service Gate in the Network Controller Host provides access to parameters that control the routing of VNP Packets between Hosts.

Link Service Management:

- Manages the life cycle and errors of communication related to Hosts and interfaces with the Link Service Gate.
- Ensures the uniqueness of the Host Alias for each registered Host.
- Is implementation dependent and out of scope of this document.

5.3.2 Commands

5.3.2.1 Overview

The Link Service Gate supports the following commands:

Table 5-7: Link Service Gate Commands

Value	Command
'10'	LINK_GET_HOST_LIST
'11'	LINK_REGISTER_HOST
'12'	LINK_DEREGISTER_HOST
'13'	LINK_SET_HOST_WHITELIST
'14'	LINK_SET_HOST_DOMAIN_WHITELIST

5.3.2.2 LINK_GET_HOST_LIST

The command LINK_GET_HOST_LIST provides the list of all Hosts registered in the VNP Network to the Link Service Gate. This command is intended to be used to discover the current configuration of the VNP Network after the registration of a new host.

This command has the following parameter:

Table 5-8: LINK_GET_HOST_LIST Command Parameter

Description	Length
Array of size N_3 registered Host Aliases (defined in section 3.7). N_3 = 0 (no command parameter) is a possible value.	N_3

When successful, the Link Service Gate shall respond with ANY_OK with parameter as listed in Table 5-9.

If there is no host related to a Host Alias requested in the command parameter, there shall be an entry for this alias in the response parameter with a zeroed structure except the Host Alias. The order of entries in the response shall be preserved.

Table 5-9: LINK_GET_HOST_LIST Response Parameter

Description	Length
Array of N_3 registered Host structures defined in Table 5-10.	N_3 x 51

If N_3 = 0 (no command parameter), then all registered Host structures shall be returned in the response.

Table 5-10: Registered Host Structure

Description	Length
Host Identifier as defined in section 3.3	16
Host Domain Identifier as defined in section 3.3	16
VPP Firmware Family Identifier as defined in section 3.3	16
Host Alias as defined in section 3.7	1
Priority of the Host Domain: 0 (lowest) to 15 (highest)	1
Informative Level of security of the Host Domain: '00' (lowest or unknown) to '07' (highest)	1

If the LINK_GET_HOST_LIST command parameter is empty (N_3 = 0), then the LINK_GET_HOST_LIST response parameter shall contain the registered Host structures of all registered Hosts. Otherwise the LINK_GET_HOST_LIST response parameter shall only contain the registered Host structures of registered Hosts having their Host Alias within the LINK_GET_HOST_LIST command parameter.

5.3.2.3 LINK_REGISTER_HOST

With the command LINK_REGISTER_HOST, a Link Application Gate may register the VPP Firmware Family Identifier of its Host and obtain its Host Alias.[2]

The registration of Hosts of the TRE Host Domain is persistent across power on/off cycles and is defined in section 3.5.

The command parameter is the following:

Table 5-11: LINK_REGISTER_HOST Command Parameters

Description	Length
Registering VPP Firmware Family Identifier (defined in section 3.3)	16

The Network Controller Host shall generate a new Host Identifier as defined in section 3.2.2.

If the Host belongs to the TRE Host Domain, the following statements apply:

- The Link Service Gate shall not support a LINK_REGISTER_HOST command from these Hosts.
- The TRE Host shall obtain its Host Alias by parsing the header of any incoming VNP Packet.
- The Network Controller Host shall initialize the Host Domain whitelist with Host Domain Identifier containing the TRE Firmware Loader Agent Host.

If the Host does not belong to the TRE Host Domain and if the command is successful, the following statements apply:

- The Network Controller Host shall initialize the Host Domain whitelist with the wildcard Host Domain Identifier.
- The Host whitelist shall be set to an empty value and the Network Controller Host shall respond with ANY_OK.
- The Network Controller Host shall use the generated Host Alias in the Header of the VNP Packet sent as a response to this command.
- The originator of this command shall extract the Alias and use it as its own Host Alias (implicit address assignment for Host outside the TRE Host Domain).

When successful, the Link Service Gate shall respond with ANY_OK with parameter as follows:

Table 5-12: LINK_REGISTER_HOST Response Parameter

Description	Length
The updated value of HOST_LIST_REVISION; see Table 5-17.	4

[2] By reading the VNP Packet Header

5.3.2.4 LINK_DEREGISTER_HOST

With the command LINK_DEREGISTER_HOST, a Link Application Gate may deregister its Host Alias.

The Link Service Gate shall not support a LINK_DEREGISTER_HOST command from a TRE Host.

The Link Service Gate shall update the whitelist of all registered Hosts, i.e. removing the Host Identifier of the deregistered Host, when it receives the deregistration event or when the Host in the TRE domain is deactivated or deleted.

The command has no parameters.

When successful, the Host is deregistered, and it shall respond ANY_OK without parameters.

The deregistration of a Host is completed when the Link Service Gate returns ANY_OK without parameters.

5.3.2.5 LINK_SET_HOST_WHITELIST

With the command LINK_SET_HOST_WHITELIST, a Link Application Gate may register its whitelist into the Link Service Gate.

Host A, prior to removing the identifier of Host B from its whitelist, should send the EVT_ADM_ALL_PIPE_SESSIONS_CLOSED event to Host B.

The command parameter is the following:

Table 5-13: LINK_SET_HOST_WHITELIST Command Parameters

Description	Length
Array of N_4 Host Identifiers of Hosts allowed to send VNP Packet(s) to the Host registering the whitelist. N_4 shall not exceed MAX_SIZE_HOST_WHITELIST, defined in section 5.3.5.	N_4 x 16

For Hosts within the TRE Host Domain, the registration of the Host whitelist is persistent across power on/off cycles.

This command shall be rejected by the Link Service Gate if the Host is not registered with the Link Service Gate.

When successful, the Network Controller Host shall respond with ANY_OK without parameter.

5.3.2.6 LINK_SET_HOST_DOMAIN_WHITELIST

With the command LINK_SET_HOST_DOMAIN_WHITELIST, a Link Application Gate may register its Host Domain Identifier(s) whitelist into the Link Service Gate.

The command parameter is the following:

Table 5-14: LINK_SET_HOST_DOMAIN_WHITELIST Command Parameters

Description	Length
Array of N_5 Host Domain Identifiers of Hosts allowed to send VNP Packet(s) to the Host registering the Host Domain whitelist. N_5 shall not exceed MAX_SIZE_HOST_DOMAIN_WHITELIST, defined in section 5.3.5.	N_5 x 16

For Host within the TRE Host Domain:

- The registration of the Host Domain whitelist is persistent across power on/off cycles.
- The Network Controller Host shall append the Host Domain Identifier of the TRE Firmware Loader Agent Host when it is not present in the Host Domain whitelist.

For Hosts outside the TRE Host Domain, the Network Controller Host shall not append the Host Domain Identifier of the TRE Firmware Loader Agent Host when it is not present in the Host Domain whitelist.

This command shall be rejected by the Link Service Gate if the Host is not registered with the Link Service Gate.

When successful, the Network Controller Host shall respond with ANY_OK without parameter.

5.3.3 Responses

The Gate supports the responses defined in Table 5-5 and Table 5-15.

Table 5-15: Link Service Gate Commands/Responses

Error Code / Command	ANY_OK	ANY_E_CMD_PAR_UNKNOWN	ANY_E_NOK	ANY_E_REG_PAR_UNKNOWN	ANY_E_CMD_NOT_SUPPORTED	ANY_E_REG_ACCESS_DENIED
LINK_GET_HOST_LIST	•	•	•	•		•
LINK_REGISTER_HOST	•	•	•	•		•
LINK_DEREGISTER_HOST	•	•	•	•		•
LINK_SET_HOST_WHITELIST	•	•	•	•		•
LINK_SET_HOST_DOMAIN_WHITELIST	•	•	•	•		•
Unknown commands					•	

5.3.4 Events

5.3.4.1 Overview

The Gate supports the following event.

Table 5-16: Link Service Gate Event

Value	Command
'01'	EVT_LINK_END_OF_OPERATION

5.3.4.2 EVT_LINK_END_OF_OPERATION

This event shall be sent by a Link Application Gate to the Link Service Gate if the Host requires no more activity (see section 3.8) and is used by Hosts Arbitration. This event has no parameter.

5.3.5 Registry

Table 5-17 lists the entries in the Link Service Gate registry.

Table 5-17: Link Service Gate Registry

TYPE	Identifier	Parameter	Access Right	Description	Length	Default
MANDATORY	'01'	MAX_SIZE_ HOST_ WHITELIST	RO	Maximum number of Host Identifiers acceptable in the LINK_SET_HOST_WHITELIST command parameter in Table 5-13. Shall be greater than or equal to 16 entries.	1	16
	'02'	MAX_SIZE_ HOST_DOMAIN_ WHITELIST	RO	Maximum number of Host Domain Identifiers acceptable in the LINK_SET_HOST_DOMAIN _WHITELIST command parameter in Table 5-13. Shall be greater than or equal to 16 entries.	1	16
	'03'	SUBSCRIBE_TO_ HOST_ REGISTRATION_ CHANGE	RW	Subscription status to the following events from the Network Controller Host (see NOTE): EVT_LINK_HOST_REGISTERED EVT_LINK_HOST_DEREGISTERED '00' – Subscription to Host registration notification is inactive. '01' – Subscription to Host registration notification is active.	1	'00'
	'04'	HOST_LIST_ REVISION	RO	A monotonic increasing counter in the range of 0 to $2^{32} – 1$ that increments whenever there is a change in the list of registered Hosts. The value is an unsigned integer.	4	'0000 0000'
	'05'	VNP_MTU	RO	Smallest MTU value of all link layers going to the Router. The value should be equal to or greater than 270 bytes.	2	270

NOTE: Host registration of certain hosts is persistent and cannot be observed with the SUBSCRIBE_TO _HOST_REGISTRATION_CHANGE mechanism (i.e. Host is not available at the time of the deregistration / registration of the other Host). In order to get the complete list of all registered Hosts, the command LINK_GET_HOST_LIST should be used.

VNP_MTU contains the smallest value in bytes of the MTU of the Link Layers between the Router and all Host Domains.

A Host shall be able to send a VNP Packet to a Host in another Host Domain without fragmentation, if the size of the VNP Packet is less than or equal to the value provided in this registry. The Router shall be able to forward to the Host in a Host Domain any VNP Packet with a size equal to or smaller than the value provided in this registry, without any further fragmentation.

5.4 Link Application Gate

5.4.1 Overview

The Link Application Gate, in every Host except the Network Controller Host, handles events related to the changes in the Link state.

5.4.2 Commands

The Link Application Gate has no dedicated command.

5.4.3 Responses

The Link Application Gate has no dedicated response.

5.4.4 Events

5.4.4.1 Overview

The Link Application Gate supports the event listed in Table 5-18.

Table 5-18: Link Application Gate Events

Value	Event
'10'	EVT_LINK_HOST_ACCESS_DENIED
'11'	EVT_LINK_HOST_NOT_REACHABLE
'12'	EVT_LINK_HOST_REGISTERED
'13'	EVT_LINK_HOST_DEREGISTERED

5.4.4.2 EVT_LINK_HOST_ACCESS_DENIED

With the event EVT_LINK_HOST_ACCESS_DENIED, the Link Service Gate notifies the Link Application Gate in Host A that the Router cannot route a VNP Packet from Host A to Host B because the Host Identifier of Host A is not in the whitelist of Host B.

The event parameter contains the Host Alias of Host B:

Table 5-19: EVT_LINK_HOST_ACCESS_DENIED Event Parameter

Description	Length
Packet header in the VNP Packet sent to Host B (see section 4.3.1)	4
Pipe Identifier in the VNP Packet sent to Host B (see section 3.10)	1

5.4.4.3 EVT_LINK_HOST_NOT_REACHABLE

With the event EVT_LINK_HOST_NOT_REACHABLE, the Link Service Gate notifies the Link Application Gate in Host A that the Router cannot route a VNP Packet from Host A to Host B because Host B is not registered in the Link Service Gate or not reachable.

The event has the following parameter.

Table 5-20: EVT_LINK_HOST_NOT_REACHABLE Event Parameter

Description	Length
Packet header in the VNP Packet sent to Host B (see section 4.3.1)	5

Host B manages this notification according to its context; this management is implementation specific and out of scope of this specification.

5.4.4.4 EVT_LINK_HOST_REGISTERED

With the event EVT_LINK_HOST_REGISTERED, the Link Service Gate notifies the Link Application Gate in Host A about the registration of an array of Hosts. This event shall be broadcasted to all registered Hosts that have requested to be notified with the SUBSCRIBE_TO_HOST_REGISTRATION_CHANGE mechanism (see section 5.3.5), using the Link Service Gate registry.

The event parameter is an array of N_6 Host Aliases.

Table 5-21: EVT_LINK_HOST_REGISTERED Event Parameter

Description	Length
Array of N_6 Host Aliases of the registered Hosts	N_6

5.4.4.5 EVT_LINK_HOST_DEREGISTERED

With the event EVT_LINK_HOST_DEREGISTERED, the Link Service Gate notifies the Link Application Gate in Host A about the deregistration of an array of Hosts. This event shall be broadcasted to all registered Hosts that have requested to be notified with the SUBSCRIBE_TO_HOST_REGISTRATION_CHANGE mechanism (see section 5.3.5), using the Link Service Gate registry.

The event parameter is an array of N_7 Host Identifiers.

Table 5-22: EVT_LINK_HOST_DEREGISTERED Event Parameter

Description	Length
Array of N_7 Host Identifiers of the deregistered Hosts	N_7 x 16

5.4.5 Registry

The Link Application Gate has no registry entry.

5.5 Administration Service Gate

5.5.1 Overview

The Administration Service Gate, in every Host, provides access to services that manage the Pipes in the network and manages the Session Layer at the Pipes' level. The Administration Service Gate shall ensure the uniqueness of the Pipe Identifiers assigned to its created Pipe Sessions.

5.5.2 Commands

The Administration Service Gate has no command.

5.5.3 Events

5.5.3.1 Overview

The Administration Service Gate supports the event listed in Table 5-23.

Table 5-23: Administration Service Gate Events

Value	Event
'10'	EVT_ADM_BIND
'11'	EVT_ADM_UNBIND
'12'	EVT_ADM_CREDIT
'13'	EVT_ADM_RECEIVED
'14'	EVT_ADM_GATE_LIST_UPDATED
'15'	EVT_ADM_HOST_PING
'16'	EVT_ADM_ALL_PIPE_SESSIONS_CLOSED
'17'	EVT_ADM_GATE_NOT_REACHABLE

5.5.3.2 EVT_ADM_BIND

The Administration Service Gates are used in a procedure to bind two Gates with a Dynamic Pipe. The identifiers of the Gates and the Pipe are provided in the parameters of this command.

With the event EVT_ADM_BIND, Host A may request Host B to create Pipe Sessions according to an array of Gate Binding Parameters.

Host A initiates the Pipe Session creation procedure by sending the EVT_ADM_BIND event.

Host B shall answer by sending the EVT_ADM_BIND event when the event has been received.

The event parameters are as follows:

Table 5-24: EVT_ADM_BIND Event Parameter

Description	Length
Array of N_8 Gate Binding Parameters as defined in Table 5-25	N_8 x 17

Table 5-25 defines the Gate Binding Parameter.

Table 5-25: Gate Binding Parameter

Description	Length
Gate Identifier G_{ID} as defined in section 3.3	16
Pipe Identifier P_{ID} as defined in section 3.10 is bound with above Gate Identifier	1

- Host A shall not bind the same Pipe Identifier to more than one Gate of another Host.
- A bound Pipe shall be unbound before being bound again to a Gate.
- Host A may unbind the Pipe by using the EVT_ADM_UNBIND event.

The size (N_8) of the Gate Binding Parameter array in the EVT_ADM_BIND event parameter exchanged between both Hosts may be different.[3] This scenario is illustrated in Figure 6-7.

EVT_ADM_BIND event with an empty parameter is allowed and shall be ignored by the receiving Host.

The bound Pipe Identifier P_{ID} shall be in the range of the Dynamic Pipe Identifier defined in section 3.10.

Host A alias within the VNP Packet header allows Host B to filter the request for binding operations from Host A.

If Host A attempts to bind an already bound Pipe Identifier with the same Gate Identifier, it has no effect. Host B, recipient of the EVT_ADM_BIND event, shall answer with the bound Pipe Identifier used for the created Pipe Session without reinitializing its Transport Layer Credit-based Data Flow Control accumulator, if required by the Gate, as stated in section 5.5.3.4.

Host B may reject the Pipe Session creation by answering Host A with a Gate Binding Parameter using the Pipe Identifier '7F'.

If Host A and Host B are in the process of setting up more than one Pipe Session for the same Gate Identifier, as defined in section 5.5.3.2, then the following rules shall apply:

- The EVT_ADM_BIND events: The first EVT_ADM_BIND event sent by Host A is associated with the first EVT_ADM_BIND event sent by Host B, (and so on) as illustrated in Figure 6-8.
- The Gate Binding Parameters order: The first Gate Identifier in the Gate Binding Parameters of the EVT_ADM_BIND event sent by Host A matches the first Gate Identifier in the Gate Binding Parameters EVT_ADM_BIND event sent by Host B (and so on) as illustrated in Figure 6-9.

[3] A Host may provide more services than another Host may support.

5.5.3.3 EVT_ADM_UNBIND

With the event EVT_ADM_UNBIND, Host A informs Host B that the Pipe Sessions assigned to an array of its N_9 bound Pipe Identifiers are deleted. All Pipe Identifiers in the EVT_ADM_UNBIND event parameter are unbound and may be rebound by Host A to other Gates.

On reception of the EVT_ADM_UNBIND event, Host B shall implicitly unbind the corresponding Pipe from Host B to Host A for each Pipe Identifier contained in the Event Parameter array, as illustrated in section 6.2.2.

The event parameter is as follows:

Table 5-26: EVT_ADM_UNBIND Event Parameter

Description	Length
Array of N_9 bound Pipe Identifiers (P_{ID})	N_9

Each Pipe Identifier P_{ID} shall be in the range of the Dynamic Pipe Identifier defined in section 3.10.

Sending an EVT_ADM_UNBIND event with a Pipe Identifier as parameter which is not involved in a Pipe Session has no effect.

Note: This includes the scenario where the same Pipe Identifier is inserted more than once as a parameter or the event is sent twice with the same Pipe Identifier.

If the Transport Layer Data Acknowledgement is required by the Gate to which the Pipe is bound, then all data sent by Host B which are not yet acknowledged by an EVT_ADM_RECEIVED event related to this Pipe might be discarded.

5.5.3.4 EVT_ADM_CREDIT

With the event EVT_ADM_CREDIT, Host A informs Host B about a credit for a $PIPE_{xx}$ receiving VNP Packet(s) from Host B. The EVT_ADM_CREDIT event parameter contains an array of Credit Parameters as defined in Table 5-28.

There is an accumulator per Pipe for each Dynamic Pipe belonging to a Pipe Session created between Gates which requires the Transport Layer Credit-based Data Flow Control mechanism.

Host B shall accumulate all Credits related to $PIPE_{xx}$ to Host A in a Credit accumulator dedicated to a Pipe Session using $PIPE_{xx}$.

Host B shall subtract from this Credit accumulator the size of the Message Fragments sent to Host A over $PIPE_{xx}$.

Host B shall not send, on a $PIPE_{xx}$, data greater than the Credit accumulator related to this $PIPE_{xx}$.

The Credit accumulator at the creation of the Pipe Session is 64.

The Credit accumulator is 0 when the Pipe Session is deleted.

With the event EVT_ADM_CREDIT, a Host A may update the Credit for multiple Pipes.

The event parameter is an array of N_{10} Credit Parameters:

Table 5-27: EVT_ADM_CREDIT Event Parameter

Description	Length
Array of N_{10} Credit Parameters	N_{10} x 3

Table 5-28: Credit Parameters

Description	Length
Pipe Identifier	1
Credit Value (Additional bytes that Host B may send to Host A) (unsigned integer)	2

The Transport Layer Credit-based Data Flow Control feature is optional at the Gate level. It is explicitly known as a definition of the Gate whether the Gate requires the Transport Layer Credit-based Data Flow Control mechanism.

5.5.3.5 EVT_ADM_RECEIVED

With the event EVT_ADM_RECEIVED, Host A reports to Host B the accumulators summing the length of the Message Fragments received by Host A. There is an accumulator per Pipe for each Dynamic Pipe belonging to a Pipe Session created between Gates which require the Transport Layer Data Acknowledgment mechanism.

Each accumulator assigned to a Pipe is reset when the Pipe Session is created or after sending the EVT_ADM_RECEIVED event.

The EVT_ADM_RECEIVED event contains an array of structures as defined in Table 5-30.

The event parameter is an array of the following structure:

Table 5-29: EVT_ADM_RECEIVED Event Parameter

Description	Length
Array of N_{11} Received Bytes structures	N_{11} x 3

Table 5-30: Received Byte Structure

Description	Length
Pipe Identifier	1
Number of received bytes by Host A (unsigned integer)	2

The Transport Layer Data Acknowledgement by using the EVT_ADM_RECEIVED event is optional at the Gate level. It is explicitly known as a definition of the Gate whether the Gate requires the Transport Layer Data Acknowledgement.

5.5.3.6 EVT_ADM_GATE_LIST_UPDATED

With the event EVT_ADM_GATE_LIST_UPDATED, Host A informs Host B about the update of the GATE_LIST registry entry in its Identity Service Gate.

Note: A Host may inform other Hosts in the VNP network to which its administration gate maintains a connection about an update of the GATE_LIST registry entry in its Identity Service Gate.

The event has no parameters.

5.5.3.7 EVT_ADM_HOST_PING

With the event EVT_ADM_HOST_PING, Host A may check if Host B is active. Host B should either return the same event, or send any other message available linked to another Pipe Session.

The event has no parameter.

5.5.3.8 EVT_ADM_ALL_PIPE_SESSIONS_CLOSED

With the event EVT_ADM_ALL_PIPE_SESSIONS_CLOSED, Host A notifies Host B that the deletion of all Pipe Sessions with Host B occurs as defined in section 3.10.3.

The event has no parameter.

5.5.3.9 EVT_ADM_GATE_NOT_REACHABLE

With the event EVT_ADM_GATE_NOT_REACHABLE, the Administration Gate of Host A shall notify the Administration Gate of Host B that there are no Pipes bound with the identifiers indicated in the event parameter to any Gate of Host A. This notification occurs when Host A is unable to dispatch one or more VNP Packets to a Gate because the Pipe Identifier in the VNP Packet is not bound to any of its Gates.

The event parameter is as follows:

Table 5-31: EVT_ADM_GATE_NOT_REACHABLE Event Parameter

Description	Length
Array of N_{12} Pipe Identifiers (P_{ID})	N_{12}

Host B manages this notification according to its context; this management is implementation specific and out of scope of this specification.

5.5.4 Registry

Table 5-32 lists the entries in the Administration Service Gate registry.

Table 5-32: Administration Service Gate Registry

Identifier	Parameter	Access Right	Comment	Length	Default
'01'	VNP_VERSION	RO	Version of VNP implemented by the Host: Major and Minor version	2	'0100'
'02'	MIN_VNP_VERSION	RO	Minimum version of VNP supported by this Host	2	'0100'

Any Host implementing VNP, according to the present document, shall set the VNP_VERSION parameter to '0100'. The VNP_VERSION parameter will be incremented for each release of this VNP specification. MIN_VNP_VERSION shall be the lowest version of VNP supported by the Host.

5.6 Administration Application Gate

5.6.1 Overview

The Administration Application Gate supports the same commands, responses, events, and registry entries as the Administration Service Gate as defined in section 5.5.

The Administration Application Gate and the Administration Service Gate are equivalent, and each Gate plays different roles when either consuming or providing a Service (see section 3.11).

5.7 Identity Service Gate

5.7.1 Overview

The Identity Service Gate provides the information listed in its Registry. This Gate requires the implementation of the Transport Layer Data Acknowledgement and Transport Layer Credit-based Data Flow Control.

5.7.2 Commands

This Service Gate has no dedicated command.

5.7.3 Responses

This Gate has no dedicated response.

5.7.4 Events

This Gate has no dedicated event.

5.7.5 Registry

The Identity Service Gate provides the list of available Services, software and hardware information about the Host.

This Gate shall be provided by all Hosts.

Table 5-33 lists the entries in the registry.

Table 5-33: Identity Service Gate Registry

TYPE	Identifier	Parameter	Access Right	Description	Length	Default
MANDATORY	'01'	VER_CURR	RO	Version of the firmware defined by the vendor for the Host. For Host of the TRE Host Domain, VERSION_SW is Current version (VER_CURR) of the firmware loaded in the execution memory of the TRE as defined in [OFL].	2	'0000'
	'02'	VENDOR_NAME	RO	Vendor name UTF8 coding of the Host. The maximum value for N_{13} shall be 20.	N_{13}	-
	'03'	MODEL_ID	RO	Model identifier assigned by the vendor.	1	'00'
	'04'	GATE_LIST	RO	Array of N_{14} Gate Identifiers of Gates providing a Service[4]	N_{14} x 16	-
	'05'	UC_IDENTIFIER	RO	Use Case Identifier (defined in section 3.3)	16	-
OPTIONAL	'06'	NB_PIPE_ SESSION_ GATE_LIST	RO	Array of Number of Pipe Sessions per Gate Identifier listed in the GATE_LIST parameter. Each number is greater than 0 and less than 255. Ordered as are the Gate Identifiers in the GATE_LIST registry.	N_{14}	-
RESERVED	'80'	-	-	Reserved for ETSI	-	-
	'81'	-	-	Reserved for ETSI	-	-

Note: The GATE_LIST registry entry may contain multiple same Gate Identifiers if the Host offers multiple Services to the same Host.

[4] The content of the GATE_LIST registry entry may be dynamic and evolves according to the Host requiring the information or any other context parameter (use case dependent) which may lead to the disclosure of different lists of Services.

5.8 Identity Application Gate

5.8.1 Overview

The Identity Application Gate consumes the information provided by the Identity Service Gate. This Gate requires the implementation of the Transport Layer Data Acknowledgement and the Transport Layer Credit-based Data Flow Control.

5.8.2 Commands

This Gate has no dedicated command.

5.8.3 Responses

This Gate has no dedicated response.

5.8.4 Events

This Gate has no dedicated event.

5.8.5 Registry

The Gate has no registry entry.

5.9 Loopback Service Gate

5.9.1 Overview

The Loopback Service Gate provides access to Services for testing each Host.

This Gate returns any data conveyed without any change.

This Gate requires the implementation of the Transport Layer Data Acknowledgement and the Transport Layer Credit-based Data Flow Control.

5.9.2 Commands

This Gate has no dedicated command.

5.9.3 Responses

This Gate has no dedicated response.

5.9.4 Events

The Gate supports the event listed in Table 5-34.

Table 5-34: Loopback Service Gate Events

Value	Event
'10'	EVT_LOOP_POST_DATA

5.9.4.1 EVT_LOOP_POST_DATA

With the event EVT_LOOP_POST_DATA, a Host copies the received data into an EVT_LOOP_ECHO_DATA event. The length of the EVT_LOOP_POST_DATA event may have any value. The Loopback Service should not have to wait until completely receiving this event in order to start sending the EVT_LOOP_ECHO_DATA event.

The event parameter is as follows:

Table 5-35: EVT_LOOP_POST_DATA Event Parameter

Description	Length
Data to echo	N_{15}

5.9.5 Registry

The Gate has no registry entry.

5.10 Loopback Application Gate

5.10.1 Overview

The Loopback Application Gate consumes the EVT_LOOP_ECHO_DATA event. This Gate requires the implementation of the Transport Layer Data Acknowledgement and the Transport Layer Credit-based Data Flow Control.

5.10.2 Commands

The Loopback Application Gate has no dedicated command.

5.10.3 Responses

The Loopback Application Gate has no dedicated response.

5.10.4 Events

5.10.4.1 Overview

The Loopback Application Gate supports the event listed in Table 5-36.

Table 5-36: Loopback Application Gate Events

Value	Event
'10'	EVT_LOOP_ECHO_DATA

5.10.4.2 EVT_LOOP_ECHO_DATA

When a Host receives the EVT_LOOP_POST_DATA, it shall echo the data back to the originating Host.

It copies the data received with the EVT_LOOP_POST_DATA into the EVT_LOOP_ECHO_DATA event parameter and sends it back.

The event parameter is as follows:

Table 5-37: EVT_LOOP_ECHO_DATA Event Parameter

Description	Length
Copy of received data	N_{16}

5.10.5 Registry

The Loopback Application Gate has no registry entry.

6 VNP Procedures

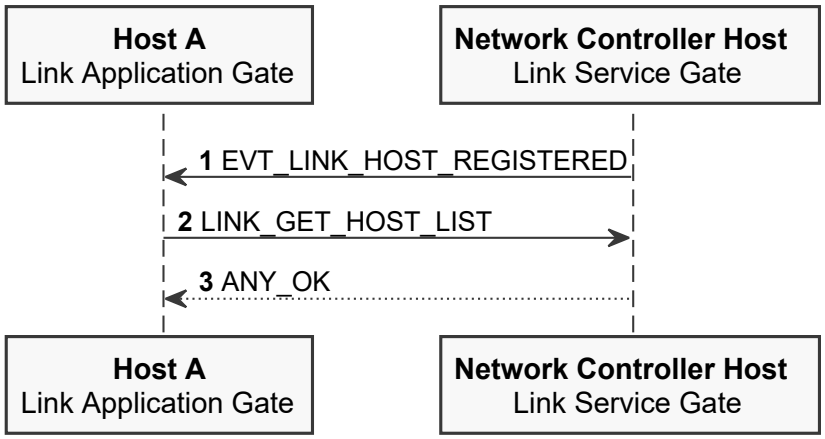
Note: For clarity, the event EVT_ADM_RECEIVED is not shown in all procedures, although the operations related to this event are mandatory for the Gates of the VNP Core Service using Dynamic Pipes. For the same reason, the event EVT_ADM_CREDIT is not shown in all procedures, although the operations related to this event are requested by a Gate.

6.1 Host Management

6.1.1 Host Discovery

Figure 6-1 shows the host discovery.

Figure 6-1: Host Discovery



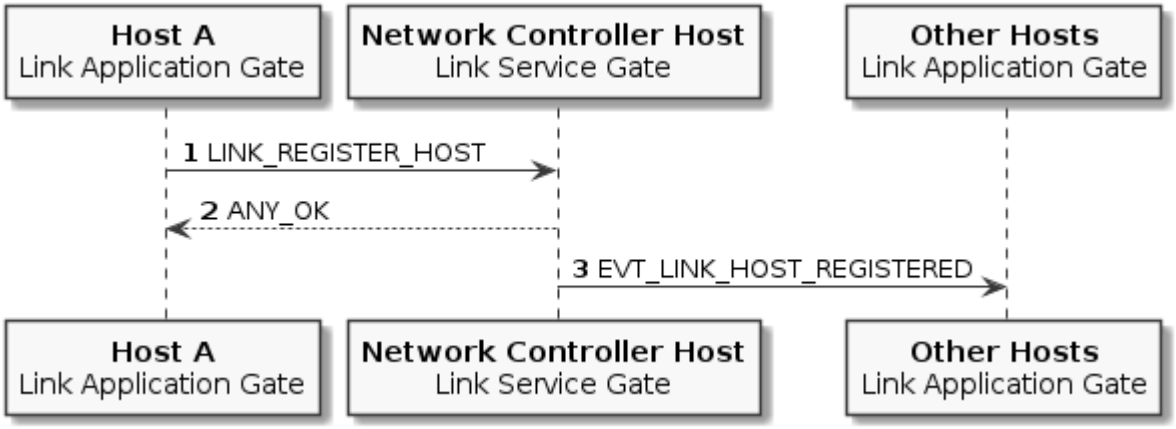
The sequence flow is as follows:

1. The Network Controller Host notifies Host A that Host B has registered. This step is optional.
2. Host A requests from the Network Controller Host the list of the registered Hosts.
3. The Network Controller Host returns the list of the registered Hosts to Host A.

6.1.2 Host Registration

Figure 6-2 illustrates the registration of Host A with the Network Controller Host and the resulting notification to other Hosts already registered with the Network Controller Host.

Figure 6-2: Host Registration



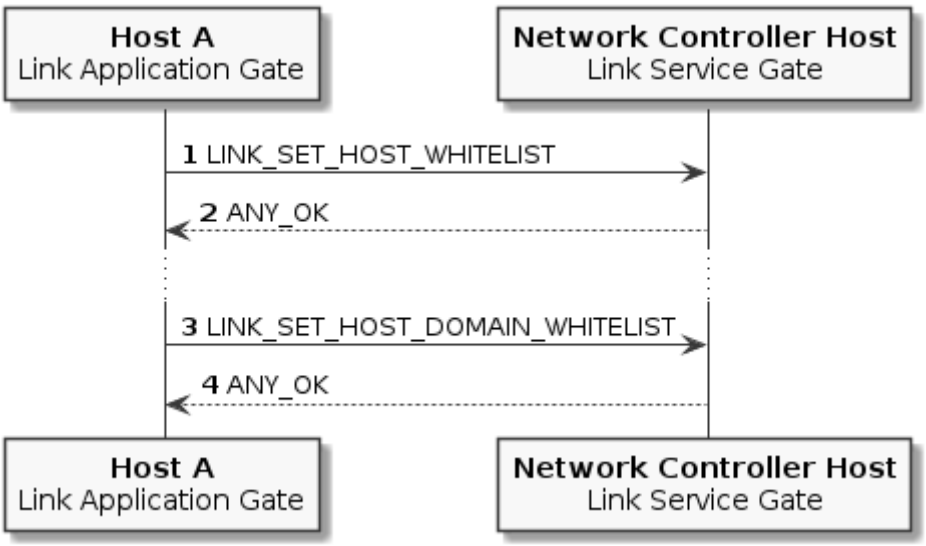
The sequence flow is the following for each registration procedure:

1. Host A registers itself with the Network Controller Host and provides its VPP Firmware Family Identifier.
2. The Network Controller Host returns ANY_OK. Host A can retrieve its Host Alias from the VNP Packet header.
3. The EVT_LINK_HOST_REGISTERED is broadcasted to the other Hosts, with the restrictions defined in section 5.4.4.4.

6.1.3 Host and Host Domain Whitelist Registration

Figure 6-3 illustrates the procedure allowing a Host and its Host Domain to register their whitelists.

Figure 6-3: Host and Host Domain Whitelist Registration



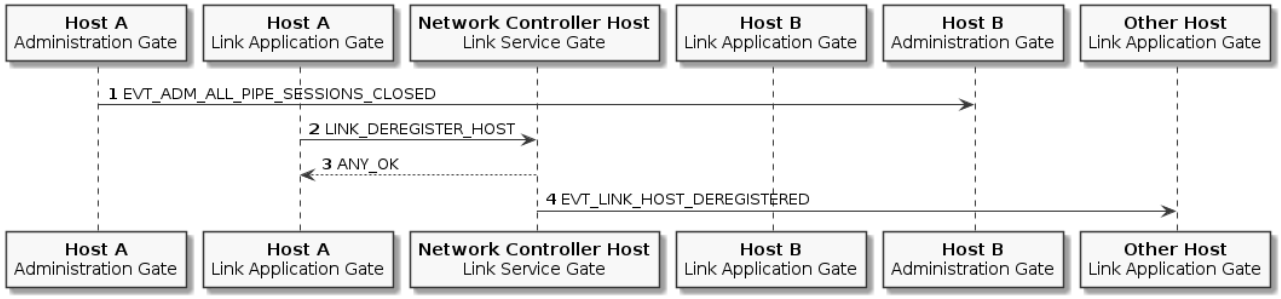
The sequence flow is the following for the whitelist registration procedure:

1. Host A requests the Network Controller Host to register its whitelist.
2. The Network Controller Host returns ANY_OK.
3. Host A requests the Network Controller Host to register its Host Domain whitelist.
4. The Network Controller Host returns ANY_OK.

6.1.4 Host Deregistration

Figure 6-4 illustrates the procedure performed for the deregistration of a Host.

Figure 6-4: Host Deregistration



The sequence flow is the following for a Host deregistration procedure:

1. Host A notifies Host B about the deletion of all Pipe Sessions between both Hosts (optional)
2. Host A requests the Network Controller Host to deregister its Host Alias.
3. The Network Controller Host returns ANY_OK.
4. The Network Controller Host broadcasts the EVT_LINK_HOST_DEREGISTERED event to all Hosts, with the restrictions defined in section 5.4.4.5.

6.2 Pipe Management

6.2.1 Pipe Binding

This section describes the following scenarios:

A: Simple Pipe Session creation between two gates.

B: Creation of five Pipe Sessions using one event exchange between two Hosts.

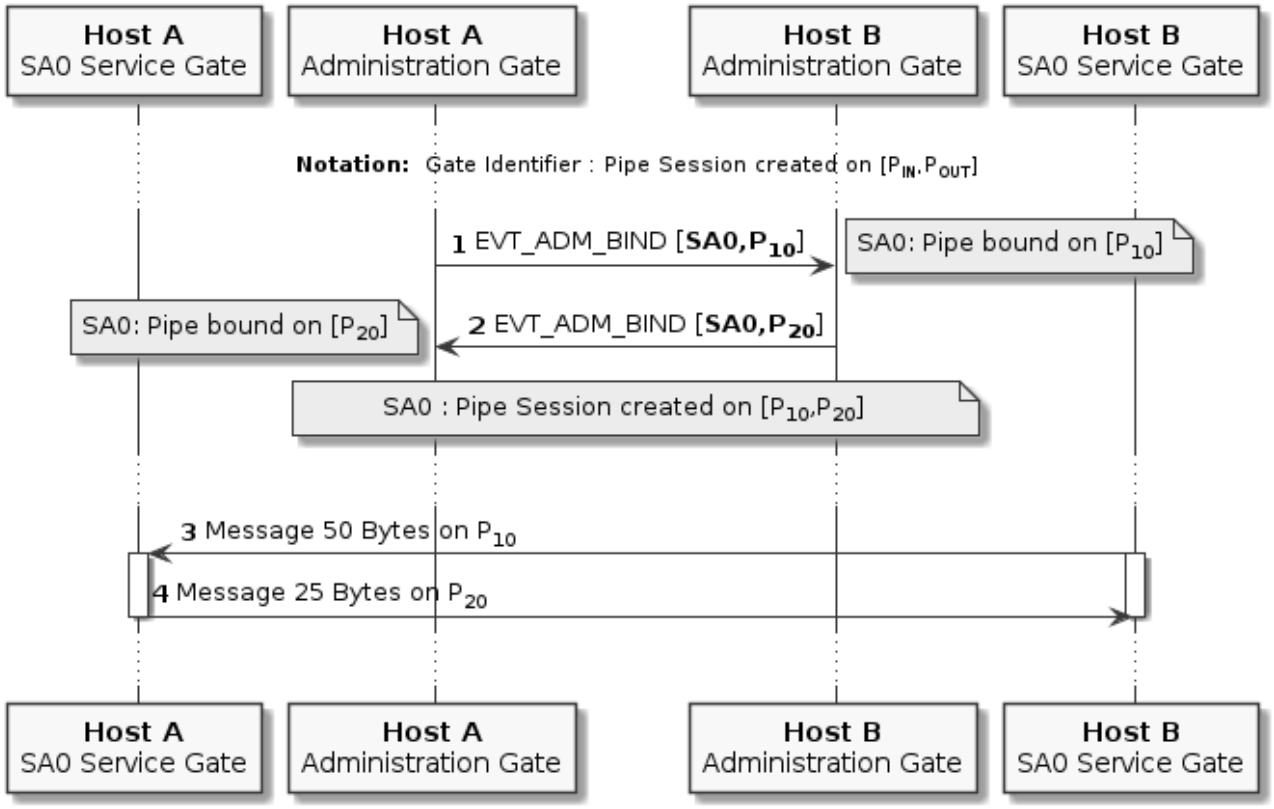
C: Creation of multiple Pipe Sessions between two Hosts using interleaved events.

D: Creation of multiple Pipe Sessions on the same gates using a pipe parameter in one event with deferred response event.

E: Creation of multiple Pipe Sessions on the same gates using multiple pipe parameters in one event.

Figure 6-5 illustrates scenario A where Host A requests Host B to create a Pipe Session between two Gates.

Figure 6-5: Pipe Binding Scenario A



The sequence flow is as follows:

1. Host A informs Host B that the Service Gate with identifier SA0, hosted by Host A, is bound on Pipe P_{10}.
2. Host B informs Host A that the Application Gate with identifier SA0, hosted by Host B, is bound on Pipe P_{20}. Pipe Session for SA0 is created:
   - SA0 Service Gate may receive VNP Packets on Pipe P_{10} and may send VNP Packets on Pipe P_{20}.

- SA0 Application Gate may receive VNP Packets on Pipe P_{20} and may send VNP Packets on Pipe P_{10}.

3. Host B sends VNP Packets to Host A on Pipe P_{10}.
4. Host A sends VNP Packets to Host B on Pipe P_{20}.

Steps 1 and 2 are mandatory. The binding procedure may occur several times after the connection of the Hosts according to the needs of its Application. A Host may send several EVT_ADM_BIND events without having to wait for the same event from the other Host.

Any of the following events:

- EVT_ADM_ALL_PIPE_SESSIONS_CLOSED or
- EVT_LINK_HOST_NOT_REACHABLE or
- EVT_LINK_HOST_ACCESS_DENIED

may occur between steps 1 and 2, and aborts the procedure.

Figure 6-6 illustrates the flow of scenario B.

Figure 6-6: Pipe Binding Scenario B

The sequence flow is as follows:

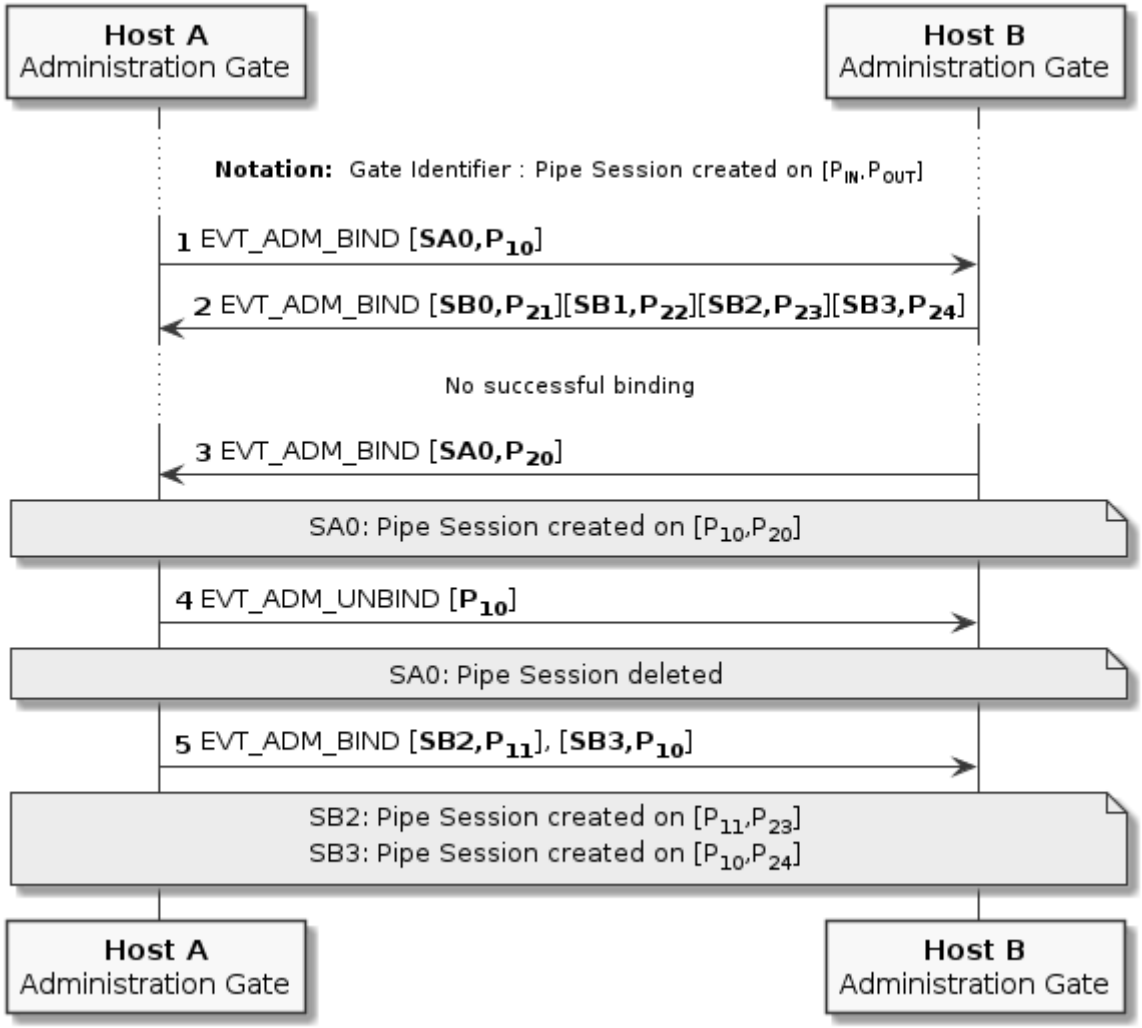
1. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
   - The Gate with Gate Identifier SB0 is bound with Pipe Identifier P_{11}.
   - The Gate with Gate Identifier SB1 is bound with Pipe Identifier P_{12}.

- The Gate with Gate Identifier SB2 is bound with Pipe Identifier P_{13}.
- The Gate with Gate Identifier SB3 is bound with Pipe Identifier P_{14}.

2. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}. The Pipe Session is created.
   - The Gate with Gate Identifier SB0 is bound with Pipe Identifier P_{21}. The Pipe Session is created.
   - The Gate with Gate Identifier SB1 is bound with Pipe Identifier P_{22}. The Pipe Session is created.
   - The Gate with Gate Identifier SB2 is bound with Pipe Identifier P_{23}. The Pipe Session is created.
   - The Gate with Gate Identifier SB3 is bound with Pipe Identifier P_{24}. The Pipe Session is created.

Figure 6-7 illustrates scenario C.

Figure 6-7: Pipe Binding Scenario C

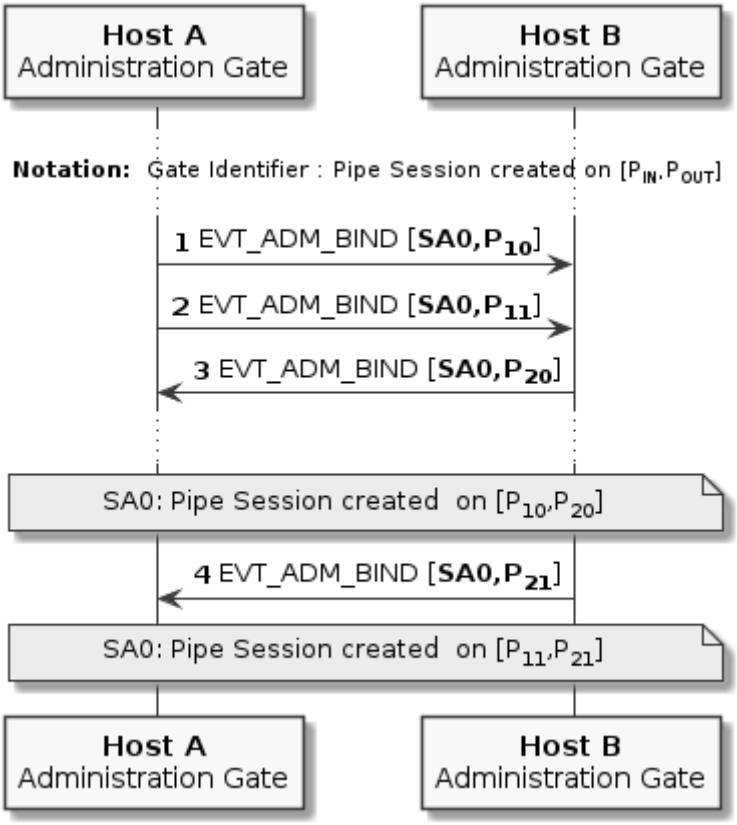


The sequence flow is as follows:

1. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
2. Host B informs Host A that:
   - The Gate with Gate Identifier SB0 is bound with Pipe Identifier P_{21}.
   - The Gate with Gate Identifier SB1 is bound with Pipe Identifier P_{22}.
   - The Gate with Gate Identifier SB2 is bound with Pipe Identifier P_{23}.
   - The Gate with Gate Identifier SB3 is bound with Pipe Identifier P_{24}.
3. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}.
   - The Pipe Session is created.
4. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is unbound with Pipe Identifier P_{10}.
5. Host A informs Host B that:
   - The Gate with Gate Identifier SB2 is bound with Pipe Identifier P_{11}.
   - The Gate with Gate Identifier SB3 is bound with Pipe Identifier P_{10}.

Figure 6-8 illustrates scenario D.

Figure 6-8: Pipe Binding Scenario D

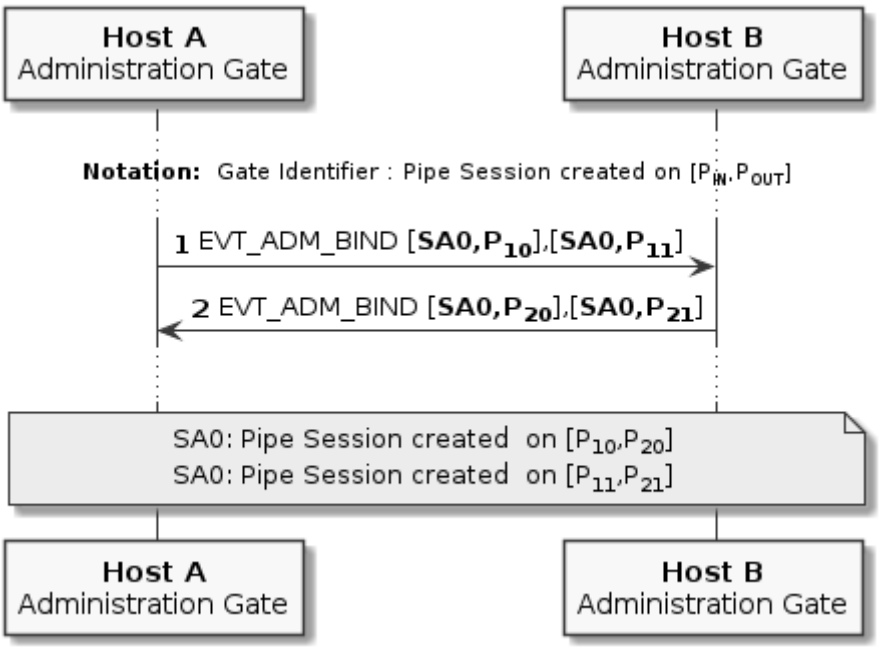


The sequence flow is as follows:

1. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
2. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{11}.
3. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}.
   - The Pipe Session is created on SA0 bound with Pipe Identifiers P_{10} and P_{20}.
4. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{21}.
   - The Pipe Session is created on SA0 bound with Pipe Identifiers P_{11} and P_{21}.

Figure 6-9 illustrates scenario E.

Figure 6-9: Pipe Binding Scenario E



The sequence flow is as follows:

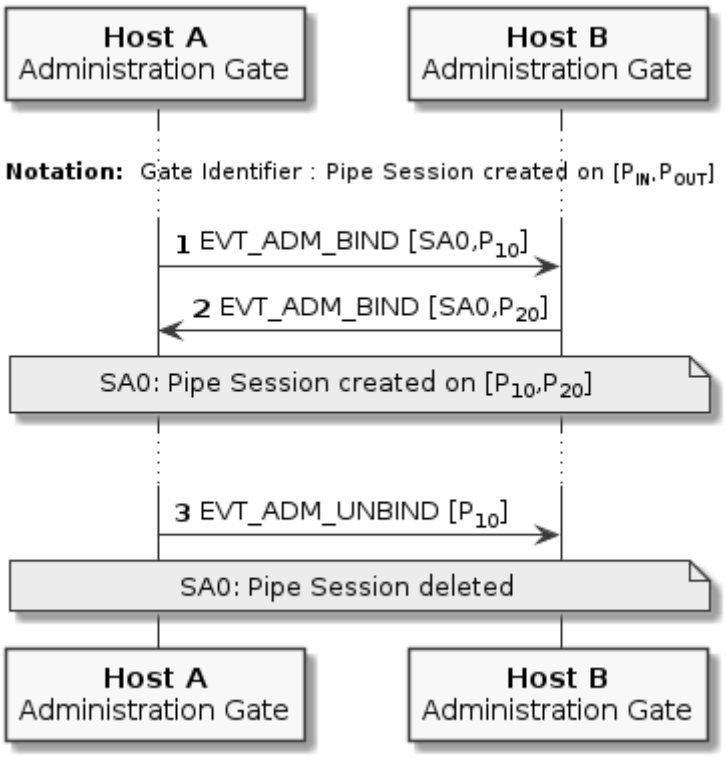
1. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{11}.
2. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}.
   - The Pipe Session is created on SA0 bound with Pipe Identifiers P_{10} and P_{20}.
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{21}.

- The Pipe Session is created on SA0 bound with Pipe Identifiers P_{11} and P_{21}.

6.2.2 Pipes Unbinding

Figure 6-10 illustrates how Host A requests the Pipe Session deletion of its Dynamic Pipe, $PIPE_{10}$, between one of its Gates and a Gate in Host B. All communications use $PIPE_{01}$.

Figure 6-10: Pipe Unbinding



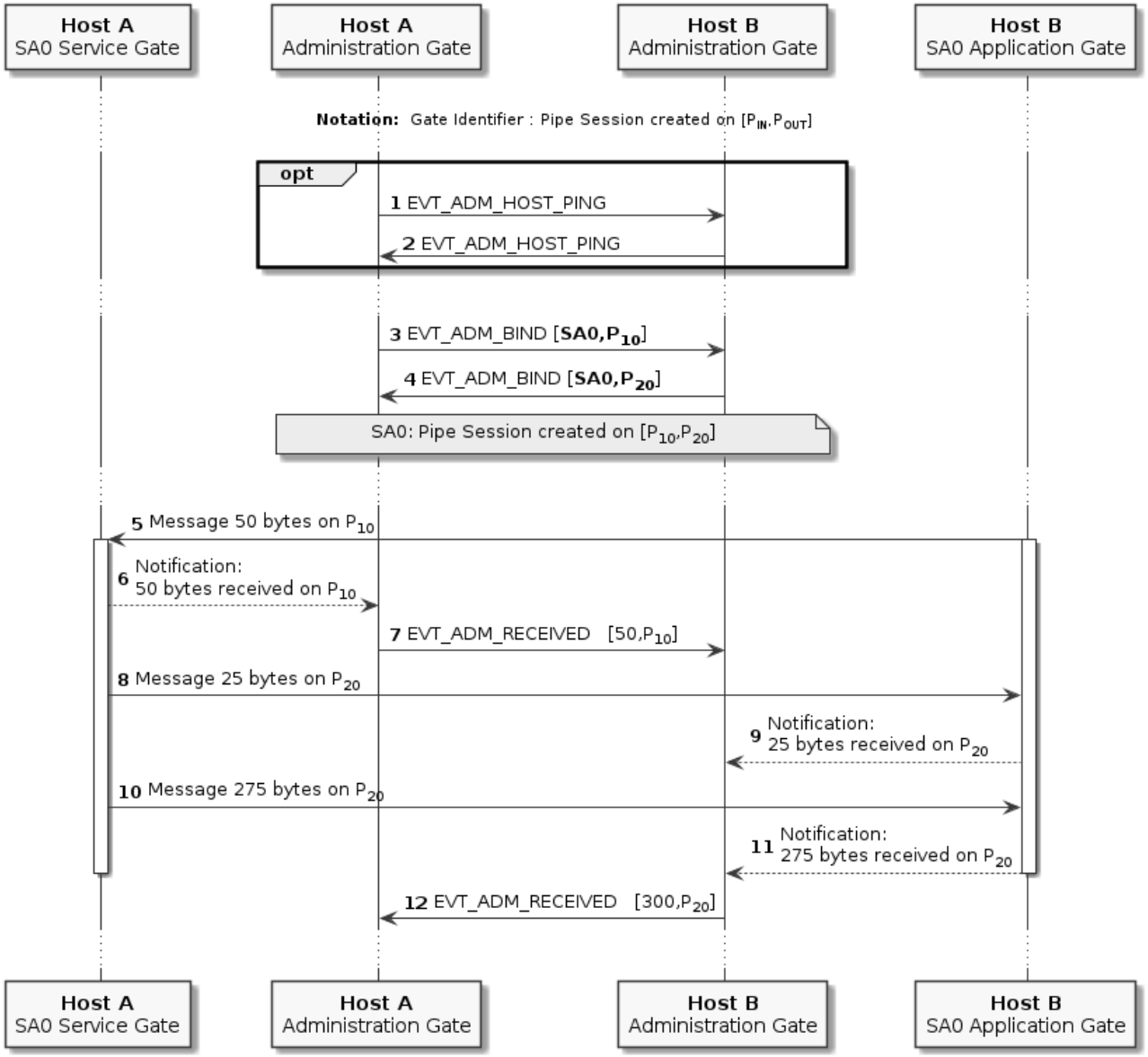
The sequence flow is as follows:

1. Host A informs Host B that the Service Gate with identifier SA0, hosted by Host A, is bound on the Pipe P_{10}.
2. Host B informs Host A that the Application Gate with identifier SA0, hosted by Host B, is bound on the Pipe P_{20}. Pipe Session for SA0 is created:
   - SA0 Service Gate may receive VNP Packets on Pipe P_{10} and may send VNP Packets on Pipe P_{20}.
   - SA0 Application Gate may receive VNP Packets on Pipe P_{20} and may send VNP Packets on Pipe P_{10}.
3. Host A notifies Host B that its Pipe Session on $PIPE_{10}$ is deleted for both Hosts. When a Pipe Session is deleted, even unilaterally by one of the two Hosts, then the dialogue between the Gates using the Pipes is globally closed as stated in section 3.10.2. Pipe P_{20} is implicitly unbound from Host A's perspective.

6.2.3 Transport Layer Data Acknowledgment in a Pipe

Figure 6-11 illustrates how the Transport Layer Data Acknowledgment is performed at the Pipe level as defined in section 5.5.3.5. The mechanism by which the Data Acknowledgment is conveyed between the SA0 Application/Service Gate and their respective Administration Gates is implementation dependent.

Figure 6-11: Transport Layer Data Acknowledgment in a Pipe



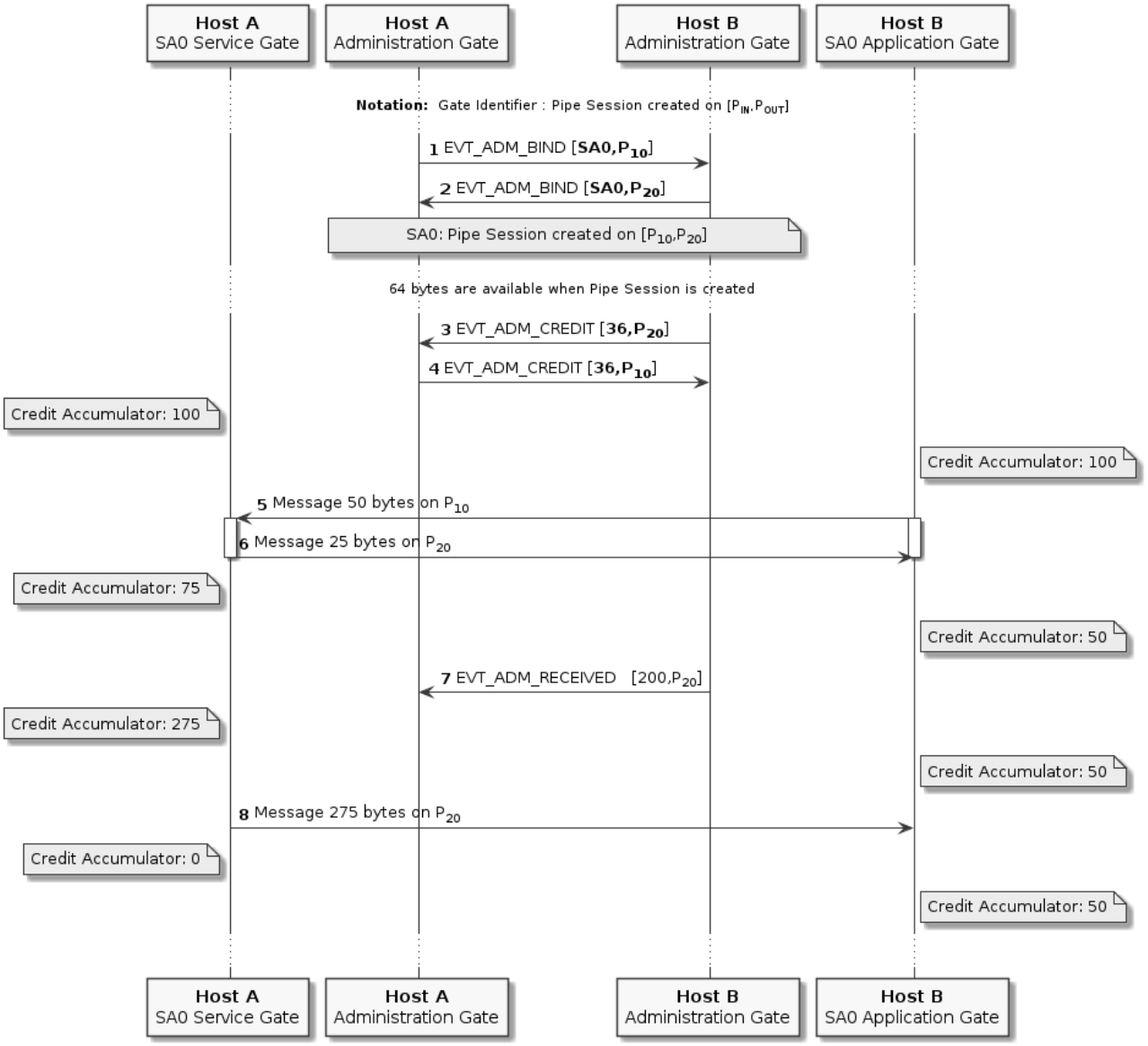
The sequence flow is as follows:

1. Host A pings Host B.
2. Host B pings Host A.
3. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
4. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}. The Pipe Session is created.
5. Host B sends 50 bytes of message fragment on the Pipe (P_{10}); then the Credit accumulator of this pipe is 50.
6. The SA0 Gate in Host A notifies the Host A Administration Gate that 50 bytes has been received.
7. The Host A Administration Gate accumulates the notification then notifies the Host B Administration Gate that 50 bytes has been received on P_{10}.
8. Host A sends 25 bytes of message fragment on the Pipe (P_{20}) to Host B.
9. The SA0 Gate in Host B notifies the Host B Administration Gate that 25 bytes has been received.
10. Host A sends 275 bytes of message fragment on the Pipe (P_{20}) to Host B.
11. The SA0 Gate in Host B notifies the Host B Administration Gate that 275 bytes has been received.
12. The Host B Administration Gate accumulates the notification then notifies the Host A Administration Gate that 300 bytes has been received on P_{20}.

6.2.4 Transport Layer Credit-based Data Flow Control in a Pipe

Figure 6-12 illustrates the procedure implementing the Transport Layer Credit-based Data Flow Control as defined in section 5.5.3.4. The mechanism used between the SA0 Application/Service Gate and their respective Administration Gate within their Hosts for getting/setting Credits is implementation dependent.

Figure 6-12: Data Flow Control in a Pipe



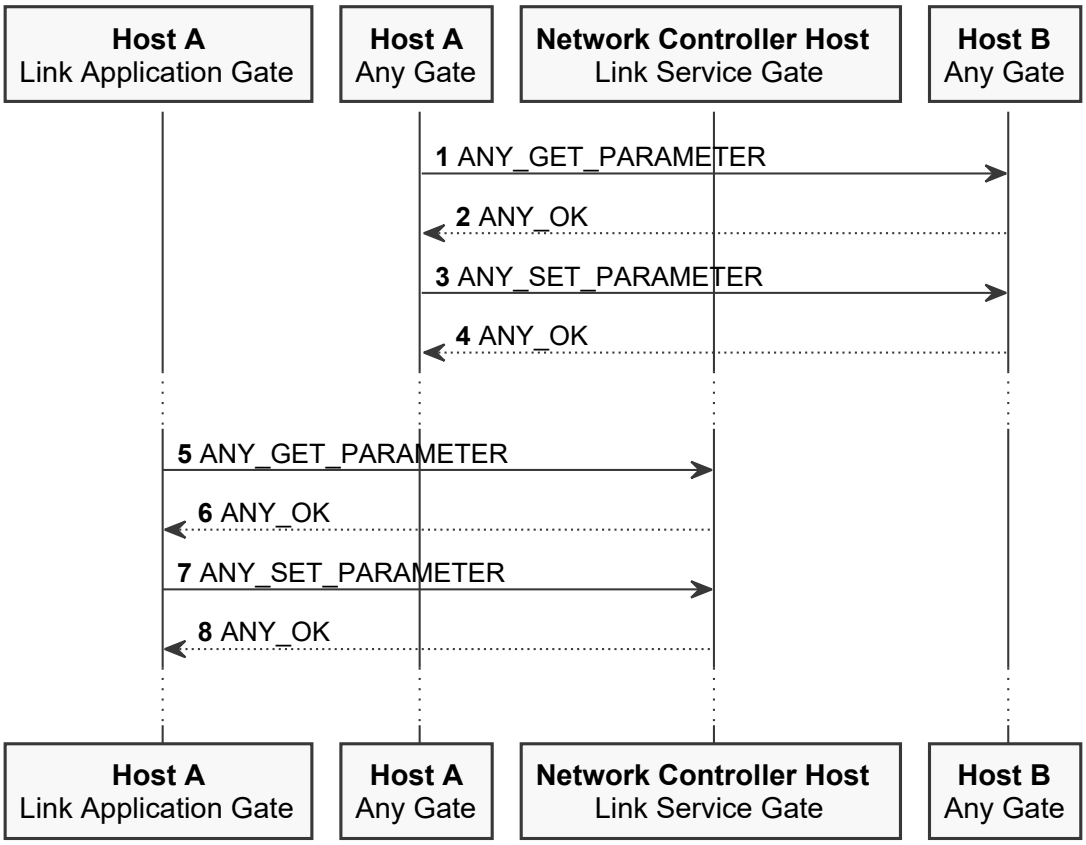
The sequence flow is as follows:

1. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
2. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}. The Pipe Session is created.
3. Host B notifies Host A about a Credit of 36 bytes for its Pipe (P_{20}). Now the Credit accumulator for this Pipe (P_{20}) is 100.
4. Host A notifies Host B about a Credit of 36 bytes for its Pipe (P_{20}). Now the Credit accumulator for this Pipe (P_{20}) is 100.
5. Host B sends 50 bytes of message fragment on the Pipe (P_{10}); then the Credit accumulator of this pipe is 50.
6. Host A sends 25 bytes of message fragment on the Pipe (P_{20}); then the Credit accumulator of this pipe is 75.
7. Host B notifies Host A about a Credit of 200 bytes for its Pipe (P_{20}). The Credit accumulator for this Pipe (P_{20}) is now 275.
8. Host A sends 275 bytes of message fragment on the Pipe (P_{20}) then the Credit accumulator of this pipe is 0. Host A shall not send additional message fragments as they will be lost.

6.3 Registry Access

Figure 6-13 shows how Host A can read/write parameters in the registry of Host B.

Figure 6-13: Registry Access



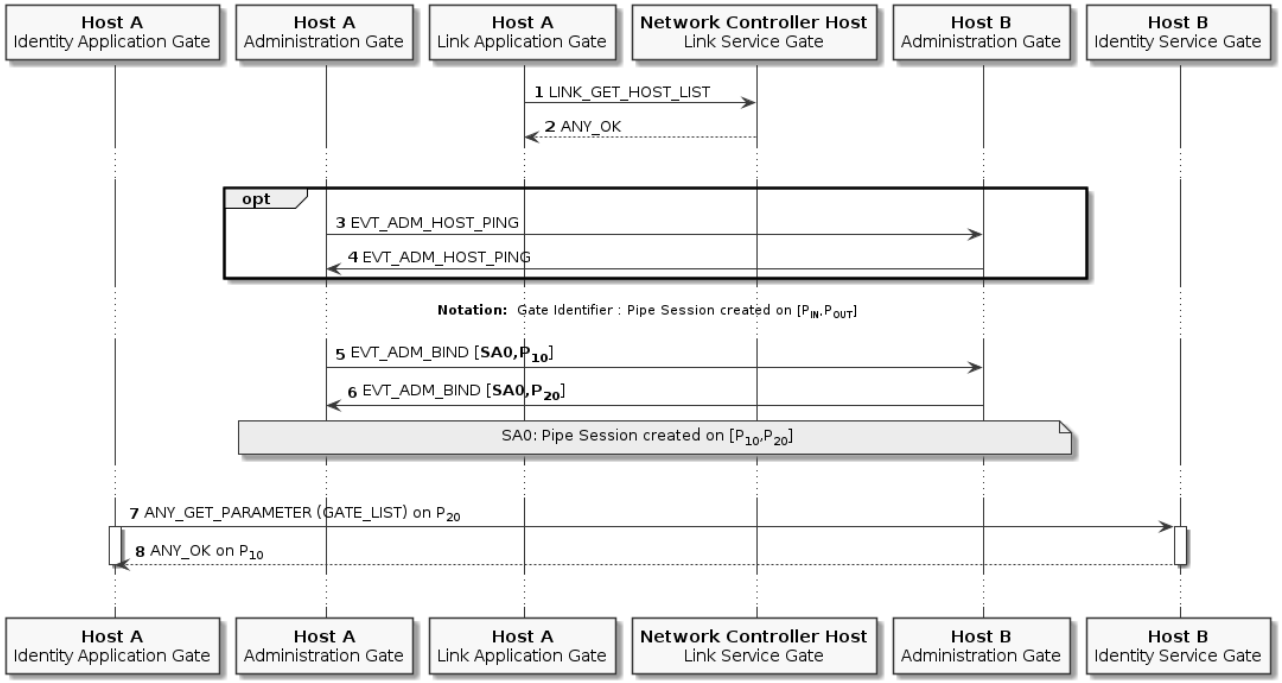
The sequence flow is as follows:

1. Host A requests a registry parameter from a Gate of Host B.
2. Host B replies with ANY_OK that includes the parameter value in its data.
3. Host A updates a registry parameter in a Gate of Host B.
4. Host B confirms that the parameter has been updated.
5. Host A requests a registry parameter from the Link Service Gate.
6. Link Service Gate replies with ANY_OK that includes the parameter value in its data.
7. Host A updates a registry parameter in a Link Service Gate.
8. Link Service Gate confirms that the parameter has been updated.

6.4 Hosts and Service Gates Discovery

Figure 6-14 shows Hosts and Service Gates discovery.

Figure 6-14: Hosts and Service Gates Discovery



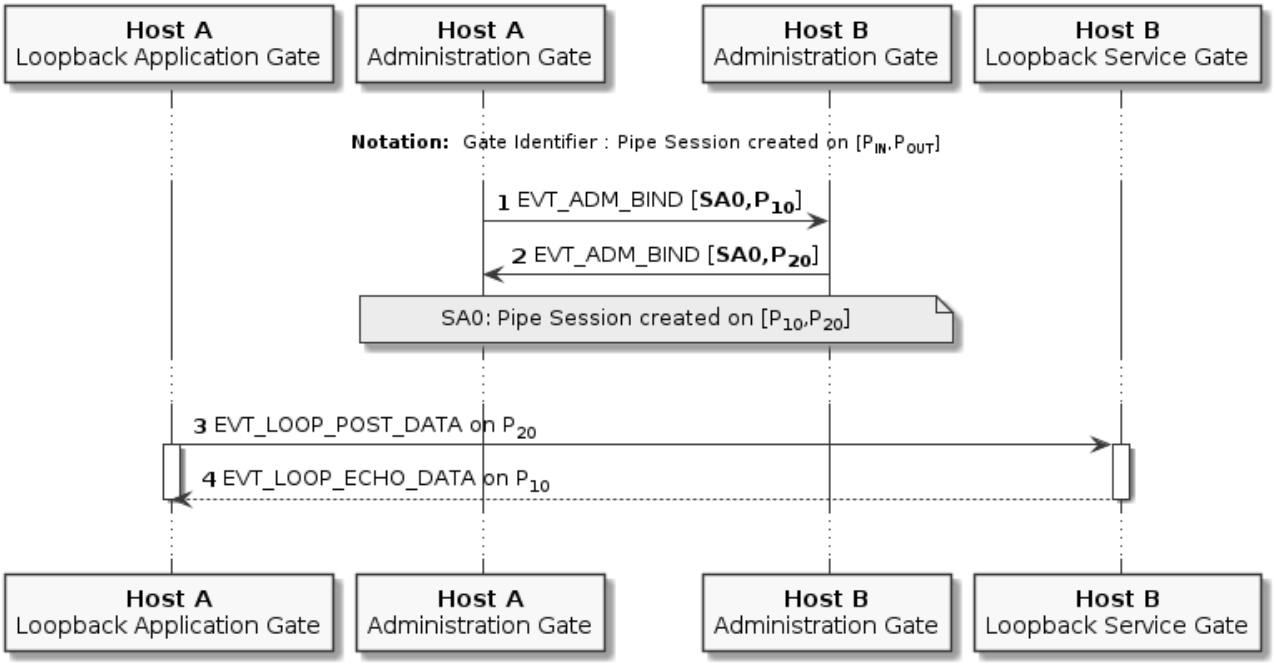
The sequence flow is as follows:

1. Host A requests the list of the registered Hosts.
2. The Network Controller Host returns the list of the registered Hosts.
3. Host A pings Host B.
4. Host B pings Host A.
5. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
6. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}. The Pipe Session is created.
7. The Host A Application Gate requests the GATE_LIST registry entry in the Host B Identity Service Gate.
8. The Host B Identity Service Gate returns the list of the Gate Identifiers providing a service.

6.5 Loopback Testing

Figure 6-15 shows how Host A can verify the connectivity to Host B.

Figure 6-15: Loopback



The sequence flow is as follows:

1. Host A informs Host B that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{10}.
2. Host B informs Host A that:
   - The Gate with Gate Identifier SA0 is bound with Pipe Identifier P_{20}. The Pipe Session is created.
3. Host A sends event EVT_LOOP_POST_DATA with data to echo on P_{20}.
4. Host B sends back event EVT_LOOP_ECHO_DATA with the same data as received in the previous step on P_{10}.

Annex A Informative: Example of VNP Flow

A.1 Overview

The next sections illustrate the use of VNP within a particular context of an OEM telecommunication device implementing a UICC. This context assumes a TRE Firmware Loader as defined in [OFL] and an OFL Agent as defined in [VOFL].

A.2 Implementation Assumptions

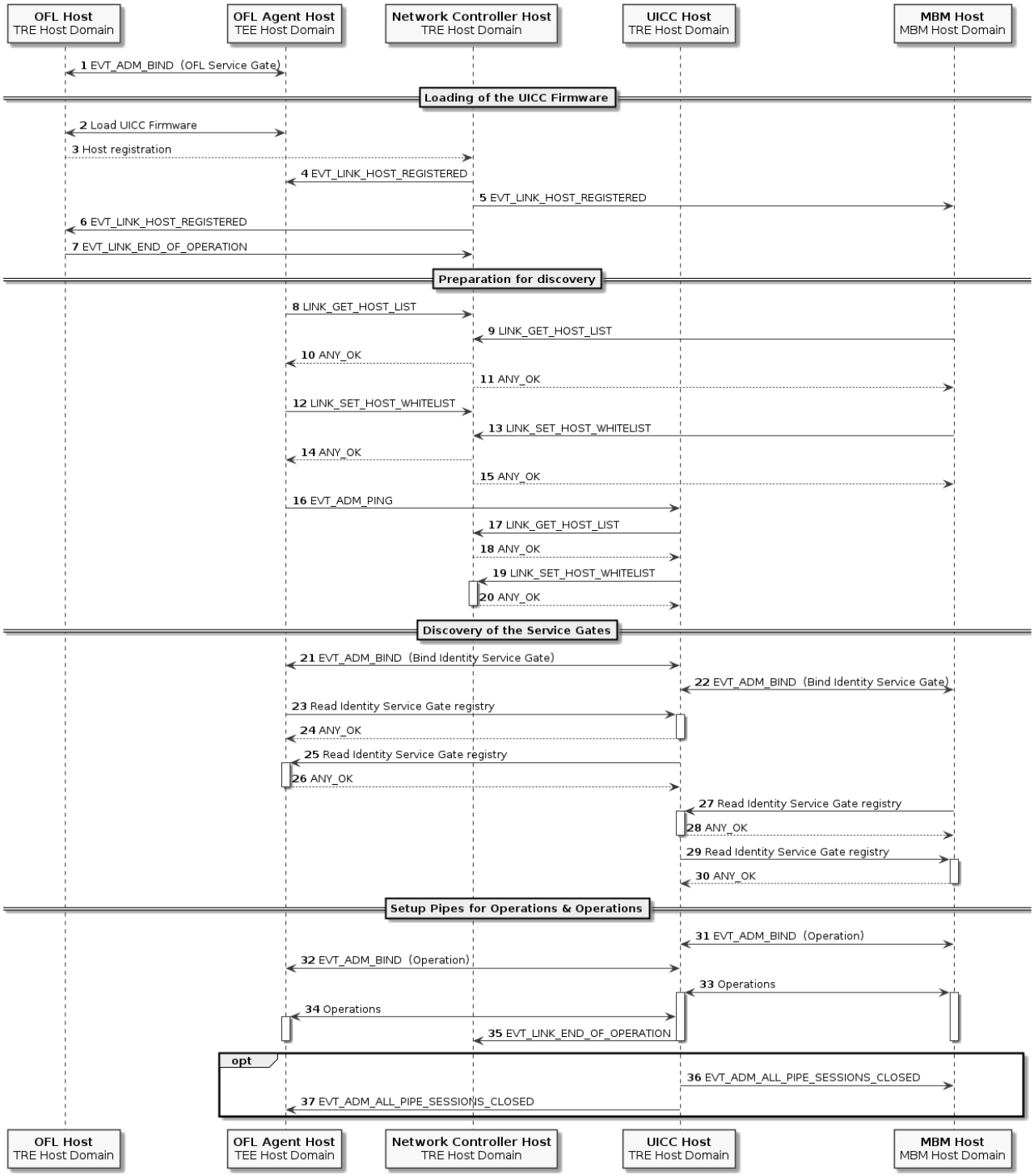
A.2.1 Host Domain

The ecosystem supports the following Hosts:

- MBM Host in the Mobile Broadband Modem Host Domain
- OFL Agent Host in the TEE Host Domain
- OFL Host in the TRE Host Domain
- UICC Hosts in the TRE Host Domain

A.3 Informative: Example of Implementation for the Telecommunication Industry

Figure A-1: Example of Procedure for a UICC Host



The sequence flow is as follows:

1. The OFL Agent Host performs a pipe binding[5] with the OFL Host.
2. The OFL Agent Host loads the UICC Firmware by using the OFL.
3. The OFL Host requires the registration of the UICC Host to the Network Controller Host.
4. The EVT_LINK_HOST_REGISTERED event is sent to the OFL Agent Host.
5. The EVT_LINK_HOST_REGISTERED event is sent to the MBM Host.
6. The EVT_LINK_HOST_REGISTERED event is sent to the OFL Host.
7. OFL Host notifies the Network Controller Host about its end of operations.
8. The OFL Agent Host requests the Hosts list from the Network Controller Host.
9. The MBM Host requests the Hosts list from the Network Controller Host.
10. The Network Controller Host returns the Hosts list to the OFL Agent Host. The OFL Agent Host is informed about the Hosts Arbitration required for the UICC Host.
11. The Network Controller Host returns the Hosts list to the MBM Host. The MBM Host is informed about the Hosts Arbitration required for the UICC Host.
12. The OFL Agent Host sets its whitelist including the Identifier of the UICC Host.
13. The MBM Host sets its whitelist including the Identifier of the UICC Host.
14. The Network Controller Host accepts the whitelist from the OFL Agent Host.
15. The Network Controller Host accepts the whitelist from the MBM Host.
16. The OFL Agent pings the UICC Host.
17. The UICC Host requests the Hosts list from the Network Controller Host.
18. The Network Controller Host returns the Hosts list to the UICC Host.
19. The UICC Host sets up its whitelist, adding the Identifier of the MBM Host.
20. The Network Controller Host confirms the registration of the UICC Host whitelist.
21. The OFL Agent Host performs a pipe binding with the UICC Host for accessing the Identity Service Gates.
22. The MBM Host and UICC Host performs a pipe binding with the UICC Host for accessing the Identity Service Gates.
23. The OFL Agent Host performs a discovery of the services available within the UICC Host.
24. The UICC Host returns its list of Service Gate Identifiers.
25. The UICC Host performs a discovery of the services available within the OFL Agent Host.
26. The OFL Agent Host returns its list of Service Gate Identifiers.
27. The MBM Host performs a discovery of the services available within the UICC Host.
28. The UICC Host returns its list of Service Gate Identifiers.
29. The UICC Host performs a discovery of the services available within the MBM Host.

[5] For simplicity, the binding procedure as described in section 6.2.1 is illustrated as two simultaneous exchanges of two EVT_ADM_BIND events. (A pipe session is created with a pair of pipes.)

30. The MBM Host returns its list of Service Gate Identifiers.
31. The MBM Host and UICC Host perform a pipe binding for accessing the Operational Service Gates.
32. The OFL Agent Host and UICC Host perform a pipe binding for accessing the Operational Service Gates.
33. The MBM Host and UICC Host perform operations.
34. The OFL Agent Host and UICC Host perform operations.
35. The UICC Host notifies the Network Controller Host about the end of its operations.
36. The UICC Host notifies the MBM Host that the communication is terminated (optional). The corresponding Pipe Session is deleted.
37. The UICC Host notifies the OFL Agent Host that the communication is terminated (optional). The corresponding Pipe Session is deleted.

Annex B Tunneling and Emulation of Other Protocols (Informative)

B.1 Overview

Tunneling is a method in computer networks that allows transport of one network protocol embedded in another network protocol; well-known examples in the internet are, e.g., IPv4 in IPv6 encapsulation or VPN tunnels. Practical examples in the context of VNP and legacy TREs (Secure Elements/Smart Cards) are the emulation of the ISO 7816-4 APDU protocol (see section B.2) and the transport of HCI Packets in HCP messages (see section B.3). Another potentially useful mechanism is the transport of Internet Protocol (IP) packets in HCP messages.

All mentioned tunneling mechanisms are optional.

B.2 ISO 7816-4 Protocol Mapping

The ISO 7816-4 protocol emulation is based on the following elements:

- The Transport Layer implements the Go-and-Wait data flow control as defined in [HCI] clause 12.4.
- The Session Layer as defined in [HCI] clauses 12.4, 12.3.2.3, and 12.2.2.2.
- The ISO 7816-4 Presentation Layer as defined in [HCI] clauses 5.2, 12.1, 12.2.2, 12.3, and 12.4.
- The application layer of the ISO 7816 application gate as defined in [HCI] clause 12.3.
- The application layer of the ISO 7816 Service gate as defined in [HCI] clause 12.2.

B.3 HCI Protocol Tunneling

The HCP Packet defined in [HCI] shall be conveyed in an HCP Message. No specific part for the transport layer is required. The specific part of the session layer is defined in clauses 8.4, 8.1, 6.1.2.3, and 6.1.2.4. This HCI tunneling mechanism allows interfacing with a Host Controller as defined in [HCI] and supporting an HCI host within a VNP Host as defined in this document.

Figure B-1: Example of HCI Tunneling

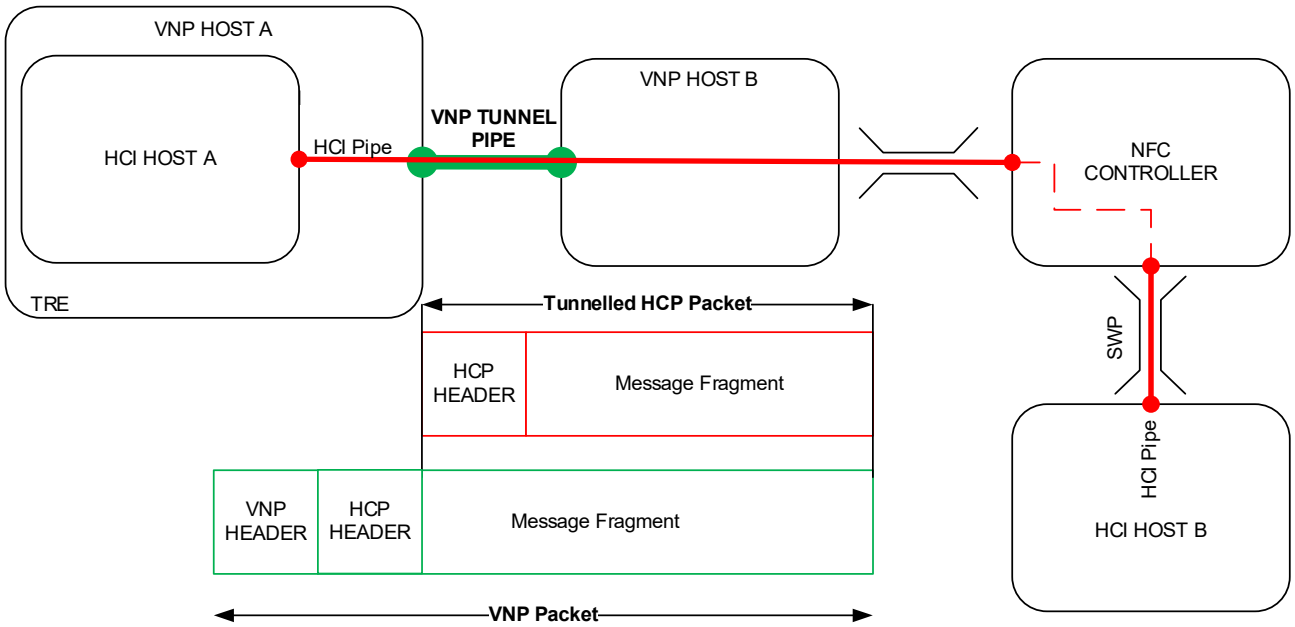


Figure B-1 illustrates an example of HCI protocol tunneling where VNP Host A, within a TRE Host Domain, runs an HCI host as defined in [HCI]. VNP Host A and the HCI Host are able to exchange HCP Packets over a pair of VNP Pipes acting as a generic Data Link Layer for the tunneled HCP Packets of the HCI Host. In this example, VNP Host B acts as proxy for interfacing with an NFC controller. The interface between Host B and the NFC Controller is out of scope of this document.