ETSI GS NFV-INF 007 V1.1.1 (2014-10)

Network Functions Virtualisation (NFV); Infrastructure; Methodology to describe Interfaces and Abstractions

Disclaimer

This document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference

DGS/NFV-INF007

Keywords

interface, NFV

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**may not**", "**need**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document describes how Network Functions Virtualisation (NFV) related interfaces and abstractions are to be derived and specified. It describes the concepts associated with these interfaces and abstractions. It covers the specification process / methodology in general. It presents a cross-cutting framework which covers compute, hypervisor and infrastructure network domains, also data, control and management planes.

The present document does not specify all the interfaces and abstractions as these are covered by other documents, e.g. the NFV INF domain specific documents. Examples of interfaces and abstractions are nevertheless supplied to illustrate the methodology.

The present document does not provide any detailed specification but makes reference to specifications developed by other bodies and to potential specifications, which, in the opinion of the NFV ISG could be usefully developed by an appropriate standards development organization (SDO). Furthermore the NFV INF domain specific documents will not provide detailed specifications either.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

Not applicable.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

unicode: unique, unified and universal encoding

zeroconf: zero configuration networking

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3GPP	3rd Generation Partnership Project
ACL	Access Control List
API	Application Programming Interface
CLR	Common Language Runtime
CPU	Central Processing Unit
EJB	Enterprise JavaBeans
ETSI	European Telecommunications Standards Institute
GS	Group Specification
HFB	Host Functional Block
IETF	Internet Engineering Task Force
ISG	Industry Specification Group
IT	Information Technology
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
JIT	Just In Time
JSON	JavaScript Object Notation
MANO	Management and Orchestration
NFV	Network Functions Virtualisation
NIC	Network Interface Card
NPU	Network Processing Unit
OS	Operating System
OVS	Open Virtual Switch
OVSDB	Open Virtual Switch Database
SATA	Serial Advanced Technology Attachment
SDN	Software Defined Networking
SDO	Standards Development Organization
SR-IOV	Single Root I/O Virtualisation
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UML	Unified Modelling Language
UTF	Unicode Transformation Format
VF	Virtual Function
VFB	Virtualised Functional Block
VM	Virtual Machine
VNF	Virtual Network Function
VNIC	Virtual Network Interface Card
VT	Virtualisation Technology
XML	eXtensible Markup Language

4 Objectives

The three key features of the NFV approach are:

1) Separation of the software defining the network function from generic high volume hardware servers, storage devices and network switches.

2) Independent modularity of the software and hardware components.

3) Automated orchestration which will automate remote installation and management of the virtual functions on the generic hardware.

4.1 Requirements

The overall vision of Network Functions Virtualisation gives rise to the following overall requirements w.r.t. interfaces and abstractions:

- Each functional block or component shall be described as an abstraction, documenting the purpose and behaviour of the functional block or component, and as a set of interfaces to the abstraction.
- The interfaces and abstractions shall be documented at a sufficiently detailed level to permit standardizing organizations to create specifications for these interfaces and abstractions.
- The NFV ISG is expected to liaise with standardizing organizations in order to ensure that such specifications are sufficiently detailed to enable interoperability between platforms and devices hosting Virtualised Network Functions that are offered by different vendors.
- The NFV ISG is expected to liaise with standardizing organizations in order to ensure that such specifications are decoupled from vendor-specific design and implementation choices within platforms and devices hosting Virtualised Network Functions.

4.2 Standardizing Organizations

The standardizing organizations that are responsible for creating detailed specifications of each interface and abstraction are listed in the overview document and in domain specific documents.

Other organizations that define methodologies relevant to interfaces and abstractions include the Object Management Group (specifically for UML) and the International Council on Systems Engineering (specifically for Systems Engineering).

5 Architectural Principles w.r.t. Interfaces and Abstractions

Many network systems, including those specified by 3GPP, IETF, and ITU-T, are specified using the principles of systems engineering. Each component of the overall system is specified as a functional block and the interactions between the functional blocks are specified as interfaces.

This clause details the basic functional block based system composition methodology and extends it to cover the process of virtualisation.

The representation of functional blocks is part of the working methods of many industries as well as different disciplines and perspectives within those industries. As a result, there is not a clear common representation of functional blocks which is unambiguous across different industries, disciplines, and perspectives.

As tools that describe functional blocks are most often used by engineers for the design, development and construction of functional blocks, quite naturally, many tools give considerable emphasis to these phases of the functional block life cycle. For example, in the construction phase, the reuse of common design features is especially important as reuse increases efficiency. Many tools therefore give considerable emphasis to the reuse of such features. In this case classification of functional blocks according to common design features is of considerable value and the natural starting point for describing functional blocks is the class. It is natural to start by representing a class of functional blocks which can be built using the same design. The class diagram can also contain hierarchy, for example an inheritance hierarchy, which can show increasing scope of design reuse at higher levels of the class hierarchy.

However, when describing the operation of functional blocks, the individual instances of functional blocks and the way individual functional blocks interact with each other are important. In this case the natural starting point is not the class but the individual instances. Classification and hierarchy of classification is much less relevant at the operations stage. More important is the way individual functional block instances are interconnected and interact.

There are of course many other aspects to functional blocks which may be important to represent in different way. For example the nature of what is passed between functional blocks may be important to differentiate. In all information and network systems, it is information that is passed between the functional blocks. However, even in this narrow category, in may be important to distinguish a continuous flow of information from discrete events. More general systems may pass fluid (pressure and flow), electricity (voltage and current), rotation (revs and torque), money, etc.

The present document is concerned with the basic characteristics of information functional blocks. We can assume that all the parameters passed between functional blocks are information of one form or another.

This clause considers some of the basic properties of information functional blocks. It focuses on the case where a functional block such as a server or a network acts as a host functional block, hosting virtual functional blocks such as virtual machines and virtual networks.

In this case, it is important to highlight the properties of functional blocks in operation and so all the discussion and diagrams in the present document show functional block instances (and not classes of functional blocks) unless otherwise stated.

5.1 System Composition using Functional Blocks

5.1.1 Functional Blocks as the Primary Specification Method

The great majority of specification of telecommunications systems specifies functional blocks using the methodology of systems engineering. A functional block is the basic unit of a system and its specification can and should be precise.

A functional block, that is a single functional block instance, consists of:

- A set of input interfaces.
- State.
- A transfer function.
- A set of output interfaces.

When describing practical functional blocks, the interfaces may be described such that an input interface is paired with an output interface. This is normally convenient when describing the interconnection between functional blocks.

When considering the functional block at its most fundamental level, the proper operation of a functional block is causal and the flow of causality from input to output is central to the methodology. In this case is normally more convenient to consider all input as separate from all outputs. A fundamental view of a functional block illustrated in Figure 1.

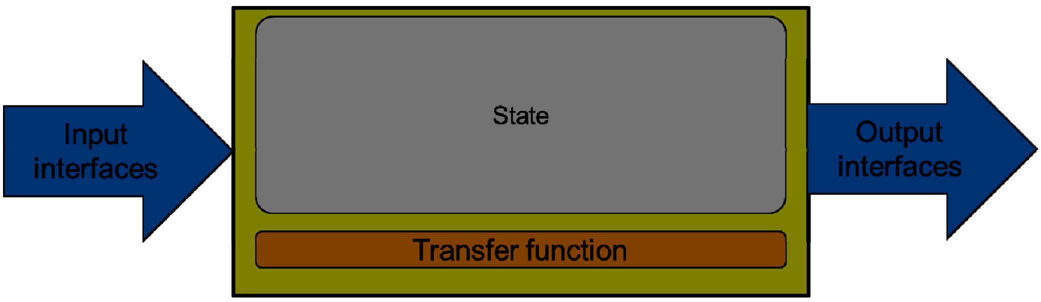

Figure 1: The fundamentals of a functional block

There a number of fundamental properties of functional blocks:

- The transfer function is fixed and defining of the functional block.
- The set of all possible values of state is fixed and is defining of the functional block.
- The set of all possible input values is fixed and is defining of the functional block.

- The set of all possible output values is fixed and is defining of the functional block.
- The transfer function is the set of mapping from a specific value of tuples of input and state to a specific value of tuple of output and state.
- The state is the ability of the evolution of the functional block to be dependent on historic inputs and not just on the current input.
- The process of acquiring the current input value, the current state value, calculating the transfer function mapping, and setting the next state value and the next output value takes a finite amount of time.
- The exact amount of time may vary with each specific mapping.

A central property of functional blocks is the complete and formal separation of the static from the dynamic. Using a more IT oriented terminology, the input, output, and internal (i.e. state) data structures and all the methods (i.e. the transfer function) and static. They shall not change. Only the values of data within the data structures can change; these values are the only things which are dynamic.

For standardized functional blocks, in order to ensure proper interoperability, the goal would normally be to fully define all the static parameters of the functional block in the standard.

5.1.2 Interconnection of Functional Blocks

Having defined what a functional block is from the inside, the next fundamental property of a functional block the ability to interconnect functional blocks. This is achieved by connecting an output interface of one functional block with the input interface of another functional block. For this to work the following shall be true.

- The data structure of the output interface of the functional block on side of the interconnection shall be compatible with the data structure input interface of the functional block on the other side of the interconnection. The output set of values shall a subset of the input set of values.

This interconnection of interfaces is called an interface binding. This arrangement is illustrated in Figure 2.

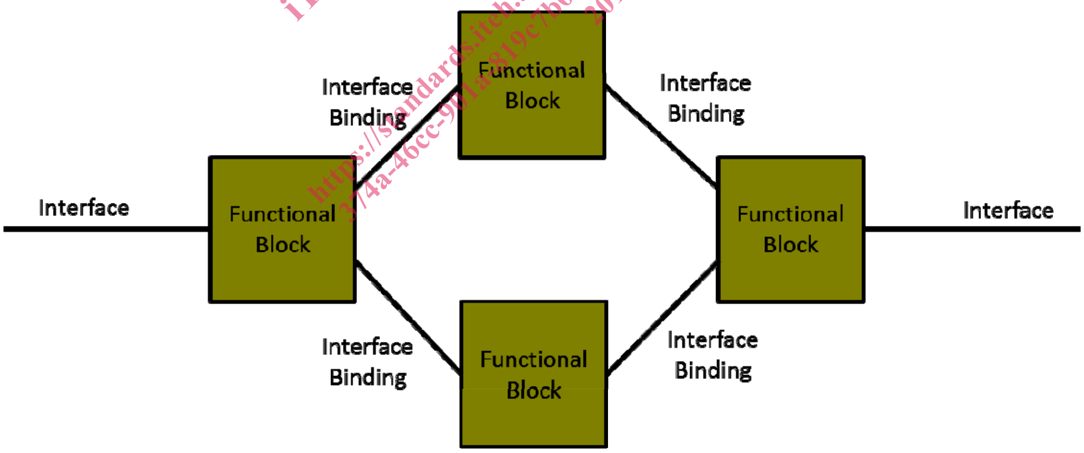

Figure 2: Interconnection of functional block with interface bindings

As illustrated in Figure 2, when a number of functional blocks are interconnected all together, the interface bindings form a topology between the functional blocks.

5.1.3 Recursive Structure of Functional Blocks

When a number of functional block are interconnected in a topology, some input interfaces and some output interfaces remain. A fundamental property of the functional block methodology is that the entity as viewed through these remaining input and output interfaces is also a functional block and meets all the properties of a single functional block.

This is illustrated in Figure 3.

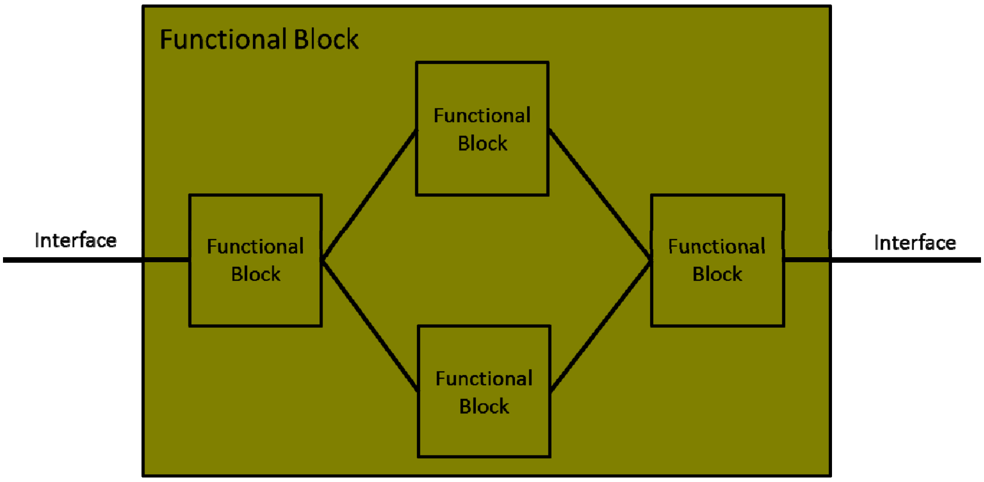

Figure 3: Recursive property of functional blocks

This means that functional blocks have a fundamental recursive property. A larger functional block can be created by aggregating a number of a smaller functional block and interconnecting them with a specific topology.

Another fundamental property of functional blocks which is immediately apparent from this recursive property is that functional blocks are inherently parallel, concurrent, and asynchronous. Any sequential and synchronous properties will arise only as a special case, normally by imposing explicit design constrains on the static properties of all the constituent functional blocks.

5.1.4 General UML Diagram for Basic Functional Block Methodology

It is possible to summarize the basic entities of the functional block methodology using a UML class diagram. UML class diagrams show classes, that is, sets of things which all have the same property. Most often classes in UML class diagrams are there to define the construction of members, often called instances, of the class. In this way, members of the class have the same properties because the class definition from the class diagram is used directly to create the instances. However, the class can also be used to categorize things even if they were not created directly by the specification. They can be classified in retrospect, rather by design.

The ability to classify in retrospect is important for many aspects of practical systems. Generally component functional blocks are designed, built, and made operational at different times and often it is not possible to change existing functional blocks when introducing new components. The UML diagrams used here show general classification, and instances of functional block classes may be classified as instances of the classes in the diagram after they have been designed, built, and made operational.

UML class diagrams show classes and relationships between classes. A relationship is shown between two classes. Two completely different type of relationship are as follows.

- Generalization (also called inheritance). This relationship shows that two classifications are different viewpoints of the same thing. The instances of the classes are the same instances. The relationship is often referred to as an 'is a' relationship. In UML, generalization is shown by an open triangular arrowhead
- Associations. These are relationship between two separate instances which interact with each other. UML allows three different strengths of association:
    - Association. General interaction between instances.
    - Aggregation. An association where the instance on one class is a component part of an instance of the other class.
    - Composition. An aggregation where the existence of the component instance depends on the existence of the aggregate instance.

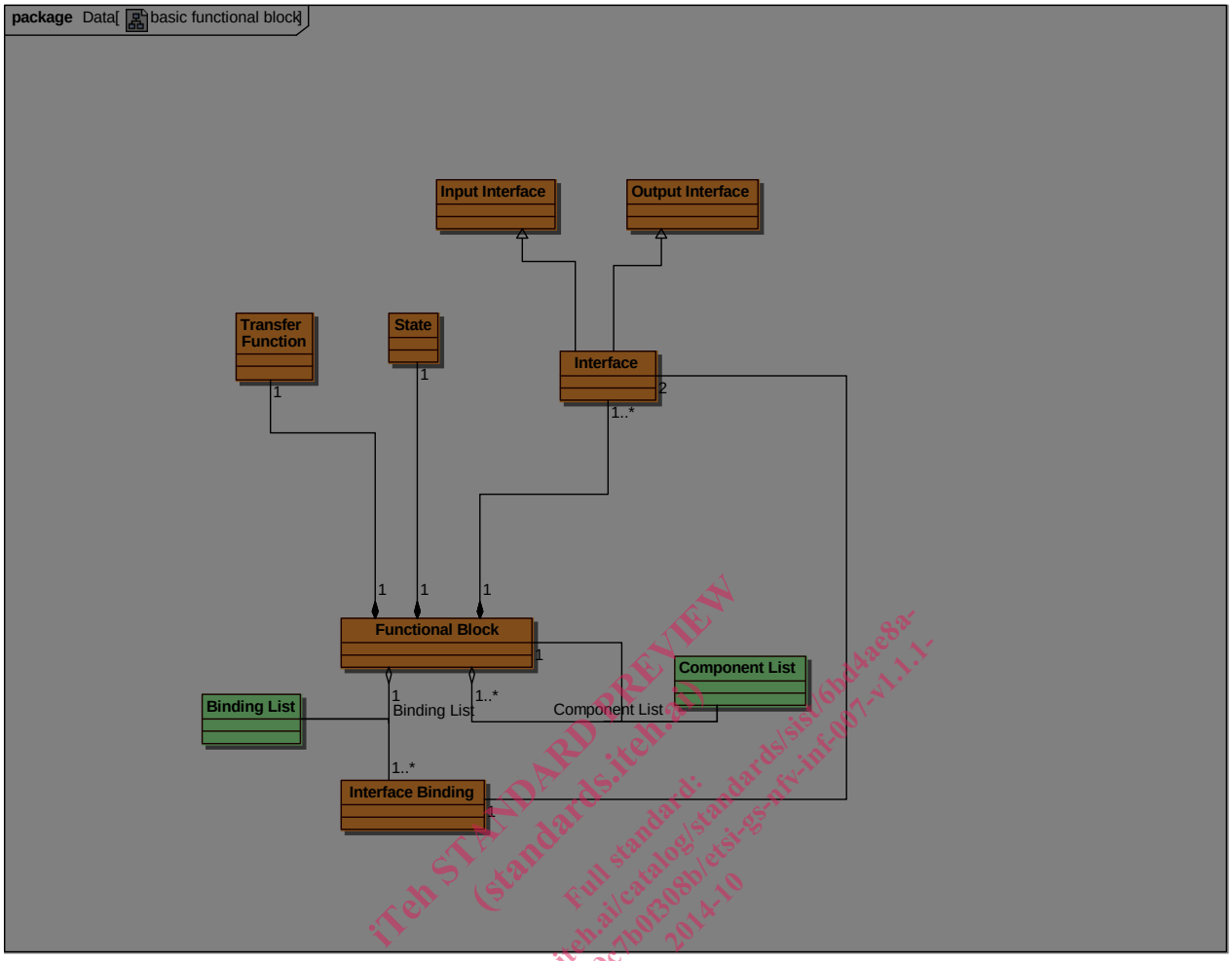

Figure 4: UML class diagram of functional blocks

Figure 4 shows a UML class diagram of functional blocks (note the functional block class is called virtual function for consistency with the following clause). The diagram specifies the three basic parts of a functional block (transfer function, state, interfaces), the ability to bind interfaces, as well as the property of recursive composition. Note that this use of composition is different to the UML definition.

5.2 Extension of Functional Block Methodology to Virtualisation

Functional block methodology does not anticipate or direct support virtualisation. However, the foundations of the methodology are very general and mathematically robust. It is still possible therefore to understand virtualisation in terms of functional blocks. This clause develops the extension of the methodology in terms which are still fully based on the same general, mathematical principles and so still retains the formal robustness of the methodology.

In summary, the essence of virtualisation is to revisit the boundary between the static and dynamic parts of the functional block specification. We will see that virtualisation uses a process with the following steps:

- A host function has some dynamic state which can be set to a value (configured) and held constant for a prescribed period of time (which will be the lifetime of the virtualised function).
- This configuration allows the host to appear to operate according to the specification of the virtualised function - this configuration of the host implements the virtualised function.

In fact, it is the case that all implementation is exactly this process. It is indeed, this mechanism of virtualisation that allows any implementer freedom to choose and optimize their own implementation of the virtual function. Moreover, all implementation independent specification is a specification of a virtual function.