Final draft ETSI ES 202 915-10 V1.4.1 (2006-10)

ETSI Standard

Open Service Access (OSA); Application Programming Interface (API); Part 10: Connectivity Manager SCF (Parlay 4)

Reference
RES/TISPAN-01028-10-OSA

Keywords
API, IDL, OSA, UML

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

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://webapp.etsi.org/IPR/home.asp).

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 ETSI Standard (ES) has been produced by ETSI Technical Committee Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN), and is now submitted for the ETSI standards Membership Approval Procedure.

The present document is part 10 of a multi-part deliverable covering Open Service Access (OSA); Application Programming Interface (API), as identified below. The API specification (ES 202 915) is structured in the following parts:

Part 1: "Overview";

Part 2: "Common Data Definitions";

Part 3: "Framework";

Part 4: "Call Control";

Part 5: "User Interaction SCF";

Part 6: "Mobility SCF";

Part 7: "Terminal Capabilities SCF";

Part 8: "Data Session Control SCF";

Part 9: "Generic Messaging SCF";

Part 10: "Connectivity Manager SCF";

Part 11: "Account Management SCF";

Part 12: "Charging SCF";

Part 13: "Policy Management SCF";

Part 14: "Presence and Availability Management SCF".

The present document has been defined jointly between ETSI, The Parlay Group (http://www.parlay.org) and the 3GPP, in co-operation with a number of JAIN™ Community (http://www.java.sun.com/products/jain) member companies.

The present document forms part of the Parlay 4.3 set of specifications.

1 Scope

The present document is part 10 of the Stage 3 specification for an Application Programming Interface (API) for Open Service Access (OSA).

The OSA specifications define an architecture that enables application developers to make use of network functionality through an open standardised interface, i.e. the OSA APIs.

The present document specifies the Connectivity Manager Service Capability Feature (SCF) aspects of the interface. All aspects of the Connectivity Manager SCF are defined here, these being:

- Sequence Diagrams.
- Class Diagrams.
- Interface specification plus detailed method descriptions.
- State Transition diagrams.
- Data Definitions.
- IDL Description of the interfaces.

The process by which this task is accomplished is through the use of object modelling techniques described by the Unified Modelling Language (UML).

2 References

The references listed in clause 2 of ES 202 915-1 contain provisions which, through reference in this text, constitute provisions of the present document.

ETSI ES 202 915-1: "Open Service Access (OSA); Application Programming Interface (API); Part 1: Overview (Parlay 4)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ES 202 915-1 and the following apply:

best effort traffic: traffic that is not carried by a VPrN established for the enterprise network by the provider

differentiated services: effort in the Internet Engineering Task Force (IETF) to provide quality of service in networks employing small well defined building blocks from which variety of service may be built

DS codepoint: marking associated with a specific VPrP

enterprise operator: administrator of the enterprise network/user of APIs

NOTE: Also referred to as operator.

operator: enterprise operator

provider network: provides a VPN and VPrP service to the enterprise network, and offers APIs for connectivity manager to the enterprise operator

NOTE: Also referred to as network service provider.

provider: entity that offers the VPN and VPrP services, and implements the APIs in the provider network

Quality of Service (QoS): collection of service levels delivered by a provider network to an enterprise network

NOTE: QoS can be characterised by various performance attributes such as: packet loss, packet delay; traffic policing measures such as maximum bandwidth and burst rate for traffic entering the providers network.

Service Access Point (SAP): enterprise network is connected to the provider network through the enterprise network service access points

NOTE: A SAP is typically the egress router from the enterprise network that connects to the provider network.

TOS bits: value held in the TOS field

NOTE: IETF defined the use the TOS field in the IPv4 packet header as a signalling mechanism aimed at providing definitions of aggregation of flows, where each aggregate is supported by the same level of QoS.

Virtual Private Network (VPN): network that uses a provider network infrastructure to connect geographically separated sites of an enterprise

NOTE: Such a network looks like a private network to the enterprise as the sites are connected using tunnelling and security technologies. With no QoS measures, VPN passes all packets among the sites with a best effort approach.

Virtual Leased Line (VLL): network that uses a provider network infrastructure to connect two geographically separated sites of an enterprise

Virtual Provisioned Network (VPrN): collection of VPrP delivered as a service to a single enterprise network

Virtual Provisioned Pipe (VPrP): service provided by the provider network to the enterprise network, which is a type of virtual leased line (VLL) provisioned with QoS levels

NOTE: VPrP carries enterprise network traffic whose packets are marked with the specific DS Codepoint that is associated with this VPrP. The enterprise operator using APIs can create on-line a VPrP.

3.2 Abbreviations

For the purposes of the present document, the abbreviations defined in ES 202 915-1 and the following apply:

CIM	Common Information Model
CM	Connectivity Manager
DiffServ	Differentiated Services
DMTF	Distributed Management Task Force
DS	Differentiated Services
QoS	Quality of Service
SAP	Service Access Point
SLA	Service Level Agreement terms
TOS	Type Of Service
VLL	Virtual Leased Line
VPN	Virtual Private Network
VPrN	Virtual Provisioned Network
VPrP	Virtual Provisioned Pipe

4 Connectivity Manager SCF

Connectivity Manager includes the APIs between the enterprise operator and the provider network for the two parties to establish QoS parameters for enterprise network packets travelling through the provider network.

The Connectivity Manager service provides tools for the enterprise operator to set up a Provisioned QoS service in the provider network. The QoS measures used in the enterprise network are outside the scope of the service. The API does not require any specific QoS method to be used in the enterprise network, nor in the provider network. However, in order for Provisioned QoS service to be applied to packets arriving from the enterprise network into the provider network, the packets have to be marked using DS Codepoint marking. Once the packets are so marked, they can enjoy the QoS service provisioned in the provider network.

APIs provide the enterprise network operator on-line access to provision quality of service measures that control the enterprise's own traffic passing through the provider network. Using APIs the operator can create virtual provisioned pipes (VPrPs) in the provider network to carry the enterprise traffic and support it with pre-specified quality of service attributes. A VPrP can be thought of as a Virtual Leased Line (VLL) provisioned to deliver pre-specified QoS. The provider may offer to the enterprise operator a set of templates that are used by the operator to specify a VPrP. For instance, the provider may offer templates for video conferencing, audio conferencing, Gold Service, Silver Service, etc. Using these templates the operator can select and provision a VPrP that specifies the quality of service attributes for this VPrP.

Elements that can be specified for a VPrP include attributes such as packet delay and packet loss. Characteristics of traffic that enters the VPrP at its access point to the provider network can be also specified with attributes such as maximum rate and burst rate.

The following is an example of a possible scenario:

- The provider prepares a template with *operator-specified* attributes, *provider-specified* attributes, and *unspecified* attributes, one for each QoS level.
- The provider generates for the enterprise network a list of all the current sites and their access points to the provider network.
- Enterprise operator logs into connectivity manager after being authenticated and authorised by the Framework service.
- Operator gets the list of the sites and service access points of the enterprise virtual private network (VPN) already provided to the enterprise by the provider.
- Enterprise operator retrieves the set of templates available to the enterprise (as supported by the SLA), selects one, and requests a template for constructing a new VPrP based upon the selected QoS.
- Enterprise operator completes the VPrP template: i.e. selects a value for delay, loss, jitter and excess traffic treatment action, enters the SLA ID against which the template could be validated, selects endpoints, load parameters and traffic flow direction, and selects the time requirements desired. The enterprise operator can choose or modify those attributes that are *operator-specified* attributes in the template. P*rovider-specified* attributes cannot be modified and are inherently part of the service.
- Enterprise operator submits the completed VPrP template for validation by the CM service. Operator creates a new VPrP with *pending-status* that holds these selections.
- The provider responds after validating the requests, which may be an approval or a denial (e.g. the requested service is not available at this access point, or at the specified time).
- If the provider approves service, the operator may send packets marked with the templates DiffServ Codepoint, which identifies together with the endpoints the VPrP that carries these packets.

Some additional clarification points:

- A VPrN is associated with a single network provider.
- A VPrP defines QoS parameters for traffic flowing through this provider network, between two specified enterprise endpoints, optionally during specified date/time period(s).
- The enterprise operator may be (provider's choice) constrained to selecting QoS parameter values from a pre-defined set of values, and selecting endpoints from a predefined set of enterprise sites; where these sets were negotiated off-line between the enterprise and the network provider and possibly documented in a Service Level Agreement (SLA).
- The CM service validates each VPrP request submitted by the enterprise operator. The validation process is not specified here. Validation against the SLA is an example of such possible validation.
- If the CM service accepts the VPrP request, it adds it to the VPrN. The DiffServ Codepoint provided by the enterprise operator is then used by the enterprise for marking all packets belonging to any traffic flow associated with the VPrP.

The following is a summary of interfaces and methods supported by connectivity manager. The syntax method (interface) is used for this description.

There are Passive CM interface functions (CM1, CM2, CM4) that are used to retrieve information (read) relative to VPN, VPrN, and QoS templates provided by the service provider, and active (CM3) functions (read/write) used to provision new services.

CM1: Retrieve information on a Virtual Private Network, its sites and their service access points:

getEnterpriseNetwork(IpConnectivityManager)
getSiteList(IpEterpriseNetwork)
getSite(IpEterpriseNetwork)
getSAPList(IpEnterpriseNetworkSite)
getSiteID(IpEnterpriseNetworkSite)
getSiteLocation(IpEnterpriseNetworkSite)
getSiteDescription(IpEnterpriseNetworkSite)
getSAPIPSubnet(IpEnterpriseNetworkSite)
getSiteIPSubnet(IpEnterpriseNetworkSite)

CM2: Retrieve QoS services offered by provider, stored in QoS templates:

getQoSMenu(IpConnectivityManager)
getTemplateList(IpQoSMenu)
getTemplate(IpQoSMenu)
getTemplateType(IpQoSTemplate)
getDescription(IpQoSTemplate)
getPipeQoSInfo(IpQoSTemplate)
getValidityInfo(IpQoSTemplate)
getProvisionedQoSInfo(IpQoSTemplate)
getDsCodepoint(IpQoSTemplate)

CM3: Set up a new Virtual Provisioned Pipe:

createVPrP(IpVPrN)
deleteVPrP(IpVPrN)
setSlaID(IpQoSTemplate)
setPipeQoSInfo(IpQoSTemplate)
setValidityInfo(IpQoSTemplate)
setProvisionedQoSInfo(IpQoSTemplate)

CM4: Retrieve information on a Virtual Provisioned Network and its Virtual Provisioned Pipes:

getVPrN(IpEterpriseNetwork)
getVPrPList(IpVPrN)
getVPrP(IpVPrN)
getVPrPID(IpVPrP)
getSlaID(IpVPrP)
getStatus(IpVPrP)
getProvisionedQoSInfo(IpVPrP)
getPipeQoSInfo(IpVPrP)
getDsCodepoint(IpVPrP)

Two typical scenarios:

1) To set up a new VPrP:

   - The enterprise operator retrieves information regarding enterprise's existing VPN (sites and SAPs), using as needed the methods listed in CM1 above.
   - Enterprise operator retrieves information on provider's offered QoS services, using methods listed in CM2 above as needed.
   - Enterprise operator submits a request to set up a new VPrP, using methods listed in CM3 above as needed.
   - Enterprise operator checks the status of the request using the methods listed in CM4 above as needed.
   - If the request was approved by the network provider, the VPrP is put in an active mode, and packets that are marked in the enterprise network with appropriate marking in their packet header will travel the provider's network through the new Virtual Provisioned Pipe that supports the requested QoS levels.

2) Retrieve information on current enterprise network services delivered to the enterprise by the provider network:

   - Enterprise operator retrieves information on current QoS services delivered to the enterprise network using CM4 methods listed above as needed.
   - Enterprise operator checks if the provider offers new QoS services using CM2 listed above as needed.

The following clauses describe each aspect of the Connectivity Manager Service Capability Feature (SCF).

The order is as follows:

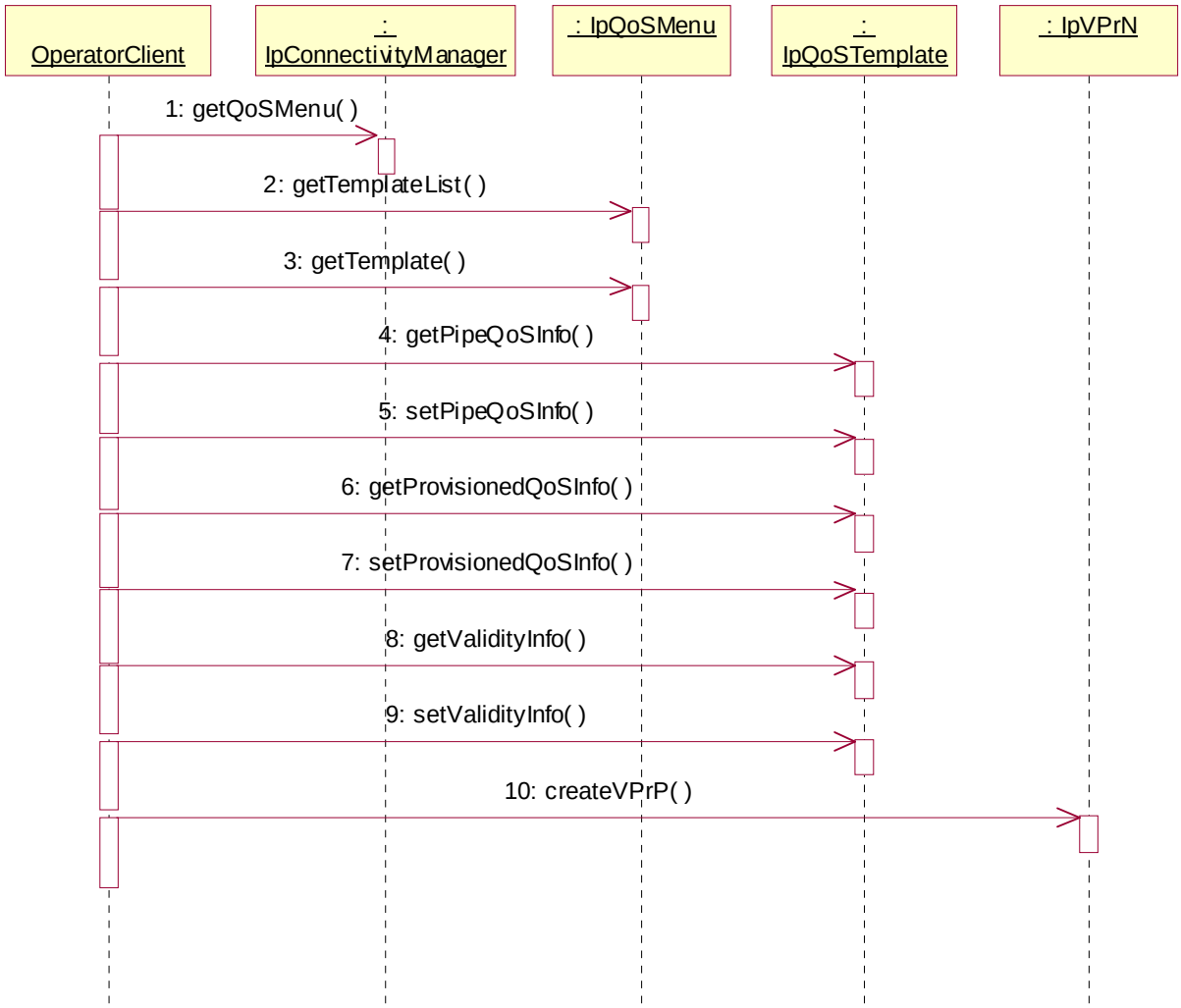
- The Sequence diagrams give the reader a practical idea of how each of the SCF is implemented.
- The Class relationships clause shows how each of the interfaces applicable to the SCF, relate to one another
- The Interface specification clause describes in detail each of the interfaces shown within the Class diagram part.
- The State Transition Diagrams (STD) show the transition between states in the SCF. The states and transitions are well-defined; either methods specified in the Interface specification or events occurring in the underlying networks cause state transitions.
- The Data Definitions clause shows a detailed expansion of each of the data types associated with the methods within the classes. Note that some data types are used in other methods and classes and are therefore defined within the Common Data types part ES 202 915-2.

An implementation of this API which supports or implements a method described in the present document, shall support or implement the functionality described for that method, for at least one valid set of values for the parameters of that method. Where a method is not supported by an implementation of a Service interface, the exception P_METHOD_NOT_SUPPORTED shall be returned to any call of that method.

5 Sequence Diagrams

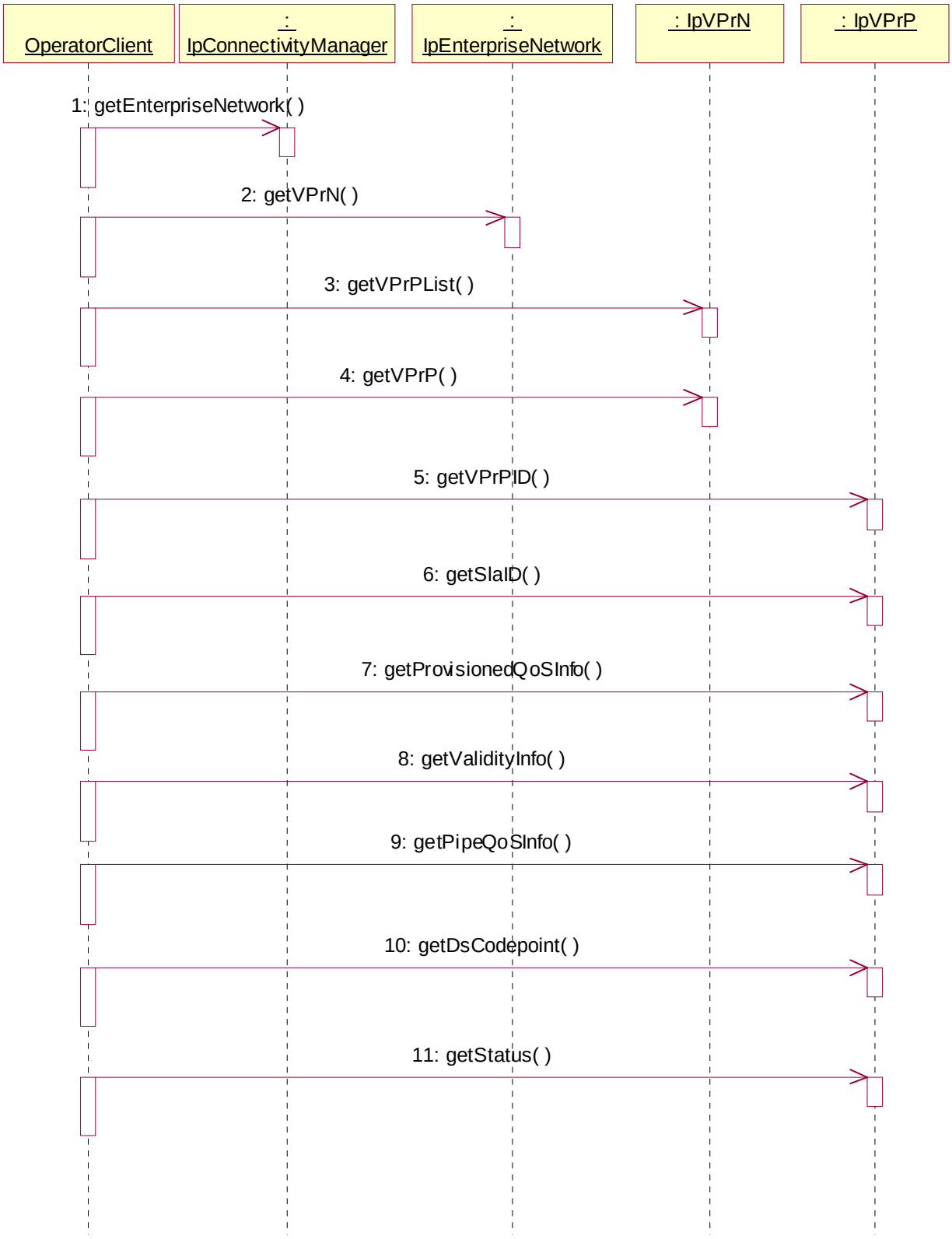
5.1 Operator Selects Service Components and creates a new VPrP

The following sequence diagram shows how an enterprise operator client collects the information required to select a service, and then selects the service parameters, and finally submits it to the connectivity manager.



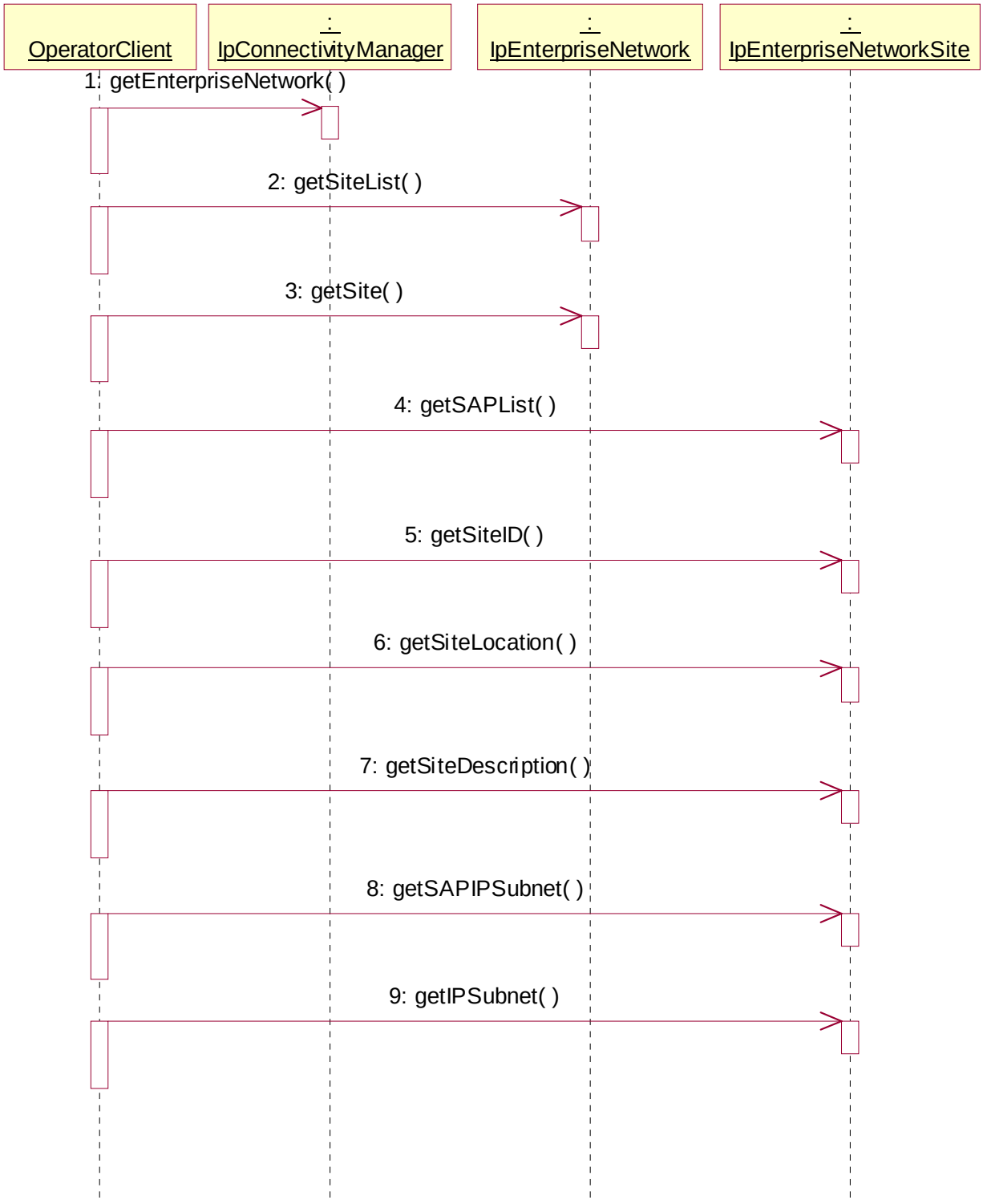
5.2 Operator Browses Virtual Provisioned Pipe

The following shows an enterprise operator client browses and collect information pertinent to an existing virtual provisioned pipe, including all the QoS parameters that have been set for this pipe.



5.3 Operator Browses SAPs and Sites

The following sequence diagram shows how an enterprise operator browses service access points and sites to retrieve information regarding a site and its SAP(s).



6 Class Diagrams

The class diagram below shows the connectivity manager interfaces. The connectivity manager interface is the entry point to this service. From this interface a reference to the enterprise network and to the Quality of Service (QoS) menu interfaces can be retrieved. The QoS menu interface provides a list of templates, each of which specifies the QoS service parameters that are offered by the service provider. The service is composed of components that are associated with a Provisioned QoS. The template interface is used to specify the service parameters that are offered by the service provider, and also, for temporary storage of parameters that the operator selects. The enterprise network interface is associated with two components: enterprise network sites, and the Virtual Provisioned Network that has been already provisioned in the provider network. The Virtual Provisioned Network interface contains references to all the Virtual Provisioned Pipes (VPrPs) already established. The QoS Menu contains references to all the QoS templates offered by the provider. Each template specifies the QoS parameters that can be set in order to create a new VPrP. Once the operator selects the QoS parameters provided in the QoS template, and submits the request to create a new VPrP, the provider validates the information submitted and if the request is approved, the new VPrP is set to an active mode.

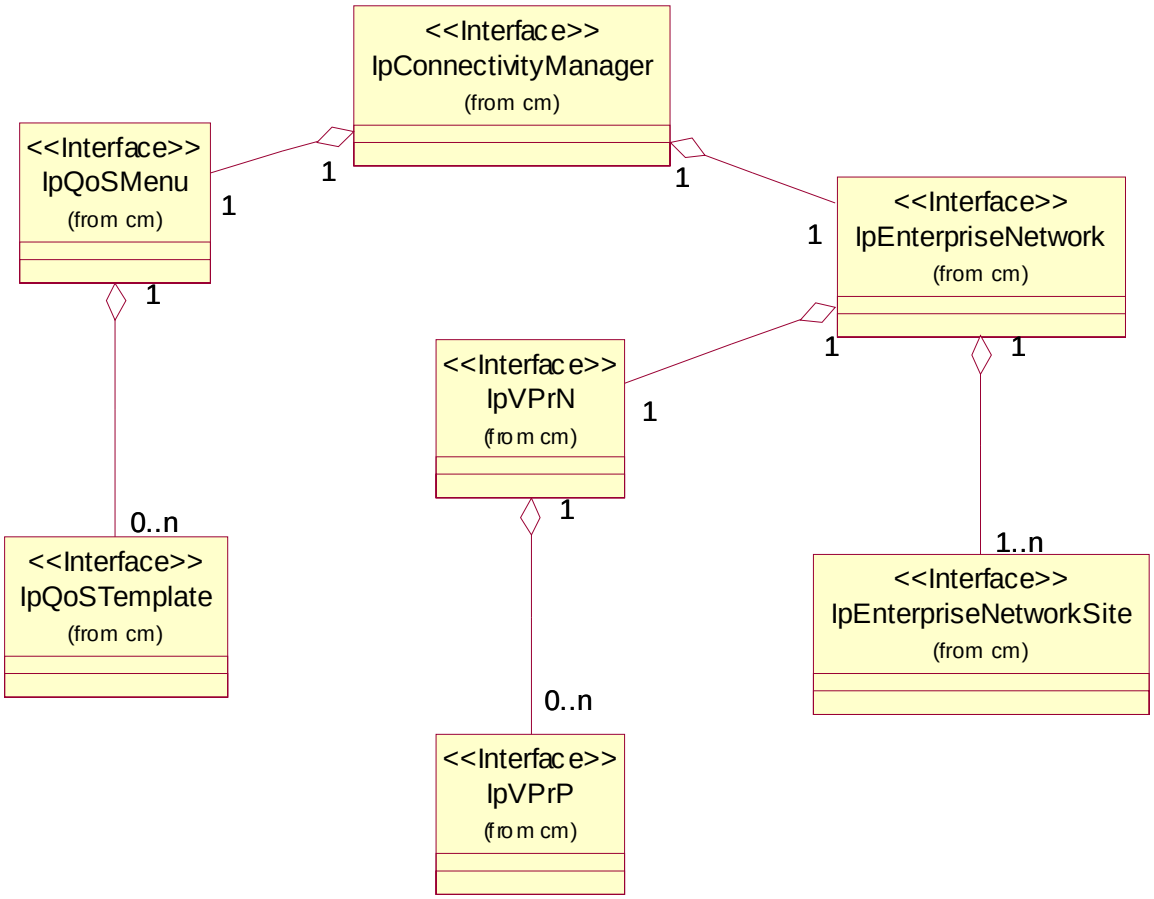


Figure 1: Connectivity Management High level class diagram

The connectivity manager class is associated with the QoS menu class and the enterprise network class as shown in the figure below. From the CM class the QoS Menu and Enterprise Network reference can be retrieved. The QoS menu provides a method (getTemplateList()), to get the list of all the QoS offered services, each of which is stored as a template (e.g., Gold template, Silver template). The QoS menu provides a method (getTemplate()) that instantiates a temporary template to retrieve the default parameters for a specific template, and store the parameters selected by the operator for a new service (VPrP). The Enterprise Network class is used to retrieve the list of sites that the operator has established for a Virtual Private Network (VPN) service with the service provider, and to get a reference to a specific site. This class also provides a reference to the Virtual Provisioned Network class that holds the information regarding the already established QoS network with the provider.

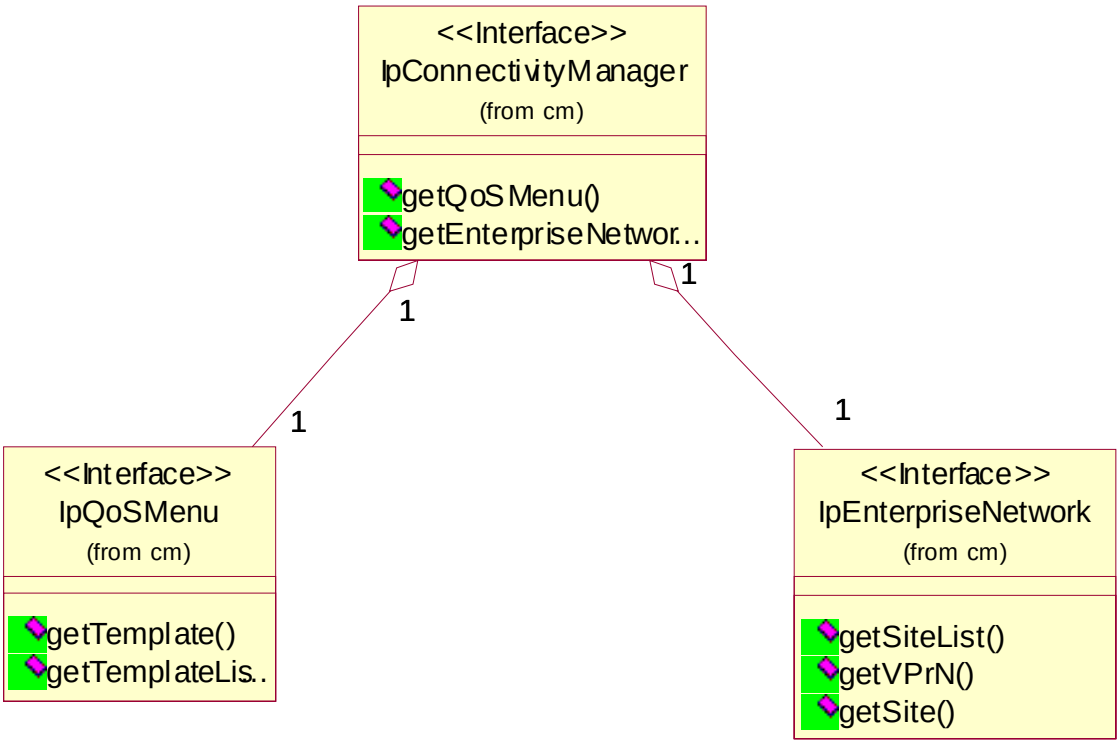


Figure 2: QoS Menu and Enterprise Network class diagram

The figure below shows QoS templates class that is used to retrieve the QoS parameters offered by the provider and select the parameters for a new VPrP and store them in this template. Most of these parameters are tagged using the TPNameDescrpTag class to indicate whether the parameter is set by the provider and cannot be changed by the operator, or is a parameter that can be set by the operator. A third option associated with each QoS parameter is the unspecified tag, that means that the operator does not offer this parameter as an option for this template. All the QoS parameters that constitute a VPrP are stored in the template class, including the source and destination site/SAP, the direction of the traffic, the provisioned QoS parameters, such as packet loss, delay, jitter, the Pipe QoS parameters, such as load conditioning.

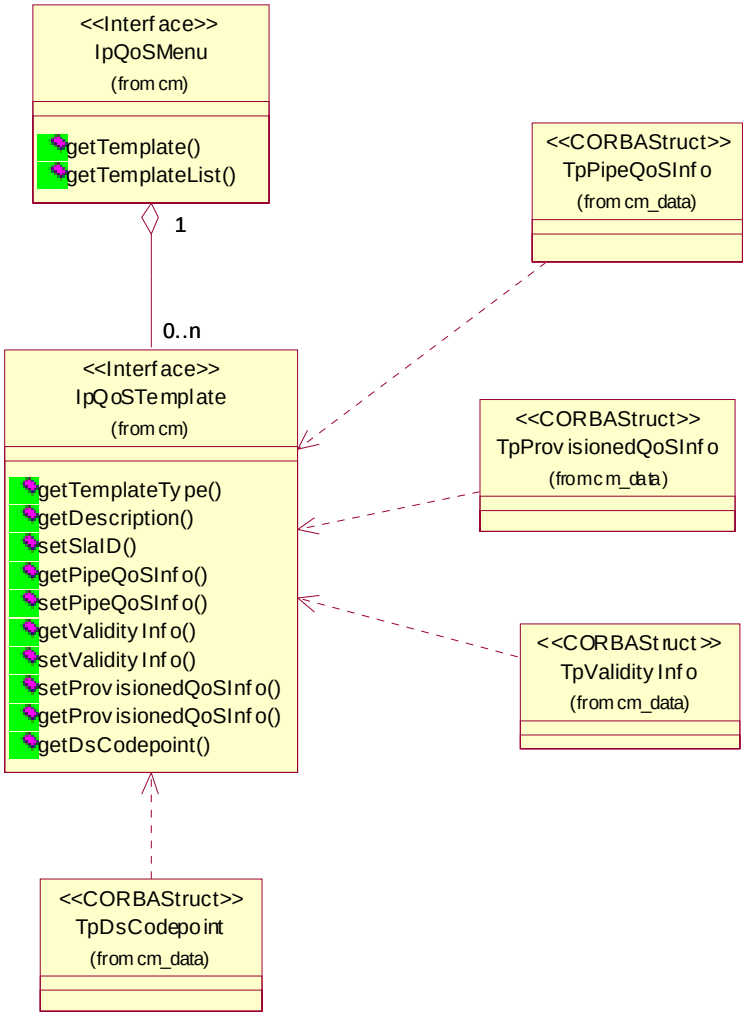


Figure 3: QoS Template class diagram

The figure below shows the data types that specify the Provisioned QoS information: as packet loss, packet delay, packet jitter. All use the Name Description Tag class to indicate whether each parameter is specified by the provider, the operator, or unspecified for this template. The value used for each of individual parameters is either an integer, or a string. For the excess load condition, one of the four actions can be applied: drop, transmit, reshape, and remark. The DS Codepoint used by this template is also shown below.

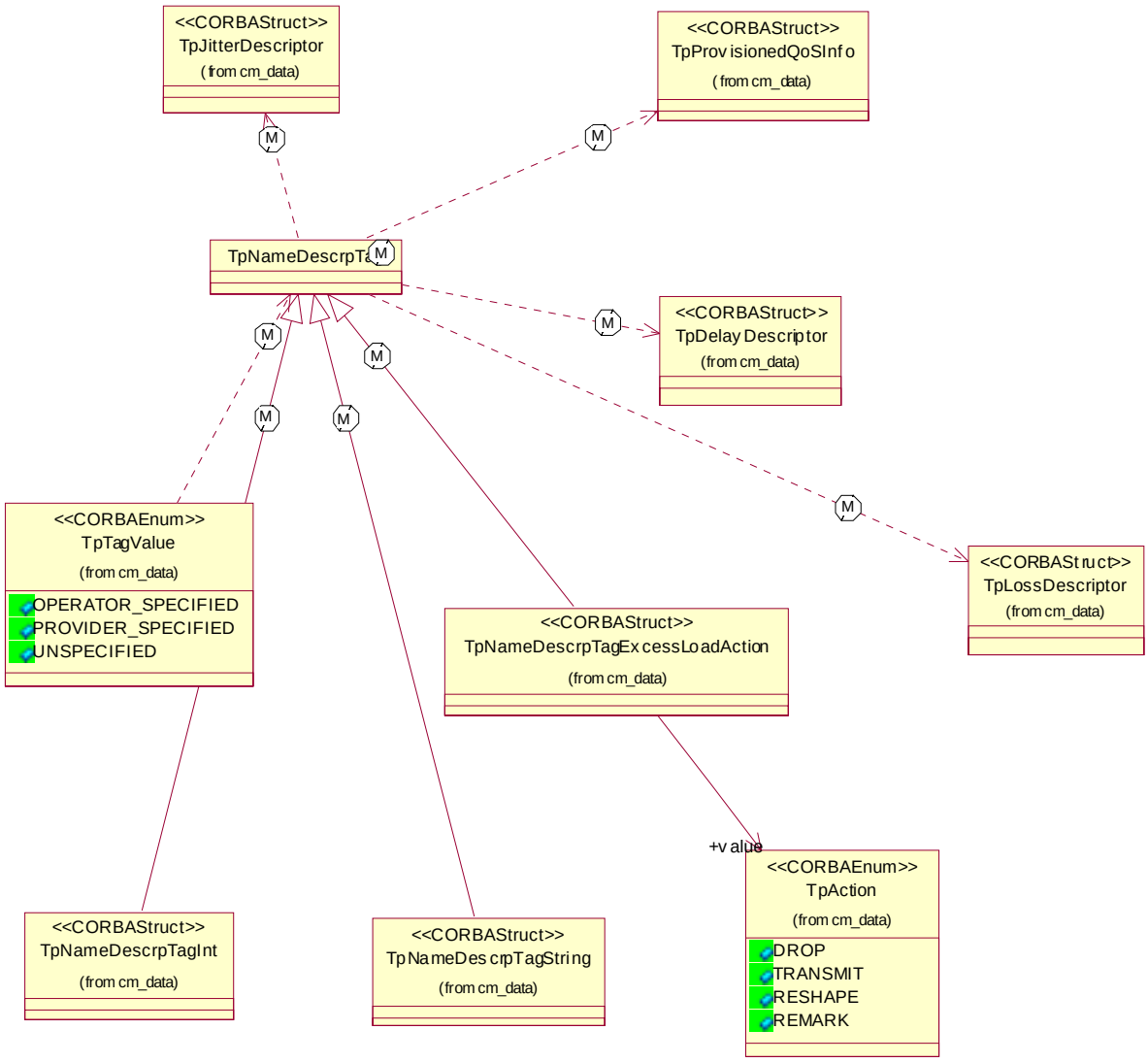


Figure 4: Provisioned QoS Data Types Classes

The figure below shows the Pipe QoS information data type classes. They include the load parameters, and the endpoint termination of the pipe. The individual values are integer, and for the directional type, either uni-directional traffic or bi-directional traffic is specified for the Pipe.

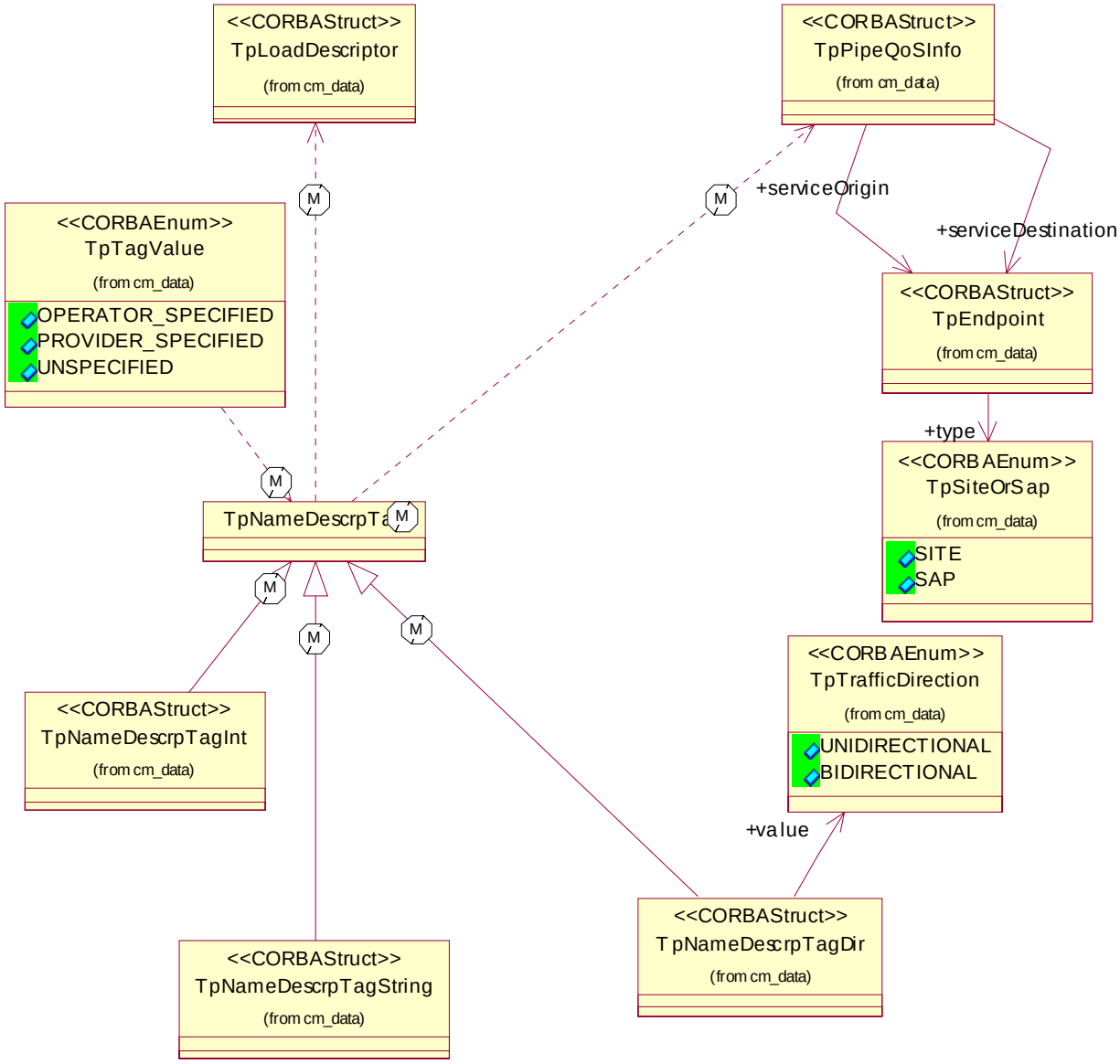


Figure 5: Pipe QoS Data Types

The figure below shows the classes that store information regarding the existing services already established between the provider and the enterprise. This includes the Virtual Private Network that is contained in the Enterprise Network Site class, and the Virtual Provisioned Network Class, and the associated Virtual Provisioned Pipe class. Also shown here is the information included for Pipe QoS, and the IP subnet information stored in these classes. However, the class that shows the information collected for Provisioned QoS is not shown here but in another class diagram below.

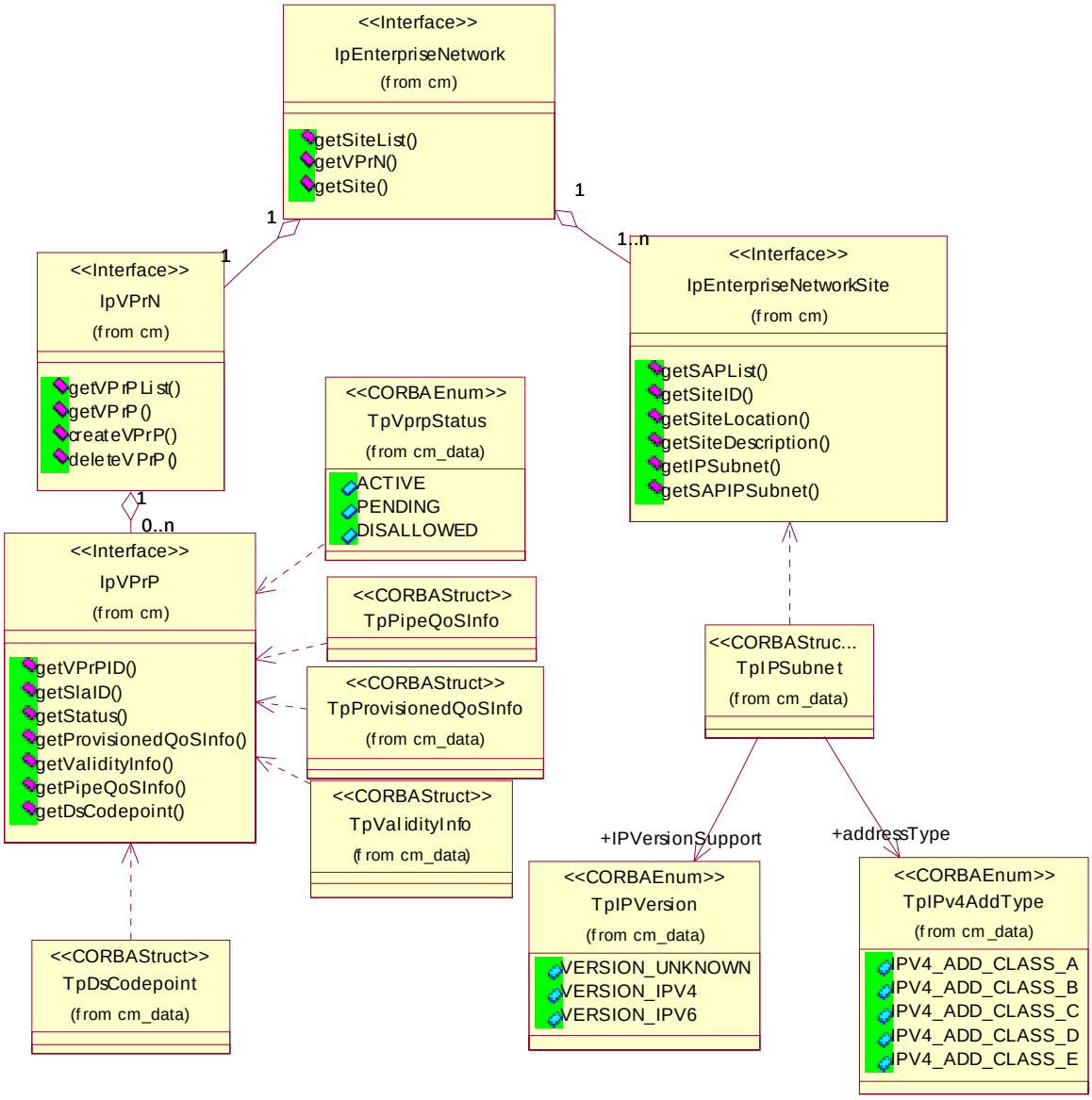


Figure 6: Existing VPN and VPrN services class diagram

The figure below shows the validity data type classes. They specify a window between two points in time, and repetitions events during a day, which days of the week, and which month of the year. They can be specified by the provider, to limit the service to certain time periods, or by the operator. The individual values are shown as subclasses to the Name Description Tag class.

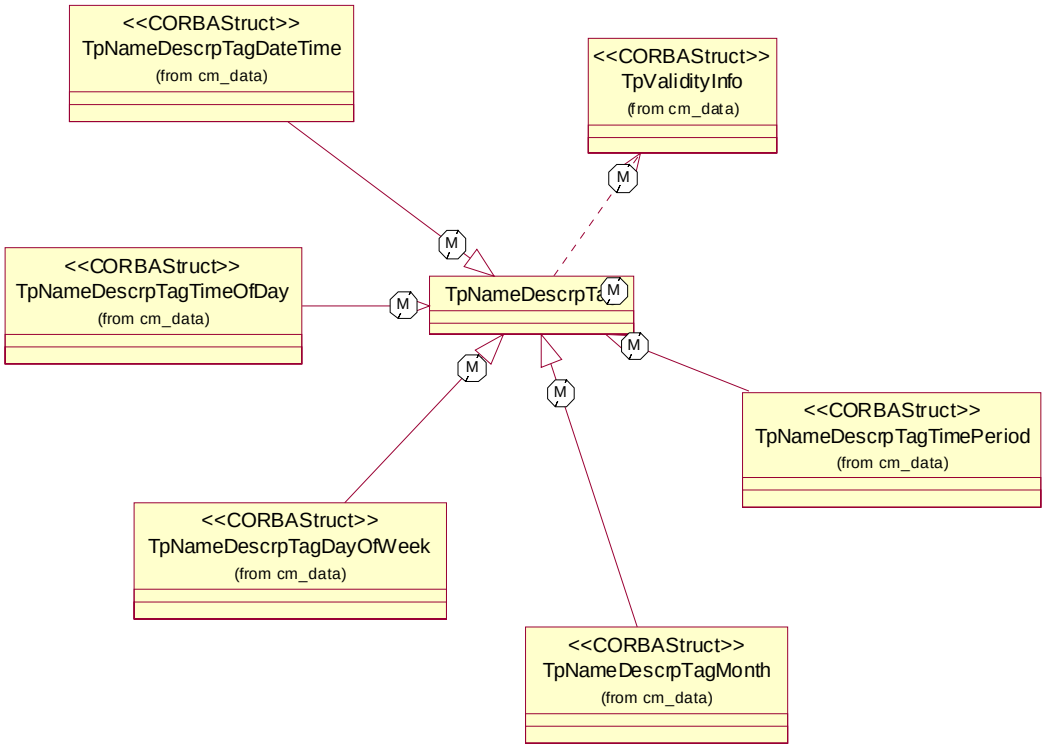


Figure 7: Validity Data Type Classes

7 The Service Interface Specifications

7.1 Interface Specification Format

This clause defines the interfaces, methods and parameters that form a part of the API specification. The Unified Modelling Language (UML) is used to specify the interface classes. The general format of an interface specification is described below.

7.1.1 Interface Class

This shows a UML interface class description of the methods supported by that interface, and the relevant parameters and types. The Service and Framework interfaces for enterprise-based client applications are denoted by classes with name Ip<name>. The callback interfaces to the applications are denoted by classes with name IpApp<name>. For the interfaces between a Service and the Framework, the Service interfaces are typically denoted by classes with name IpSvc<name>, while the Framework interfaces are denoted by classes with name IpFw<name>.

7.1.2 Method descriptions

Each method (API method "call") is described. Both synchronous and asynchronous methods are used in the API. Asynchronous methods are identified by a 'Req' suffix for a method request, and, if applicable, are served by asynchronous methods identified by either a 'Res' or 'Err' suffix for method results and errors, respectively. To handle responses and reports, the application or service developer must implement the relevant IpApp<name> or IpSvc<name> interfaces to provide the callback mechanism.

7.1.3 Parameter descriptions

Each method parameter and its possible values are described. Parameters described as 'in' represent those that must have a value when the method is called. Those described as 'out' are those that contain the return result of the method when the method returns.

7.1.4 State Model

If relevant, a state model is shown to illustrate the states of the objects that implement the described interface.

7.2 Base Interface

7.2.1 Interface Class IpInterface

All application, framework and service interfaces inherit from the following interface. This API Base Interface does not provide any additional methods.

<<Interface>> IpInterface

7.3 Service Interfaces

7.3.1 Overview

The Service Interfaces provide the interfaces into the capabilities of the underlying network - such as call control, user interaction, messaging, mobility and connectivity management.

The interfaces that are implemented by the services are denoted as 'Service Interface'. The corresponding interfaces that must be implemented by the application (e.g. for API callbacks) are denoted as 'Application Interface'.

7.4 Generic Service Interface

7.4.1 Interface Class IpService

Inherits from: IpInterface;

All service interfaces inherit from the following interface.

<<Interface>> IpService
setCallback (appInterface : in IpInterfaceRef) : void setCallbackWithSessionID (appInterface : in IpInterfaceRef, sessionID : in TpSessionID) : void

7.4.1.1 Method setCallback()

This method specifies the reference address of the callback interface that a service uses to invoke methods on the application. It is not allowed to invoke this method on an interface that uses SessionIDs.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

Raises

`TpCommonExceptions, P_INVALID_INTERFACE_TYPE`

7.4.1.2 Method setCallbackWithSessionID()

This method specifies the reference address of the application's callback interface that a service uses for interactions associated with a specific session ID: e.g. a specific call, or call leg. It is not allowed to invoke this method on an interface that does not use SessionIDs.

Parameters

`appInterface : in IpInterfaceRef`

Specifies a reference to the application interface, which is used for callbacks.

`sessionID : in TpSessionID`

Specifies the session for which the service can invoke the application's callback interface.

Raises

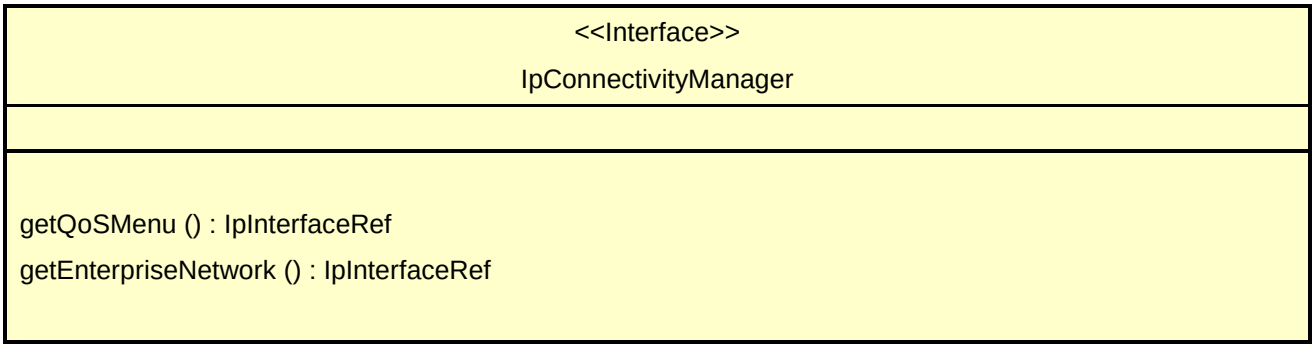
`TpCommonExceptions, P_INVALID_SESSION_ID, P_INVALID_INTERFACE_TYPE`

8 Connectivity Manager Interface Classes

8.1 Interface Class IpConnectivityManager

Inherits from: IpService.

The service Connectivity Manager Interface is the entry point to the Connectivity Manager service. After the enterprise operator client is authenticated and authorised, the client application discovers the Connectivity Manager interface, then the operator can use this interface to step through the process of provisioning a new VPrP. This interface has two methods, one to get the handle to the menu of QoS services offered by the provider, and the other one is a handle to the enterprise network interface that holds information about current services that the provider network delivers to the enterprise network.



8.1.1 Method getQoSMenu()

A client uses this method to get a reference to the QoS menu interface.

Returns menuRef : This parameter is a reference to the QoS menu interface. If no menu is found, P_UNKNOWN_MENU exception is raised.

Parameters
No Parameters were identified for this method.

Returns

IpInterfaceRef

Raises

TpCommonExceptions, P_UNKNOWN_MENU

8.1.2 Method getEnterpriseNetwork()

This method is used to get a handle to the enterprise network interface, which holds information regarding network services that are already provisioned for the enterprise network in the provider network.

Returns enterpriseNetworkRef : This parameter is a reference to the enterprise network interface. . If enterprise network is not found, P_UNKNOWN_ENTERPRISE_NETWORK exception is raised.

Parameters
No Parameters were identified for this method.

Returns

IpInterfaceRef

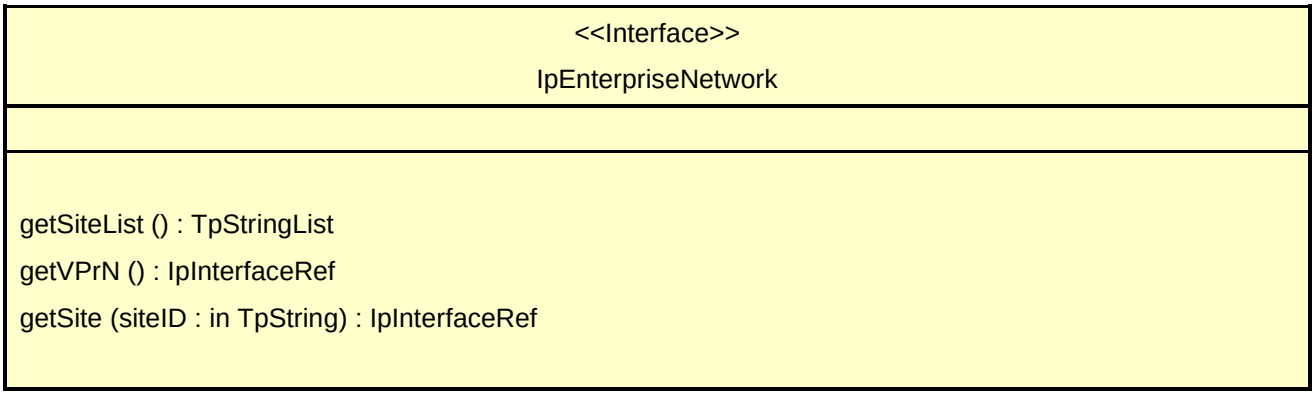
Raises

TpCommonExceptions, P_UNKNOWN_ENTERPRISE_NETWORK

8.2 Interface Class IpEnterpriseNetwork

Inherits from: IpService.

This interface stores enterprise network information maintained by the provider as it relates to the virtual private network service and the virtual provisioned network service that the enterprise had already established with the service provider network. The enterprise operator can only retrieve but not change the information stored with this interface. The methods of this interface enable the enterprise operator to obtain the handle to the interface that holds information regarding an existing VPrN, to list the sites connected to the VPN, and get the handle to a specific site interface that store information about the site.



8.2.1 Method getSiteList()

This method is used to get the list of enterprise network site IDs. These IDs identify the sites that are inter-connected through the provider network. These IDs were set when the VPN was provisioned in the provider network in the provider network.

Returns siteList : This parameter lists the site IDs (e.g., research, marketing, Middletown Building D5, London) of the enterprise network that are serviced by the provider network. If no site is found, then a P_UNKNOWN_SITES exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`TpStringList`

Raises

`TpCommonExceptions, P_UNKNOWN_SITES`

8.2.2 Method getVPrN()

This method is used to get a handle to the interface that holds information regarding a previously provisioned Virtual Private Network (VPrN).

Returns vPrNRef : This parameter is a handle to the VPrN interface that holds information about previously provisioned VPrN.

If no VPrN is found for this enterprise network, then a P_UNKNOWN_VPRN exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`IpInterfaceRef`

Raises

`TpCommonExceptions, P_UNKNOWN_VPRN`

8.2.3 Method getSite()

This method is used to get a handle to an interface that holds information about a specific site.

Returns siteRef : This parameter is a reference to the site interface.

Parameters

`siteID: in TpString`

This parameter is the ID given to a particular site. The ID is not assigned via OSA APIs, but previously when a new VPN (or VLL) is established for the enterprise on the provider network. These ID are typically names that refer to objects that are meaningful in the context of the enterprise network, such as: Marketing, New York, or Bulling 4. This site ID can be used as an endpoint of a provisioned virtual provisioned pipe (VPrP).

- If the string representation of the siteID does not obey the rules for site identification, then a P_ILLEGAL_SITE_ID exception is raised.
- If the site ID representation is legal but there is no site with this ID, then P_UNKNOWN_SITE_ID exception is raised.

Returns

`IpInterfaceRef`

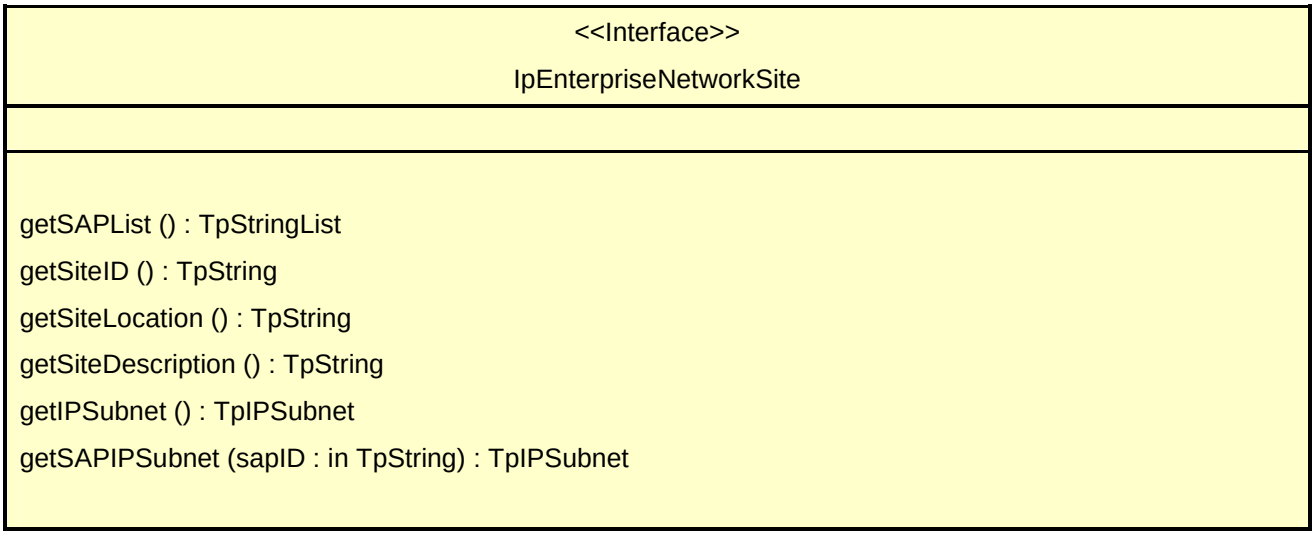
Raises

`TpCommonExceptions, P_ILLEGAL_SITE_ID, P_UNKNOWN_SITE_ID`

8.3 Interface Class IpEnterpriseNetworkSite

Inherits from: IpEnterpriseNetwork.

This interface stores enterprise network site information maintained by the provider.



8.3.1 Method getSAPList()

This method is used to get the list of SAP IDs of the enterprise VPN (i.e. on the provider network) that have previously been established for this site with the provider network.

Returns sapList : This parameter is a list of SAP IDs. This SAP ID can be used as an endpoint of a provisioned virtual provisioned pipe (VPrP).

If no SAPs are found for this site, then P_UNKNOWN_SAPS exception is raised.

Parameters

No Parameters were identified for this method.

Returns

`TpStringList`

Raises

`TpCommonExceptions, P_UNKNOWN_SAPS`

8.3.2 Method getSiteID()

This method is used to get the site ID for this site.

Returns siteID : This parameter holds the value for the site ID.

If no site ID is found for this site, then P_UNKNOWN_SITE_ID exception is raised.

Parameters
No Parameters were identified for this method.

Returns

TpString

Raises

TpCommonExceptions, P_UNKNOWN_SITE_ID

8.3.3 Method getSiteLocation()

This method is used to get the site location.

Returns siteLocation : This parameter holds the value for the site location.

If no site location is found for this site, then P_UNKNOWN_SITE_LOCATION exception is raised.

Parameters
No Parameters were identified for this method.

Returns

TpString

Raises

TpCommonExceptions, P_UNKNOWN_SITE_LOCATION

8.3.4 Method getSiteDescription()

This method is used to get the description associated with this site.

Returns siteDescription : This parameter is a string that holds the site description.

If no description is found for this site, then P_UNKNOWN_SITE_DESCRIPTION exception is raised.

Parameters
No Parameters were identified for this method.

Returns

TpString

Raises

TpCommonExceptions, P_UNKNOWN_SITE_DESCRIPTION

8.3.5 Method getIPSubnet()

This method is used to get IP subnet information for this site.

Returns ipSubnet : This parameter lists the subnet information.

If no IP Subnet information is found for this site, then a P_UNKNOWN_IPSUBNET exception is raised.

Parameters
No Parameters were identified for this method.

Returns

TpIPSubnet

Raises

TpCommonExceptions, P_UNKNOWN_IPSUBNET

8.3.6 Method getSAPIPSubnet()

This method is used to get the IP address of the SAP on the enterprise network.

Returns ipSubnet : This parameter holds the IP address information for the SAP. This TpIPSubnet data type follows the DMTF CIM specification for IP sub-net.

Parameters

sapID : in TpString

This parameter holds the IP address information for the SAP.
- If the string representation of the sapID does not obey the rules for site identification, then a P_ILLEGAL_SITE_ID exception is raised.
- If the site ID representation is legal but there is no site with this ID, then P_UNKNOWN_SAP exception is raised.
- If no IP Subnet information is found for this SAP, then a P_UNKNOWN_IPSUBNET exception is raised.

Returns

TpIPSubnet

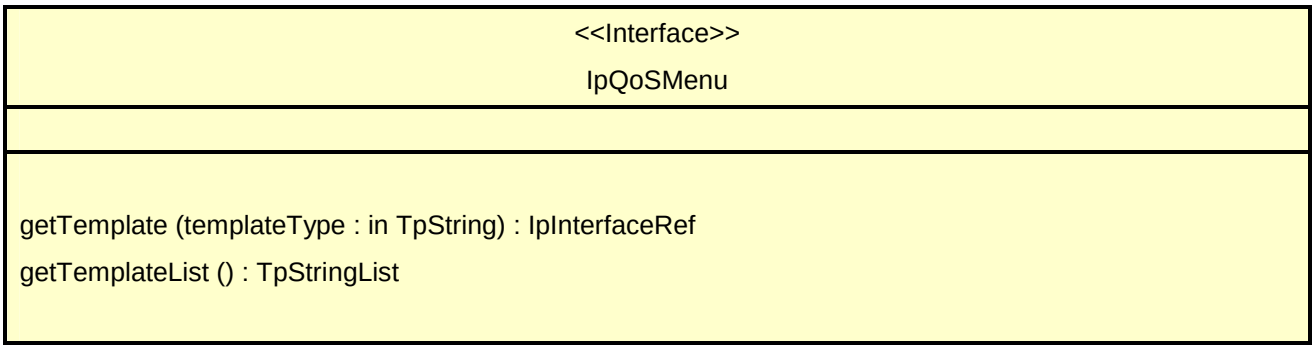
Raises

TpCommonExceptions, P_ILLEGAL_SITE_ID, P_UNKNOWN_SAP, P_UNKNOWN_IPSUBNET

8.4 Interface Class IpQoSMenu

Inherits from: IpService.

This interface holds the QoS menu offered by the provider. Each QoS service offered (e.g., Gold, Silver) is specified in a separate template. The template specifies the parameters and their default values from which the operator may choose to create a VPrP. When the operator asks for a specific template from the list of templates (getTemplate method), a temporary template interface is created. This temporary template interface holds all the parameters (e.g., all the Gold parameters) and their default values offered by the provider for this template.



8.4.1 Method getTemplate()

This method is used to get an interface reference to a specific template. The provider creates a temporary copy of the original template that contains all the QoS parameters for this template (e.g., Gold).

Returns templateRef : This parameter contains a reference to the template interface. Note that if the reference to this temporary template is lost, there is no way to recall it. To create a new temporary template this method has to be applied again, however, any values that were set in the old temporary template are lost.

Parameters

templateType : in TpString

This parameter contains template type.

Returns

IpInterfaceRef

Raises

TpCommonExceptions

8.4.2 Method getTemplateList()

This method is used to get a list of templates, each of which specifies a QoS service, such as Gold or Silver.

Returns templateList : This parameter contains a list of QoS service templates IDs.

If no templates are found, then a P_UNKNOWN_TEMPLATES exception is raised.

Parameters

No Parameters were identified for this method.

Returns

TpStringList

Raises

TpCommonExceptions, P_UNKNOWN_TEMPLATES

8.5 Interface Class IpQoSTemplate

Inherits from: IpService.

This interface provides access to a specific QoS template, such as Gold, offered by the provider. This interface provides get methods to discover the QoS service details, and set methods to set the requested values for a new VPrP. The template specifies the QoS parameters and their default values. Each service template parameter is tagged by the service provider to indicate one of the following:

-· Provider specified: the value cannot be modified for this template.

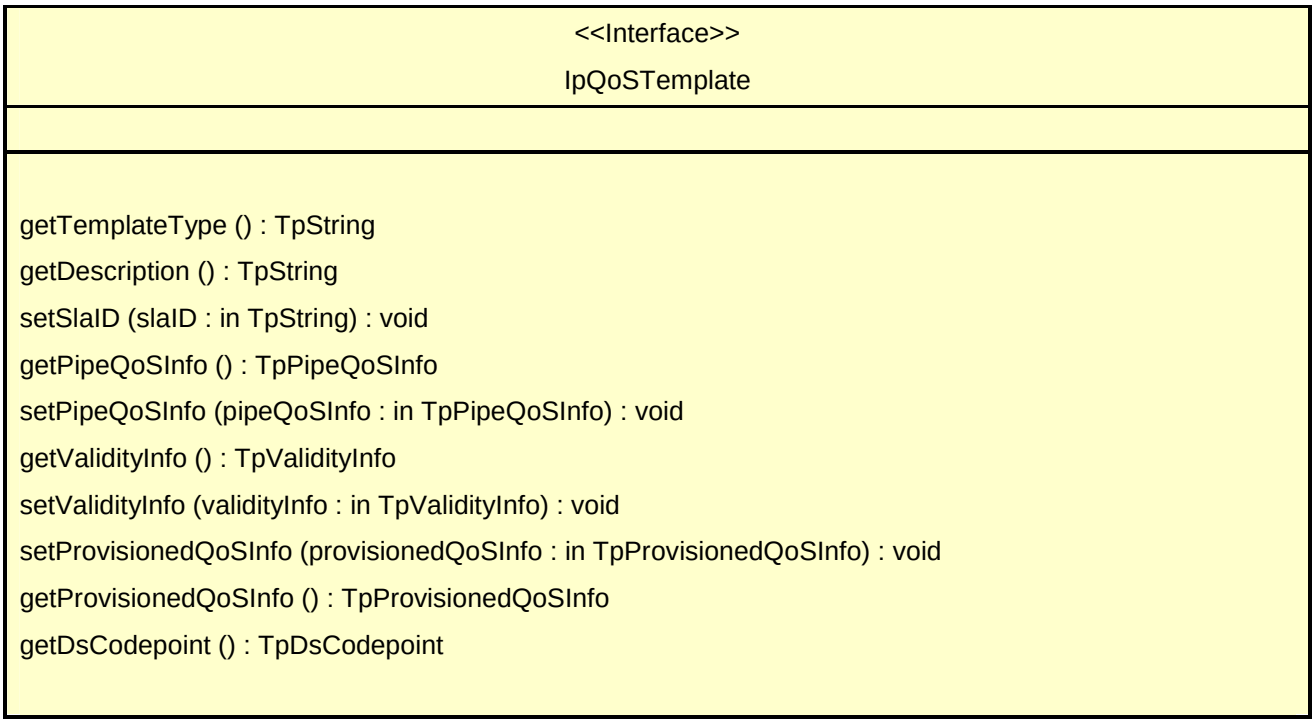
-· Enterprise operator specified: operator may change the default value set by the provider. The default value can be blank to indicate that there is no default value for this parameter, and the user can change it according to advice typically given by the description parameters.

-· Unspecified: the parameter is not used for this template, and the Enterprise operator cannot change it.

For example, maximum delay for a Gold template may be provider specified, while maximum bandwidth may be Enterprise operator specified, meaning that its values can be changed by the setProvisionedQoSInfo, or by setPipeQoSInfo methods. Guidance how to change the default values may be provided by the template description parameter.

The tag of a parameter cannot be changed by any set method of this interface, i.e., if a parameter is tagged unspecified, or provider specified, Enterprise operator cannot override the value of this tag to say operator specified.

This template is a temporary interface created as a copy of the original template that stores all the template parameters. The temporary interface is created with the getTemplate method of IpQoSMenu interface. The values passed to this template interface by the set methods replace (if permitted by the tags) the default values stored in this template interface, i.e., a get following a set method to this template interface will fetch the new values set by the Enterprise operator. Once a new VPrP is created by the create method in IpVPrN, the temporary interface might not be accessible anymore.



8.5.1 Method getTemplateType()

This method is used to get the template type, e.g., Gold.

Returns templateType : This parameter contains the template type.

If template type is not found, then P_UNKNOWN_TEMPLATE_TYPE exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`TpString`

Raises

`TpCommonExceptions, P_UNKNOWN_TEMPLATE_TYPE`

8.5.2 Method getDescription()

This method is used to get a description of the QoS service stored in this template interface. Connectivity manager APIs support default values set by the provider for each QoS parameter, i.e., a template (e.g., Gold template) may have a set of default values (e.g., a default value for minimum delay, a default value for maximum delay, etc.). If the network service provider allows (using the tags described above) the enterprise operator to change a specific default value, the provider can use this description to advise the user the conditions under which they can be changed, and the alternate values that can be used.

Returns description : This parameter contains a description of the service for this template and may also be used to convey any advice to the user such as what values can be selected instead of default values.

If the description is not found, then P_UNKNOWN_DESCRIPTION exception is raised. Note that if the description is found, but it contains no description, this should not raise the exception.

Parameters
No Parameters were identified for this method.

Returns

TpString

Raises

TpCommonExceptions, P_UNKNOWN_DESCRIPTION

8.5.3 Method setSlaID()

This method is used to store an existing service level agreement (SLA) identifier associated it with a specific VPrP. SLA ID is optional and is not required to be part of every VPrP. Each time this method is performed, the new value replaces the old value in the template.

Parameters

slaID : in TpString

This parameter contains the SLA ID. If the string representation of the SLA ID does not obey the rules for SLA identification, then a P_ILLEGAL_SLA_ID exception is raised.

Raises

TpCommonExceptions, P_ILLEGAL_SLA_ID

8.5.4 Method getPipeQoSInfo()

This method is used to get pipe QoS information consisting of load parameters, direction of the traffic, and the endpoint (SAP or site) of a virtual provisioned pipe offered by this template.

Returns pipeQoSInfo : This parameter includes the pipe QoS default parameters for this template. The endpoints are not specified in the getpipeQoS method. The directionality and load parameters are tagged, and can be set by the provider or left for the operator to be set.

· If no pipe QoS information is found, then P_UNKNOWN_PIPEQOSINFO exception is raised.

Parameters

No Parameters were identified for this method.

Returns

TpPipeQoSInfo

Raises

TpCommonExceptions, P_UNKNOWN_PIPEQOSINFO

8.5.5 Method setPipeQoSInfo()

This method is used to request pipe QoS parameters consisting of load parameters, direction of the traffic, and the endpoint (SAP or site) of the virtual provisioned pipe, as selected by the operator from the set of values offered by the provider. To modify any default value, the tag has to be set to OperatorSpecified. The parameters name, description, and tag are ignored with this method.

Parameters

pipeQoSInfo : in TpPipeQoSInfo

This parameter includes the virtual provisioned pipe information regarding the flow direction, the load on the endpoint of the pipe, and the load on the endpoints.

- If a parameter is tagged with providerSpecified, or unspecified, then the P_ILLEGAL_TAG exception is raised.
- If a value requested for a specific parameter by this method is not consistent with the advice given by the provider for choosing parameter values, the P_ILLEGAL_VALUE is raised. This is an optional exception that would be applied if the provider can verify consistency on the fly. Otherwise, the new VPrP request will be denied by setting its status flag to disallowed.
- If a combination of requested parameters is illegal, then P_ILLEGAL_COMBINATION is raised.

Raises

TpCommonExceptions, P_ILLEGAL_TAG, P_ILLEGAL_VALUE, P_ILLEGAL_COMBINATION

8.5.6 Method getValidityInfo()

The operator uses this method to get the default time period set by the provider for the template. Applying a logical AND operation of all the components of this parameter evaluates the valid period(s).

Returns validityInfo : This parameter provides the default validity information.

· If no validity information is found for this template, then P_UNKNOWN_VALIDITY_INFO exception is raised. Note that if the validity information is found and validitySpecified, is FALSE this exception is not raised.

Parameters

No Parameters were identified for this method.

Returns

TpValidityInfo

Raises

TpCommonExceptions, P_UNKNOWN_VALIDITY_INFO

8.5.7 Method setValidityInfo()

The operator uses this method to set the required time period for a new VPrP. The requested time and the default time set by the provider are all ANDed together to determine the final valid time period for this VPrP. Note that only those components that are tagged as operator specified can be set by the operator.

Parameters

validityInfo : in TpValidityInfo

This parameter provides the requested validity information for a new VPrP.

- If a parameter is tagged with providerSpecified, or unspecified, then the P_ILLEGAL_TAG exception is raised.
- If a value requested for a specific parameter by this method is not consistent with the advice given by the provider for choosing parameter values, the P_ILLEGAL_VALUE is raised. This is an optional exception that would be applied if the provider can verify consistency on the fly. Otherwise, the new VPrP request will be denied by setting its status flag to disallowed.
- If a combination of requested parameters is illegal, then P_ILLEGAL_COMBINATION is raised. For example, if the specified time duration is longer than 24 hours for a time-of-day parameter, or the integer value representing day-of-week or month-of-year is outside the permitted range.

Raises

TpCommonExceptions, P_ILLEGAL_TAG, P_ILLEGAL_VALUE, P_ILLEGAL_COMBINATION

8.5.8 Method setProvisionedQoSInfo()

The Enterprise operator uses this method to set the requested values for the QoS parameters. The values passed by this method replace the default values in the temporary template interface. Tag values associated with each parameter can be set only by the provider and cannot be changed by the operator. If tag values are included in this method, they should be ignored. Only those parameters that are tagged with the value operator specified can be modified using this method. With this method, the name and description parameters are ignored.

Parameters

`provisionedQoSInfo : in TpProvisionedQoSInfo`

This parameter consists of delay, loss, jitter, and exceed load action parameters.

· If a parameter from the delayParameters, lossParameters, jitterParameters, or excessLoadAction is tagged in the template with the values provider specified, or unspecified, then the P_ILLEGAL_TAG exception is raised.

· If a value requested for a specific parameter by this method is not consistent with the advice given by the provider for choosing parameter values, the P_ILLEGAL_VALUE is raised. This is an optional exception that would be applied if the provider can verify consistency on the fly. Otherwise, the new VPrP request will be denied by setting its status flag to disallowed.

· If a combination of requested parameters or parameter values is illegal, then P_ILLEGAL_COMBINATION is raised. An example of an illegal combination is maximum delay parameter and delay priority, as only one of the two can be used.

Raises

`TpCommonExceptions, P_ILLEGAL_TAG, P_ILLEGAL_VALUE, P_ILLEGAL_COMBINATION`

8.5.9 Method getProvisionedQoSInfo()

This method is used to get the default values associated with this template (e.g., delay default value, loss default value).

Returns provisionedQoSInfo:

This parameter consists of delay, loss, jitter, and exceed load action parameters.

· If no QoS information is found, then P_UNKNOWN_QOS_INFO exception is raised.

The Provisioned QoS Information has the following information

The Delay descriptor lists the delay default values, i.e., default values for mean delay, maximum delay, minimum delay, and delay priority. A provider may choose to tag any number of delay parameters as provider specified, Enterprise operator specified, or unspecified. For example, a Gold template may have a default value just for the mean delay, leaving the other parameters either unspecified, or some set to enterprise operator specified.

The loss descriptor lists the packet loss default values, i.e., mean loss, maximum loss, minimum loss, and loss priority. A provider may choose to tag any number of loss parameters as provider specified, Enterprise operator specified, or unspecified. For example, a Gold template may have a default value just for the mean loss only, leaving the other parameters either unspecified, or some be Enterprise operator specified.

The jitter descriptor lists the jitter default values (the delay measured between arriving packets), that is, mean jitter, maximum jitter, minimum jitter, and jitter priority. A provider may choose to tag any number of jitter parameters as provider specified Enterprise operator specified, or unspecified. For example, a Gold template may have a default value just for the mean jitter only, leaving the other parameters either unspecified, or some may be Enterprise operator specified.

The excess load action parameter specifies the policing treatment for traffic that exceeds the load parameters set for the virtual provisioned pipe. This policing function can take the following actions when the provider network detects that the traffic trying to enter the VPrP exceeds the load parameters specified in the pipe QoS loads parameters:

·Drop: drop packets (i.e., do not ever transmit them) that exceed the load traffic parameters that were set for the VPrP

- Transmit: transmit packets even though transmitting them will create a load in excess of the load traffic parameters that were set for the VPrP

- Reshape: reshape the entering traffic by trying to keep the packet (and not drop them yet) waiting for the entering traffic load to come down below the load conditions set for the VPrP, and if it does, transmit the packets then.

- Remark : remark the packet for a lower QoS service, then transmit them (i.e., transfer the packet through some other less demanding VPrP). This may result in increased packet loss (i.e., the excess packets may have now higher probability of being dropped before reaching their SAP or Site destination), or increased packet delay and / or packet jitter.

Parameters
No Parameters were identified for this method.

Returns

`TpProvisionedQoSInfo`

Raises

`TpCommonExceptions, P_UNKNOWN_QOS_INFO`

8.5.10 Method getDsCodepoint()

This method is used to get the DiffServ Codepoint of QoS service offered by the template.

Returns dsCodepoint : This parameter holds the DS Codepoint for the VPrP. It has two parameters: match and mask to enable the provider to locate the bit string in any location in the 6-bit long field. The actual Codepoint is a result of an AND operation bit by bit of the two parameters.

· If no DS Codepoint is found, then P_UNKNOWN_DSCODEPOINT exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`TpDsCodepoint`

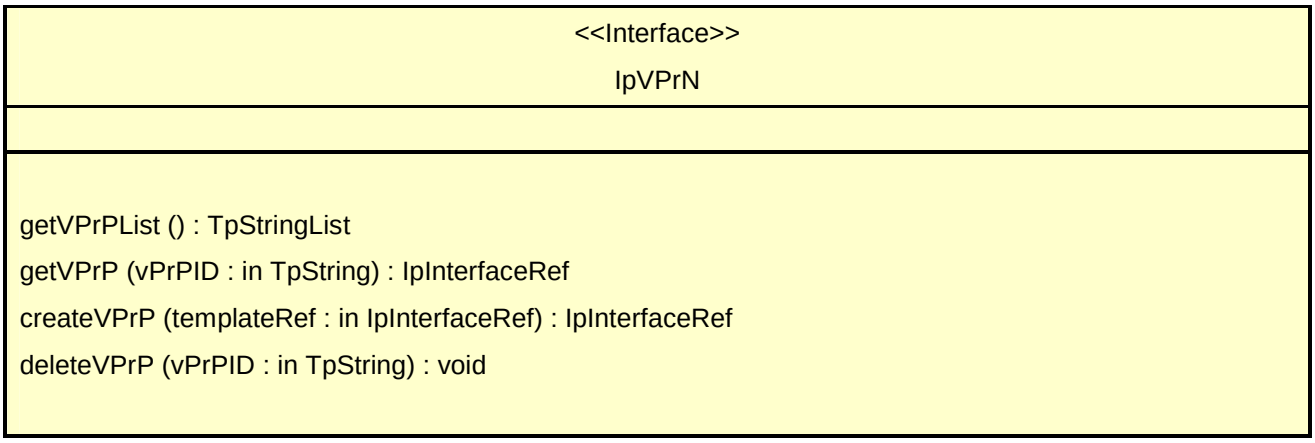
Raises

`TpCommonExceptions, P_UNKNOWN_DSCODEPOINT`

8.6 Interface Class IpVPrN

Inherits from: IpService.

The enterprise operator can create a new virtual provisioned pipe (VPrP) in an existing virtual private network (VPN) with this VPN interface. Such a pipe is extended between specific SAPs/sites. Each such pipe is associated with QoS parameters identified by a specific DiffServ Codepoint. A packet that arrives at the SAP/site with a specific Codepoint, is "directed" to the virtual provisioned pipe that supports the QoS parameters provisioned for this pipe. The collection of all the virtual provisioned pipes (VPrPs), provisioned within the enterprise VPN, constitutes the virtual provisioned network (VPrN). Enterprise operator can create new VPrPs and delete existing VPrP using this interface. This interface provides also methods to get the list of already provisioned VPrPs, and a handle to a specific VPrP interface that holds information for this VPrP.



8.6.1 Method getVPrPList()

This method is used to get the list of VPrP IDs for the already established virtual provisioned pipes for the enterprise network. Each pipe is assigned an ID at the provisioning of the pipe.

Returns vPrPList : This parameter lists the IDs of all the virtual provisioned pipes established in the virtual provisioned network.

· If no VPrP is found, then the P_UNKNOWN_VPRP exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`TpStringList`

Raises

`TpCommonExceptions, P_UNKNOWN_VPRP`

8.6.2 Method getVPrP()

This method is used to get a handle to the virtual provisioned pipe.

Returns vPrPRef : This parameter is the reference to a provision virtual provisioned pipe interface.

Parameters

`vPrPID : in TpString`

This parameter is virtual provisioned pipe ID, a unique ID across all VPrNs (of different enterprises) in the provider network. This ID is assigned by the provider when a new VPrP is created by the create method of this interface.

- If the string representation of the vPrPID does not obey the rules for site identification, then a P_ILLEGAL_VPRP_ID exception is raised.
- If the VPrP ID representation is legal but there is no VPrP with this ID, then the P_UNKNOWN_VPRP_ID exception is raised.
- Note that as soon a request to create a new VPrP (see IpVPrN interface) is submitted by the enterprise client, a new VPrP interface should be created by the provider. However, the provider might be in a situation where the evaluation of the request for a new VPrP has not been completed yet. In such a case, until the provider makes a decision, the status of the new VPrP should be set to Pending. The P_UNKNOWN_VPRP_ID exception should not be raised in this case.

Returns

`IpInterfaceRef`

Raises

TpCommonExceptions, P_ILLEGAL_VPRPID, P_UNKNOWN_VPRP_ID

8.6.3 Method createVPrP()

This method is used to create a new virtual provisioned pipe, which includes the pipe QoS information, the provisioned QoS information, the SLA ID, and the selected pairs of SAP/Sites. The method returns a reference to the new virtual provisioned pipe interface that is added to an existing VPrN. This VPrP needs to be accessed in order to find the status of the request to create a new VPrP. The status can have one of the following values: Pending, Active, or Disallowed. The enterprise operator should delete disallowed VPrPs. The provider may remove VPrPs with a disallowed status, if it stays in this status for some pre-agreed length of time.

Returns vPrPRef : This parameter is the handle to the new VPrP interface created as a response to the createVPrP method. The new VPrP interface may not include yet the decision of the provider to the request to create a new VPrP. However, if the request is granted, the status flag of the VPrP is set to Active. If the request is denied, the status flag is set to DISALLOWED. The status of the new VPrP is held in the status parameter of the VPrP, which should be Pending if the processing of the request has not been completed by the provider.

Parameters

templateRef : in IpInterfaceRef

This parameter is a reference to the template interface that holds all the requested QoS parameters for a new VPrP. The requested QoS parameter values, stored in the template interface, are used by the provider to provision a new VPrP for the enterprise network.

- If the reference representation of the templateRef does not obey the rules for reference values, then a P_ILLEGAL_REF_VALUE exception is raised.
- If the reference representation is legal but there is no interface with this reference, then P_UNKNOWN_INTERFACE exception is raised.
- If the one of the parameter values requested in the template is not consistent with default values set by the provider or the advice given in one of its description parameters, this is considered to be an inconsistent VPrP. The provider can deny a request for an inconsistent VPrP. The reason of the denial would be specified in the denial reason parameter for that VPrP. Since it is not required that the provider renders a decision in real time, no exception parameter is defined for this createVPrP method for requesting an inconsistent VPrP.

Returns

IpInterfaceRef

Raises

TpCommonExceptions, P_ILLEGAL_REF_VALUE, P_UNKNOWN_INTERFACE

8.6.4 Method deleteVPrP()

This method is used to delete a virtual provisioned pipe. A VPrP may have one of the following status values: Active, Pending, Disallowed. The reasons for deleting a VPrP may vary. Here are some examples. If a VPrP is active, the delete method is used when the VPrP is not needed anymore. If the VPrP is pending approval, one can still delete the VPrP. If the VPrP is disallowed, the VPrP should be deleted, as it does not serve any useful purpose anymore.

Parameters

vPrPID : in TpString

This parameter specifies the ID of the VPrP to be deleted. If the VPrP cannot be deleted, then P_CANT_DELETE_VPRP exception is raised. If the VPrP ID is not found, then P_UNKNOWN_VPRP_ID exception is raised.

Raises

TpCommonExceptions, P_CANT_DELETE_VPRP, P_UNKNOWN_VPRP_ID

8.7 Interface Class IpVPrP

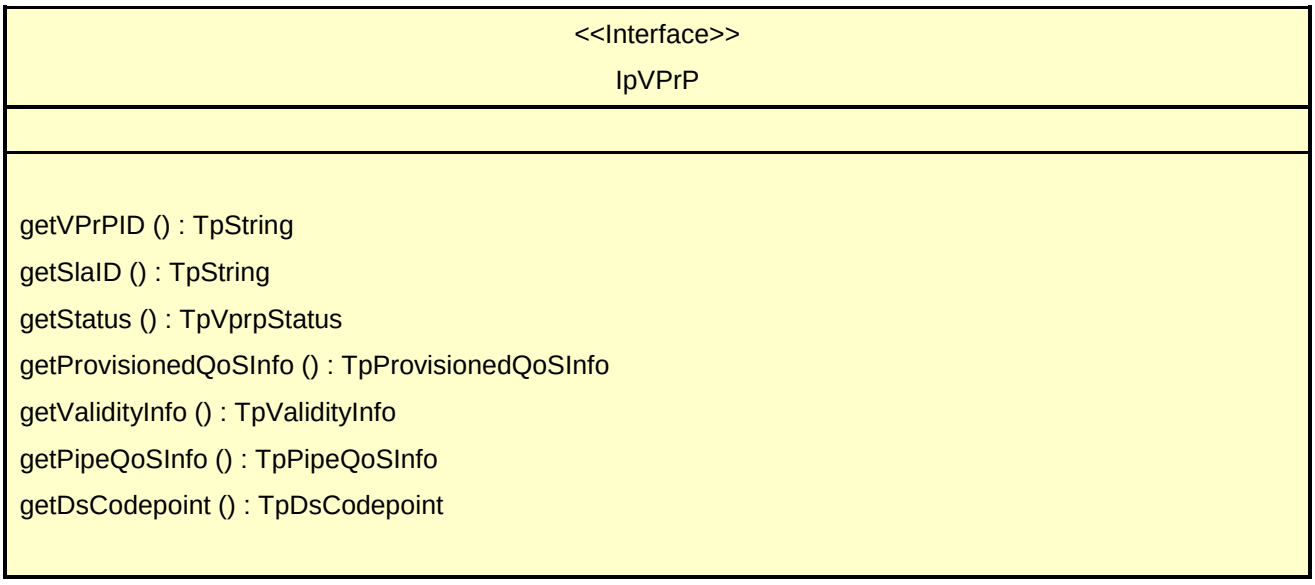
Inherits from: IpService.

The virtual provisioned pipe interface provides information on a VPrP whose status can be in one of the following states:

- Active: a previously established VPrP, which indicates that a previous request to create the VPrP was granted by the provider. Packet that belong to this VPrP and meet the validity time requirements, are admitted to the enterprise VPrN
- Pending: a request to create a new VPrP is still pending response from the provider, indicating that the provider is still processing the request to create a new VPrP. Packet that belong to this VPrP are not admitted to the enterprise VPrN
- Disallowed: a request to create a new VPrP was denied. A description parameter may include the reason for the denial. This is a disallowed VPrP and packet that belong to this VPrP are not admitted to the enterprise VPrN.

A VPrP is composed of the following elements, each of which provides the following Provisioned QoS measures:

Element type	Element	QoS measure
Endpoint	A SAP or a site	Pipe QoS information based on VPrP endpoints: load parameters, policing action. Applied to traffic entering the enterprise VPrN.
Packet header marking	TOS bits marked with DS Codepoint	Provisioned QoS information. One of the QoS levels established by the provider in its backbone network to be applied to the packets carrying this marking.
Time Validity requirement	A time that is applied to VPrP.	Packets are admitted to the VPrN only if they also meet validity time requirements.



8.7.1 Method getVPrPID()

This method is used to get the ID of the virtual provisioned pipe.

Returns vPrPID : This parameter is the ID of the virtual provisioned pipe.

· If no VPrP ID is found, then P_UNKNOWN_VPRP_ID exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`TpString`

Raises

`TpCommonExceptions, P_UNKNOWN_VPRP_ID`

8.7.2 Method getSlaID()

This method is used to get the ID of the service level agreement (SLA) if such was associated with the virtual provisioned pipe at the provisioning of the VPrP.

Returns slaID : This parameter is the identifier for the service level description.

· If no SLA ID is found, then P_UNKNOWN_SLA_ID exception is raised.

· Note that SLA ID is optional. If no SLA ID was entered when this VPrP was created, this exception will be raised.

Parameters
No Parameters were identified for this method.

Returns

`TpString`

Raises

`TpCommonExceptions, P_UNKNOWN_SLA_ID`

8.7.3 Method getStatus()

This method is used to get the status of the virtual provisioned pipe. It can be used to convey, for example, the status of an outstanding previously submitted provisioning request (which the provider could not verify in real time).

Returns status : This parameter is used to convey status of the virtual provisioned pipe. The semantics of each of these states is specified above.

· If status information is not found, Then P_UNKNOWN_STATUS exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`TpVprpStatus`

Raises

`TpCommonExceptions, P_UNKNOWN_STATUS`

8.7.4 Method getProvisionedQoSInfo()

This method is used to get the provisioned QoS information set for this virtual provisioned pipe. This method is the same method with the same parameters as in the QoS Template interface. The only difference is that the tag value is meaningless here. The values for an Active VPrP are the values provisioned in the provider network, and the values for a Pending VPrP are the requested values.

Returns provisionedQoSInfo : This parameter consists of delay, loss, jitter, and exceed load action parameters. The tag value (provider specified, operator specified, or unspecified) is used only with the template interface, and is meaningless for this VPrP interface. Delay priority, Loss priority, and Jitter priority are used to specify the priority of this VPrP relative to other VPrPs, instead of specifying explicit values for these parameters.

· If no QoS information is found, then P_UNKNOWN_QOS_INFO exception is raised.

Parameters
No Parameters were identified for this method.

Returns

TpProvisionedQoSInfo

Raises

TpCommonExceptions, P_UNKNOWN_QOS_INFO

8.7.5 Method getValidityInfo()

This method is used to get the time period for which the VPrP is valid. For the VPrP to be included in the VPrN, the VPrP has to be in active status and valid.

Returns validityInfo : This parameter defines the time periods for which this VPrP is valid. The valid periods are evaluated by applying a logical AND operation of all the components of this parameter, without regard who specified the valid time (i.e., specified by operator, or provider).

· If no validity information is found for this VPrP, then P_UNKNOWN_VALIDITY_INFO exception is raised. Note that if the validity information is found and validitySpecified is FALSE this exception is not raised.

Parameters
No Parameters were identified for this method.

Returns

TpValidityInfo

Raises

TpCommonExceptions, P_UNKNOWN_VALIDITY_INFO

8.7.6 Method getPipeQoSInfo()

This method is used to get and set the pipe QoS information for this VPrP. For an Active VPrP the values of these parameters are provisioned in the provider network. For a Pending VPrP, the values are the requested values. The tag value is meaningless for this interface.

Returns pipeQoSInfo : This parameter consists of load parameters, direction of the traffic, and the endpoints (SAP or site) of the virtual provisioned pipe. This data type is defined in the QoS Template.

If no pipe QoS information is found, then P_UNKNOWN_PIPEQOSINFO exception is raised.

Parameters
No Parameters were identified for this method.

Returns

TpPipeQoSInfo

Raises

TpCommonExceptions, P_UNKNOWN_PIPEQOSINFO

8.7.7 Method getDsCodepoint()

This method is used to get the DiffServ Codepoint of the virtual provisioned pipe. This DS Codepoint is populated in the TOS bits of the packet header to identify to the service provider network the specific VPrP to service this packet entering the provider network.

Returns dsCodepoint : This parameter holds the DS Codepoint for the VPrP. It has two parameters: match and mask to enable the provider to locate the bit string in any location in the 6-bit long field. The actual Codepoint is a result of an AND operation bit by bit of the two parameters.

· If no DS Codepoint is found, then P_UNKNOWN_DSCODEPOINT exception is raised.

Parameters
No Parameters were identified for this method.

Returns

`TpDsCodepoint`

Raises

`TpCommonExceptions, P_UNKNOWN_DSCODEPOINT`

9 State Transition Diagrams

There are no State Transition Diagrams for the Connectivity Manager SCF.

10 Data Definitions

Data type specifications can be found below. There is an extensive use of a specific data type for connectivity manager. This data type is discussed here to clarify how different interfaces and methods use it.

The data type `TpNameDescrpTag` is not explicitly specified (however, it is shown in the class diagrams document) as it is inherited by various other data types, such as `TpNameDescrpTagDateTime`, `TpNameDescrpTagdayofWeek`, `TpNameDescrpTagint`, `TpNameDescrpTagString`, and so forth. The `TpNameDescrpTag` class includes three parameters that are inherited by all of its subclasses data types: The `name` parameter, the `description` parameter, and the `tag` parameter. These parameters were defined for the `get` method in the template interface, and were reused by other methods, with some semantic modifications. The three parameters are:

- The `name` parameter which names the parameter in a template.
- The `description` parameter describes the parameter as it relates to the QoS service, and could add explanations and restrictions regarding the use of a parameter, such as what values can be selected by the operator for the parameter in a given template.
- The `tag` parameter has the following values and interpretation:
  - `Provider specified`: the operator cannot modify the value of this parameter as set in the template interface.
  - `Operator specified`: enterprise operator may change the default value set by the provider in the template. The default value can be blank to indicate that there is no default value for this parameter. The operator may change it according to advice, if provided in the description parameters.
  - `Unspecified`: the parameter is not used for this template, and the enterprise operator cannot change it.

Various interfaces and methods reuse these data types, since they require only a slight semantic modification while reusing the entire structure. The following specifies the modified semantics as used by the various methods.

For the QoS template interface and all the `set` methods that use the `TpNameDescrpTag` data type, the tag values should be ignored by the CM implementation, as only the provider sets the tag value in all cases, and the operator cannot change it. For any parameter that is set to `operator specified`, the `set` method can modify the values set in the template, but not the associated tag. If the tag is set to `provider specified`, or `unspecified`, and the operator tries to modify it with a `set` method, the P_NOT_OPERATOR_SPECIFIED_VALUE exception is raised. However, the `description` can be modified by the `set` method if the tag is `operator specified`. The operator can use this feature to enter notes that would be later shown when retrieving VPrP information. However, the `name` parameter can not be changed as it might be unique in the provider's network.

For the VPrP interface, the tag value is irrelevant for all the `get` methods and may be ignored by the operator client.

All data types referenced but not defined in this clause are common data definitions which may be found in ES 202 915-2.

10.1 Connectivity Manager Data Types

10.1.1 TpIPSubnet

Defines a `Sequence of Data Elements.`

Sequence Element Name	Sequence Element Type	Description
subnetNumber	TpString	IP address, IPv4 example 2.3.4.15
subnetMask	TpString	IPv4 mask example, 255.255.255.0
addressType	TpIPv4AddType	Class for the Ipv4 address format
IPVersionSupport	TpIPVersion	Version supported
NOTE: The TpIPSubnet data type follows the DMTF CIM specification for IP sub-net.		

10.1.2 TpIPv4AddType

Name	Value	Description
IPV4_ADD_CLASS_A	0	Address Class A
IPV4_ADD_CLASS_B	1	Address Class b
IPV4_ADD_CLASS_C	2	Address Class c
IPV4_ADD_CLASS_D	3	Address Class d
IPV4_ADD_CLASS_E	4	Address Class e

10.1.3 TpIPVersion

Name	Value	Description
VERSION_UNKNOWN	0	Unknown IP addressing format
VERSION_IPV4	1	IPv4 addressing format
VERSION_IPV6	2	IPv6 addressing format

10.1.4 TpVprpStatus

Name	Value	Description
ACTIVE	0	A VPrP status indicating that this VPrP is a previously established VPrP. Also means that a previous request to create the VPrP was granted by the provider. Packet that belong to this VPrP and meet the validity time requirements, are admitted to the enterprise VPrN.
PENDING	1	A VPrP status indicating that this VPrP is a request to create a new VPrP, and it is still pending a response from the provider .i.e. the provider is still processing the request to create a this new VPrP. Packet that belong to this VPrP are not admitted to the enterprise VPrN.
DISALLOWED	2	A VPrP status indicating that this VPrP is a request to create a new VPrP, however, the request was denied. A description parameter may include the reason for the denial. This is an disallowed VPrP and packet that belong to this VPrP are not admitted to the enterprise VPrN.

10.1.5 TpDsCodepoint

Sequence Element Name	Sequence Element Type	Description
match	TpString	Marking significant bits. 6-bit long. Valid characters are 0 and 1.
mask	TpString	Identifies significant part (1s) of for marking Codepoint. 6-bit long. Valid characters are 0 and 1.

10.1.6 TpProvisionedQoSInfo

Sequence Element Name	Sequence Element Type	Description
delayDescriptor	TpDelayDescriptor	Delay parameters
lossDescriptor	TpLossDescriptor	Loss parameters
jitterDescriptor	TpJitterDescriptor	Jitter parameters
excessLoadAction	TpNameDescrpTagExcessLoadAction	Excess load action parameters
description	TpNameDescrpTagString	Operator can add text using the Set method, if tag is Operator Specified

10.1.7 TpDelayDescriptor

Sequence Element Name	Sequence Element Type	Description
meanDelay	TpNameDescrpTagInt	in milliseconds
measurementPeriod	TpNameDescrpTagInt	in milliseconds
maxDelay	TpNameDescrpTagInt	in milliseconds
minDelay	TpNameDescrpTagInt	in milliseconds
delayPriority	TpNameDescrpTagInt	higher value indicates higher priority
description	TpNameDescrpTagString	Operator can add text using the Set method, if tag is Operator Specified

10.1.8 TpLossDescriptor

Sequence Element Name	Sequence Element Type	Description
meanLoss	TpNameDescrpTagInt	per this many packets, one packet is lost
measurementPeriod	TpNameDescrpTagInt	in milliseconds
maxLoss	TpNameDescrpTagInt	per this many packets, one packet is lost
minLoss	TpNameDescrpTagInt	per this many packets, one packet is lost
lossPriority	TpNameDescrpTagInt	higher value indicates higher priority
description	TpNameDescrpTagString	Operator can add text using the Set method, if tag is Operator Specified.

10.1.9 TpJitterDescriptor

Sequence Element Name	Sequence Element Type	Description
meanJitter	TpNameDescrpTagInt	in milliseconds.
measurementPeriod	TpNameDescrpTagInt	in milliseconds.
maxJitter	TpNameDescrpTagSInt	in milliseconds.
minJitter	TpNameDescrpTagInt	in milliseconds.
jitterPriority	TpNameDescrpTagInt	higher value indicates higher priority.
description	TpNameDescrpTagString	Operator can add text using the Set method, if tag is Operator Specified.

10.1.10 TpNameDescrpTagInt

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider.
description	TpString	Description set by provider.
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpInt32	Can be set by operator or provider, depending on the tag value.

10.1.11 TpNameDescrpTagString

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider.
description	TpString	Description set by provider
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpString	Can be set by operator or provider, depending on the tag value.

10.1.12 TpTagValue

Name	Value	Description
PROVIDER_SPECIFIED	0	The operator cannot modify the value of this parameter as set in the template interface.
OPERATOR_SPECIFIED	1	Enterprise operator may change the default value set by the provider in the template. The default value can be blank to indicate that there is no default value for this parameter. The operator may change it according with advice, if provided in the description parameters.
UNSPECIFIED	2	Parameter is not used for this template, and the enterprise operator cannot change it.

10.1.13 TpNameDescrpTagExcessLoadAction

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider.
description	TpString	Description set by provider.
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpAction	Can be set by operator or provider, depending on the tag value.

10.1.14 TpAction

Name	Value	Description
DROP	0	Drop packets (i.e. do not ever transmit them) that exceed the load traffic parameters that were set for the VPrP
TRANSMIT	1	Transmit packets even though transmitting them will create a load in excess of the load traffic parameters that were set for the VPrP
RESHAPE	2	Reshape the entering traffic by trying to keep the packet (and not drop them yet) waiting for the entering traffic load to come down below the load conditions set for the VPrP, and if it does, transmit the packets then.
REMARK	3	Remark the packet for a lower QoS service, then transmit them (i.e. transfer the packet through some other less demanding VPrP). This may result in increased packet loss (i.e. the excess packets may have now higher probability of being dropped before reaching their SAP or Site destination), or increased packet delay and / or packet jitter

10.1.15 TpPipeQoSInfo

Sequence Element Name	Sequence Element Type	Description
directionality	TpNameDescrpTagDir	Specifies whether traffic is uni- or bi-directional.
serviceOrigin	TpEndpoint	
serviceDestination	TpEndpoint	
forwardLoad	TpLoadDescriptor	Traffic flowing from service origin to service destination.
reverseLoad	TpLoadDescriptor	Traffic flowing from service destination to service source.
description	TpNameDescrpTagString	

10.1.16 TpNameDescrpTagDir

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider.
description	TpString	Description set by provider.
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpTrafficDirection	Can be set by operator or provider, depending on the tag value.

10.1.17 TpTrafficDirection

Name	Value	Description
UNIDIRECTIONAL	0	The traffic for this VPrP is unidirectional.
BIDIRECTIONAL	1	The traffic for this VPrP is bi-directional.

10.1.18 TpEndpoint

Sequence Element Name	Sequence Element Type	Description
type	TpSiteOrSap	Specifies whether the endpoint is a site or a SAP.
id	TpString	Endpoint name.

10.1.19 TpSiteOrSap

Name	Value	Description
SITE	0	Endpoint is a site
SAP	1	Endpoint is a SAP

10.1.20 TpLoadDescriptor

Sequence Element Name	Sequence Element Type	Description
meanBandwidth	TpNameDescrpTagInt	Bytes per second
measurmentInterval	TpNameDescrpTagInt	milliseconds
maxBandwidth	TpNameDescrpTagInt	Bytes per second
minBandwidth	TpNameDescrpTagInt	Bytes per second
bandwidthShare	TpNameDescrpTagInt	1/100 of a percent, e.g. 500 is 0,5 %
bandwidthWeight	TpNameDescrpTagInt	
burstSize	TpNameDescrpTagInt	Bytes
description	TpNameDescrpTagString	

10.1.21 TpValidityInfo

Sequence Element Name	Sequence Element Type	Description
validFrom	TpNameDescrpTagDateTime	valid start time
validPeriod	TpNameDescrpTagTimePeriod	valid Duration
validDailyFrom	TpNameDescrpTagTimeOfDay	daily start time
validDailyPeriod	TpNameDescrpTagTimePeriod	Daily Duration
validDayOfWeek	TpNameDescrpTagDayOfWeek	days of the week
validMonth	TpNameDescrpTagMonth	months in a year
description	TpNameDescrpTagString	description

Specifies the validity period for a VPrP

10.1.22 TpNameDescrpTagDateTime

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider.
description	TpString	Description set by provider.
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpDateAndTime	Can be set by operator or provider, depending on the tag value. The TpDateAndTime data type is a common data type in accordance with ISO 8601.

Specifies a date and the time on that day for a valid period to start.

10.1.23 TpNameDescrpTagTimePeriod

Sequence Element Name	Sequence Element Type	Description
duration	TpInt32	Time measured from a start time specified by another parameter for the validity information. Measured in Seconds.

10.1.24 TpNameDescrpTagTimeOfDay

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider.
description	TpString	Description set by provider.
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpString	String with the following format HH:MM. HH two digit hour, and MM two digit minutes. Time of the day to be applied to validity information provided by other validity parameters.

Specifies a time in a day, where the day is specified by another validity parameter. For example, if validity information includes Mondays and Thursday of the week, this daily time applies to these two days. The valid time window will start on Mondays and Thursday s at the time specified by this parameter.

10.1.25 TpNameDescrpTagDayOfWeek

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider.
description	TpString	Description set by provider.
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpInt32	Day of the week to be applied to validity information provided by other validity parameters. Next table below shows the value assigned to each day. The value for this parameter can be in the range of 1 to 127.

The value parameter specifies information regarding the days of the week for the validity time period. For multiple days, this parameter holds the total value of the days, see examples below.

In the following table the value corresponding to the valid days is obtained by adding the values for each day. For example:

1 = Mondays

2 = Tuesdays

3 = 2 + 1 = Mondays and Tuesdays

4 = Wednesdays

5 = 4 + 1 = Mondays and Wednesdays

96 = 32 + 64 = Saturday + Sunday = weekend.

Values for Days Of Week

Day	Value
Monday	1
Tuesday	2
Wednesday	4
Thursday	8
Friday	16
Saturday	32
Sunday	64

10.1.26 TpNameDescrpTagMonth

Sequence Element Name	Sequence Element Type	Description
name	TpString	Name set by provider
description	TpString	Description set by provider
tag	TpTagValue	Tag set by provider. Cannot be overwritten by operator.
value	TpInt32	Months of the year to be applied to validity information provided by other validity parameters. Next table below shows the value assigned to each Month. The value for this parameter must be in the range of 1 to 4095.

In the following table the value corresponding to the valid month is obtained by adding the values for each valid month. For example:

1 = January

2 = February

3 = 2 + 1 = January and February

4 = March

5 = 4 + 1 = March and January

Values for Months of Year

Month	Sequence Element Type
January	1
February	2
March	4
April	8
May	16
June	32
July	64
August	128
September	256
October	512
November	1024
December	2048

11 Exception Classes

The following are the list of exception classes which are used in this interface of the API.

Name	Description
P_CANT_DELETE_VPRP	Virtual Provisioned Pipe cannot be deleted
P_ILLEGAL_COMBINATION	Illegal combination of requested parameters
P_ILLEGAL_REF_VALUE	Illegal Reference Value
P_ILLEGAL_SITE_ID	Illegal Enterprise Network Site ID
P_ILLEGAL_SLA_ID	Illegal format for Service Level Agreement ID
P_ILLEGAL_TAG	Illegal or unspecified Tag used
P_ILLEGAL_VALUE	Value of parameter is illegal
P_ILLEGAL_VPRPID	Attempt to lock an already locked mailbox
P_UNKNOWN_DESCRIPTION	No description field found
P_UNKNOWN_DSCODEPOINT	No DiffServ Codepoint found
P_UNKNOWN_ENTERPRISE_NETWORK	No Enterprise Network found
P_UNKNOWN_INTERFACE	A negative number of properties was requested
P_UNKNOWN_IPSUBNET	No IP Subnet found for this site
P_UNKNOWN_MENU	No menu found / unknown menu
P_UNKNOWN_PIPEQOSINFO	No PIPE QoS information found
P_UNKNOWN_QOS_INFO	No QoS information found
P_UNKNOWN_SAP	No site found with the specified SAP ID
P_UNKNOWN_SAPS	No Service Access Point found
P_UNKNOWN_SITE_DESCRIPTION	No site description found for this site
P_UNKNOWN_SITE_ID	No Enterprise Network Site with this ID found
P_UNKNOWN_SITE_LOCATION	No site location found for this site
P_UNKNOWN_SITES	No enterprise network site found
P_UNKNOWN_SLA_ID	No SLA ID found
P_UNKNOWN_STATUS	Status information not found
P_UNKNOWN_TEMPLATE_TYPE	Template Type not found
P_UNKNOWN_TEMPLATES	No Templates found
P_UNKNOWN_VALIDITY_INFO	No Validity Information found
P_UNKNOWN_VPRN	No Virtual Private Network found
P_UNKNOWN_VPRP	No Virtual Provisioned Pipe found
P_UNKNOWN_VPRP_ID	Illegal VPrP ID

Each exception class contains the following structure:

Structure Element Name	Structure Element Type	Structure Element Description
ExtraInformation	TpString	Carries extra information to help identify the source of the exception, e.g. a parameter name.

Annex A (normative): OMG IDL Description of Connectivity Manager SCF

The OMG IDL representation of this interface specification is contained in a text file (cm.idl) contained in archive es_20291510IDL.zip.

This archive can be found in es_20291510v010401m0.zip which accompanies the present document.

Annex B (informative): W3C WSDL Description of Connectivity Manager SCF

Significant changes have occurred in Web Services technologies and understanding of how to best apply Web Services as a realisation of OSA. These changes are not reflected and therefore this realisation is removed. A future activity may provide a replacement for the content of this annex, reflective of current technology and usage expected.

Annex C (informative): Java™ API Description of the Connectivity Management SCF

The Java™ API realisation of this interface specification is produced in accordance with the Java™ Realisation rules defined in ES 202 915-1. These rules aim to deliver for Java™, a developer API, provided as a realisation, supporting a Java™ API that represents the UML specifications. The rules support the production of both J2SE™ and J2EE™ versions of the API from the common UML specifications.

The J2SE™ representation of this interface specification is provided as Java™ Code, contained in archive 20291510J2SE.zip.

The J2EE™ representation of this interface specification is provided as Java™ Code, contained in archive 20291510J2EE.zip.

Both these archives can be found in es_20291510v010401m0.zip which accompanies the present document.

Annex D (informative): Record of changes

The following is a list of the changes made to the present document for each release. The list contains the names of all changed, deprecated, added or removed items in the specifications and not the actual changes. Any type of change information that is important to the reader is put in the final clause of this annex.

Changes are specified as changes to the prior major release, but every minor release will have its own part of the table allowing the reader to know when the actual change was made.

D.1 Interfaces

D.1.1 New

Identifier	Comments
Interfaces added in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Interfaces added in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Interfaces added in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Interfaces added in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.1.2 Deprecated

Identifier	Comments
Interfaces deprecated in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Interfaces deprecated in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Interfaces deprecated in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Interfaces deprecated in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.1.3 Removed

Identifier	Comments
Interfaces removed in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Interfaces removed in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Interfaces removed in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Interfaces removed in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.2 Methods

D.2.1 New

Identifier	Comments
Methods added in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Methods added in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Methods added in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Methods added in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.2.2 Deprecated

Identifier	Comments
Methods deprecated in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Methods deprecated in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Methods deprecated in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Methods deprecated in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.2.3 Modified

Identifier	Comments
Methods modified in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Methods modified in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Methods modified in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Methods modified in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.2.4 Removed

Identifier	Comments
Methods removed in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Methods removed in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Methods removed in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Methods removed in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.3 Data Definitions

D.3.1 New

Identifier	Comments
Data Definitions added in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Data Definitions added in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Data Definitions added in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Data Definitions added in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.3.2 Modified

Identifier	Comments
Data Definitions modified in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Data Definitions modified in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
TpNameDescrpTagMonth	Elements re-ordered in IDL to match documentation
Data Definitions modified in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Data Definitions modified in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.3.3 Removed

Identifier	Comments
Data Definitions removed in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
TpStringList	Moved to Common Data Part 2
Data Definitions removed in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Data Definitions removed in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Data Definitions removed in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.4 Service Properties

D.4.1 New

Identifier	Comments
Service Properties added in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Service Properties added in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Service Properties added in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Service Properties added in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.4.2 Deprecated

Identifier	Comments
Service Properties deprecated in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Service Properties deprecated in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Service Properties deprecated in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Service Properties deprecated in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.4.3 Modified

Identifier	Comments
Service Properties modified in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Service Properties modified in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Service Properties modified in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Service Properties modified in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.4.4 Removed

Identifier	Comments
Service Properties removed in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Service Properties removed in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Service Properties removed in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Service Properties removed in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.5 Exceptions

D.5.1 New

Identifier	Comments
Exceptions added in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Exceptions added in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Exceptions added in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Exceptions added in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.5.2 Modified

Identifier	Comments
Exceptions modified in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Exceptions modified in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Exceptions modified in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Exceptions modified in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.5.3 Removed

Identifier	Comments
Exceptions removed in ES 202 915-10 version 1.1.1 (Parlay 4.0)	
Exceptions removed in ES 202 915-10 version 1.2.1 (Parlay 4.1)	
Exceptions removed in ES 202 915-10 version 1.3.1 (Parlay 4.2)	
Exceptions removed in ES 202 915-10 version 1.4.1 (Parlay 4.3)	

D.6 Others

ES 202 915-10 V1.3.1:

New annex C added, together with J2EE™ and J2SE™ Java™ code.

ES 202 915-10 V1.4.1:

WSDL code removed from annex B.

History

Document history		
V1.1.1	January 2003	Publication
V1.2.1	August 2003	Publication
V1.3.1	March 2005	Publication
V1.4.1	October 2006	Membership Approval Procedure MV 20061215: 2006-10-17 to 2006-12-15