Abstract

This document provides a gap-analysis on how the ForCES architecture meets the requirements for CUSP. In addition it provides a ForCES XML model that implements the current proposed CUSP information model.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 31, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
# Terminology and Conventions

## Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## Definitions

This document reiterates the terminology defined by the ForCES architecture in various documents for the sake of clarity.
FE Model - The ForCES model used for describing resources to be managed/controlled. This includes three components; the modeling of individual Logical Functional Block (LFB model), the logical interconnection between LFBs (LFB topology), and the FE-level attributes, including LFB components, capabilities and events. The FE model provides the basis to define the information elements exchanged between CEs and FEs in the ForCES protocol [RFC5810].

LFB (Logical Functional Block) Class - A template that represents a resource that is being modeled. Most LFBs relate to packet processing in the data path; however, that is not always the case. LFB classes are the basic building blocks of the FE model.

LFB Instance - A runtime instantiation of an LFB class.

ForCES Component - A ForCES Component is a well-defined, uniquely identifiable and addressable ForCES model building block. A component has a 32-bit ID, name, type, and an optional synopsis description. These are often referred to simply as components.

ForCES Protocol - Protocol that runs in the Fp reference points in the ForCES Framework [RFC3746].

ForCES Protocol Layer (ForCES PL) - A layer in the ForCES protocol architecture that defines the ForCES protocol messages, the protocol state transfer scheme, and the ForCES protocol architecture itself as defined in the ForCES Protocol Specification [RFC5810].

ForCES Protocol Transport Mapping Layer (ForCES TML) - A layer in ForCES protocol architecture that uses the capabilities of existing transport protocols to specifically address protocol message transportation issues, such as how the protocol messages are mapped to different transport media (like TCP, IP, ATM, Ethernet, etc.), and how to achieve and implement reliability, ordering, etc. The ForCES SCTP TML [RFC5811] describes a TML that is mandated for ForCES.

2. Introduction

The ForCES architecture comprises:

1. Data model definition [RFC5812] serving as a basis for the architecture constructs acted on by the protocol.

2. The ForCES protocol (PL) [RFC5810] which acts on the model component constructs for the purpose of control/management.
3. A transport mapping layer (TML) which takes the PL constructs and maps them to underlying convenient transport(s) and then delivers them to the target end points. Currently there is only one standardized TML based on SCTP; [RFC5811]. However more could be defined — example QUIC [I-D.ietf-quic-transport] appears to be a very good fit.

This document presents the ForCES architecture as a basis for CUSP. For contextual overview, any performance numbers or prescribed "experience" in this document are based on deployment experience over many years at large and small deployment environments for embedded, cloud as well as data centre environments. Some of these deployments (still operational at time of writing) have been publicly hinted at in media [media1], [media2].

3. ForCES overview

In this section we present a quick overview of the ForCES architecture. The reader is encouraged to read the relevant documents, in particular 5810 [RFC5810], 5812 [RFC5812] and 5811 [RFC5811].

The origins of ForCES lie in desire to separate control and datapath; where "datapath" was intended to be packet processing resources. Over time, however, due to the convenience of ForCES architecture it has been used for controlling and managing arbitrary (other than packet processing) resources. As long as one can abstract the resources using the ForCES model, the protocol semantics allows using ForCES protocol to control and manage said resources.

In the case of the CUSP BNG information model, as we will show later the attributes such as interfaces, user statistics and QoS parameters can all be modeled as resources.

3.1. ForCES protocol

The ForCES protocol features can be summarized as:

1. Transport independence. The ForCES protocol is intended to run on a variety of chosen protocols. This was design intent to separate the PL from the different transports for given resources and environments. As an example, one of the original desires was to directly run over PCI-Express.

2. Simplified ForCES layer when possible:

* Security is left up to the transport choice keeping the ForCES layer simple. As an example, RFC 5811 defines running
ForCES on top of SCTP with IPSEC for security. TLS has been introduced also in deployments over time with SCTP, although no standards docs have been published for this.

* Optional configurable Controller high availability. FEs(resource owners) when desired can connect to multiple controllers in both cold or hot standby mode [RFC5810], [RFC7121].

* Controller-controller connectivity could be taken care of by other existing technologies. For example, in known deployments of ForCES controller to controller connectivity and mastership delegation has been taken care by using specialized control applications that interact with external consensus-driven infrastructure implementations such as the various Raft-based[XXX: Ref here] utilities.

3. Transport requirements. Deployment experience with ForCES as depicted in the SCTP TML (RFC 5811 [RFC5811]) has shown an absolute need for a variety of shades of reliability. Requests from the control to the resource must be reliably delivered, pronto (think a FIB update). So are the responses back; however, transactions like synchronous reports of statistics could afford to be lost once in a while; in other words retransmitting such stats is not useful since they are monotonically incrementing. This helps reduce noise in situations of congestion. Likewise, packet redirects coming from outside the box could afford to be lost since the end peer will end up retransmitting anyways. (Think OSPF link state updates for example or BGP on top of TCP).

4. Node overload. Deployment experience has shown the need to protect against node overload in a work-conserving mode (thus optimal resource usage). The SCTP TML prioritizes both compute and wire resources to favor requests made to the controlled-resource as well as responses back to the controller over events; whereas, events are prioritized over redirect packets. With this approach, there is no need to provide hacks and workarounds like non-work conserving approaches (such as rate limiting redirects to the controller). As an example, delivering from the resource owner (FE) a SET response to a request to update a FIB entry is considered more important than delivering an event for a link going down (then back up); the link event will be prioritized over an externally sourced BGP packet requiring further controller processing.

5. Transactional capabilities in the form of 2 phase commits.
6. Wire Serialization. Encoding on the wire is binary. The data model is sufficient to describe the content on the wire.

7. Transactional scalability, low latency and high throughput. The original desire of ForCES was to be able to achieve very high throughput transactions for the purpose of updating one or more datapath tables (depending on the table contents, achieving 10s to 100s of thousands of table updates/second is possible). It has been demonstrated in deployments that ForCES supports these throughput numbers along with supporting handling of millions of events per seconds. The choice of making the underlying protocol as binary, topped with allowing for command batching allows the protocol to achieve these goals.

8. Various execution modes for transactions (Execute-all-or-none, Execute-until-error, Execute-all-despite-errors). The default mode of execution in ForCES is the atomic Execute-all-or-none where the resource controller(CE) asks the resource owner(FE) to run a transaction and abort if anything fails. The reader is referred to look at RFC 5810 for more details.

9. Communication methods. ForCES provides two communication methods for a controller to receive data from the device, namely request/response and publish/subscribe. ForCES allows the controller at any time to access (request) any resource, and allows for a controller to subscribe to any supported resources events.

10. API affinity. The ForCES architecture provides (very) few simple protocol verbs which act upon a multitude of nouns as represented by the ForCES model. This allows powerful programmatic interfaces i.e. the "API" construct boils down to a formulation of operations in the form of a "verb" or a command acting on a set "nouns" (describing a resource path) followed by "optional arguments" in the form of data. The grammar could be described as:

<Command> <Resource path> [Data]

In other words, the ForCES semantic allows composing of rich services on top of the basic grammar. The expressive simplicity of the protocol is achieved due to the few verbs which act on the agreed-to modeled LFB components. The protocol is totally agnostic of the nature of the resource being controlled/managed. It is up to the modeler to describe the resource in the manner that is fitting (although frowned-upon, one could describe the resource model exactly as it is implemented and reduce the generalities and therefore translation overhead). The model is
highly extensible and for this reason, the knowledge of the resource control is offloaded to the service layer and a basic infrastructure is all that is needed.

The ForCES verbs are: {GET, SET, DELETE, REPORT and a few helpers).

11. Traffic sensitive protocol level connectivity detection. ForCES layer heartbeats could be sent in either or both directions. Heartbeats, however are only sent when the link is idle.

12. Dynamic association of FEs to CEs. FEs register and unregister to controllers and advertise their capabilities and capacities.

3.2. ForCES Model

The ForCES Model features can be summarized as:

1. Data model modularity through LFB class definition.

2. Data model type definitions via atomic types, complex/compound types, grouping of compound types in the form of structures and indexed/keyed tables which then end up composing addressable components within an LFB class.

3. Hierarchical/tree definition of control/config/state components which is acted on by a controller via the ForCES protocol.

4. Information-modeled metadata and expectations.

5. Built in LFB class resource capability definition/advertisement.

6. Publish/subscribe LFB event model with expressive trigger and report definitions. Filters include:

   * Hysteresis - used to suppress generation of notifications for oscillations around a condition value. Example, "generate an event if the link laser goes below 10 or goes above 15".

   * Count - used to suppress event generation around occurrence count. Example, "report the event to the client only after 5 consecutive occurrences".

   * Time interval - used to suppress event generation around a time limit. Example, "Generate an event if table foo row hasn’t been used in the last 10 minutes".
There could be multiple filters defined per event. Example of compound filtering: "Generate an event when the count reaches 5 or every 10 minutes when there is at least one event".

7. Data model flexibility/extensibility through augmentations, and inheritance.

8. Backward and forward compatibility via LFB design and versioning rules.


4. Meeting CUSP requirements

This section describes how ForCES meets the CUSP requirements[XXX: ref to latest cusp draft]. We hope to convince the reader that there already exists a robust IETF architecture which has a large deployment experience that meets all the CUSP requirements.

4.1. Transmit information tables

One of the main requirements for the CUSP protocol is the ability to efficiently transmit information tables. In a few words, ForCES is a wire-optimized protocol able to efficiently send and receive data. The following list addresses each of the stated requirements in [XXX: ref to latest cusp draft].

REQ1: "The separation protocol SHOULD be lightweight to support at least 2000 bytes for at least 2000 users per second."

For the protocol throughput, this translates to support at least 4Mbytes per second. There are no limitations in the protocol or architecture that constrain throughput. Typical throughput constraints observed in deployments tend to be wire bandwidth or in certain cases accessing ASIC for setting or dumping tables. ForCES deployments in real production, depending on environment have been demonstrated to support 10s to 100s thousands of table updates and millions of event per second.

REQ2: "The separation protocol SHOULD support data encoded as either XML or binary."

ForCES encodes formalized modeled data values as binary to be efficient on the wire as discussed earlier. RFC 5812 [RFC5812] defines the data model which is carried in the protocol RFC 5810 [RFC5810]. While it is not advisable for reasons of efficiency, one could define content in UTF-8 XML.
REQ3: "Batching and command grouping ability SHOULD be provided. Also the protocol MUST have an all-or-nothing semantics."

As has been discussed in previous sections of this document, in regards to batching, ForCES inherently provides batching of protocol messages as well as pipelining. Regarding command grouping, ForCES messages are ordered and are received in the order they are sent. Finally, ForCES has an Execution Mode flag that supports, among others, the execute-all-or-none semantic.

REQ4: "The protocol SHOULD be able to support at least hundreds of devices and tens of thousands of ports. In effect the protocol field sizes SHOULD correspond to large numbers."

ForCES uses unsigned integers of 32 bit size for identifiers. As such, for a specific level of hierarchy, ForCES can address up to 4,294,967,296 unique object, more than enough for this requirement.

The authors believes that the ForCES framework meets the requirement to efficiently transmit information tables

4.2. Message Priority

REQ5: "The separation protocol MUST provide a means to express the protocol message priorities."

As has been discussed earlier, the ForCES transport protocol (TML) is responsible for priority levels and currently supports up to 8 different priority levels. The current implementation of TML, SCTP-TML supports three priority channels, a high priority, reliable channel, a medium priority, semi-reliable channel and a low priority, unreliable channel.

The authors believe that ForCES meets the protocol message priority requirements.

4.3. Reliability

REQ6: "The separation protocol MUST provide a heartbeat monitoring mechanism which SHOULD have the ability to distinguish whether the interruption is an actual failure."

ForCES provides a parametrizable heartbeat mechanism that allows the controller to modify the frequency of the heartbeats and a piggybacking mechanism where protocol
messages are considered heartbeats themselves and as such reduce the number of individual heartbeats. ForCES also allows the controller to turn off the heartbeat mechanism, making the device ignore the heartbeats, for example in the case of a planned update while the controller is being updated. ForCES is able to support these heartbeat mechanism features since these parameters have been modeled as resources and as such are made available to the controller for configuration.

The authors believe ForCES meets the requirements for reliability.

4.4. Support for secure communication

REQ7: "The separation protocol MUST protect against man-in-the-middle attacks and security in a variety of scenarios. TLS and IPsec are good candidates."

As has been discussed earlier, ForCES’s TML is responsible for security services and is expected to employ TLS or IPsec. Specifically, the current defined TML, SCTP-TML, utilizes IPsec. ForCES has also been deployed with the use of TLS with SCTP.

The authors believe ForCES meets the security requirements.

4.5. Version negotiation

REQ8: "The separation protocol MUST provide some mechanisms to perform version negotiation for protocol versions."

ForCES provides different layers of version negotiation. There is a protocol field that specifies the current version of the ForCES protocol with which the controller and the device can agree to. In regards to the various supported resource (LFB) models, after the association has been established between the controller and the device, the controller can request and discover the current supported version of LFBs classes.

The authors believe that ForCES meets the requirements version negotiation requirements.

4.6. Capability Exchange

REQ9: "The protocol MUST provide mechanisms to exchange the device’s capabilities."

Haleplidis & Hadi Salim Expires October 31, 2019 [Page 10]
ForCES provides a means to model each resource’s capabilities in detail. The capabilities can then be queried at any time by the controller. Thus the controller is able to determine, after the association has been complete, the capabilities of each resource.

The authors believe that ForCES meets the protocol capability exchange requirement.

### 4.7. CP primary/backup capability

REQ10: "CUSP should provide mechanisms to support backup CPs. It should provide a mechanism to determine which CP is the primary and a mechanism to switch between primary and backup."

The ForCES architecture has been engineered with high availability mechanisms such as cold and hot standby, as defined by RFC7121 [RFC7121]. ForCES allows a single master writer and multiple backups which can act as readers (1+M). This was done for the sake of simplicity of the HA mode. ForCES allows the controller to specify which is the master and the order a backup will be selected from a pool of backup controllers. When the master is down, the device sends out an event and selects the next best candidate.

The authors believe that ForCES meets the protocol primary and backup requirement.

### 4.8. Event Notification

REQ11: "The protocol SHOULD be able to asynchronously notify the controller of events. Examples are sending responses back, user tracing, statistics collection and report user detected."

The ForCES architecture exposes a publish/subscribe event model. The events are published using the ForCES model on a per resource level. An event definition constitutes of the event target, meaning the monitored value, the trigger and the reported value. ForCES allows the controller to subscribe to any event defined by any resource. In addition ForCES also allows filtering of events for further refinement.

The authors believe this requirement is fully met by ForCES.
4.9. Query Statistics

REQ12: "The CUSP protocol MUST provide a way to query statistics."

As discussed earlier, ForCES provides two distinct approaches to receiving statistics. Either by the controller polling the device to get the updated values, or the controller can subscribe to event notification and receive periodic updated statistics.

The authors believe this requirement is fully met by ForCES.

4.10. Beyond the CUSP requirements

One requirement that believe is important but is not specified in the CUSP document is the separation of the protocol and the data model.

A singular protocol with varying data models makes it feasible to cater for various access technologies: it allows a single control interface for multi-access devices that provide termination of subscribers over fixed access nodes (DSLAM and OLTs), fixed-wireless and hybrid access. ForCES is an excellent fit for reasons described thus far.

5. Control and user plane information models

In this section we provide individual ForCES based XML models for different parts of the CUSP information model

5.1. Information model for the Control Plane

5.1.1. User related information

```xml
<dataTypeDefs>
  <!-- User ID -->
  <dataTypeDef>
    <name>UserID</name>
    <synopsis>Identification of a user</synopsis>
    <typeRef>uint32</typeRef>
  </dataTypeDef>
  <!-- MAC address -->
  <dataTypeDef>
    <name>IEEEMac</name>
    <synopsis>MAC address</synopsis>
    <typeRef>byte[6]</typeRef>
  </dataTypeDef>
  <!-- Access Type -->
  <dataTypeDef>
```

Haleplidis & Hadi Salim Expires October 31, 2019 [Page 12]
<name>AccessDataType</name>
<synopsis>Indicates the protocol being used for the user’s access</synopsis>
<atomic>
  <baseType>uint16</baseType>
  <specialValues>
    <specialValue value="0">
      <name>PPPoE</name>
      <synopsis/>
    </specialValue>
    <specialValue value="1">
      <name>IPoE</name>
      <synopsis/>
    </specialValue>
  </specialValues>
</atomic>
</dataTypeDef>

<!-- User Basic Information -->
<dataTypeDef>
  <name>UserBasicRow</name>
  <synopsis>User Basic Information</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>MacAddress</name>
      <synopsis>The MAC Address of the user</synopsis>
      <typeRef>IEEE1MAC</typeRef>
    </component>
    <component componentID="3">
      <name>AccessType</name>
      <synopsis>The access type of the user</synopsis>
      <optional/>
      <typeRef>AccessDataType</typeRef>
    </component>
    <component componentID="4">
      <name>SessionID</name>
      <synopsis>Identifier of PPPoE session</synopsis>
      <optional/>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="5">
      <name>InnerVLANID</name>
      <synopsis>Inner VLAN ID</synopsis>
      <optional/>
    </component>
  </struct>
</dataTypeDef>
<typeRef>uint16</typeRef>
</component>

<component componentID="6">
  <name>OuterVLANID</name>
  <synopsis>Outer VLAN ID</synopsis>
  <optional/>
  <typeRef>uint16</typeRef>
</component>

<component componentID="7">
  <name>UserInterface</name>
  <synopsis>Binding interface of a specific user</synopsis>
  <typeRef>uint32</typeRef>
</component>

</struct>
</dataTypeDef>

<dataTypeDef>
  <name>IPv4InfoRow</name>
  <synopsis>IPv4 User Information</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>UserIPv4</name>
      <synopsis>A user’s IPv4 Address</synopsis>
      <typeRef>byte[4]</typeRef>
    </component>
    <component componentID="3">
      <name>MaskLength</name>
      <synopsis>The subnet mask length</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="4">
      <name>Gateway</name>
      <synopsis>The user’s gateway</synopsis>
      <typeRef>byte[4]</typeRef>
    </component>
    <component componentID="5">
      <name>VRF</name>
      <synopsis>Identifier of VRF instance</synopsis>
      <typeRef>uint32</typeRef>
    </component>
  </struct>
</dataTypeDef>

<!-- IPv6 Selectory Type -->
<dataTypeDef>
  <name>IPv6SelectorType</name>
  <synopsis>IPv6 Type selector</synopsis>
  <atomic>
    <baseType>uchar</baseType>
    <specialValues>
      <specialValue value="0">
        <name>IPv6Address</name>
        <synopsis>Value contains an IPv6 Address</synopsis>
      </specialValue>
      <specialValue value="1">
        <name>PDAddress</name>
        <synopsis>Value contains PD address</synopsis>
      </specialValue>
    </specialValues>
  </atomic>
</dataTypeDef>

<dataTypeDef>
  <name>IPv6InfoRow</name>
  <synopsis>IPv6 User Information</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>IPv6Type</name>
      <synopsis>IPv6 Type selector</synopsis>
      <typeRef>IPv6SelectorType</typeRef>
    </component>
    <component componentID="3">
      <name>IPv6orPD</name>
      <synopsis>An IPv6 address or a PD</synopsis>
      <typeRef>byte[16]</typeRef>
    </component>
    <component componentID="4">
      <name>PrefixLen</name>
      <synopsis>The prefix length for either an IPv6 Address or Prefix Delegation</synopsis>
      <optional/>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="5">
      <name>VRF</name>
      <synopsis>Identifier of VRF instance</synopsis>
      <typeRef>uint32</typeRef>
    </component>
  </struct>
</dataTypeDef>
<!-- Rate and Size for committed and peak-->
<dataTypeDef>
  <name>RateSizeType</name>
  <synopsis>Rate and size for committed and peak</synopsis>
  <struct>
    <component componentID="1">
      <name>CIR</name>
      <synopsis>Commited Information Rate</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="2">
      <name>PIR</name>
      <synopsis>Peak Information Rate</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="3">
      <name>CBS</name>
      <synopsis>Commited Burst Size</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="4">
      <name>PBS</name>
      <synopsis>Peak Burst Size</synopsis>
      <typeRef>uint32</typeRef>
    </component>
  </struct>
</dataTypeDef>

<dataTypeDef>
  <name>QoSRow</name>
  <synopsis>User’s QoS</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>RateSize</name>
      <synopsis>Rate and size for committed and peak</synopsis>
      <typeRef>RateSizeType</typeRef>
    </component>
    <component componentID="3">
      <name>QoSProfile</name>
      <typeRef></typeRef>
    </component>
  </struct>
</dataTypeDef>
<synopsis>Standard profile from operator</synopsis>
	<typeRef>uint32</typeRef>
	</component>
</struct>
</dataTypeDef>
</dataTypeDefs>

<LFBClassDefs>
  <LFBClassDef LFBClassID="2001">
    <name>ControlPlaneUser</name>
    <synopsis>The control plane components for users</synopsis>
    <version>1.0</version>
    <components>
      <component componentID="1">
        <name>UserInfo</name>
        <synopsis>User Informational model</synopsis>
        <array>
          <struct>
            <component componentID="1">
              <name>UserBasic</name>
              <synopsis>User Basic Information</synopsis>
              <typeRef>UserBasicRow</typeRef>
            </component>
            <component componentID="2">
              <name>IPv4Info</name>
              <synopsis>Optional IPv4 information</synopsis>
              <optional/>
              <typeRef>IPv4InfoRow</typeRef>
            </component>
            <component componentID="3">
              <name>IPv6Info</name>
              <synopsis>Optional IPv6 information</synopsis>
              <optional/>
              <typeRef>IPv6InfoRow</typeRef>
            </component>
            <component componentID="4">
              <name>QoSInfo</name>
              <synopsis>Optional QoS Profile</synopsis>
              <optional/>
              <typeRef>QoSRow</typeRef>
            </component>
          </struct>
        </array>
      </component>
    </components>
  </LFBClassDef>
</LFBClassDefs>
5.1.2. Interface related information

<dataTypeDefs>
<!-- Interface Related Information -->
</dataTypeDefs>

<!-- Service Data Type -->
<dataTypeDef>
  <name>ServiceDataType</name>
  <synopsis>Service Type</synopsis>
  <struct>
    <component componentID="1">
      <name>PPPonly</name>
      <synopsis>If true, interface only supports PPP user</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="2">
      <name>IPv4Trig</name>
      <synopsis>If true, interface supports the user being triggered to connect to internet via IPv4</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="3">
      <name>IPv6Trig</name>
      <synopsis>If true, interface supports the user being triggered to connect to internet via IPv6</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="4">
      <name>NDTrig</name>
      <synopsis>If true, interface supports the user being triggered to connect to Internet via neighbor discovery</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="5">
      <name>ARPProxy</name>
      <synopsis>If true, ARP Proxy is enabled on this interface</synopsis>
      <typeRef>uint16</typeRef>
    </component>
  </struct>
</dataTypeDef>

<!-- interface info row -->
<dataTypeDef>
</dataTypeDef>
<name>InterfaceInfoRow</name>
<synopsis>Interface information</synopsis>
<struct>
  <component componentID="1">
    <name>IfIndex</name>
    <synopsis>Index assigned to an interface</synopsis>
    <typeRef>uint32</typeRef>
  </component>
  <component componentID="2">
    <name>bas_enable</name>
    <synopsis>Bas enable flag</synopsis>
    <typeRef>uint16</typeRef>
  </component>
  <component componentID="3">
    <name>ServiceType</name>
    <synopsis>Service Type</synopsis>
    <typeRef>ServiceDataType</typeRef>
  </component>
</struct>
</dataTypeDef>
</dataTypeDefs>
</LFBClassDefs>

Figure 2: Interface related information

5.1.3. Device related information

<!-- Address Field -->
<dataTypeDefs>
  <!-- Address Field -->
  <dataTypeDef>
    <name>AddressRow</name>
    <synopsis>Address field distribute table row
    <typeRef>AddressDataType</typeRef>
  </dataTypeDef>
</dataTypeDefs>
</synopsis>
<struct>
  <component componentID="1">
    <name>AddressSegment</name>
    <synopsis>Address Segment</synopsis>
    <typeRef>byte[4]</typeRef>
  </component>
  <component componentID="2">
    <name>AddressSegmentMask</name>
    <synopsis>Address Segment Mask</synopsis>
    <typeRef>byte[4]</typeRef>
  </component>
  <component componentID="3">
    <name>AddressSegmentVRF</name>
    <synopsis>Address Segment VRF instance identifier</synopsis>
    <typeRef>uint32</typeRef>
  </component>
  <component componentID="4">
    <name>IfIndex</name>
    <synopsis>Index assigned to an interface</synopsis>
    <typeRef>uint32</typeRef>
  </component>
  <component componentID="5">
    <name>maskLength</name>
    <synopsis>Length of the mask</synopsis>
    <typeRef>uint32</typeRef>
  </component>
</struct>
</dataTypeDef>
</dataTypeDefs>
<LFBClassDefs>
  <LFBClassDef LFBClassID="2003">
    <name>ControlPlaneDevice</name>
    <synopsis>The control plane components for device</synopsis>
    <version>1.0</version>
    <components>
      <component componentID="1">
        <name>Device</name>
        <synopsis>Device Information</synopsis>
        <array>
          <typeRef>AddressRow</typeRef>
        </array>
      </component>
    </components>
  </LFBClassDef>
</LFBClassDefs>
Figure 3: Device related information

5.2. Information model for User Plane

<dataTypeDefs>
  <!-- User ID -->
  <dataTypeDef>
    <name>UserID</name>
    <synopsis>Identification of a user</synopsis>
    <typeRef>uint32</typeRef>
  </dataTypeDef>
  <!-- MAC address -->
  <dataTypeDef>
    <name>IEEEMac</name>
    <synopsis>MAC address</synopsis>
    <typeRef>byte[6]</typeRef>
  </dataTypeDef>
  <!-- IfTypeDataType -->
  <dataTypeDef>
    <name>IfTypeDataType</name>
    <synopsis>Type of Interface</synopsis>
    <atomic>
      <baseType>uint32</baseType>
      <specialValues>
        <specialValue value="0">
          <name>Ethernet</name>
          <synopsis>An ethernet interface</synopsis>
        </specialValue>
        <specialValue value="1">
          <name>GE</name>
          <synopsis>A Gigabit Ethernet interface</synopsis>
        </specialValue>
        <specialValue value="2">
          <name>EtherTrunk</name>
          <synopsis>An EtherTrunk interfaces</synopsis>
        </specialValue>
      </specialValues>
    </atomic>
  </dataTypeDef>
  <!-- LinkTypeDataType -->
  <dataTypeDef>
    <name>LinkTypeDataType</name>
    <synopsis>Link Types</synopsis>
  </dataTypeDef>
</dataTypeDefs>
<atomic>
  <baseType>uint32</baseType>
  <specialValues>
    <specialValue value="0">
      <name>PointToPoint</name>
      <synopsis>A point to point link</synopsis>
    </specialValue>
    <specialValue value="1">
      <name>Broadcast</name>
      <synopsis>A broadcast link</synopsis>
    </specialValue>
    <specialValue value="2">
      <name>Multipoint</name>
      <synopsis>A multipoint link</synopsis>
    </specialValue>
  </specialValues>
</atomic>
</dataTypeDef>
</!-- IfStateType -->
<dataTypeDef>
  <name>IfStateType</name>
  <synopsis>Current operational state of the interface</synopsis>
  <atomic>
    <baseType>uchar</baseType>
    <specialValues>
      <specialValue value="1">
        <name>Up</name>
        <synopsis>Up</synopsis>
      </specialValue>
      <specialValue value="2">
        <name>Down</name>
        <synopsis>Down</synopsis>
      </specialValue>
      <specialValue value="3">
        <name>Testing</name>
        <synopsis>In testing mode</synopsis>
      </specialValue>
      <specialValue value="4">
        <name>Unknown</name>
        <synopsis>Status cannot be determined</synopsis>
      </specialValue>
      <specialValue value="5">
        <name>Dormant</name>
        <synopsis>Dormant</synopsis>
      </specialValue>
    </specialValues>
  </atomic>
</dataTypeDef>
<specialValue value="6">
  <name>NotPresent</name>
  <synopsis>Component is missing</synopsis>
</specialValue>
</atomic>
</dataTypeDef>
<!-- Port Resources of UP Informational model -->
<dataTypeDef>
  <name>PortResourcesRow</name>
  <synopsis>Port resources table row</synopsis>
  <struct>
    <component componentID="1">
      <name>IfIndex</name>
      <synopsis>Index assigned to an interface</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="2">
      <name>IfName</name>
      <synopsis>Name of the interface</synopsis>
      <typeRef>string[64]</typeRef>
    </component>
    <component componentID="3">
      <name>IfType</name>
      <synopsis>Type of Interface</synopsis>
      <typeRef>IfTypeDataType</typeRef>
    </component>
    <component componentID="4">
      <name>LinkType</name>
      <synopsis>Link Types</synopsis>
      <typeRef>LinkTypeDataType</typeRef>
    </component>
    <component componentID="5">
      <name>MacAddress</name>
      <synopsis>The MAC Address of the link</synopsis>
      <typeRef>IEEEMAC</typeRef>
    </component>
    <component componentID="6">
      <name>IfState</name>
      <synopsis>Current operational state of the interface</synopsis>
      <typeRef>IfStateType</typeRef>
    </component>
  </struct>
</dataTypeDef>
<!-- Statistics data Type -->
<dataTypeDef>
<name>StatDataType</name>
<synopsis>Traffic Type</synopsis>
<atomic>
  <baseType>uint32</baseType>
  <specialValues>
    <specialValue value="0">
      <name>IPv4</name>
      <synopsis>IPv4 traffic</synopsis>
    </specialValue>
    <specialValue value="1">
      <name>IPv6</name>
      <synopsis>IPv6 traffic</synopsis>
    </specialValue>
  </specialValues>
</atomic>
</dataTypeDef>

<!-- Traffic Statistics Informational model -->
<dataTypeDef>
  <name>TrafficStatisticsRow</name>
  <synopsis>Traffic stats per user</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>StatType</name>
      <synopsis>Traffic Type</synopsis>
      <typeRef>StatDataType</typeRef>
    </component>
    <component componentID="3">
      <name>IngressPackets</name>
      <synopsis>Ingress packet stats</synopsis>
      <typeRef>uint64</typeRef>
    </component>
    <component componentID="4">
      <name>IngressByte</name>
      <synopsis>Ingress byte stats</synopsis>
      <typeRef>uint64</typeRef>
    </component>
    <component componentID="5">
      <name>EgressPackets</name>
      <synopsis>Egress packet stats</synopsis>
      <typeRef>uint64</typeRef>
    </component>
    <component componentID="6">
      <name>EgressBytes</name>
    </component>
  </struct>
</dataTypeDef>
6. ForCES XML model for CUSP info model

In this section we provide the whole ForCES based XML model of the information model of the CUSP BNG.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LFBLibrary xmlns="urn:ietf:params:xml:ns:forces:lfbmodel:1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" provides="vbng">
    <frameDefs>
        <frameDef>
            <name>Arbitrary</name>
            <synopsis>Any kind of packet</synopsis>
        </frameDef>
    </frameDefs>
</LFBLibrary>
```
<dataTypeDefs>
<!-- User ID -->
<dataTypeDef>
  <name>UserID</name>
  <synopsis>Identification of a user</synopsis>
  <typeRef>uint32</typeRef>
</dataTypeDef>

<!-- MAC address -->
<dataTypeDef>
  <name>IEEEMac</name>
  <synopsis>MAC address</synopsis>
  <typeRef>byte[6]</typeRef>
</dataTypeDef>

<!-- Access Type -->
<dataTypeDef>
  <name>AccessDataType</name>
  <synopsis>Indicates the protocol being used for the user’s access</synopsis>
  <atomic>
    <baseType>uint16</baseType>
    <specialValues>
      <specialValue value="0">PPPoE</specialValue>
      <synopsis>An ethernet interface</synopsis>
      <specialValue value="1">IPoE</specialValue>
      <synopsis>A Gigabit Ethernet interface</synopsis>
    </specialValues>
  </atomic>
</dataTypeDef>

<!-- IfTypeDataType -->
<dataTypeDef>
  <name>IfTypeDataType</name>
  <synopsis>Type of Interface</synopsis>
  <atomic>
    <baseType>uint32</baseType>
    <specialValues>
      <specialValue value="0">Ethernet</specialValue>
      <synopsis>An ethernet interface</synopsis>
      <specialValue value="1">GE</specialValue>
      <synopsis>A Gigabit Ethernet interface</synopsis>
    </specialValues>
  </atomic>
</dataTypeDef>
<specialValue value="2">
  <name>EtherTrunk</name>
  <synopsis>An EtherTrunk interfaces</synopsis>
</specialValue>
</specialValues>
</atomic>
</dataTypeDef>
<!-- LinkTypeDataType -->
<dataTypeDef>
  <name>LinkTypeDataType</name>
  <synopsis>Link Types</synopsis>
  <atomic>
    <baseType>uint32</baseType>
    <specialValues>
      <specialValue value="0">
        <name>PointToPoint</name>
        <synopsis>A point to point link</synopsis>
      </specialValue>
      <specialValue value="1">
        <name>Broadcast</name>
        <synopsis>A broadcast link</synopsis>
      </specialValue>
      <specialValue value="2">
        <name>Multipoint</name>
        <synopsis>A multipoint link</synopsis>
      </specialValue>
    </specialValues>
  </atomic>
</dataTypeDef>
<!-- IfStateType -->
<dataTypeDef>
  <name>IfStateType</name>
  <synopsis>Current operational state of the interface</synopsis>
  <atomic>
    <baseType>uchar</baseType>
    <specialValues>
      <specialValue value="1">
        <name>Up</name>
        <synopsis>Up</synopsis>
      </specialValue>
      <specialValue value="2">
        <name>Down</name>
        <synopsis>Down</synopsis>
      </specialValue>
    </specialValues>
  </atomic>
</dataTypeDef>
<specialValue value="3">
  <name>Testing</name>
  <synopsis>In testing mode</synopsis>
</specialValue>
<specialValue value="4">
  <name>Unknown</name>
  <synopsis>Status cannot be determined</synopsis>
</specialValue>
<specialValue value="5">
  <name>Dormant</name>
  <synopsis>Dormant</synopsis>
</specialValue>
<specialValue value="6">
  <name>NotPresent</name>
  <synopsis>Component is missing</synopsis>
</specialValue>
</specialValues>
</atomic>
</dataTypeDef>

<!-- Port Resources of UP Informational model -->
<dataTypeDef>
  <name>PortResourcesRow</name>
  <synopsis>Port resources table row</synopsis>
  <struct>
    <component componentID="1">
      <name>IfIndex</name>
      <synopsis>Index assigned to an interface</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="2">
      <name>IfName</name>
      <synopsis>Name of the interface</synopsis>
      <typeRef>string[64]</typeRef>
    </component>
    <component componentID="3">
      <name>IfType</name>
      <synopsis>Type of Interface</synopsis>
      <typeRef>IfTypeDataType</typeRef>
    </component>
    <component componentID="4">
      <name>LinkType</name>
      <synopsis>Link Types</synopsis>
      <typeRef>LinkTypeDataType</typeRef>
    </component>
    <component componentID="5">
      <name>MacAddress</name>
    </component>
  </struct>
</dataTypeDef>
<synopsis>The MAC Address of the link</synopsis>
<typeRef>IEEEMAC</typeRef>
</component>
</struct>
</dataTypeDef>
 setDataDef
<dataTypeDef>
<name>StatDataType</name>
<synopsis>Traffic Type</synopsis>
<atomic>
<baseType>uint32</baseType>
<specialValues>
<specialValue value="0">
<name>IPv4</name>
<synopsis>IPv4 traffic</synopsis>
</specialValue>
<specialValue value="1">
<name>IPv6</name>
<synopsis>IPv6 traffic</synopsis>
</specialValue>
</specialValues>
</atomic>
</dataTypeDef>
<!-- Statistics data Type -->
<!-- Traffic Statistics Informational model -->
<dataTypeDef>
<name>TrafficStatisticsRow</name>
<synopsis>Traffic stats per user</synopsis>
<struct>
<component componentID="1">
<name>userID</name>
<synopsis>The user id</synopsis>
<typeRef>UserID</typeRef>
</component>
<component componentID="2">
<name>StatType</name>
<synopsis>Traffic Type</synopsis>
<typeRef>StatDataType</typeRef>
</component>
<component componentID="3">
<name>IngressPackets</name>
<synopsis>Ingress packet stats</synopsis>
<typeRef>uint64</typeRef>
</component>
</struct>
</dataTypeDef>
<component componentID="4">
  <name>IngressByte</name>
  <synopsis>Ingress byte stats</synopsis>
  <typeRef>uint64</typeRef>
</component>

<component componentID="5">
  <name>EgressPackets</name>
  <synopsis>Egress packet stats</synopsis>
  <typeRef>uint64</typeRef>
</component>

<component componentID="6">
  <name>EgressBytes</name>
  <synopsis>Egress byte stats</synopsis>
  <typeRef>uint64</typeRef>
</component>

</struct>
</dataTypeDef>

<!-- User Basic Information -->
<dataTypeDef>
  <name>UserBasicRow</name>
  <synopsis>User Basic Information</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>MacAddress</name>
      <synopsis>The MAC Address of the user</synopsis>
      <typeRef>IEEEMAC</typeRef>
    </component>
    <component componentID="3">
      <name>AccessType</name>
      <synopsis>The Access Type of the user</synopsis>
      <optional/>
      <typeRef>AccessDataType</typeRef>
    </component>
    <component componentID="4">
      <name>SessionID</name>
      <synopsis>Identifier of PPPoE session</synopsis>
      <optional/>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="5">
      <name>InnerVLANID</name>
      <synopsis>Inner VLAN ID</synopsis>
    </component>
  </struct>
</dataTypeDef>
<optional/>
<typeRef>uint16</typeRef>
</component>
<component componentID="6">
<name>OuterVLANID</name>
<synopsis>Outer VLAN ID</synopsis>
<optional/>
<typeRef>uint16</typeRef>
</component>
<component componentID="7">
<name>UserInterface</name>
<synopsis>Binding interface of a specific user</synopsis>
<typeRef>uint32</typeRef>
</component>
</struct>
</dataTypeDef>
</dataTypeDef>
<dataTypeDef>
<name>IPv4InfoRow</name>
<synopsis>IPv4 User Information</synopsis>
<struct>
<component componentID="1">
<name>userID</name>
<synopsis>The user id</synopsis>
<typeRef>UserID</typeRef>
</component>
<component componentID="2">
<name>UserIPv4</name>
<synopsis>A user's IPv4 Address</synopsis>
<typeRef>byte[4]</typeRef>
</component>
<component componentID="3">
<name>MaskLength</name>
<synopsis>The subnet mask length</synopsis>
<typeRef>uint32</typeRef>
</component>
<component componentID="4">
<name>Gateway</name>
<synopsis>The user's gateway</synopsis>
<typeRef>byte[4]</typeRef>
</component>
<component componentID="5">
<name>VRF</name>
<synopsis>Identifier of VRF instance</synopsis>
<typeRef>uint32</typeRef>
</component>
</struct>
</dataTypeDef>
<!-- IPv6 Selectory Type -->
<dataTypeDef>
  <name>IPv6SelectorType</name>
  <synopsis>IPv6 Type selector</synopsis>
  <atomic>
    <baseType>uchar</baseType>
    <specialValues>
      <specialValue value="0">
        <name>IPv6Address</name>
        <synopsis>Value contains an IPv6 Address</synopsis>
      </specialValue>
      <specialValue value="1">
        <name>PDAddress</name>
        <synopsis>Value contains PD address</synopsis>
      </specialValue>
    </specialValues>
  </atomic>
</dataTypeDef>

<dataTypeDef>
  <name>IPv6InfoRow</name>
  <synopsis>IPv6 User Information</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>IPv6Type</name>
      <synopsis>IPv6 Type selector</synopsis>
      <typeRef>IPv6SelectorType</typeRef>
    </component>
    <component componentID="3">
      <name>IPv6orPD</name>
      <synopsis>An IPv6 address or a PD</synopsis>
      <typeRef>byte[16]</typeRef>
    </component>
    <component componentID="4">
      <name>PrefixLen</name>
      <synopsis>The prefix length for either an IPv6 Address or Prefix Delegation</synopsis>
      <optional/>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="5">
      <name>VRF</name>
      <synopsis>Identifier of VRF instance</synopsis>
      <typeRef>uint32</typeRef>
    </component>
  </struct>
</dataTypeDef>
<!-- Rate and Size for committed and peak-->
<dataTypeDef>
  <name>RateSizeType</name>
  <synopsis>Rate and size for committed and peak</synopsis>
  <struct>
    <component componentID="1">
      <name>CIR</name>
      <synopsis>Commited Information Rate</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="2">
      <name>PIR</name>
      <synopsis>Peak Information Rate</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="3">
      <name>CBS</name>
      <synopsis>Commited Burst Size</synopsis>
      <typeRef>uint32</typeRef>
    </component>
    <component componentID="4">
      <name>PBS</name>
      <synopsis>Peak Burst Size</synopsis>
      <typeRef>uint32</typeRef>
    </component>
  </struct>
</dataTypeDef>

<dataTypeDef>
  <name>QoSRow</name>
  <synopsis>User’s QoS</synopsis>
  <struct>
    <component componentID="1">
      <name>userID</name>
      <synopsis>The user id</synopsis>
      <typeRef>UserID</typeRef>
    </component>
    <component componentID="2">
      <name>RateSize</name>
      <synopsis>Rate and size for committed and peak</synopsis>
      <typeRef>RateSizeType</typeRef>
    </component>
  </struct>
</dataTypeDef>
<name>QoSProfile</name>
  <synopsis>Standard profile from operator</synopsis>
  <typeRef>uint32</typeRef>
</component>
</struct>
</dataTypeDef>

<!-- Interface Related Information -->
<!-- Service Data Type -->
<dataTypeDef>
  <name>ServiceDataType</name>
  <synopsis>Service Type</synopsis>
  <struct>
    <component componentID="1">
      <name>PPPonly</name>
      <synopsis>If true, interface only supports PPP user</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="2">
      <name>IPv4Trig</name>
      <synopsis>If true, interface supports the user being triggered to connect to internet via IPv4</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="3">
      <name>IPv6Trig</name>
      <synopsis>If true, interface supports the user being triggered to connect to internet via IPv6</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="4">
      <name>NDTrig</name>
      <synopsis>If true, interface supports the user being triggered to connect to Internet via neighbor discovery</synopsis>
      <typeRef>uint16</typeRef>
    </component>
    <component componentID="5">
      <name>ARPProxy</name>
      <synopsis>If true, ARP Proxy is enabled on this interface</synopsis>
      <typeRef>uint16</typeRef>
    </component>
  </struct>
</dataTypeDef>

<!-- interface info row -->
<dataTypeDef>
    <name>InterfaceInfoRow</name>
    <synopsis>Interface information</synopsis>
    <struct>
      <component componentID="1">
        <name>IfIndex</name>
        <synopsis>Index assigned to an interface</synopsis>
        <typeRef>uint32</typeRef>
      </component>
      <component componentID="2">
        <name>bas_enable</name>
        <synopsis>Bas enable flag</synopsis>
        <typeRef>uint16</typeRef>
      </component>
      <component componentID="3">
        <name>ServiceType</name>
        <synopsis>Service Type</synopsis>
        <typeRef>ServiceDataType</typeRef>
      </component>
    </struct>
  </dataTypeDef>

  <!-- Address Field -->
  <dataTypeDef>
    <name>AddressRow</name>
    <synopsis>Address field distribute table row</synopsis>
    <struct>
      <component componentID="1">
        <name>AddressSegment</name>
        <synopsis>Address Segment</synopsis>
        <typeRef>byte[4]</typeRef>
      </component>
      <component componentID="2">
        <name>AddressSegmentMask</name>
        <synopsis>Address Segment Mask</synopsis>
        <typeRef>byte[4]</typeRef>
      </component>
      <component componentID="3">
        <name>AddressSegmentVRF</name>
        <synopsis>Address Segment VRF instance identifier</synopsis>
        <typeRef>uint32</typeRef>
      </component>
      <component componentID="4">
        <name>IfIndex</name>
        <synopsis>Index assigned to an interface</synopsis>
        <typeRef>uint32</typeRef>
      </component>
    </struct>
  </dataTypeDef>
<component componentID="5">
  <name>maskLength</name>
  <synopsis>Length of the mask</synopsis>
  <typeRef>uint32</typeRef>
</component>
</struct>
</dataTypeDef>
</dataTypeDefs>
<LFBClassDefs>
<br LFBClassID="2001">
  <name>ControlPlaneUser</name>
  <synopsis>The control plane components for users</synopsis>
  <version>1.0</version>
  <components>
    <component componentID="1">
      <name>UserInfo</name>
      <synopsis>User Informational model</synopsis>
      <array>
        <struct>
          <component componentID="1">
            <name>UserBasic</name>
            <synopsis>User Basic Information</synopsis>
            <typeRef>UserBasicRow</typeRef>
          </component>
          <component componentID="2">
            <name>IPv4Info</name>
            <synopsis>Optional IPv4 information</synopsis>
            <optional/>
            <typeRef>IPv4InfoRow</typeRef>
          </component>
          <component componentID="3">
            <name>IPv6Info</name>
            <synopsis>Optional IPv6 information</synopsis>
            <optional/>
            <typeRef>IPv6InfoRow</typeRef>
          </component>
          <component componentID="4">
            <name>QoSInfo</name>
            <synopsis>Optional QoS Profile</synopsis>
            <optional/>
            <typeRef>QoSRow</typeRef>
          </component>
        </struct>
      </array>
    </component>
  </components>
</LFBClassDef>
<LFBClassDef LFBClassID="2002">
  <name>ControlPlaneInterface</name>
  <synopsis>The control plane components for interfaces</synopsis>
  <version>1.0</version>
  <components>
    <component componentID="1">
      <name>Interface</name>
      <synopsis>Interface Information</synopsis>
      <array>
        <typeRef>InterfaceInfoRow</typeRef>
      </array>
    </component>
  </components>
</LFBClassDef>

<LFBClassDef LFBClassID="2003">
  <name>ControlPlaneDevice</name>
  <synopsis>The control plane components for device</synopsis>
  <version>1.0</version>
  <components>
    <component componentID="1">
      <name>Device</name>
      <synopsis>Device Information</synopsis>
      <array>
        <typeRef>AddressRow</typeRef>
      </array>
    </component>
  </components>
</LFBClassDef>

<LFBClassDef LFBClassID="2011">
  <name>User</name>
  <synopsis>The user plane components</synopsis>
  <version>1.0</version>
  <components>
    <component componentID="1">
      <name>PortResourceTable</name>
      <synopsis>The port resource table for available network resources</synopsis>
      <array>
        <typeRef>PortResourcesRow</typeRef>
      </array>
    </component>
    <component componentID="2">
      <name>StatisticsTable</name>
      <synopsis>Stats per user</synopsis>
      <array>
      </array>
    </component>
  </components>
</LFBClassDef>
<typeRef>TrafficStatisticsRow</typeRef>
</array>
</component>
</components>
</LFBClassDef>
</LFBClassDefs>
</LFBLibrary>

Figure 5: ForCES based CUSP Info Model

7. Acknowledgements

Thanks to Joel Halpern and Diego Lopez for discussions (during IETF 104) that led to the creation of this document.

8. IANA Considerations

TBD

9. Security Considerations

TBD

10. References

10.1. Normative References

[I-D.ietf-quic-transport]


10.2. Informative References


Authors’ Addresses

Evangelos Haleplidis
Mojatatu Networks
Suite 200, 15 Fitzgerald Road
Ottawa, Ontario K2H 9G1
Canada

Email: ehalep@mojatatu.com
Jamal Hadi Salim
Mojatatu Networks
Suite 200, 15 Fitzgerald Road
Ottawa, Ontario  K2H 9G1
Canada

Email: hadi@mojatatu.com