VNF

Management Requirements

for

OpenECOMP

Revision	**1.0**
Revision Date	**2/1/2017**

Document Revision History

Date	Revision	Description
2/1/2017	1.0	Initial publication defining VNF Management Requirements for OpenECOMP

Table of Contents

1. Introduction

This document is part of a hierarchy of documents that describes the overall Requirements and Guidelines for OpenECOMP. The diagram below identifies where this document fits in the hierarchy.

OpenECOMP Requirements and Guidelines				
VNF Guidelines for Network Cloud and OpenECOMP				Future OpenECOMP Subject Documents
VNF Cloud Readiness Requirements for OpenECOMP	VNF Management Requirements for OpenECOMP	VNF Heat Template Requirements for OpenECOMP	Future VNF Requirements Documents	Future Requirements Documents

Document summary:

VNF Guidelines for Network Cloud and OpenECOMP

- Describes VNF environment and overview of requirements

VNF Cloud Readiness Requirements for OpenECOMP

- Cloud readiness requirements for VNFs (Design, Resiliency, Security, and DevOps)

VNF Management Requirements for OpenECOMP

- Requirements for how VNFs interact and utilize OpenECOMP

VNF Heat Template Requirements for OpenECOMP

- Provides recommendations and standards for building Heat templates compatible with OpenECOMP– initial implementations of Network Cloud are assumed to be OpenStack based.

The OpenECOMP (Enhanced Control, Orchestration, Management and Policy) platform is the part of the larger Network Function Virtualization/Software Defined Network (NFV/SDN) ecosystem that is responsible for the efficient control, operation and management of Virtual Network Function (VNF) capabilities and functions. It specifies standardized abstractions and interfaces that enable efficient interoperation of the NVF/SDN ecosystem components. It enables product/service independent capabilities for design, creation and runtime lifecycle management (includes all aspects of installation, change management, assurance, and retirement) of resources in NFV/SDN environment (see ECOMP white paper[1]). These capabilities are provided using two major architectural frameworks: (1) a Design Time Framework to design, define and program the platform (uniform onboarding), and (2) a Runtime Execution Framework to execute the logic programmed in the design environment (uniform delivery and runtime lifecycle management). The platform delivers an integrated information model based on the VNF package to express the characteristics and behavior of these resources in the Design Time Framework. The information model is utilized by Runtime Execution Framework to manage the runtime lifecycle of the

[1] ECOMP (Enhanced Control Orchestration, Management & Policy) Architecture White Paper (http://about.att.com/content/dam/snrdocs/ecomp.pdf)

VNFs. The management processes are orchestrated across various modules of OpenECOMP to instantiate, configure, scale, monitor, and reconfigure the VNFs using a set of standard APIs provided by the VNF developers.

2. Design Definition

The OpenECOMP Design Time Framework provides the ability to design NFV resources including VNFs, Services, and products. The vendor must provide VNF packages that include a rich set of recipes, management and functional interfaces, policies, configuration parameters, and infrastructure requirements that can be utilized by the OpenECOMP Design module to onboard and catalog these resources. Initially this information may be provided in documents, but in the near future a method will be developed to automate as much of the transfer of data as possible to satisfy its long term requirements.

The current VNF Package Requirement is based on a subset of the Requirements contained in the ETSI Document: ETSI GS NFV-MAN 001 v1.1.1 and GS NFV IFA011 V0.3.0 (2015-10) - Network Functions Virtualization (NFV), Management and Orchestration, VNF Packaging Specification.

Table 1. VNF Package

Principle	Description	Type	ID #
2.0.1 Resource Description	The VNF Vendor must provide a Manifest File that contains a list of all the components in the VNF package.	Must	10010
	The package must include VNF Identification Data to uniquely identify the resource for a given Vendor. The identification data must include: an identifier for the VNF, the name of the VNF as was given by the VNF Vendor, VNF description, VNF Vendor, and version.	Must	10020
	The VNF Vendor must provide documentation describing VNF Management APIs. The document must include information and tools for: • OpenECOMP to deploy and configure (initially and ongoing) the VNF application(s) (e.g., NETCONF APIs). Includes description of configurable parameters for the VNF and whether the parameters can be configured after VNF instantiation. • OpenECOMP to monitor the health of the VNF (conditions that require healing and/or scaling responses). Includes a description of: o Parameters that can be monitored for the VNF and event records (status, fault, flow, session, call, control plane, etc.) generated by the VNF after instantiation. o Runtime lifecycle events and related actions (e.g., control responses, tests) which can be performed for the VNF.	Must	10030
	The VNF package must include documentation describing VNF Functional APIs that are utilized to build network and application services. Provides the externally exposed functional inputs and outputs for the VNF, including interface format and protocols supported.	Must	10040

	The VNF Vendor must provide documentation describing VNF Functional Capabilities that are utilized to operationalize the VNF and compose complex services.	Must	10050
	The VNF Vendor must provide information regarding any dependency with other VNFs and resources.	Must	10060
2.0.2 Resource Configuration	The VNF Vendor must provide a Resource/Device YANG model as a foundation for creating the YANG model for configuration. This will include VNF attributes/parameters and valid values/attributes configurable by policy.	Must	10070
	The VNF Package must include configuration scripts for boot sequence and configuration.	Must	10080
	The VNF Vendor must provide configurable parameters (if unable to conform to YANG model) including VNF attributes/parameters and valid values, dynamic attributes and cross parameter dependencies (e.g., customer provisioning data).	Must	10090
2.0.3 Resource Control Loop	The VNF Vendor must provide documentation for the VNF Policy Description to manage the VNF runtime lifecycle. The document must include a description of how the policies (conditions and actions) are implemented in the VNF.	Must	10100
	The VNF Package must include documentation describing the fault, performance, capacity events/alarms and other event records that are made available by the VNF. The document must include: • A unique identification string for the specific VNF, a description of the problem that caused the error, and steps or procedures to perform Root Cause Analysis and resolve the issue. • All events, severity level (e.g., informational, warning, error) and descriptions including causes/fixes if applicable for the event. • All events (fault, measurement for VNF Scaling, Syslogs, State Change and Mobile Flow), that need to be collected at each VM, VNFC (defined in *VNF Guidelines for Network Cloud and OpenECOMP*) and for the overall VNF.	Must	10110
	The VNF Vendor must provide an XML file that contains a list of VNF error codes, descriptions of the error, and possible causes/corrective action.	Must	10120
	Provide documentation describing all parameters that are available to monitor the VNF after instantiation (includes all counters, OIDs, PM data, KPIs, etc., that must be collected for reporting purposes. The documentation must include a list of: • Monitoring parameters/counters exposed for virtual resource management and VNF application management. • KPIs and metrics that need to be collected at each VM for capacity planning purposes. • For each KPI, provide lower and upper limits. • When relevant, provide a threshold crossing alert point for each KPI at which time scaling rules will apply. • For each KPI, identify the suggested actions that need to be performed when a threshold crossing alert event is recorded.	Must	10130

	• Describe any requirements for the monitoring component of tools for Network Cloud automation and management to provide these records to components of the VNF. • When applicable, provide calculators needed to convert raw data into appropriate reporting artifacts.		
	The VNF Package must include documentation describing supported VNF scaling capabilities and capacity limits (e.g., number of users, bandwidth, throughput, concurrent calls).	Must	10140
	The VNF Package must include documentation describing the characteristics for the VNF reliability and high availability.	Must	10150
2.0.4 Compute, Network, Storage Requirements	The VNF Package must include VNF topology that describes basic network and application connectivity internal and external to the VNF including Link type, KPIs, Bandwidth, QoS (if applicable) for each interface.	Must	10160
	The VNF Package must include VM requirements via a Heat template that provides the necessary data for: • VM specifications for all VNF components - for hypervisor, CPU, memory, storage. • Network connections, interface connections, internal and external to VNF. • High availability redundancy model. • Static scaling/growth VM specifications. Note1: Must comply with the *Heat Template Requirements for Virtual Network Functions*. Note2: Must comply with the Network Cloud Specifications defined in *Example Implementation of Network Cloud.*	Must	10170
	The VNF Vendor must provide the binaries and images needed to instantiate the VNF (VNF and VNFC images).	Must	10180
	The VNF Vendor must describe scaling capabilities to manage scaling characteristics of the VNF.	Must	10190
2.0.5 Testing	The VNF Package must include documentation describing the tests that were conducted by the Vendor and the test results.	Must	10200
	The VNF Vendor must provide their testing scripts to support testing.	Must	10210
	The VNF Vendor must provide software components that can be packaged with/near the VNF, if needed, to simulate any functions or systems that connect to the VNF system under test. This component is necessary only if the existing testing environment does not have the necessary simulators.	Must	10220
2.0.6 Licensing Guidelines	VNFs must provide metrics (e.g., number of sessions, number of subscribers, number of seats, etc.) to OpenECOMP for tracking every license.	Must	10230
	Contract shall define the reporting process and the available reporting tools. The vendor will have to agree to the process that can be met by Service Provider reporting infrastructure.	Must	10240
	VNF vendors shall enumerate all of the open source licenses their VNF(s) incorporate.	Must	10250
	Audits of Service Provider’s business must not be required.	Must	10260

	Vendor functions and metrics that require additional infrastructure such as a vendor license server for deployment shall not be supported.	Must	10270
	Provide clear measurements for licensing purposes to allow automated scale up/down by the management system.	Must	10280
	The vendor must provide the ability to scale up a vendor supplied product during growth and scale down a vendor supplied product during decline without “real-time” restrictions based upon vendor permissions.	Must	10290
	A universal license key must be provided per VNF to be used as needed by services (i.e., not tied to a VM instance) as the recommended solution. The vendor may provide pools of Unique VNF License Keys, where there is a unique key for each VNF instance as an alternate solution. Licensing issues should be resolved without interrupting in-service VNFs.	Must	10300

3. Configuration Management

OpenECOMP interacts directly with VNFs through its Network and Application Adapters to perform configuration activities within NFV environment. These activities include service and resource configuration/reconfiguration, automated scaling of resources, service and resource removal to support runtime lifecycle management of VNFs and services. The Adapters employ a model driven approach along with standardized APIs provided by the VNF developers to configure resources and manage their runtime lifecycle.

3.1 NETCONF Standards and Capabilities

OpenECOMP Controllers and their Adapters utilize device YANG model and NETCONF APIs to make the required changes in the VNF state and configuration. The VNF providers must provide the Device YANG model and NETCONF server supporting NETCONF APIs to comply with target OpenECOMP and industry standards.

Table 2. VNF Configuration

Principle	Description	Type	ID #
3.1.1 Configuration Management	Virtual Network functions (VNFs) must include a NETCONF server enabling runtime configuration and lifecycle management capabilities. The NETCONF server embedded in VNFs shall provide a NETCONF interface fully defined by supplied YANG models.	Must	11010
3.1.2 NETCONF Server Requirements	NETCONF server connection parameters shall be configurable during virtual machine instantiation through Heat templates where SSH keys, usernames, passwords, SSH service and SSH port numbers are Heat template parameters.	Must	11020
	Following protocol operations must be implemented: **close-session()**- Gracefully close the current session. **commit(confirmed, confirm-timeout)** - Commit candidate configuration datastore to the running configuration.	Must	11030

	copy-config(target, source) - Copy the content of the configuration datastore source to the configuration datastore target. **delete-config(target) -** Delete the named configuration datastore target. **discard-changes()** - Revert the candidate configuration datastore to the running configuration **edit-config(target, default-operation, test-option, error-option, config)** - Edit the target configuration datastore by merging, replacing, creating, or deleting new config elements. **get(filter)** - Retrieve (a filtered subset of a) the running configuration and device state information. This should include the list of VNF supported schemas. **get-config(source, filter)** - Retrieve a (filtered subset of a) configuration from the configuration datastore source. **kill-session(session)** - Force the termination of **session**. **lock(target)** - Lock the configuration datastore target. **unlock(target)** - Unlock the configuration datastore target.		
	Following protocol operations should be implemented: **copy-config(target, source) -** Copy the content of the configuration datastore source to the configuration datastore target. **delete-config(target) -** Delete the named configuration datastore target. **get-schema(identifier, version, format) -** Retrieve the Yang schema.	Should	11040
	All configuration data shall be editable through a NETCONF <*edit-config*> operation. Proprietary NETCONF RPCs that make configuration changes are not sufficient.	Must	11050
	By default, the entire configuration of the VNF must be retrievable via NETCONF's <get-config> and <edit-config>, independently of whether it was configured via NETCONF or other mechanisms.	Must	11060
	The **:partial-lock** and **:partial-unlock** capabilities, defined in RFC 5717 must be supported. This allows multiple independent clients to each write to a different part of the <running> configuration at the same time.	Must	11070
	The ***:rollback-on-error*** value for the <error-option> parameter to the <edit-config> operation must be supported. If any error occurs during the requested edit operation, then the target database (usually the running configuration) will be left affected. This provides an 'all-or-nothing' edit mode for a single <edit-config> request.	Must	11080
	The server must support the **:startup** capability. It will allow the running configuration to be copied to this special database. It can also be locked, and unlocked.	Must	11090
	The **:url** value must be supported to specify protocol operation source and target parameters. The capability URI for this feature will indicate which schemes (e.g., file, https, sftp) that the server supports within a particular URL value. The 'file' scheme allows for	Must	11100

	editable local configuration databases. The other schemes allow for remote storage of configuration databases.		
	At least one of the capabilities ***:candidate*** or ***:writable-running*** must be implemented. If both **:candidate** and **:writable-running** are provided then two locks should be supported.	Must	11110
	The server must fully support the XPath 1.0 specification for filtered retrieval of configuration and other database contents. The 'type' attribute within the <filter> parameter for <get> and <get-config> operations may be set to 'xpath'. The 'select' attribute (which contains the XPath expression) will also be supported by the server. A server may support partial XPath retrieval filtering, but it cannot advertise the **:xpath** capability unless the entire XPath 1.0 specification is supported.	Must	11120
	The ***:validate*** capability must be implemented.	Must	11130
	If ***:candidate*** is supported, ***:confirmed-commit*** must be implemented.	Must	11140
	The ***:with-defaults*** capability [RFC6243] shall be implemented.	Must	11150
	Data model discovery and download as defined in [RFC6022] shall be implemented.	Must	11160
	NETCONF Event Notifications [RFC5277] should be implemented.	Should	11170
	All data models shall be defined in YANG [RFC6020], and the mapping to NETCONF shall follow the rules defined in this RFC.	Must	11180
	The data model upgrade rules defined in [RFC6020] section 10 should be followed. All deviations from section 10 rules shall be handled by a built-in automatic upgrade mechanism.	Must	11190
	The VNF must support parallel and simultaneous configuration of separate objects within itself.	Must	11200
	Locking is required if a common object is being manipulated by two simultaneous NETCONF configuration operations on the same VNF within the context of the same writable running data store (e.g., if an interface parameter is being configured then it should be locked out for configuration by a simultaneous configuration operation on that same interface parameter).	Must	11210
	Locking must be applied based on the sequence of NETCONF operations, with the first configuration operation locking out all others until completed.	Must	11220
	If a VNF needs to lock an object for configuration, the lock must be permitted at the finest granularity to avoid blocking simultaneous configuration operations on unrelated objects (e.g., BGP configuration should not be locked out if an interface is being configured, Entire Interface configuration should not be locked out if a non-overlapping parameter on the interface is being configured). The granularity of the lock must be able to be specified via a restricted or full XPath expression.	Must	11230
	All simultaneous configuration operations should guarantee the VNF configuration integrity (for example: if a change is attempted to the BUM filter rate from multiple interfaces on the same EVC,	Must	11240

	then they need to be sequenced in the VNF without locking either configuration method out)		
	To prevent permanent lock-outs, locks must be released: a. when/if a session applying the lock is terminated (e.g., SSH session is terminated) b. the corresponding <partial-unlock> operation succeeds c. a user configured timer has expired forcing the NETCONF SSH Session termination (i.e., product must expose a configuration knob for a user setting of a lock expiration timer) Additionally, to guard against hung NETCONF sessions, another NETCONF session should be able to initiate the release of the lock by killing the session owning the lock, using the <kill-session> operation.	Must	11250
	The VNF should support simultaneous <commit> operations within the context of this locking requirements framework.	Must	11260
	The supplied YANG code and associated NETCONF servers shall support all operations, administration and management (OAM) functions available from the supplier for VNFs.	Must	11270
	Sub tree filtering must be supported.	Must	11280
	Heartbeat via a <get> with null filter shall be supported.	Must	11290
	Get-schema (ietf-netconf-monitoring) must be supported to pull YANG model over session.	Must	11300

	The supplied YANG code shall be validated using the open source pyang[2] program using the following commands:
	`$ pyang --verbose --strict <YANG-file-name(s)>`
	`$ echo $!`
		Must	11310

	The `echo` command must return a zero value otherwise the validation has failed.	Must	11320
	The supplier shall demonstrate mounting the NETCONF server on OpenDaylight (client) and: • Modify, update, change, rollback configurations using each configuration data element. • Query each state (non-configuration) data element. • Execute each YANG RPC. • Receive data through each notification statement.	Must	11330

The following table provides the Yang models that suppliers must conform, and those where applicable, that suppliers need to use.

Table 3. YANG Models

RFC	Description	Type	ID #
RFC 6020	YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)	Must	12010

[2] https://github.com/mbj4668/pyang

RFC 6022	YANG module for NETCONF monitoring	Must	12020
RFC 6470	NETCONF Base Notifications	Must	12030
RFC 6244	An Architecture for Network Management Using NETCONF and YANG	Must	12040
RFC 6087	Guidelines for Authors and Reviewers of YANG Data Model Documents	Must	12050
RFC 6991	Common YANG Data Types	Should	12060
RFC 6536	NETCONF Access Control Model	Should	12070
RFC 7223	A YANG Data Model for Interface Management	Should	12080
RFC 7224	IANA Interface Type YANG Module	Should	12090
RFC 7277	A YANG Data Model for IP Management	Should	12100
RFC 7317	A YANG Data Model for System Management	Should	12110
RFC 7407	A YANG Data Model for SNMP Configuration	Should	12120

The NETCONF server interface shall fully conform to the following NETCONF RFCs.

Table 4. NETCONF RFCs

RFC	Description	Type	ID #
RFC 4741	NETCONF Configuration Protocol	Must	12130
RFC 4742	Using the NETCONF Configuration Protocol over Secure Shell (SSH)	Must	12140
RFC 5277	NETCONF Event Notification	Must	12150
RFC 5717	Partial Lock Remote Procedure Call	Must	12160
RFC 6241	NETCONF Configuration Protocol	Must	12170
RFC 6242	Using the Network Configuration Protocol over Secure Shell	Must	12180

3.2 VNF REST APIs

Healthcheck is a command for which no NETCONF support exists. Therefore, this must be supported using a RESTful interface which we have defined.

The VNF must provide two REST formatted RPCs to support Healthcheck queries via the GET method over HTTP(s).

Table 5. VNF REST APIs

Principal	Description	Type	ID #

3.2.1
REST APIs
	GET /check - The **/check** RPC, executes a vendor-defined VNF Healthcheck over the scope of the entire VNF (e.g if there are multiple VMs, then run a health check, as appropriate, for all VMs). /check returns a 200 OK if the test passes and a 50x response if the test fails. The precise failure code may depend upon type of failure (process error, overload etc.). A JSON object is returned indicating state, scope identifier, time-stamp and info field as well as an optional fault field.
	For example:
	503 Threshold Exceeded
	{
	"identifier": "scope represented",
	"info": "System threshold exceeded details",
	"fault":
	{
	"cpuOverall": 0.80,
	"cpuThreshold": 0.45
	},
	"time": "01-01-1000:0000"
	}
		Must	12190

	GET /status - The **/status** RPC returns a 200 OK code and state of the VNF (resources utilized) in the form of a nested JSON response (multiple resources for each VM within the VNF).
	For example:
	{
	"identifier": "scope represented",
	"stats":
	{
	"vm_123":
	{
	"cpuOverall": 0.32
	"usedMemory": 1000
	"totalMemory": 2000
	}
	},
	"time": "01-01-1000:0000"
	}
		Must	12200

3.3 OpenECOMP Controller APIs and Behavior

OpenECOMP Controllers support the following operations which act directly upon the VNF. Most of these utilize the NETCONF interface. There are additional commands in use but these either act internally on Controller itself or depend upon network cloud components for implementation. Those actions do not put any special requirement on the VNF provider.

The following table summarizes how the VNF must act in response to commands from OpenECOMP.

Table 6. OpenECOMP Controller APIs

Action	Description	VNF Action	NETCONF COMMANDs
Action Status	Queries OpenECOMP Controller for the current state of a previously submitted runtime LCM (Lifecycle Management) action.	Checks if VNF is busy. Current operation depends on a completion code from any previous operation. In the future a positive acknowledgement of busy status may be useful to handle ambiguous conditions. However, at this time none is being used.	<none>
Audit	Compare active configuration against a configuration stored in OpenECOMP's configuration store.	Retrieve running configuration and device state information. Get-config updates the config tree which can then be compared to the stored current config in the OpenECOMP database.	get-config
Check Lock	Returns true when the given VNF has been locked.	VnfLock may have been used to lock the VNF. There is currently no way to query lock state in NETCONF so locked state is managed internally by OpenECOMP.	<none>
Configure	Configures the target VNF or VNFC.	The <edit-config> operation loads all or part of a specified configuration data set to the specified target VNF.	edit-config, commit
Health Check	Executes a VNF health check and returns the result. A health check is VNF-specific.	The OpenECOMP health check interface is defined over REST and requires the target VNF to expose a standardized HTTP(S) interface for that purpose. Return the health status of the VNF by performing (via any vendor-specific means) internal checks of needed resources, process states, etc. The specific errors returned can be used to indicate the source of the problem. OpenECOMP will generate error events for all reported health problems.	**REST API** GET /check GET /status
Live Upgrade	Upgrades the target VNF to a new version without interrupting VNF operation.	Supported today on some VNFs via CLI only (the CLI use is an interim solution)	load, restart
<mount>	This is an internal Controller operation used to create config-tree and operations tree in the controller.	OpenECOMP must retrieve a schema definition from the VNF. The NETCONF server returns the	get, get-schema

		requested schema. During session establishment OpenECOMP issues a NETCONF <get> command which will retrieve all running configuration parameters, all running operational parameters and a list of NETCONF schemas. OpenECOMP retrieves the schemas to create a Yang model describing the parameters used by the VNF and legal values for each parameter (patterns or ranges). The schemas tell OpenECOMP what parameters can be set and what constitute legal values for those parameters.	
Config Modify	The ConfigModify LCM action affects only a subset of the total configuration data of a VNF. It can be used to change specific parameters across a number of separate instances for the same VnfcType without changing instance specific values of each. It can also be used to make successive changes to a number of parameters where those changes are considered cumulative. Thus each ConfigModify invocation leaves previous values untouched and only edits the parameters which are sent to OpenECOMP.	The <edit-config> operation loads only a part of the full set of configuration parameters to the specified target configuration without changing any existing parameters.	edit-config, commit
Config Save	Saves a VNF's running configuration into the configuration store in OpenECOMP, for later retrieval.	(optional) If copy-config to a local file is supported by the VNFC this command is used to store the running config locally in order to save time on any subsequent Reconfigure. To support this action, the VNF must allow <copy-config> to save to a local file and must support subsequent retrieval of the copied configuration back to the running configuration. If this capability is not supported, OpenECOMP will still function, but updates will take longer.	copy-config, delete-config
Reconfigure	Reconfigure a VNF to some previously stored baseline configuration stored by a previous ConfigSetBaseline.	If a previous config has been saved locally, and designated as the baseline configuration, use quick restore (<copy-config> from file). If the restore fails, fallback to a process of changing the configuration value by value using <edit-config> and referencing the SQL values stored by APP-C.	edit-config or copy-config
Config Restore	Reconfigure a VNF to some previously stored baseline configuration stored by a previous ConfigSetBaseline.	If a previous config has been saved locally use quick restore (<copy-config> from file). If the restore fails, fallback to a process of changing the configuration value by value using <edit-config>.	edit-config or copy-config

Sync	Updates the current configuration of a VNF in OpenECOMP's SQL configuration storage repository by uploading the running config. Useful if the current and running configurations do not match as determined by a previous Audit call.	Retrieve running config from VNF	get, get-config
VNFLock	Lock or Unlock a VNF to ensure exclusive access during a series of critical steps.	The lock operation allows the client to lock the configuration system of a device.	lock, unlock

4. Monitoring & Management

This section addresses data collection and event processing functionality that is directly dependent on the interfaces provided by the VNFs' APIs. These can be in the form of Asynchronous interfaces for event, fault notifications, and autonomous data streams. They can also be Synchronous interfaces for on-demand requests to retrieve various performance, usage, and other event information.

The target direction for VNF interfaces is to employ APIs that are implemented utilizing standardized messaging and modeling protocols over standardized transports. Migrating to a virtualized environment presents a tremendous opportunity to eliminate the need for proprietary interfaces for vendor equipment while removing the traditional boundaries between Network Management Systems and Element Management Systems. Additionally, VNFs provide the ability to instrument the networking applications by creating event records to test and monitor end-to-end data flow through the network, similar to what physical or virtual probes provide without the need to insert probes at various points in the network. The VNF vendors must be able to provide the aforementioned set of required data directly to the OpenECOMP collection layer using standardized interfaces.

4.1 Transports and Protocols Supporting Resource Interfaces

Delivery of data from VNFs to OpenECOMP must use the same common transport mechanisms and protocols for all VNFs. Transport mechanisms and protocols have been selected to enable both high volume and moderate volume datasets, as well as asynchronous and synchronous communications over secure connections. The specified encoding provides self-documenting content, so data fields can be changed as needs evolve, while minimizing changes to data delivery.

The term 'Event Record' is used throughout this document to represent various forms instrumentation/telemetry made available by the VNF including, faults, status events and various other types of VNF measurements and logs. Headers received by themselves must be used as heartbeat indicators. The common structure and delivery protocols for other types of data will be given in future versions of this document as we get more insight into data volumes and required processing.

In the following guidelines we provide options for encoding, serialization and data delivery. Agreements between Service Providers and VNF vendors shall determine which encoding, serialization and delivery method to use for particular data sets. The selected methods must be agreed to prior to the on-boarding of the VNF into OpenECOMP design studio.

Table 7. Monitoring & Management

Principle	Description	Type	ID #
4.1.1 Encoding and Serialization	Content delivered from VNFs to OpenECOMP is to be encoded and serialized using JSON (option 1). High-volume data is to be encoded and serialized using Avro, where Avro data format are described using JSON (option 2)[3]. • JSON plain text format is preferred for moderate volume data sets (option 1), as JSON has the advantage of having well-understood simple processing and being human-readable without additional decoding. Examples of moderate volume data sets include the fault alarms and performance alerts, heartbeat messages, measurements used for VNF scaling and syslogs. • Binary format using Avro is preferred for high volume data sets (option 2) such as mobility flow measurements and other high-volume streaming events (such as mobility signaling events or SIP signaling) or bulk data, as this will significantly reduce the volume of data to be transmitted. As of the date of this document, all events are reported using plain text JSON and REST. • Avro content is self-documented, using a JSON schema. The JSON schema is delivered along with the data content (http://avro.apache.org/docs/current/). This means the presence and position of data fields can be recognized automatically, as well as the data format, definition and other attributes. Avro content can be serialized as JSON tagged text or as binary. In binary format, the JSON schema is included as a separate data block, so the content is not tagged, further compressing the volume. For streaming data, Avro will read the schema when the stream is established and apply the schema to the received content. • In the future, we may consider support for other types of encoding & serialization (e.g., gRPC) based on industry demand.	Must	13010
4.1.2 Reporting Frequency	The frequency that asynchronous data is delivered will vary based on the content and how data may be aggregated or grouped together. For example, alarms and alerts are expected to be delivered as soon as they appear. In contrast, other content, such as performance measurements, KPIs or reported network signaling may have various ways of packaging and delivering content. Some content should be streamed immediately; or content may be monitored over a time interval, then packaged as collection of records and delivered as block; or data may be collected until a package of a certain size has been collected; or content may be summarized statistically over a time interval, or computed as a KPI, with the summary or KPI being delivered. • We expect the reporting frequency to be configurable depending on the virtual network function's needs for management. For example, Service Provider may choose to vary the frequency of collection between normal and trouble-shooting scenarios.	Must	13020

[3] This option is not currently supported in OpenECOMP and it is currently under consideration.

	• Decisions about the frequency of data reporting will affect the size of delivered data sets, recommended delivery method, and how the data will be interpreted by OpenECOMP. However, this should not affect deserialization and decoding of the data, which will be guided by the accompanying JSON schema.		
4.1.3 Addressing and Delivery Protocol	OpenECOMP destinations can be addressed by URLs for RESTful data PUT. Future data sets may also be addressed by host name and port number for TCP streaming, or by host name and landing zone directory for SFTP transfer of bulk files. • REST using HTTPS delivery of plain text JSON is preferred for moderate sized asynchronous data sets, and for high volume data sets when feasible. • VNFs must have the capability of maintaining a primary and backup DNS name (URL) for connecting to OpenECOMP collectors, with the ability to switch between addresses based on conditions defined by policy such as time-outs, and buffering to store messages until they can be delivered. At its discretion, the service provider may choose to populate only one collector address for a VNF. In this case, the network will promptly resolve connectivity problems caused by a collector or network failure transparently to the VNF. • VNFs will be configured with initial address(es) to use at deployment time. After that the address(es) may be changed through OpenECOMP-defined policies delivered from OpenECOMP to the VNF using PUTs to a RESTful API, in the same way that other controls over data reporting will be controlled by policy. • Other options are expected to include: o REST delivery of binary encoded data sets. o TCP for high volume streaming asynchronous data sets and for other high volume data sets. TCP delivery can be used for either JSON or binary encoded data sets. o SFTP for asynchronous bulk files, such as bulk files that contain large volumes of data collected over a long time interval or data collected across many VNFs. This is not preferred. Preferred is to reorganize the data into more frequent or more focused data sets, and deliver these by REST or TCP as appropriate. o REST for synchronous data, using RESTCONF (e.g., for VNF state polling). • The OpenECOMP addresses as data destinations for each VNF must be provided by OpenECOMP Policy, and may be changed by Policy while the VNF is in operation. We expect the VNF to be capable of redirecting traffic to changed destinations with no loss of data, for example from one REST URL to another, or from one TCP host and port to another.	Must	13030
4.1.4 Asynchronous and	VNFs are to deliver asynchronous data as data becomes available, or according to the configured frequency. The delivered data must be encoded using JSON or Avro, addressed and delivered as described in the previous paragraphs.	Must	13040

Synchronous Data Delivery	VNFs are to respond to data requests from OpenECOMP as soon as those requests are received, as a synchronous response.	Must	13050
	Synchronous communication must leverage the RESTCONF/NETCONF framework used by the OpenECOMP configuration subsystem. This shall include using YANG configuration models and RESTCONF (https://tools.ietf.org/html/draft-ietf-netconf-restconf-09#page-46).	Must	13060
	The VNF must respond with content encoded in JSON, as described in the RESTCONF specification. This way the encoding of a synchronous communication will be consistent with Avro.	Must	13070
	OpenECOMP may request the VNF to deliver the current data for any of the record types defined in Section 4.2 below. The VNF must respond by returning the requested record, populated with the current field values. (Currently the defined record types include fault fields, mobile flow fields, measurements for VNF scaling fields, and syslog fields. Other record types will be added in the future as they become standardized and are available).	Must	13080
	OpenECOMP may request the VNF to deliver granular data on device or subsystem status or performance, referencing the YANG configuration model for the VNF. The VNF must respond by returning the requested data elements.	Must	13090
	YANG models can be translated to and from JSON (https://trac.tools.ietf.org/id/draft-lhotka-netmod-yang-json-00.html), meaning YANG configuration and content can be represented via JSON, consistent with Avro, as described in “Encoding and Serialization” section.	Must	13100
4.1.5 Security	VNFs must support secure connections and transports.	Must	13110
	Access to OpenECOMP and to VNFs, and creation of connections, must be controlled through secure credentials, log-on and exchange mechanisms.	Must	13120
	Data in motion must be carried only over secure connections.	Must	13130
	Service Providers require that any content containing Sensitive Personal Information (SPI) or certain proprietary data must be encrypted, in addition to applying the regular procedures for securing access and delivery.	Must	13140

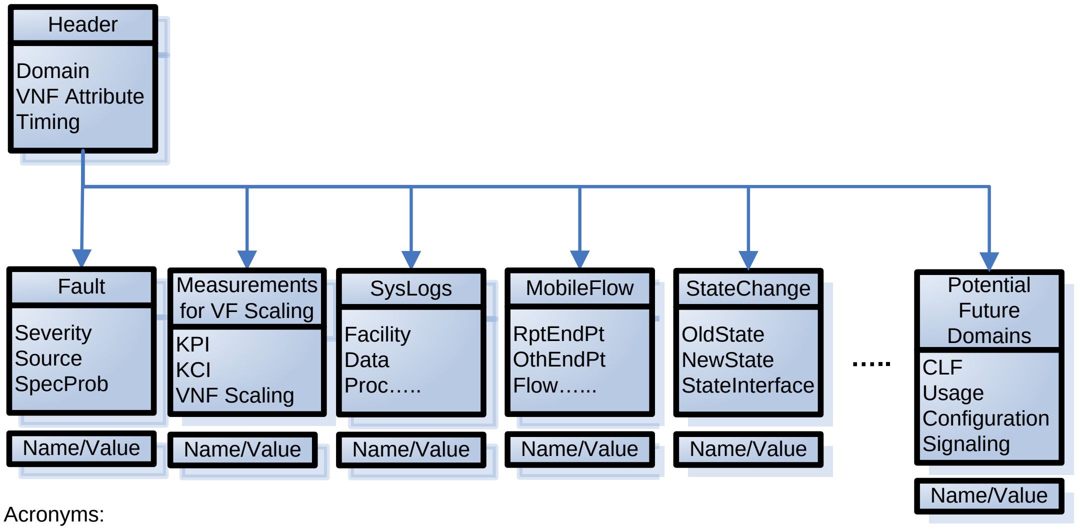
4.2 Data Model for Event Records

This section describes the data model for the collection of telemetry data from VNFs by Service Providers (SPs) to manage VNF health and runtime lifecycle. This data model is referred to as the VNF Event Streaming (VES) specifications. OPNFV has a VES project[4] that provides a holistic solution for OpenStack’s internal telemetry to manage Application (VNFs), Physical and virtual infrastructure (compute, storage, network devices), and virtual infrastructure managers (cloud controllers, SDN controllers). Note that any configurable parameters for these data records (e.g., frequency, granularity, policy-based configuration) will be managed using the “Configuration” framework described in the prior sections.

The Data Model consists of:

[4] https://wiki.opnfv.org/display/PROJ/VNF+Event+Stream

- Common Header Record: This precedes each of the domain-specific records.
- Domain Specific Event Records. This version of the document specifies the model for Fault, Performance, Syslog, State Change, and Mobile Flow records. In the future, these will be extended to support other types of records (e.g., Signaling or control plane messages, probe-less monitoring records, Status Records, Security records, etc.). Each of these records allows additional fields (name value pairs) for extensibility. The VNF vendors can use these VNF-specific additional fields to provide additional information that may be relevant to the managing systems.



Acronyms:
CLF – Control Loop Functions
KCI – Key Capacity Indicators
KPI – Key Performance Indicators
VNF – Virtual Network Function

Figure 1. Data Model for Event Records

4.3 Event Records - Data Structure Description

The data structure for event records consists of a Header Block and zero (heartbeat would only have header) or more event domain blocks (e.g., Common Fault Event domain, Common Performance Event domain, Common Syslog Event domain, Specialized Mobile Flow Event Domain, etc.). The tables in Appendix A present the details for the Common Header and other specific record types.

4.3.1 Common Event Header

The common header that precedes any of the domain-specific records contains information identifying the type of record to follow, information about the sender and other identifying characteristics related to timestamp, sequence number, etc. The table A.1 in Appendix A describes the structure for the common header.

4.3.2 Event Data Structure – Fault Fields

The Fault Record, describing a condition in the Fault domain, contains information about the fault such as the entity under fault, the severity, resulting status, etc. The table A.2 in Appendix A describes the structure for the fault record.

4.3.3 Event Data Structure – Measurements for VNF Scaling Fields

The VNF Scaling Record contains information about VNF resource structure and its condition to help in the management of the resources for purposes of elastic scaling. The table A.3 in Appendix A describes the structure for the VNF Scaling record.

4.3.4 Event Data Structure – Syslog Fields

The Syslog Record provides a structure for communicating any type of information that may be logged by the VNF. It can contain information about system internal events, status, errors, etc. The table A.4 in Appendix A describes the structure for the Syslog record.

4.3.5 Event Data Structure – State Change Fields

The State change domain provides a structure for communicating information about data flow through the VNF. It can contain information about state change related to Physical device that is reported by VNF. As an example when cards or port name of the entity that has changed state. The table A.5 in Appendix A describes the structure of the State Change record.

4.3.6 Event Data Structure – Mobile Flow Fields

The Mobile Flow Record provides a structure for communicating information about data flow through the VNF. It can contain information about connectivity and data flows between serving elements for mobile service, such as between LTE reference points, etc. The table A.6 in Appendix A describes the structure for the Mobile Flow record.

Appendix A – Data Record Format

The following provides additional information on the event record formats for the following data structures (for complete information, please refer to AT&T Service Specification; Service: VES Event Listener, revision 4.0, dated Jan 5th, 2017):

- Common Event Header
- Fault Fields
- Measurements for VF Scaling Fields
- Syslog Fields
- State Change Fields
- Mobile Flow Fields

A.1 EVENT RECORDS – Common Event Header

Field	Type	Required?	Description
version	number	No	Version of the event header (currently: 2.0)
eventType	string	No	Unique event topic name
domain	string	Yes	Event domain enumeration: 'fault', 'heartbeat', 'measurementsForVfScaling', 'mobileFlow', 'other', 'stateChange', 'syslog', 'thresholdCrossingAlert'
eventId	string	Yes	Event key that is unique to the event source
sourceId	string	No	UUID identifying the entity experiencing the event issue (note: the AT&T internal enrichment process shall ensure that this field is populated)
sourceName	string	Yes	Name of the entity experiencing the event issue
functionalRole	string	Yes	Function of the event source e.g., eNodeB, MME, PCRF
reportingEntityId	string	No	UUID identifying the entity reporting the event, for example an OAM VM (note: the AT&T internal enrichment process shall ensure that this field is populated)
reportingEntityName	string	Yes	Name of the entity reporting the event, for example, an OAM VM
priority	string	Yes	Processing priority enumeration: 'High', 'Medium', 'Normal', 'Low'
startEpochMicrosec	number	Yes	the earliest unix time aka epoch time associated with the event from any component--as microseconds elapsed since 1 Jan 1970 not including leap seconds
lastEpochMicrosec	number	Yes	the latest unix time aka epoch time associated with the event from any component--as microseconds elapsed since 1 Jan 1970 not including leap seconds
sequence	integer	Yes	Ordering of events communicated by an event source instance (or 0 if not needed)
internalHeader Fields	object	No	Fields (not supplied by event sources) that the VES Event Listener service can use to enrich the event if needed for efficient internal processing. This is an empty object which is intended to be defined separately by each provider implementing the VES Event Listener.

A.2 EVENT RECORDS – Fault Fields

Field	Type	Required?	Description

faultFieldsVersion	number	No	Version of the faultFields block (currently: 1.1)
eventSeverity	string	Yes	Event severity or priority enumeration: ‘CRITICAL’, ‘MAJOR’, ‘MINOR’, ‘WARNING’, ‘NORMAL’
eventSourceType	string	Yes	Examples: ‘other’, ‘router’, ‘switch’, ‘host’, ‘card’, ‘port’, ‘slotThreshold’, ‘portThreshold’, ‘virtualMachine’, ‘virtualNetworkFunction’
alarmCondition	string	Yes	Alarm condition reported by the device
specificProblem	string	Yes	Short description of the alarm or problem
vfStatus	string	Yes	Virtual function status enumeration: ‘Active’, ‘Idle’, ‘Preparing to terminate’, ‘Ready to terminate’, ‘Requesting Termination’
alarmtInterfaceA	string	No	Card, port, channel or interface name of the device generating the alarm
alarmAdditional Information	Name-value pair object array	No	Expressed as an array of name-value pairs which can be used to describe additional Information related to Alarm, such as Repair Action, Remedy code….May by serialized alarm payload: varbind list, original syslog message, notification parameters, etc. when event is generated via other means, should provide raw detail out of element.

A.3 EVENT RECORDS – Measurements for VF Scaling Fields

Field	Type	Required ?	Description
measurementsForVfScalingFieldsVersion	number	No	Version of the measurementsForVfScalingFields block (currently: 1.1)
additionalMeasurements	object array	No	Expressed as an array of measurementGroup objects, each of which contains a measurement group along with an array of name-value pair fields. Can be used to provide additional measurement fields
aggregateCpuUsage	number	No	Aggregate CPU usage of the VM on which the VNFC reporting the event is running
codecUsageArray	Array	No	Expressed as an array of codecsInUse objects, each of which contains a string identifying the codec, along with a number indicating the number of such codecs in use.
concurrentSessions	number	No	Peak concurrent sessions for the VM or VNF (depending on the context) over the measurementInterval
configuredEntities	number	No	Depending on the context over the measurementInterval: peak total number of users, subscribers, devices, adjacencies, etc., for the VM, or peak total number of subscribers, devices, etc., for the VNF
cpuUsageArray	object array	No	Expressed as an array of cpuUsage objects, each of which contains a string identifying the cpu, along with a number indicating the cpu usage percentage.
errors	object	No	Provides receive and transmit errors and discards
featureUsageArray	object array	No	Expressed as an array of featuresInUse objects, each of which contains a string identifying the feature, along with a number indicating the number of times the feature was used.
filesystemUsageArray	object array	No	Expressed as an array of filesystemUsage objects, each of which contains a string identifying the filesystem, along with numbers indicating the configured and used block and ephemeral capacity in GB, along with the input-output operations per second for block and ephemeral storage.
latencyDistribution	object array	No	Expressed as an array of latencyBucketMeasure objects, defined by two numbers indicating the low end and high end of the latency bucket (in ms), plus a number indicating the number of counts in that bucket.
meanRequestLatency	number	No	Mean seconds required to respond to each request for the VM on which the VNFC reporting the event is running
measurementInterval	number	Yes	Interval over which measurements are being reported in seconds
memoryConfigured	number	No	Memory in MB configured in the VM on which the VNFC reporting the event is running
memoryUsed	number	No	Memory usage in MB of the VM on which the VNFC reporting the event is running
numberOfMediaPortsInUse	Number	No	Number of media ports in use
requestRate	number	No	Peak rate of service requests per second to the VNF over the measurementInterval
vnfcScalingMetric	number	No	Represents busy-ness of the VNF from 0 to 100 as reported by the VNFC

vNicUsageArray	object array	No	Expressed as an array of vNicUsage objects, each of which contains a string identifying the vNic, along with numbers indicating the unicast, multicast, broadcast and total number of packets received and sent, plus the total number of bytes in and out of the vNic (in MB).

A.4 EVENT RECORDS – Syslog Fields

Field	Type	Required?	Description
syslogFieldsVersion	number	No	Version of the syslogFields block (currently: 2.0)
additionalFields	Name-value pair object array	No	Expressed as an array of name-value pairs which can be used to describe additional syslog fields if needed
eventSourceHost	string	No	Hostname of the device
eventSourceType	string	Yes	Examples: ‘other’, ‘router’, ‘switch’, ‘host’, ‘card’, ‘port’, ‘slotThreshold’, ‘portThreshold’, ‘virtualMachine’, ‘virtualNetworkFunction’
syslogFacility	number	No	Numeric code from 0 to 23 for facility: 0 kernel messages 1 user-level messages 2 mail system 3 system daemons 4 security/authorization messages 5 messages generated internally by syslogd 6 line printer subsystem 7 network news subsystem 8 UUCP subsystem 9 clock daemon 10 security/authorization messages 11 FTP daemon 12 NTP subsystem 13 log audit 14 log alert 15 clock daemon (note 2) 16 local use 0 (local0) 17 local use 1 (local1) 18 local use 2 (local2) 19 local use 3 (local3) 20 local use 4 (local4) 21 local use 5 (local5) 22 local use 6 (local6) 23 local use 7 (local7)
syslogMsg	string	Yes	Syslog message

syslogPri	number	No	0-192 Combined Severity and Facility
syslogProc	string	No	Identifies the application that originated the message
syslogProcId	number	No	A change in the value of this field indicates a discontinuity in syslog reporting
syslogSData	string	No	Syslog structured data consisting of a structured data Id followed by a set of key value pairs (see below for an example) **Note: SD-ID may not be present if syslogSdId is populated
syslogSdId	string	No	0-32 char in format name@number, ie ourSDID@32473
syslogSev	string	No	Numerical Code for Severity (derived from syslogPri: remaider of syslogPri / 8) 0 Emergency: system is unusable 1 Alert: action must be taken immediately 2 Critical: critical conditions 3 Error: error conditions 4 Warning: warning conditions 5 Notice: normal but significant condition 6 Informational: informational messages 7 Debug: debug-level messages
syslogTag	string	Yes	MsgId indicating the type of message such as ‘TCPOUT’ or ‘TCPIN’; ‘NILVALUE’ should be used when no other value can be provided
syslogVer	number	No	IANA assigned version of the syslog protocol specification (typically ‘1’)

A.5 EVENT RECORDS – State Change Fields

Field	Type	Required?	Description
stateChangeFieldsVersion	number	No	Version of the stateChangeFields block (currently: 1.1)
additionalFields	Name-value pair object array	No	Expressed as an array of name-value pairs which can be used to describe additional state change fields if needed
newState	string	Yes	New state of the entity: ‘inService’, ‘maintenance’, ‘outOfService’
oldState	string	Yes	Previous state of the entity: ‘inService’, ‘maintenance’, ‘outOfService’
stateInterface	string	Yes	Card or port name of the entity that changed state

A.6 EVENT RECORDS – Mobile Flow Fields

Field	Type	Required?	Description
mobileFlowFieldsVersion	number	No	Version of the mobileFlowFields block (currently: 1.2)
additionalFields	field	No	Additional mobileFlow fields if needed Similar to adddiotnalFileds in fault domain
applicationType	string	No	Application type inferred
appProtocolType	string	No	Application protocol
appProtocolVersion	string	No	Application version
cid	string	No	Cell Id
connectionType	string	No	Abbreviation referencing a 3GPP reference point e.g., S1-U, S11, etc
ecgi	string	No	Evolved Cell Global Id
flowDirection	string	Yes	Flow direction, indicating if the reporting node is the source of the flow or destination for the flow
gtpPerFlowMetrics	object	Yes	Mobility GTP Protocol per flow metrics (see below)
gtpProtocolType	string	No	GTP protocol
gtpVersion	string	No	GTP protocol version
httpHeader	string	No	HTTP request header, if the flow connects to a node referenced by HTTP
Imei	string	No	IMEI for the subscriber UE used in this flow, if the flow connects to a mobile device
Imsi	string	No	IMSI for the subscriber UE used in this flow, if the flow connects to a mobile device
ipProtocolType	string	Yes	IP protocol type e.g., TCP, UDP, RTP...
ipVersion	string	Yes	IP protocol version e.g., IPv4, IPv6
Lac	string	No	Location area code
Mcc	string	No	Mobile country code
Mnc	string	No	Mobile network code
msisdn	string	No	MSISDN for the subscriber UE used in this flow, as an integer, if the flow connects to a mobile device

otherEndpointIpAddress	string	Yes	IP address for the other endpoint, as used for the flow being reported on
otherEndpointPort	string	Yes	IP Port for the reporting entity, as used for the flow being reported on
otherFunctionalRole	string	No	Functional role of the other endpoint for the flow being reported on e.g., MME, S-GW, P-GW, PCRF...
Rac	string	No	Routing area code
radioAccessTechnology	string	No	Radio Access Technology e.g., 2G, 3G, LTE
reportingEndpointIpAddr	string	Yes	IP address for the reporting entity, as used for the flow being reported on
reportingEndpointPort	string	Yes	IP port for the reporting entity, as used for the flow being reported on
Sac	string	No	Service area code
samplingAlgorithm	string	No	Integer identifier for the sampling algorithm or rule being applied in calculating the flow metrics if metrics are calculated based on a sample of packets, or 0 if no sampling is applied
Tac	string	No	Transport area code
tunnelId	string	No	Tunnel identifier
vlanId	string	No	VLAN identifier used by this flow
gtpPerFlowMetrics Object (referenced above)			
avgBitErrorRate	number	Yes	Average bit error rate
avgPacketDelayVariation	number	Yes	Average packet delay variation or jitter in milliseconds for received packets: Average difference between the packet timestamp and time received for all pairs of consecutive packets
avgPacketLatency	number	Yes	Average delivery latency
avgReceiveThroughput	number	Yes	Average receive throughput
avgTransmitThroughput	number	Yes	Average transmit throughput
durConnectionFailedStatus	number	No	Duration of failed state in milliseconds, computed as the cumulative time between a failed echo request and the next following successful error request, over this reporting interval
durTunnelFailedStatus	number	No	Duration of errored state, computed as the cumulative time between a tunnel error indicator and the next following non-errored indicator, over this reporting interval
flowActivatedBy	string	No	Endpoint activating the flow
flowActivationEpoch	number	Yes	Time the connection is activated in the flow (connection) being reported on, or transmission time of the first packet if activation time is not available
flowActivationMicrosec	number	Yes	Integer microseconds for the start of the flow connection
flowActivationTime	datetime	No	Time the connection is activated in the flow being reported on, or transmission time of the first packet if activation time is not available; with RFC 2822 compliant format: ‘Sat, 13 Mar 2010 11:29:05 -0800’
flowDeactivatedBy	string	No	Endpoint deactivating the flow
flowDeactivationEpoch	number	Yes	Time for the start of the flow connection, in integer UTC epoch time aka UNIX time

flowDeactivationMicrosec	number	Yes	Integer microseconds for the start of the flow connection
flowDeactivationTime	datetime	Yes	Transmission time of the first packet in the flow connection being reported on; with RFC 2822 compliant format: ‘Sat, 13 Mar 2010 11:29:05 -0800’
flowStatus	string	Yes	Connection status at reporting time as a working / inactive / failed indicator value
gtpConnectionStatus	string	No	Current connection state at reporting time
gtpTunnelStatus	string	No	Current tunnel state at reporting time
ipTosCountList	associative array	No	Array of key: value pairs where the keys are drawn from the IP Type-of-Service identifiers which range from '0' to '255', and the values are the count of packets that had those ToS identifiers in the flow
ipTosList	string	No	Array of unique IP Type-of-Service values observed in the flow where values range from '0' to '255'
largePacketRtt	number	No	large packet round trip time
largePacketThreshold	number	No	large packet threshold being applied
maxPacketDelayVariation	number	Yes	Maximum packet delay variation or jitter in milliseconds for received packets: Maximum of the difference between the packet timestamp and time received for all pairs of consecutive packets
maxReceiveBitRate	number	No	maximum receive bit rate"
maxTransmitBitRate	number	No	maximum transmit bit rate
mobileQciCosCountList	associative array	No	array of key: value pairs where the keys are drawn from LTE QCI or UMTS class of service strings, and the values are the count of packets that had those strings in the flow
mobileQciCosList	string	No	Array of unique LTE QCI or UMTS class-of-service values observed in the flow
numActivationFailures	number	Yes	Number of failed activation requests, as observed by the reporting node
numBitErrors	number	Yes	number of errored bits
numBytesReceived	number	Yes	number of bytes received, including retransmissions
numBytesTransmitted	number	Yes	number of bytes transmitted, including retransmissions
numDroppedPackets	number	Yes	number of received packets dropped due to errors per virtual interface
numGtpEchoFailures	number	No	Number of Echo request path failures where failed paths are defined in 3GPP TS 29.281 sec 7.2.1 and 3GPP TS 29.060 sec. 11.2
numGtpTunnelErrors	number	No	Number of tunnel error indications where errors are defined in 3GPP TS 29.281 sec 7.3.1 and 3GPP TS 29.060 sec. 11.1
numHttpErrors	number	No	Http error count
numL7BytesReceived	number	Yes	number of tunneled layer 7 bytes received, including retransmissions
numL7BytesTransmitted	number	Yes	number of tunneled layer 7 bytes transmitted, excluding retransmissions
numLostPackets	number	Yes	number of lost packets
numOutOfOrderPackets	number	Yes	number of out-of-order packets
numPacketErrors	number	Yes	number of errored packets

numPacketsReceivedExclRetrans	number	Yes	number of packets received, excluding retransmission
numPacketsReceivedInclRetrans	number	Yes	number of packets received, including retransmission
numPacketsTransmittedInclRetrans	number	Yes	number of packets transmitted, including retransmissions
numRetries	number	Yes	number of packet retries
numTimeouts	number	Yes	number of packet timeouts
numTunneledL7BytesReceived	number	Yes	number of tunneled layer 7 bytes received, excluding retransmissions
roundTripTime	number	Yes	Round Trip time
tcpFlagCountList	associative array	No	Array of key: value pairs where the keys are drawn from TCP Flags and the values are the count of packets that had that TCP Flag in the flow
tcpFlagList	string	No	Array of unique TCP Flags observed in the flow
timeToFirstByte	number	Yes	Time in milliseconds between the connection activation and first byte received