AT&T Service Specification

Service: *VES Event Listener*

|  |  |
| --- | --- |
| Document Number | xxxx |
| Revision | 5.3 |
| Revision Date | June 22, 2017 |
| Author | Rich Erickson |

© 2017, AT&T Intellectual Property.  All other rights reserved. AT&T and AT&T logos are trademarks of AT&T Intellectual Property. All marks, trademarks, and product names used in this document are the property of their respective owners.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:  This product includes software developed by the AT&T.
4. Neither the name of AT&T nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY AT&T INTELLECTUAL PROPERTY ''AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL AT&T INTELLECTUAL PROPERTY BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This document and the information contained herein (collectively, the "**Information**") is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) ("**You**" and "**Your**") by AT&T, on behalf of itself and its affiliates ("**AT&T**") for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely on a belief, that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU "AS IS" AND "WITH ALL FAULTS." AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

© 2017 AT&T Intellectual Property

All rights reserved.

AT&T and AT&T logos are trademarks of AT&T Intellectual Property.

All marks, trademarks, and product names used in this document are the property of their respective owners.

Change Log for the latest version of the document (for historical change logs, see the Appendix)

| Date | Revision | Description |
| --- | --- | --- |
| 6/22/2017 | v5.3 | * JSON Schema: created v28.3 by correcting an error in the sipSignalingFields: changed vnfVendorNameFields to vendorVnfNameFields. Embedded the new schema at the top of section 4. |

[1 Introduction 1](#_Toc485913929)

[1.1 Event Registration 1](#_Toc485913930)

[1.2 Naming Standards for eventName 1](#_Toc485913931)

[1.3 Support for Protocols Other Than HTTPS 2](#_Toc485913932)

[1.4 Versioning 3](#_Toc485913933)

[2 Security 4](#_Toc485913934)

[2.1.1 Sample Request and Response 4](#_Toc485913935)

[2.1.1.1 Sample Request 4](#_Toc485913936)

[2.1.1.2 Sample Success Response 5](#_Toc485913937)

[3 Resource Structure 6](#_Toc485913938)

[4 Common Event Format 7](#_Toc485913939)

[4.1 Common Event Datatypes 7](#_Toc485913940)

[4.1.1 Command List Processing Datatypes 7](#_Toc485913941)

[4.1.1.1 Datatype: command 7](#_Toc485913942)

[4.1.1.2 Datatype: commandList 7](#_Toc485913943)

[4.1.1.3 Datatype: eventDomainThrottleSpecification 8](#_Toc485913944)

[4.1.1.4 Datatype: eventDomainThrottleSpecificationList 8](#_Toc485913945)

[4.1.1.5 Datatype: eventThrottlingState 8](#_Toc485913946)

[4.1.1.6 Datatype: suppressedNvPairs 8](#_Toc485913947)

[4.1.2 Common Event Datatypes 9](#_Toc485913948)

[4.1.2.1 Datatype: event 9](#_Toc485913949)

[4.1.2.2 Datatype: eventList 9](#_Toc485913950)

[4.1.2.3 Datatype: field 9](#_Toc485913951)

[4.1.2.4 Datatype: jsonObject 10](#_Toc485913952)

[4.1.2.5 Datatype: jsonObjectInstance 10](#_Toc485913953)

[4.1.2.6 Datatype: key 10](#_Toc485913954)

[4.1.2.7 Datatype: namedArrayOfFields 11](#_Toc485913955)

[4.1.2.8 Datatype: requestError 11](#_Toc485913956)

[4.1.2.9 Datatype: vendorVnfNameFields 11](#_Toc485913957)

[4.1.3 ‘Common Event Header’ Datatypes 12](#_Toc485913958)

[4.1.3.1 Datatype: commonEventHeader 12](#_Toc485913959)

[4.1.3.2 Datatype: internalHeaderFields 13](#_Toc485913960)

[4.2 Technology Independent Datatypes 13](#_Toc485913961)

[4.2.1 ‘Fault’ Domain Datatypes 13](#_Toc485913962)

[4.2.1.1 Datatype: faultFields 13](#_Toc485913963)

[4.2.2 ‘Heartbeat’ Domain Datatypes 14](#_Toc485913964)

[4.2.2.1 Datatype: heartbeatFields 14](#_Toc485913965)

[4.2.3 ‘Measurements For VF Scaling’ Domain Datatypes 14](#_Toc485913966)

[4.2.3.1 Datatype: codecsInUse 14](#_Toc485913967)

[4.2.3.2 Datatype: cpuUsage 14](#_Toc485913968)

[4.2.3.3 Datatype: diskUsage 15](#_Toc485913969)

[4.2.3.4 Datatype: featuresInUse 18](#_Toc485913970)

[4.2.3.5 Datatype: filesystemUsage 18](#_Toc485913971)

[4.2.3.6 Datatype: latencyBucketMeasure 19](#_Toc485913972)

[4.2.3.7 Datatype: measurementsForVfScalingFields 19](#_Toc485913973)

[4.2.3.8 Datatype: memoryUsage 20](#_Toc485913974)

[4.2.3.9 Datatype: vNicPerformance 21](#_Toc485913975)

[4.2.4 ‘Other’ Domain Datatypes 23](#_Toc485913976)

[4.2.4.1 Datatype: otherFields 23](#_Toc485913977)

[4.2.5 ‘State Change’ Domain Datatypes 24](#_Toc485913978)

[4.2.5.1 Datatype: stateChangeFields 24](#_Toc485913979)

[4.2.6 ‘Syslog’ Domain Datatypes 24](#_Toc485913980)

[4.2.6.1 Datatype: syslogFields 24](#_Toc485913981)

[4.2.7 ‘Threshold Crossing Alert’ Domain Datatypes 26](#_Toc485913982)

[4.2.7.1 Datatype: counter 26](#_Toc485913983)

[4.2.8 Datatype: thresholdCrossingAlertFields 26](#_Toc485913984)

[4.3 Technology Specific Datatypes 27](#_Toc485913985)

[4.3.1 ‘Mobile Flow’ Domain Datatypes 27](#_Toc485913986)

[4.3.1.1 Datatype: gtpPerFlowMetrics 27](#_Toc485913987)

[4.3.1.2 Datatype: mobileFlowFields 30](#_Toc485913988)

[4.3.2 ‘SipSignaling’ Domain Datatypes 32](#_Toc485913989)

[4.3.2.1 Datatype: sipSignalingFields 32](#_Toc485913990)

[4.3.3 ‘Voice Quality’ Domain Datatypes 32](#_Toc485913991)

[4.3.3.1 Datatype: endOfCallVqmSummaries 32](#_Toc485913992)

[4.3.3.2 Datatype: voiceQualityFields 33](#_Toc485913993)

[5 Exceptions 34](#_Toc485913994)

[5.1 RESTful Web Services Exceptions 34](#_Toc485913995)

[5.2 Service Exceptions 34](#_Toc485913996)

[5.3 Policy Exceptions 35](#_Toc485913997)

[6 RESTful Web Services Definition 37](#_Toc485913998)

[6.1 REST Operation Overview 37](#_Toc485913999)

[6.1.1 REST Operation Summary 37](#_Toc485914000)

[6.1.2 Api Version 37](#_Toc485914001)

[6.1.3 Commands Toward Event Source Clients 37](#_Toc485914002)

[6.1.4 Buffering of Events 39](#_Toc485914003)

[6.2 Operation: publishAnyEvent 39](#_Toc485914004)

[6.2.1 Functional Behavior 39](#_Toc485914005)

[6.2.2 Call Flow 40](#_Toc485914006)

[6.2.3 Input Parameters 40](#_Toc485914007)

[6.2.4 Output Parameters 41](#_Toc485914008)

[6.2.5 HTTP Status Codes 42](#_Toc485914009)

[6.2.6 Sample Request and Response 42](#_Toc485914010)

[6.2.6.1 Sample Request 42](#_Toc485914011)

[6.2.6.2 Sample Success Response #1 43](#_Toc485914012)

[6.2.6.3 Sample Success Response #2 43](#_Toc485914013)

[6.2.6.4 Sample Error Responses 44](#_Toc485914014)

[6.3 Operation: publishEventBatch 45](#_Toc485914015)

[6.3.1 Functional Behavior 45](#_Toc485914016)

[6.3.2 Call Flow 46](#_Toc485914017)

[6.3.3 Input Parameters 46](#_Toc485914018)

[6.3.4 Output Parameters 47](#_Toc485914019)

[6.3.5 HTTP Status Codes 48](#_Toc485914020)

[6.3.6 Sample Request and Response 48](#_Toc485914021)

[6.3.6.1 Sample Request 48](#_Toc485914022)

[6.3.6.2 Sample Success Response #1 50](#_Toc485914023)

[6.3.6.3 Sample Success Response #2 50](#_Toc485914024)

[6.3.6.4 Sample Error Responses 51](#_Toc485914025)

[6.4 Operation: provideThrottlingState 52](#_Toc485914026)

[6.4.1 Functional Behavior 52](#_Toc485914027)

[6.4.2 Call Flow 52](#_Toc485914028)

[6.4.3 Input Parameters 52](#_Toc485914029)

[6.4.4 Output Parameters 53](#_Toc485914030)

[6.4.5 HTTP Status Codes 53](#_Toc485914031)

[6.4.6 Sample Request and Response 54](#_Toc485914032)

[6.4.6.1 Sample Request 54](#_Toc485914033)

[6.4.6.2 Sample Success Response 55](#_Toc485914034)

[6.4.6.3 Sample Error Responses 55](#_Toc485914035)

[7 Appendix: Historical Change Log 57](#_Toc485914036)

Figures

[Figure 1 – REST Resource Structure 6](#_Toc485223912)

[Figure 2 - publishAnyEvent Call Flow 40](#_Toc485223913)

[Figure 3 – publishEventBatch Call Flow 46](#_Toc485223914)

[Figure 4 - provideClientThrottlingState Call Flow 52](#_Toc485223915)

Tables

[Table 1 - Service Exceptions 35](#_Toc485223916)

[Table 2 - Policy Exceptions 36](#_Toc485223917)

[Table 3 - REST Operation Summary 37](#_Toc485223918)

# Introduction

This document describes the RESTful interface for the VES (Virtual function Event Streaming) Event Listener. The VES Event Listener is capable of receiving any event sent in the VES Common Event Format. The Common Event Format is a JSON structure consisting of a required Common Event Header Block accompanied by zero or more event domain blocks. A JSON Schema of the VES Common Event Format is provided in Section 4 of this document.

It should be understood that events are well structured packages of information, identified by an eventName, which are asynchronously communicated to subscribers who are interested in the eventName. Events can convey measurements, faults, syslogs, threshold crossing alerts and others types of information. Events are simply a way of communicating well-structured packages of information to one or more instances of an Event Listener service.

This document describes a RESTful connectionless push event listener that is capable of receiving single events or batches of events in the Common Event Format. In future, additional documents may describe other transports which make use of persistent TCP connections for high volumes of streaming events.

## Event Registration

All events must be compliant with the common event format, but specific events identified by their eventNames, may require that certain fields, which are optional in the common event format, be present when they are published. For example, a specific eventName may require that specific name-value pairs be present in the extensible structures provided within the Common Event Format.

Events are registered using an extensible YAML format (defined in a separate document), which specifies, for each eventName, the fields that are required, what field values may be sent, and any special handling that should be performed on those eventNames.

## Naming Standards for eventName

To prevent naming collisions, eventNames sent as part of the commonEventHeader, should conform to the following naming convention designed to summarize the purpose and type of the event, and to ensure the uniqueness of the eventName:

{DomainAbbreviation}\_{AsdcModel or ApplicationPlatform}\_{DescriptionOfInfoBeingConveyed}

Domain abbreviations are derived from the ‘domain’ field in the commonEventHeader, as specified below:

* ‘Fault’ for the fault domain
* ‘Heartbeat’ for the heartbeat domain
* ‘Mfvs’ for the measurementsForVfScaling domain
* ‘MobileFlow’ for the mobileFlow domain
* ‘Other’ for the other domain
* ‘SipSignaling’ for the sipSignaling domain
* ‘StateChange’ for the stateChange domain
* ‘Syslog’ for the syslog domain
* ‘Tca’ for the thresholdCrossingAlert domain
* ‘voiceQuality’ for the voiceQuality domain

ASDC (the AT&T Service Design and Creation environment) defines and catalogs specific services, VNFs, VF modules and other entities, which are generically referred to as ‘ASDC models’. The ASDC model that an event is associated with should be indicated in the second subfield within the eventName. If the event is not associated with an ASDC model but is instead being generated by an application platform like MSO, then a string identifying the Application Platform may be used instead. In either case, all subfield names should be converted to camel case format (with no spaces, hyphens or underscores).

The final subfield of the eventName name should describe, in a compact camel case format (with no spaces, hyphens or underscores), the specific information being conveyed by the event. In some cases, this final subfield will not be required (e.g., in the case of Heartbeats or in the case of an event source which, for a domain like syslog, defines only one eventName to support it):

Examples of eventNames following the naming standards are provided below:

* Fault\_MobileCallRecording\_PilotNumberPoolExhaustion
* Heartbeat\_vIsbcMmc
* Other\_WanBonding\_InstantiationPart1Complete
* Syslog\_vDbe
* Tca\_vDbe\_CpuThresholdExceeded
* Other\_Mso\_InstantiationPhase1Complete

Any questions about the naming of eventNames should be resolved as part of service and resource onboarding to the AT&T Service Design and Creation environment (i.e., ASDC).

## Support for Protocols Other Than HTTPS

This API specification describes an HTTPS RESTful interface using the JSON content-type.

Alternative specifications may be provided in future using Websockets, which would establish a permanent TCP socket, or Apache Avro which provides a binary format over an RPC protocol to be defined. Both would leverage the JSON schema provided in this document.

## Versioning

Three types of version numbers supported by this specification:

* The API specification itself is versioned. Going forward, the major number of the specification version will be incremented whenever any change could break an existing client (e.g., a field name is deleted or changed). All other changes to the spec (e.g., a field name is added or text changes are made to the specification itself) will increment only the minor number. Note that the major number appears in REST resource URLs as v# (where ‘#’ is the major number).
* The JSON schema is versioned. Going forward, the major number of the JSON schema will be incremented whenever any change could break an existing client (e.g., a field name is deleted or changed). All other changes to the schema (e.g., a field name is added or text changes are made to the field descriptions) will increment only the minor number.
* The field blocks are versioned. Field blocks include the commonEventHeader and the domain blocks (e.g., the faultFields block). Going forward, the major number of each field block will be incremented whenever any change to that block could break an existing client (e.g., a field name is deleted or changed). All other changes to that block (e.g., a field name is added or text changes are made to the field descriptions) will increment only the minor number.

# Security

Event sources must identify themselves to the VES Event Listener.

Event source credentials are passed using HTTP [Basic Authentication](http://tools.ietf.org/html/rfc2617).

Credentials must not be passed on the query string. Credentials must be sent in an Authorization header as follows:

1. The username and password are formed into one string as “username:password”
2. The resulting string is Base64 encoded to produce the encoded credential.
3. The encoded credential is communicated in the header after the string “Authorization: Basic “

Because the credentials are merely encoded but not encrypted, HTTPS (rather than HTTP) should be used. HTTPS will also encrypt and protect event contents.

Examples are provided below.

### Sample Request and Response

#### Sample Request

|  |
| --- |
| POST /eventListener/v5 HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: 12345 {  "event": {  "commonEventHeader": {  "version": 3.0,  "domain": "heartbeat",  "eventName": "Heartbeat\_vIsbcMmc",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546015",  "sequence": 0,  "priority": "Normal",  "reportingEntityId": "cc305d54-75b4-431b-adb2-eb6b9e541234",  "reportingEntityName": "EricssonOamVf",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  "sourceName": "ibcx0001vm002ssc001",  "nfNamingCode": "ibcx",  "nfcNamingCode": "ssc",  "startEpochMicrosec": 1413378172000000,  "lastEpochMicrosec": 1413378172000000  }  }  } |

#### Sample Success Response

|  |
| --- |
| HTTPS/1.1 202 Accepted |

# Resource Structure

REST resources are defined with respect to a ServerRoot:

ServerRoot = https://{Domain}:{Port}/{optionalRoutingtPath}

The resource structure is provided below:

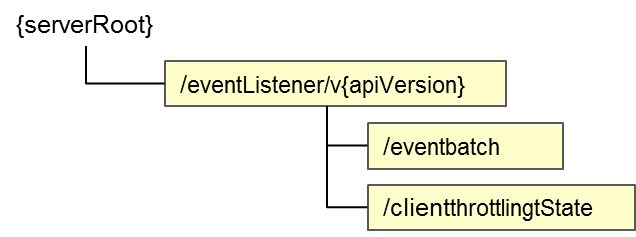


Figure – REST Resource Structure

The {Domain} or FQDN above is typically provisioned into each eventsource when it is instantiated. The {Port} above is typically 8443.

# Common Event Format

A JSON schema describing the Common Event Format is provided below and is reproduced in the tables that follow.



## Common Event Datatypes

### Command List Processing Datatypes

#### Datatype: command

The command datatype is used by an event collector to request changes in the behavior of an event source (for more information, see 6.1.3); it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| commandType | string | Yes | Enumeration: ‘heartbeatIntervalChange’, ‘measurementIntervalChange’,  ‘provideThrottlingState’, ‘throttllingSpecification’ |
| eventDomainThrottle Specification | eventDomainThrottleSpecification | No | If commandType is ‘throttlingSpecification’, the fields to suppress within an event domain |
| heartbeatInterval | integer | No | If commandType is ‘heartbeatIntervalChange’, the heartbeatInterval duration to use in seconds |
| measurementInterval | integer | No | If commandType is ‘measurementIntervalChange’, the measurementInterval duration to use in seconds |

#### Datatype: commandList

The commandList datatype is an array of commands from an event collector toward an event source; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| commandList | Command [ ] | Yes | List of commands from an event collector toward an event source |

#### Datatype: eventDomainThrottleSpecification

The eventDomainThrottleSpecification datatype specifies what fields to suppress within an event domain; it consists of the following fields common to all events:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| eventDomain | string | Yes | Event domain enum from the commonEventHeader domain field |
| suppressedFieldNames | string [ ] | No | List of optional field names in the event block that should not be sent to the Event Listener |
| suppressedNvPairsList | suppressedNvPairs [ ] | No | Optional list of specific NvPairsNames to suppress within a given Name-Value Field |

#### Datatype: eventDomainThrottleSpecificationList

The eventDomainThrottleSpecificationList datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| eventDomainThrottleSpecificationList | eventDomainThrottleSpecification [ ] | Yes | Array of eventDomainThrottleSpecifications |

#### Datatype: eventThrottlingState

The eventThrottlingState datatype reports the throttling in force at the event source; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| eventThrottlingMode | string | Yes | Enumeration: ‘normal’, ‘throttled’ |
| eventDomainThrottleSpecificationList | eventDomainThrottleSpecificationList | No | A list of eventDomainThrottleSpecifications currently in force at the event source, if the eventManagerMode is ‘throttled’ |

#### Datatype: suppressedNvPairs

The suppressedNvPairs datatype is a list of specific NvPairsNames to suppress within a given Name-Value Field (for event throttling); it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| nvPairFieldName | string | Yes | Name of the field within which are the nvpair names to suppress |
| suppressedNvPairNames | string [ ] | Yes | Array of nvpair names to suppress (within the nvpairFieldName) |

### Common Event Datatypes

#### Datatype: event

The event datatype consists of the following fields which constitute the ‘root level’ of the common event format:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| commonEventHeader | commonEventHeader | Yes | Fields common to all events |
| faultFields | faultFields | No | Fields specific to fault events |
| heartbeatFields | heartbeatFields | No | Fields specific to heartbeat events |
| measurementsForVfScalingFields | measurementsForVfScalingFields | No | Fields specific to measurementsForVfScaling events |
| mobileFlowFields | mobileFlowFields | No | Fields specific to mobility flow events |
| otherFields | otherFields | No | Fields specific to other types of events |
| sipSignalingFields | sipSignalingFields | No | Fields specific to sipSignaling events |
| stateChangeFields | stateChangeFields | No | Fields specific to state change events |
| syslogFields | syslogFields | No | Fields specific to syslog events |
| thresholdCrossingAlertFields | thresholdCrossingAlertFields | No | Fields specific to threshold crossing alert events |
| voiceQualityFields | voiceQualityFields | No | Fields specific to voiceQuality events |

#### Datatype: eventList

The eventList datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| eventList | event [ ] | Yes | Array of events |

#### Datatype: field

The field datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| name | string | Yes | Name of the field |
| value | string | Yes | Value of the named field |

#### Datatype: jsonObject

The jsonObject datatype provides a json object schema, name and other meta-information along with one or more object instances that conform to the schema:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| objectInstances | JsonObjectInstance [ ] | Yes | Contains one or more instances of the json object |
| objectName | string | Yes | Name of the json object |
| objectSchema | string | No | json schema for the object |
| objectSchemaUrl | string | No | URL to the json schema for the object |
| nfSubscribedObjectName | string | No | Name of the object associated with the nfSubscriptionId |
| nfSubscriptionId | string | No | Identifies an openConfig telemetry subscription on a network function, which configures the network function to send complex object data associated with the jsonObject |

#### Datatype: jsonObjectInstance

The jsonObjectInstance datatype provides meta-information about an instance of a jsonObject along with the actual object instance:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| objectInstance | object | Yes | Contains an instance conforming to the jsonObject schema |
| objectInstanceEpochMicrosec | number | No | the unix time, aka epoch time, associated with this objectInstance--as microseconds elapsed since 1 Jan 1970 not including leap seconds |
| objectKeys | key [ ] | No | An ordered set of keys that identifies this particular instance of jsonObject (e.g., that places it in a hierarchy) |

#### Datatype: key

The key datatype is a tuple which provides the name of a key along with its value and relative order; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| keyName | string | Yes | Name of the key |
| keyOrder | Integer | No | Relative sequence or order of the key (with respect to other keys) |
| keyValue | string | No | Value of the key |

#### Datatype: namedArrayOfFields

The namedArrayOfFields datatype is an array of name value pairs along with a name for the array; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| name | string | Yes | Name for the array of name-value pairs |
| arrayOfFields | field [ ] | Yes | Name-value pairs |

#### Datatype: requestError

The requestError datatype defines the standard request error data structure:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| messageId | string | Yes | Unique message identifier of the format ‘ABCnnnn’ where ‘ABC’ is either ‘SVC’ for Service Exceptions or ‘POL’ for Policy Exception. Exception numbers may be in the range of 0001 to 9999 where 0001 to 2999 are defined by OMA (see section 5.1) and 3000-9999 are available and undefined. |
| text | string | Yes | Message text, with replacement variables marked with %n, where n is an index into the list of <variables> elements, starting at 1 |
| url | string | No | Hyperlink to a detailed error resource e.g., an HTML page for browser user agents |
| variables | string | No | List of zero or more strings that represent the contents of the variables used by the message text |

#### Datatype: vendorVnfNameFields

The vendorVnfNameFields provides vendor, vnf and vfModule identifying information:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| vendorName | string | Yes | VNF vendor name |
| vfModuleName | string | No | The ASDC vfModuleName for the vfModule generating the event |
| vnfName | string | No | The ASDC modelName for the VNF generating the event |

### ‘Common Event Header’ Datatypes

#### Datatype: commonEventHeader

The commonEventHeader datatype consists of the following fields common to all events:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| version | number | Yes | Version of the event header (currently: 3.0) |
| eventName | string | Yes | Unique event name (see section 1 for more information) |
| domain | string | Yes | Event domain enumeration: ‘fault’, ‘heartbeat’, ‘measurementsForVfScaling’, ‘mobileFlow’, ‘other’, ‘sipSignaling’, ‘stateChange’, ‘syslog’, ‘thresholdCrossingAlert’, ‘voiceQuality’ |
| eventId | string | Yes | Event key that is unique to the event source |
| eventType | string | No | For example: ‘applicationVnf’, ‘guestOS’, ‘hostOS’, ‘platform’ |
| nfcNamingCode | string | No | Network function component type: 3 characters (aligned with vfc naming standards) |
| nfNamingCode | string | No | Network function type: 4 characters (aligned with vnf naming standards) |
| 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 |
| 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 EMS name. May be the same as the sourceName. For synthetic events generated by DCAE, it is the name of the app generating the event. |
| 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 | internalHeader Fields | 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. |

#### Datatype: internalHeaderFields

The internalHeaderFields datatype is an undefined object which can contain arbitrarily complex JSON structures. It is intended to be defined separately by each provider implementing the VES Event Listener. The fields in internalHeaderFields are not provided by any event source but instead are added by the VES Event Listener service itself as part of an event enrichment process necessary for efficient internal processing of events received by the VES Event Listener:

## Technology Independent Datatypes

### ‘Fault’ Domain Datatypes

#### Datatype: faultFields

The faultFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| faultFieldsVersion | number | Yes | Version of the faultFields block (currently: 2.0) |
| eventSeverity | string | Yes | Event severity enumeration: ‘CRITICAL’, ‘MAJOR’, ‘MINOR’, ‘WARNING’, ‘NORMAL’ |
| eventSourceType | string | Yes | Examples: ‘card’, ‘host’, ‘other’, ‘port’, ‘portThreshold’, ‘router’, ‘slotThreshold’, ‘switch’, ‘virtualMachine’, ‘virtualNetworkFunction’ |
| eventCategory | string | No | Event category, for example: ‘license’, ‘link’, ‘routing’, ‘security’, ‘signaling’ |
| alarmCondition | string | Yes | Alarm condition reported by the device (e.g., ‘tpLgCgiNotInConfig’) |
| specificProblem | string | Yes | Short description of the alarm or problem (e.g., ‘This event is sent when the LG is asked to perform a location for a CGI that is not in its configuration’) |
| vfStatus | string | Yes | Virtual function status enumeration: ‘Active’, ‘Idle’, ‘Preparing to terminate’, ‘Ready to terminate’, ‘Requesting Termination’ |
| alarmInterfaceA | string | No | Card, port, channel or interface name of the device generating the alarm |
| alarmAdditional Information | field [ ] | No | Additional alarm information (note: for SNMP mapping to VES, for name use OID of varbind, for value use incoming data for that varbind) |

### ‘Heartbeat’ Domain Datatypes

#### Datatype: heartbeatFields

The heartbeatFields datatype is an optional field block for fields specific to heartbeat events; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| heartbeatFieldsVersion | number | Yes | Version of the heartbeatFields block (currently: 1.0) |
| additionalFields | field [ ] | No | Additional expansion fields if needed |
| heartbeatInterval | Integer | Yes | Current heartbeatInterval in seconds |

### ‘Measurements For VF Scaling’ Domain Datatypes

#### Datatype: codecsInUse

The codecsInUse datatype consists of the following fields describing the number of times an identified codec was used over the measurementInterval:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| codecIdentifer | string | Yes | Description of the codec |
| numberInUse | integer | Yes | Number of such codecs in use |

#### Datatype: cpuUsage

The cpuUsage datatype defines the usage of an identifier CPU and consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| cpuIdentifier | string | Yes | CPU Identifier |
| cpuIdle | number | No | Percentage of CPU time spent in the idle task |
| cpuUsageInterrupt | number | No | Percentage of time spent servicing interrupts |
| cpuUsageNice | number | No | Percentage of time spent running user space processes that have been niced |
| cpuUsageSoftIrq | number | No | Percentage of time spent handling soft irq interrupts |
| cpuUsageSteal | number | No | Percentage of time spent in involuntary wait which is neither user, system or idle time and is effectively time that went missing |
| cpuUsageSystem | number | No | Percentage of time spent on system tasks running the kernel |
| cpuUsageUser | number | No | Percentage of time spent running un-niced user space processes |
| cpuWait | number | No | Percentage of CPU time spent waiting for I/O operations to complete |
| percentUsage | number | Yes | Aggregate cpu usage of the virtual machine on which the VNFC reporting the event is running |

#### Datatype: diskUsage

The diskUsage datatype defines the usage of a disk and consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| diskIdentifier | string | Yes | Disk Identifier |
| diskIoTimeAvg | number | No | Milliseconds spent doing input/output operations over 1 sec; treat this metric as a device load percentage where 1000ms matches 100% load; provide the average over the measurement interval |
| diskIoTimeLast | number | No | Milliseconds spent doing input/output operations over 1 sec; treat this metric as a device load percentage where 1000ms matches 100% load; provide the last value measurement within the measurement interval |
| diskIoTimeMax | number | No | Milliseconds spent doing input/output operations over 1 sec; treat this metric as a device load percentage where 1000ms matches 100% load; provide the maximum value measurement within the measurement interval |
| diskIoTimeMin | number | No | Milliseconds spent doing input/output operations over 1 sec; treat this metric as a device load percentage where 1000ms matches 100% load; provide the minimum value measurement within the measurement interval |
| diskMergedReadAvg | number | No | Number of logical read operations that were merged into physical read operations, e.g., two logical reads were served by one physical disk access; provide the average measurement within the measurement interval |
| diskMergedReadLast | number | No | Number of logical read operations that were merged into physical read operations, e.g., two logical reads were served by one physical disk access; provide the last value measurement within the measurement interval |
| diskMergedReadMax | number | No | Number of logical read operations that were merged into physical read operations, e.g., two logical reads were served by one physical disk access; provide the maximum value measurement within the measurement interval |
| diskMergedReadMin | number | No | Number of logical read operations that were merged into physical read operations, e.g., two logical reads were served by one physical disk access; provide the minimum value measurement within the measurement interval |
| diskMergedWriteAvg | number | No | Number of logical write operations that were merged into physical write operations, e.g., two logical writes were served by one physical disk access; provide the average measurement within the measurement interval |
| diskMergedWriteLast | number | No | Number of logical write operations that were merged into physical write operations, e.g., two logical writes were served by one physical disk access; provide the last value measurement within the measurement interval |
| diskMergedWriteMax | number | No | Number of logical write operations that were merged into physical write operations, e.g., two logical writes were served by one physical disk access; provide the maximum value measurement within the measurement interval |
| diskMergedWriteMin | number | No | Number of logical write operations that were merged into physical write operations, e.g., two logical writes were served by one physical disk access; provide the minimum value measurement within the measurement interval |
| diskOctetsRead Avg | number | No | Number of octets per second read from a disk or partition; provide the average measurement within the measurement interval |
| diskOctetsRead  Last | number | No | Number of octets per second read from a disk or partition; provide the last measurement within the measurement interval |
| diskOctetsRead Max | number | No | Number of octets per second read from a disk or partition; provide the maximum measurement within the measurement interval |
| diskOctetsRead Min | number | No | Number of octets per second read from a disk or partition; provide the minimum measurement within the measurement interval |
| diskOctetsWrite Avg | number | No | Number of octets per second written to a disk or partition; provide the average measurement within the measurement interval |
| diskOctetsWrite Last | number | No | Number of octets per second written to a disk or partition; provide the last measurement within the measurement interval |
| diskOctetsWriteMax | number | No | Number of octets per second written to a disk or partition; provide the maximum measurement within the measurement interval |
| diskOctetsWriteMin | number | No | Number of octets per second written to a disk or partition; provide the minimum measurement within the measurement interval |
| diskOpsReadAvg | number | No | Number of read operations per second issued to the disk; provide the average measurement within the measurement interval |
| diskOpsReadLast | number | No | Number of read operations per second issued to the disk; provide the last measurement within the measurement interval |
| diskOpsReadMax | number | No | Number of read operations per second issued to the disk; provide the maximum measurement within the measurement interval |
| diskOpsReadMin | number | No | Number of read operations per second issued to the disk; provide the minimum measurement within the measurement interval |
| diskOpsWriteAvg | number | No | Number of write operations per second issued to the disk; provide the average measurement within the measurement interval |
| diskOpsWriteLast | number | No | Number of write operations per second issued to the disk; provide the last measurement within the measurement interval |
| diskOpsWrite Max | number | No | Number of write operations per second issued to the disk; provide the maximum measurement within the measurement interval |
| diskOpsWriteMin | number | No | Number of write operations per second issued to the disk; provide the minimum measurement within the measurement interval |
| diskPendingOperationsAvg | number | No | Queue size of pending I/O operations per second; provide the average measurement within the measurement interval |
| diskPendingOperationsLast | number | No | Queue size of pending I/O operations per second; provide the last measurement within the measurement interval |
| diskPendingOperationsMax | number | No | Queue size of pending I/O operations per second; provide the maximum measurement within the measurement interval |
| diskPendingOperationsMin | number | No | Queue size of pending I/O operations per second; provide the minimum measurement within the measurement interval |
| diskTimeReadAvg | number | No | Milliseconds a read operation took to complete; provide the average measurement within the measurement interval |
| diskTimeRead Last | number | No | Milliseconds a read operation took to complete; provide the last measurement within the measurement interval |
| diskTimeRead Max | number | No | Milliseconds a read operation took to complete; provide the maximum measurement within the measurement interval |
| diskTimeRead Min | number | No | Milliseconds a read operation took to complete; provide the minimum measurement within the measurement interval |
| diskTimeWrite Avg | number | No | Milliseconds a write operation took to complete; provide the average measurement within the measurement interval |
| diskTimeWrite Last | number | No | Milliseconds a write operation took to complete; provide the last measurement within the measurement interval |
| diskTimeWrite Max | number | No | Milliseconds a write operation took to complete; provide the maximum measurement within the measurement interval |
| diskTimeWrite Min | number | No | Milliseconds a write operation took to complete; provide the minimum measurement within the measurement interval |

#### Datatype: featuresInUse

The featuresInUse datatype consists of the following fields which describe the number of times an identified feature was used over the measurementInterval:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| featureIdentifer | string | Yes | Description of the feature |
| featureUtilization | integer | Yes | Number of times the identified feature was used |

#### Datatype: filesystemUsage

The filesystemUsage datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| filesystemName | string | Yes | File system name |
| blockConfigured | number | Yes | Configured block storage capacity in GB |
| blockIops | number | Yes | Block storage input-output operations per second |
| blockUsed | number | Yes | Used block storage capacity in GB |
| ephemeralConfigured | number | Yes | Configured ephemeral storage capacity in GB |
| ephemeralIops | number | Yes | Ephemeral storage input-output operations per second |
| ephemeralUsed | number | Yes | Used ephemeral storage capacity in GB |

#### Datatype: latencyBucketMeasure

The latencyBucketMeasure datatype consists of the following fields which describe the number of counts falling within a defined latency bucket:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| countsInTheBucket | number | Yes | Number of counts falling within a defined latency bucket |
| highEndOfLatencyBucket | number | No | High end of bucket range (typically in ms) |
| lowEndOfLatencyBucket | number | No | Low end of bucket range (typically in ms) |

#### Datatype: measurementsForVfScalingFields

The measurementsForVfScalingFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| measurementsForVfScalingVersion | number | Yes | Version of the measurementsForVfScalingFields block (currently: 2.0) |
| additionalFields | field [ ] | No | Additional measurement fields if needed |
| additionalMeasurements | namedArrayOfFields [ ] | No | Array of named name-value-pair arrays if needed |
| additionalObjects | jsonObject [ ] | No | Array of JSON objects described by name, schema and other meta-information, if needed |
| codecUsageArray | codecsInUse [] | No | Array of codecs in use |
| concurrentSessions | integer | No | Peak concurrent sessions for the VM or VNF (depending on the context) over the measurementInterval |
| configuredEntities | integer | 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 | cpuUsage [] | No | Usage of an array of CPUs |
| diskUsageArray | diskUsage [] | No | Usage of an array of disks |
| featureUsageArray | featuresInUse [] | No | Array of features in use |
| filesystemUsageArray | filesystemUsage [] | No | Filesystem usage of the VM on which the VNFC reporting the event is running |
| latencyDistribution | latencyBucketMeasure [ ] | No | Array of integers representing counts of requests whose latency in milliseconds falls within per-VNF configured ranges; where latency is the duration between a service request and its fulfillment. |
| 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 |
| memoryUsageArray | memoryUsage [] | No | Memory usage of an array of VMs |
| numberOfMediaPortsInUse | integer | No | Number of media ports in use |
| requestRate | number | No | Peak rate of service requests per second to the VNF over the measurementInterval |
| vnfcScalingMetric | integer | No | Represents busy-ness of the VNF from 0 to 100 as reported by the VNFC |
| vNicPerformanceArray | vNicPerformance [ ] | No | Performance metrics of an array of virtual network interface cards |

#### Datatype: memoryUsage

The memoryUsage datatype defines the memory usage of a virtual machine and consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| memoryBuffered | number | No | Kibibytes of temporary storage for raw disk blocks |
| memoryCached | number | No | Kibibytes of memory used for cache |
| memoryConfigured | number | No | Kibibytes of memory configured in the virtual machine on which the VNFC reporting the event is running |
| memoryFree | number | Yes | Kibibytes of physical RAM left unused by the system |
| memorySlabRecl | number | No | The part of the slab that can be reclaimed such as caches measured in kibibytes |
| memorySlabUnrecl | number | No | The part of the slab that cannot be reclaimed even when lacking memory measure in kibibytes |
| memoryUsed | number | Yes | Total memory minus the sum of free, buffered, cached and slab memory measured in kibibytes |
| vmIdentifier | string | Yes | Virtual Machine identifier associated with the memory metrics |

#### Datatype: vNicPerformance

The vNicPerformance datatype consists of the following fields which describe the performance and errors of an of an identified virtual network interface card:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| receivedBroadcastPacketsAccumulated | number | No | Cumulative count of broadcast packets received as read at the end of the measurement interval |
| receivedBroadcastPacketsDelta | number | No | Count of broadcast packets received within the measurement interval |
| receivedDiscardedPacketsAccumulated | number | No | Cumulative count of discarded packets received as read at the end of the measurement interval |
| receivedDiscardedPacketsDelta | number | No | Count of discarded packets received within the measurement interval |
| receivedErrorPacketsAccumulated | number | No | Cumulative count of error packets received as read at the end of the measurement interval |
| receivedErrorPacketsDelta | number | No | Count of error packets received within the measurement interval |
| receivedMulticastPacketsAccumulated | number | No | Cumulative count of multicast packets received as read at the end of the measurement interval |
| receivedMulticastPacketsDelta | number | No | Count of multicast packets received within the measurement interval |
| receivedOctetsAccumulated | number | No | Cumulative count of octets received as read at the end of the measurement interval |
| receivedOctetsDelta | number | No | Count of octets received within the measurement interval |
| receivedTotalPacketsAccumulated | number | No | Cumulative count of all packets received as read at the end of the measurement interval |
| receivedTotalPacketsDelta | number | No | Count of all packets received within the measurement interval |
| receivedUnicastPacketsAccumulated | number | No | Cumulative count of unicast packets received as read at the end of the measurement interval |
| receivedUnicastPacketsDelta | number | No | Count of unicast packets received within the measurement interval |
| transmittedBroadcastPacketsAccumulated | number | No | Cumulative count of broadcast packets transmitted as read at the end of the measurement interval |
| transmittedBroadcastPacketsDelta | number | No | Count of broadcast packets transmitted within the measurement interval |
| transmittedDiscardedPacketsAccumulated | number | No | Cumulative count of discarded packets transmitted as read at the end of the measurement interval |
| transmittedDiscardedPacketsDelta | number | No | Count of discarded packets transmitted within the measurement interval |
| transmittedErrorPacketsAccumulated | number | No | Cumulative count of error packets transmitted as read at the end of the measurement interval |
| transmittedErrorPacketsDelta | number | No | Count of error packets transmitted within the measurement interval |
| transmittedMulticastPacketsAccumulated | number | No | Cumulative count of multicast packets transmitted as read at the end of the measurement interval |
| transmittedMulticastPacketsDelta | number | No | Count of multicast packets transmitted within the measurement interval |
| transmittedOctetsAccumulated | number | No | Cumulative count of octets transmitted as read at the end of the measurement interval |
| transmittedOctetsDelta | number | No | Count of octets transmitted within the measurement interval |
| transmittedTotalPacketsAccumulated | number | No | Cumulative count of all packets transmitted as read at the end of the measurement interval |
| transmittedTotalPacketsDelta | number | No | Count of all packets transmitted within the measurement interval |
| transmittedUnicastPacketsAccumulated | number | No | Cumulative count of unicast packets transmitted as read at the end of the measurement interval |
| transmittedUnicastPacketsDelta | number | No | Count of unicast packets transmitted within the measurement interval |
| valuesAreSuspect | string | Yes | Enumeration: ‘true’ or ‘false’. If ‘true’ then the vNicPerformance values are likely inaccurate due to counter overflow or other condtions. |
| vNicIdentifier | string | Yes | vNic identification |

### ‘Other’ Domain Datatypes

#### Datatype: otherFields

The otherFields datatype defines fields for events belonging to the 'other' domain of the commonEventHeader domain enumeration; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| otherFieldsVersion | number | Yes | Version of the otherFields block (currently: 1.1) |
| hashOfNameValuePairArrays | namedArrayOfFields [ ] | No | Array of named name-value-pair arrays |
| jsonObjects | jsonObject [ ] | No | Array of JSON objects described by name, schema and other meta-information |
| nameValuePairs | field [ ] | No | Array of name-value pairs |

### ‘State Change’ Domain Datatypes

#### Datatype: stateChangeFields

The stateChangeFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| stateChangeFieldsVersion | number | Yes | Version of the stateChangeFields block (currently: 2.0) |
| additionalFields | field [ ] | No | Additional stateChange 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 |

### ‘Syslog’ Domain Datatypes

#### Datatype: syslogFields

The syslogFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| syslogFieldsVersion | number | Yes | Version of the syslogFields block (currently: 3.0) |
| additionalFields | string | No | Additional syslog fields if needed, provided as name=value delimited by a pipe ‘|’ symbol, for example: “name1=value1|name2=value2|…” |
| eventSourceHost | string | No | Hostname of the device |
| eventSourceType | string | Yes | Examples: ‘other’, ‘router’, ‘switch’, ‘host’, ‘card’, ‘port’, ‘slotThreshold’, ‘portThreshold’, ‘virtualMachine’, ‘virtualNetworkFunction’ |
| syslogFacility | integer | 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 | integer | 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,  i.e., ourSDID@32473 |
| syslogSev | string | No | Level-of-severity enumeration in quotes below:      ‘Emergency’: system is unusable      ‘Alert’            : action must be taken immediately      ‘Critical’        : critical conditions      ‘Error’            : error conditions      ‘Warning’      : warning conditions      ‘Notice’         : normal but significant condition      ‘Info’              : Informational: informational messages      ‘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’) |

Example of syslogSData:

STRUCTURED-DATA = NILVALUE / 1\*SD-ELEMENT

SD-ELEMENT = "[" SD-ID \*(SP SD-PARAM) "]"

SD-PARAM = PARAM-NAME "=" %d34 PARAM-VALUE %d34

SD-ID = SD-NAME

PARAM-NAME = SD-NAME

PARAM-VALUE = UTF-8-STRING ; characters '"', '\' and

; ']' MUST be escaped.

SD-NAME = 1\*32PRINTUSASCII

; except '=', SP, ']', %d34 (")

### ‘Threshold Crossing Alert’ Domain Datatypes

#### Datatype: counter

The counter datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| name | string | Yes | Name of the counter |
| value | string | Yes | Current value of the counter |
| threshholdCrossed | string | Yes | Last threshold that was crossed |
| criticality | string | Yes | Enumeration: ‘CRIT’, ‘MAJ’ |

### Datatype: thresholdCrossingAlertFields

The thresholdCrossingAlertFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| thresholdCrossing FieldsVersion | number | Yes | Version of the thresholdCrossingAlertFields block (currently: 2.0) |
| additionalFields | field [ ] | No | Additional threshold crossing alert fields if needed |
| additionalParameters | counter [ ] | Yes | Array of performance counters |
| alertAction | string | Yes | Enumeration: ‘SET’, ‘CONT’, ‘CLEAR’ |
| alertDescription | string | Yes | Unique short alert description (e.g., NE-CPUMEM) |
| alertType | string | Yes | Enumeration: ‘CARD-ANOMALY’, ‘INTERFACE-ANOMALY’, ELEMENT-ANOMALY’, ‘SERVICE-ANOMALY’ |
| alertValue | string | No | Calculated API value (if applicable) |
| associatedAlertIdList | string [ ] | No | List of eventIds associated with the event being reported |
| collectionTimestamp | string | Yes | Time when the performance collector picked up the data; with RFC 2822 compliant format: ‘Sat, 13 Mar 2010 11:29:05 -0800’ |
| dataCollector | string | No | Specific performance collector instance used |
| elementType | string | No | Type of network element (internal AT&T field) |
| eventSeverity | string | Yes | Event severity or priority enumeration: ‘CRITICAL’, ‘MAJOR’, ‘MINOR’, ‘WARNING’, ‘NORMAL’ |
| eventStartTimestamp | string | Yes | Time closest to when the measurement was made; with RFC 2822 compliant format: ‘Sat, 13 Mar 2010 11:29:05 -0800’ |
| interfaceName | string | No | Physical or logical port or card (if applicable) |
| networkService | string | No | Network name (internal AT&T field) |
| possibleRootCause | string | No | Reserved for future use |

## Technology Specific Datatypes

### ‘Mobile Flow’ Domain Datatypes

#### Datatype: gtpPerFlowMetrics

The gtpPerFlowMetrics datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| 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 | string | 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 | string | 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 retrie |
| 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 |

#### Datatype: mobileFlowFields

The mobileFlowFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| mobileFlowFieldsVersion | number | Yes | Version of the mobileFlowFields block (currently: 2.0) |
| additionalFields | field [ ] | No | Additional mobileFlow fields if needed |
| 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 | gtpPer FlowMetrics | Yes | Mobility GTP Protocol per flow metrics |
| 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 | integer | 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 | integer | Yes | IP port for the reporting entity, as used for the flow being reported on |
| sac | string | No | Service area code |
| samplingAlgorithm | integer | 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 |

### ‘SipSignaling’ Domain Datatypes

#### Datatype: sipSignalingFields

The sipSignalingFields datatype communicates information about sip signaling messages, parameters and signaling state; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| sipSignalingFieldsVersion | number | Yes | Version of the sipSignalingFields block (currently: 1.0) |
| additionalInformation | field [ ] | No | Additional sipSignaling fields |
| compressedSip | string | No | The full SIP request/response including headers and bodies |
| correlator | string | Yes | Constant across all events on this call |
| localIpAddress | string | Yes | Ip address on VNF |
| localPort | string | Yes | Port on VNF |
| remoteIpAddress | string | Yes | IP address of peer endpoint |
| remotePort | string | Yes | Port of peer endpoint |
| summarySip | string | No | The SIP Method or Response (‘INVITE’, ‘200 OK’, ‘BYE’, etc) |
| vendorVnfNameFields | vendorVnfNameFields | Yes | Vendor, VNF and VfModule names |

### ‘Voice Quality’ Domain Datatypes

#### Datatype: endOfCallVqmSummaries

The endOfCallVqmSummaries datatype provides end of call voice quality metrics; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| adjacencyName | string | Yes | Adjacency name |
| endpointDescription | string | Yes | Enumeration: ‘Caller’, ‘Callee’ |
| endpointJitter | number | No | Endpoint jitter |
| endpointRtpOctetsDiscarded | number | No | Endpoint RTP octets discarded |
| endpointRtpOctetsReceived | number | No | Endpoint RTP octets received |
| endpointRtpOctetsSent | number | No | Endpoint RTP octets sent |
| endpointRtpPacketsDiscarded | number | No | Endpoint RTP packets discarded |
| endpointRtpPacketsReceived | number | No | Endpoint RTP packets received |
| endpointRtpPacketsSent | number | No | Endpoint RTP packets sent |
| localJitter | number | No | Local jitter |
| localRtpOctetsDiscarded | number | No | Local RTP octets discarded |
| localRtpOctetsReceived | number | No | Local RTP octets received |
| localRtpOctetsSent | number | No | Local RTP octets sent |
| localRtpPacketsDiscarded | number | No | Local RTP packets discarded |
| localRtpPacketsReceived | number | No | Local RTP packets received |
| localRtpPacketsSent | number | No | Local RTP packets sent |
| mosCqe | number | No | Decimal range from 1 to 5 (1 decimal place) |
| packetsLost | number | No | Packets lost |
| packetLossPercent | number | No | Calculated percentage packet loss based on endpoint RTP packets lost (as reported in RTCP) and local RTP packets sent. Direction is based on endpoint description (Caller, Callee). Decimal (2 decimal places) |
| rFactor | number | No | rFactor from 0 to 100 |
| roundTripDelay | number | No | Round trip delay in milliseconds |

#### Datatype: voiceQualityFields

The voiceQualityFields datatype provides statistics related to customer facing voice products; consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| voiceQualityFieldsVersion | number | Yes | Version of the voiceQualityFields block (currently: 1.0) |
| additionalInformation | field [ ] | No | Additional voice quality fields |
| calleeSideCodec | string | Yes | Callee codec for the call |
| callerSideCodec | string | Yes | Caller codec for the call |
| correlator | string | Yes | Constant across all events on this call |
| endOfCallVqmSummaries | endOfCallVqm Summaries | No | End of call voice quality metric summaries |
| phoneNumber | string | No | Phone number associated with the correlator |
| midCallRtcp | string | Yes | Base64 encoding of the binary RTCP data (excluding Eth/IP/UDP headers) |
| vendorVnfNameFields | vendorVnfNameFields | Yes | Vendor, VNF and VfModule names |

# Exceptions

## RESTful Web Services Exceptions

RESTful services generate and send exceptions to clients in response to invocation errors. Exceptions send HTTP status codes (specified later in this document for each operation). HTTP status codes may be followed by an optional JSON exception structure described below. Two types of exceptions may be defined: service exceptions and policy exceptions.

| **Field Name** | **Data Type** | **Required?** | **Description** |
| --- | --- | --- | --- |
| messageId | xs:string | Yes | Unique message identifier of the format ‘ABCnnnn’ where ‘ABC’ is either ‘SVC’ for Service Exceptions or ‘POL’ for Policy Exception.  Exception numbers may be in the range of 0001 to 9999 where :   * 0001 to 2999 are defined by OMA (see OMA’s [Common definitions for RESTful Network APIs](http://technical.openmobilealliance.org/Technical/release_program/docs/REST_NetAPI_Common/V1_0-20120417-C/OMA-TS-REST_NetAPI_Common-V1_0-20120417-C.pdf) for details) * 3000-9999 are available and undefined |
| text | xs:string | Yes | Message text, with replacement variables marked with %n, where n is an index into the list of <variables> elements, starting at 1 |
| variables | xs:string [0..unbounded] | No | List of zero or more strings that represent the contents of the variables used by the message text. |
| url | xs:anyUrl | No | Hyperlink to a detailed error resource (e.g., an HTML page for browser user agents). |

## Service Exceptions

When a service is not able to process a request, and retrying the request with the same information will also result in a failure, and the issue is not related to a service policy issue, then the service will issue a fault using the service exception fault message. Examples of service exceptions include invalid input, lack of availability of a required resource or a processing error.

A service exception uses the letters 'SVC' at the beginning of the message identifier. ‘SVC’ service exceptions used by the VES Event Listener API are defined below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *MessageId* | *Description / Comment* | *Text* | *Variables* | *Parent HTTP Code* |
| SVC0001 | General service error (see SVC2000) | <custom error message> | None | 400 |
| SVC0002 | Bad parameter | Invalid input value for message part %1 | %1: message part | 400 |
| SVC1000 | No server resources | No server resources available to process the request | None | 500 |
| SVC2000 | More elaborate version of SVC0001 | The following service error occurred: %1. Error code is %2. | %1: human readable description of the error  %2: error code | 400 |

Table - Service Exceptions

## Policy Exceptions

When a service is not able to complete because the request fails to meet a policy criteria, then the service will issue a fault using the policy exception fault message. To clarify how a policy exception differs from a service exception, consider that all the input to an operation may be valid as meeting the required input for the operation (thus no service exception), but using that input in the execution of the service may result in conditions that require the service not to complete. Examples of policy exceptions include privacy violations, requests not permitted under a governing service agreement or input content not acceptable to the service provider.

A Policy Exception uses the letters 'POL' at the beginning of the message identifier. ‘POL’ policy exceptions used by the VES Event Listener API are defined below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *MessageId* | *Description / Comment* | *Text* | *Variables* | *Parent HTTP Code* |
| POL0001 | General policy error (see POL2000) | A policy error occurred. | None | 401 |
| POL1009 | User not provisioned for service | User has not been provisioned for service | None | 401 |
| POL1010 | User suspended from service | User has been suspended from service | None | 401 |
| POL2000 | More elaborate version of POL0001 | The following policy error occurred: %1. Error code is %2. | %1: human readable description of the error  %2: error code | 401 |
| POL9003 | Message size exceeds limit | Message content size exceeds the allowable limit | None | 400 |

Table - Policy Exceptions

# RESTful Web Services Definition

## REST Operation Overview

### REST Operation Summary

| **Operation Action** | **HTTP**  **Verb** | **Resource URL relative to {ServerRoot}, which is defined in section 3** |
| --- | --- | --- |
| publishAnyEvent | POST | /eventListener/v{apiVersion} |
| publishEventBatch | POST | /eventListener/v{apiVersion}/eventBatch |
| provideClientThrottlingState | POST | /eventListener/v{apiVersion}/clientThrottlingState |

Table - REST Operation Summary

### Api Version

apiVersion is used to describe the major version number of the event listener API (which is the same as the major version number of this specification). When this number changes, the implication is: clients of older versions will break in some way, if they try to use the new API without modification (e.g., unmodified v1 clients would not be able to use v2 without error).

### Commands Toward Event Source Clients

Note: Vendors are not currently required to implement support for command processing; in addition, command processing may be supported by an App-C interface in future.

This specification supports commands from event consumers back toward event source clients. This enables the event consumer (e.g., AT&T event collectors) to command event sources to change their measurement intervals or throttle the information they are sending to the event consumer. Note that commands are sent as part of the synchronous response to events sent by the event source toward the event consumer. This is done so that the event source does not need to host a service to listen for commands from events consumers. The following commands are currently supported:

| **Command** | **Description** |
| --- | --- |
| heartbeatInterval Change | Commands the event source to change the interval (in seconds) it waits between heartbeat events sent to the VES Event Listener. If ‘0’ is provided, the event source should return to its default heartbeatInterval. |
| measurementIntervalChange | Commands the event source to change its measurementInterval to the number provided (in seconds). If ‘0’ is provided, the event source should return to its default measurementInterval. |
| provideThrottlingState | Commands the event source to invoke the provideThrottlingState operation on the event consumer. |
| throttlingSpecification | Commands the event source to throttle events as specified by the provided eventDomainThrottlingSpecification. This specification identifies the fields to suppress within the domain and even supports identification of subfields to suppress within objects or name-value pair structures. Note that required fields should not be suppressed and may result in errors being thrown by the event consumer back toward the event source when events without the required fields are sent to the event consumer. Other notes for event sources:   * the default throttling state is \*off\* for all domains * the throttling state for a domain is altered only by receipt of an eventDomainThrottleSpecification for that domain * the presence of the optional suppressedFieldNames replaces any existing list of suppressed field names * if suppressedFieldNames is not provided, then any existing list of suppressed field names shall be discarded * the presence of the optional suppressedNvPairsList replaces the any existing list of suppressed name-value pairs * if suppressedNvPairsList is not provided, then any existing list of suppressed name-value pairs shall be discarded |

### Buffering of Events

{ServerRoot} is defined in section 3 of this document, which defines the REST resource URL. One or more FQDNs may be provisioned in an event source when it is instantiated or updated. If an event source is unable to reach any of the provisioned FQDNs, it should buffer the event data specified below, up to a maximum of 1 hour, until a connection can be established and the events can be successfully delivered to the VES Event Listener service.

During such an outage, only the following events should be buffered:

* Faults with eventSeverity of “MINOR”, “MAJOR” or “CRITICAL”
* Syslogs with syslogSev of 0-5
* All MeasurementsForVfScaling events

VNFs acting as event sources should not send syslog events to the VES Event Listener during debug mode (which is controlled via the Netconf management interface), but should store syslog events locally for access, and possible FTP transfer, via the VNF console (e.g., command line interface).

If the internal event source event buffer or local storage should overflow, then the event source should send a Fault event, and should discard events in a first-in, first-out (FIFO) manner (i.e., discard oldest events first).

## Operation: publishAnyEvent

### Functional Behavior

Allows authorized clients to publish any single event to the VES event listener.

* Supports only secure HTTPS (one way SSL) access.
* Uses the HTTP verb POST
* Supports JSON content types
* Provides HTTP response codes as well as Service and Policy error messages
* Allows the event collector to use the HTTP response to command the event source to throttle event messages it may send in the future.

### Call Flow

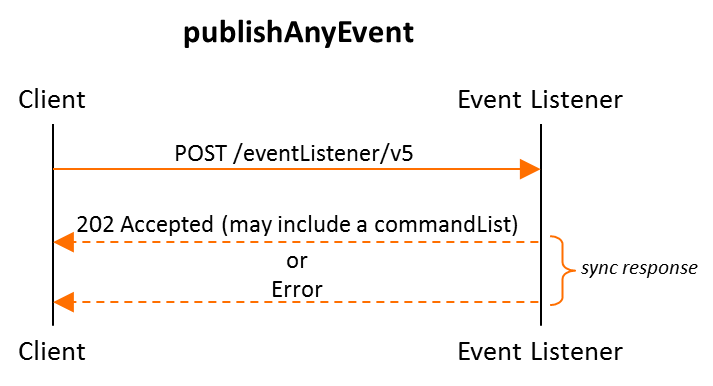


Figure - publishAnyEvent Call Flow

### Input Parameters

Header Fields (note: all parameter names shall be treated as case-insensitive):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| Accept | string | No | Determines the format of the body of the response. Valid values are:   * application/json |
| Authorization | string | Yes | The username and password are formed into one string as “username:password”. This string is then Base64 encoded to produce the encoded credential which is communicated in the header after the string “Authorization: Basic “. See examples below. If the Authorization header is missing, then an HTTP 400 Invalid Request message shall be returned. If the string supplied is invalid, then an HTTP 401 Unauthorized message shall be returned. |
| Content-length | integer | No | Note that content length is limited to 1Megabyte. |
| Content-type | string | Yes | Must be set to one of the following values:   * application/json |

Body Fields:

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| Event | event | Yes | Contains the JSON structure of the common event format. |

### Output Parameters

Header fields:

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| Content-length | integer | No | Used only in error conditions. |
| Content-type | string | No | Used only in error conditions |
| Date | datetime | Yes | Date time of the response in GMT |

Body Fields (for success responses without a commandList): no content is provided and the header fields are not required.

Body Fields (for success responses with one or more commands from the event collector toward the event source):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| commandList | commandList | No | Array of commands (e.g., measurement Interval changes and/or what fields to suppress within specified event domains and/or a request to report the state of event throttling by event domain that is currently in force in the event source). Note: for ‘provideThrottlingState’ commands, the client should subsequently provide the throttling state by calling the provideThrottlingState operation. |

Body Fields (for error Responses):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| requestError | requestError | Yes (for errors) | Used only in error conditions. |

### HTTP Status Codes

|  |  |  |
| --- | --- | --- |
| *Code* | *Reason Phrase* | *Description* |
| 202 | Accepted | The request has been accepted for processing |
| 400 | Bad Request | Many possible reasons not specified by the other codes (e.g., missing required parameters or incorrect format). The response body may include a further exception code and text. HTTP 400 errors may be mapped to SVC0001 (general service error), SVC0002 (bad parameter), SVC2000 (general service error with details) or PO9003 (message content size exceeds the allowable limit). |
| 401 | Unauthorized | Authentication failed or was not provided. HTTP 401 errors may be mapped to POL0001 (general policy error) or POL2000 (general policy error with details). |
| 404 | Not Found | The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. |
| 405 | Method Not Allowed | A request was made of a resource using a request method not supported by that resource (e.g., using PUT on a REST resource that only supports POST). |
| 500 | Internal Server Error | The server encountered an internal error or timed out; please retry (general catch-all server-side error).HTTP 500 errors may be mapped to SVC1000 (no server resources). |

### Sample Request and Response

#### Sample Request

|  |
| --- |
| POST /eventListener/v5 HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: 12345 {  "event": {  "commonEventHeader": {  "version": 3.0,  "domain": "fault",  "eventName": "Fault\_MobileCallRecording\_PilotNumberPoolExhaustion",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546015",  "sequence": 0,  "priority": "High",  "reportingEntityId": "cc305d54-75b4-431b-adb2-eb6b9e541234",  "reportingEntityName": "EricssonOamVf",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  "sourceName": "scfx0001vm002cap001",  "nfNamingCode": "scfx",  "nfcNamingCode": "ssc",  "startEpochMicrosec": 1413378172000000,  "lastEpochMicrosec": 1413378172000000  },  "faultFields": {  "faultFieldsVersion": 2.0,  "alarmCondition": "PilotNumberPoolExhaustion",  "eventSourceType": "other",  "specificProblem": "Calls cannot complete - pilot numbers are unavailable",  "eventSeverity": "CRITICAL",  "vfStatus": "Active",  "alarmAdditionalInformation": [  {  "name": "PilotNumberPoolSize",  "value": "1000"  }  ]  }  }  } |

#### Sample Success Response #1

For success responses without a provided command list:

|  |
| --- |
| HTTPS/1.1 202 Accepted |

#### Sample Success Response #2

For success responses with a provided command list:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {      “commandList”: [          {              “commandType”: “throttlingSpecification”,              “eventDomainThrottleSpecification”: {                  “eventDomain”: “fault”,                  “suppressedFieldNames”: [                      “alarmInterfaceA”,                      “alarmAdditionalInformation”                  ]              }          },          {              “commandType”: “throttlingSpecification”,              “eventDomainThrottleSpecification”: {                  “eventDomain”: “thresholdCrossingAlert”,                  “suppressedFieldNames”: [                      “associatedAlertIdList”,                      “possibleRootCause”                  ],                  “suppressedNvPairs” {                      “nvPairFieldName”: additionalParameters”,                      “suppressedNvPairNames”: [                          “someCounterName”,                          “someOtherCounterName”                      ]                  }              }          },          {              “commandType”: “measurementIntervalChange”,              “measurementInterval”: 600          },          {              “commandType”: “heartbeatIntervalChange”,              “heartbeatInterval”: 90          },          {              “commandType”: “provideThrottlingState”          }      ]  } |

#### Sample Error Responses

##### Sample Policy Exception

|  |
| --- |
| HTTPS/1.1 400 Bad Request  content-type: application/json  content-length: 12345  Date: Thu, 04 Jun 2009 02:51:59 GMT  {  “requestError”: {  “policyException”: {  “messageId”: “POL9003”,  “text”: “Message content size exceeds the allowable limit”,  }  }  } |

##### Sample Service Exception

|  |
| --- |
| HTTPS/1.1 400 Bad Request  content-type: application/json  content-length: 12345  Date: Thu, 04 Jun 2009 02:51:59 GMT  {  “requestError”: {  “serviceException”: {  “messageId”: “SVC2000”,  “text”: “Missing Parameter: %1. Error code is %2”  “variables”: [  “severity”,  “400”  ]  }  }  } |

## Operation: publishEventBatch

### Functional Behavior

Allows authorized clients to publish any single event to the VES event listener.

* Supports only secure HTTPS (one way SSL) access.
* Uses the HTTP verb POST
* Supports JSON content types
* Provides HTTP response codes as well as Service and Policy error messages
* Allows the event collector to use the HTTP response to command the event source to throttle event messages it may send in the future.

### Call Flow

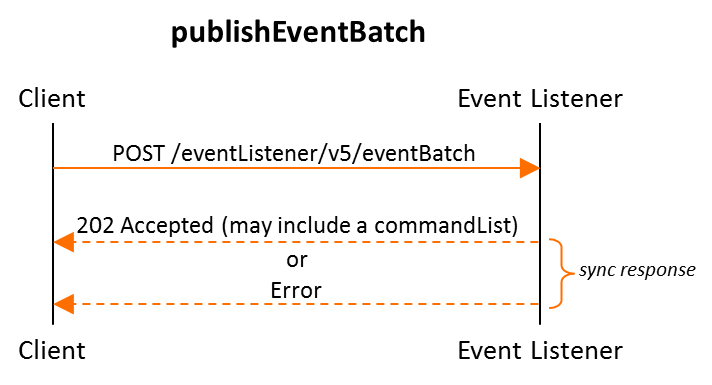


Figure – publishEventBatch Call Flow

### Input Parameters

Header Fields (note: all parameter names shall be treated as case-insensitive):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| Accept | string | No | Determines the format of the body of the response. Valid values are:   * application/json |
| Authorization | string | Yes | The username and password are formed into one string as “username:password”. This string is then Base64 encoded to produce the encoded credential which is communicated in the header after the string “Authorization: Basic “. See examples below. If the Authorization header is missing, then an HTTP 400 Invalid Request message shall be returned. If the string supplied is invalid, then an HTTP 401 Unauthorized message shall be returned. |
| Content-length | integer | No | Note that content length is limited to 1Megabyte. |
| Content-type | string | Yes | Must be set to one of the following values:   * application/json |

Body Fields:

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| eventList | eventList | Yes | Array of events conforming to the common event format. |

### Output Parameters

Header fields:

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| Content-length | integer | No | Used only in error conditions. |
| Content-type | string | No | Used only in error conditions |
| Date | datetime | Yes | Date time of the response in GMT |

Body Fields (for success responses without a commandList): no content is provided and the header fields are not required.

Body Fields (for success responses with one or more commands from the event collector toward the event source):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| commandList | commandList | No | Array of commands (e.g., measurement Interval changes and/or what fields to suppress within specified event domains and/or a request to report the state of event throttling by event domain that is currently in force in the event source). Note: for ‘provideThrottlingState’ commands, the client should subsequently provide the throttling state by calling the provideThrottlingState operation. |

Body Fields (for error Responses):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| requestError | requestError | Yes (for errors) | Used only in error conditions. |

### HTTP Status Codes

|  |  |  |
| --- | --- | --- |
| *Code* | *Reason Phrase* | *Description* |
| 202 | Accepted | The request has been accepted for processing |
| 400 | Bad Request | Many possible reasons not specified by the other codes (e.g., missing required parameters or incorrect format). The response body may include a further exception code and text. HTTP 400 errors may be mapped to SVC0001 (general service error), SVC0002 (bad parameter), SVC2000 (general service error with details) or PO9003 (message content size exceeds the allowable limit). |
| 401 | Unauthorized | Authentication failed or was not provided. HTTP 401 errors may be mapped to POL0001 (general policy error) or POL2000 (general policy error with details). |
| 404 | Not Found | The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. |
| 405 | Method Not Allowed | A request was made of a resource using a request method not supported by that resource (e.g., using PUT on a REST resource that only supports POST). |
| 500 | Internal Server Error | The server encountered an internal error or timed out; please retry (general catch-all server-side error).HTTP 500 errors may be mapped to SVC1000 (no server resources). |

### Sample Request and Response

#### Sample Request

|  |
| --- |
| POST /eventListener/v5/eventBatch HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: 12345 {  "eventList": [  {  "commonEventHeader": {  "version": 3.0,  "domain": "fault",  "eventName": "Fault\_MobileCallRecording\_PilotNumberPoolExhaustion",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546015",  "sequence": 0,  "priority": "High",  "reportingEntityId": "cc305d54-75b4-431b-adb2-eb6b9e541234",  "reportingEntityName": "EricssonOamVf",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  "sourceName": "scfx0001vm002cap001",  "nfNamingCode": "scfx",  "nfcNamingCode": "ssc",  "startEpochMicrosec": 1413378172000000,  "lastEpochMicrosec": 1413378172000000  },  "faultFields": {  "faultFieldsVersion": 2.0,  "alarmCondition": "PilotNumberPoolExhaustion",  "eventSourceType": "other",  "specificProblem": "Calls cannot complete - pilot numbers are unavailable",  "eventSeverity": "CRITICAL",  "vfStatus": "Active",  "alarmAdditionalInformation": [  {  "name": "PilotNumberPoolSize",  "value": "1000"  }  ]  }  },  {  "commonEventHeader": {  "version": 3.0,  "domain": "fault",  "eventName": "Fault\_MobileCallRecording\_RecordingServerUnreachable",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546025",  "sequence": 0,  "priority": "High",  "reportingEntityId": "cc305d54-75b4-431b-adb2-eb6b9e541234",  "reportingEntityName": "EricssonOamVf",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  "sourceName": "scfx0001vm002cap001",  "nfNamingCode": "scfx",  "nfcNamingCode": "ssc",  "startEpochMicrosec": 1413378172000010,  "lastEpochMicrosec": 1413378172000010  },  "faultFields": {  "faultFieldsVersion": 2.0,  "alarmCondition": "RecordingServerUnreachable",  "eventSourceType": "other",  "specificProblem": "Recording server unreachable",  "eventSeverity": "CRITICAL",  "vfStatus": "Active"  }  }  ]  } |

#### Sample Success Response #1

For success responses without a provided commandList:

|  |
| --- |
| HTTPS/1.1 202 Accepted |

#### Sample Success Response #2

For success responses with a provided commandList:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {      “commandList”: [          {              “commandType”: “throttlingSpecification”,              “eventDomainThrottleSpecification”: {                  “eventDomain”: “fault”,                  “suppressedFieldNames”: [                      “alarmInterfaceA”,                      “alarmAdditionalInformation”                  ]              }          },          {              “commandType”: “throttlingSpecification”,              “eventDomainThrottleSpecification”: {                  “eventDomain”: “thresholdCrossingAlert”,                  “suppressedFieldNames”: [                      “associatedAlertIdList”,                      “possibleRootCause”                  ],                  “suppressedNvPairs” {                      “nvPairFieldName”: additionalParameters”,                      “suppressedNvPairNames”: [                          “someCounterName”,                          “someOtherCounterName”                      ]                  }              }          },          {              “commandType”: “measurementIntervalChange”,              “measurementInterval”: 600          },          {              “commandType”: “heartbeatIntervalChange”,              “heartbeatInterval”: 90          },          {              “commandType”: “provideThrottlingState”          }      ]  } |

#### Sample Error Responses

##### Sample Policy Exception

|  |
| --- |
| HTTPS/1.1 400 Bad Request  content-type: application/json  content-length: 12345  Date: Thu, 04 Jun 2009 02:51:59 GMT  {  “requestError”: {  “policyException”: {  “messageId”: “POL9003”,  “text”: “Message content size exceeds the allowable limit”,  }  }  } |

##### Sample Service Exception

|  |
| --- |
| HTTPS/1.1 400 Bad Request  content-type: application/json  content-length: 12345  Date: Thu, 04 Jun 2009 02:51:59 GMT  {  “requestError”: {  “serviceException”: {  “messageId”: “SVC2000”,  “text”: “Missing Parameter: %1. Error code is %2”  “variables”: [  “severity”,  “400”  ]  }  }  } |

## Operation: provideThrottlingState

### Functional Behavior

Allows authorized event source clients to report the state of event throttling by event domain that is currently in force in the event source.

* Supports only secure HTTPS (one way SSL) access.
* Uses the HTTP verb POST
* Supports application/json content types
* Provides HTTP response codes as well as Service and Policy error messages

### Call Flow

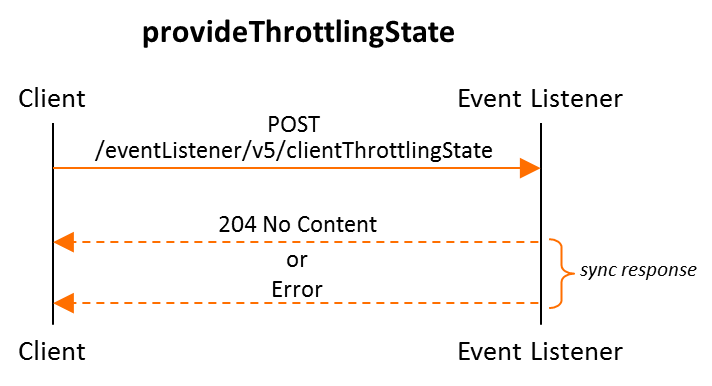


Figure - provideClientThrottlingState Call Flow

### Input Parameters

Header Fields (note: all parameter names shall be treated as case-insensitive):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| Accept | string | No | Determines the format of the body of the response. Valid values are:   * application/json |
| Authorization | string | Yes | The username and password are formed into one string as “username:password”. This string is then Base64 encoded to produce the encoded credential which is communicated in the header after the string “Authorization: Basic “. See examples below. If the Authorization header is missing, then an HTTP 400 Invalid Request message shall be returned. If the string supplied is invalid, then an HTTP 401 Unauthorized message shall be returned. |
| Content-length | integer | No | Note that content length is limited to 1Megabyte. |
| Content-type | string | Yes | Must be set to one of the following values:   * application/json |

Body Fields:

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| eventThrottlingState | eventThrottlingState | Yes | Consists of an eventThrottlingMode enumeration which can be ‘normal’ or ‘throttled’ followed by an optional array of eventDomainThrottleSpecification structures |

### Output Parameters

The only output parameters are an HTTP response code and message.

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| Content-length | integer | No | Used only in error conditions. |
| Content-type | string | No | Used only in error conditions. |

Body Fields:

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| requestError | requestError | No | Used only in error conditions. |

### HTTP Status Codes

|  |  |  |
| --- | --- | --- |
| *Code* | *Reason Phrase* | *Description* |
| 204 | No Content | The throttling state update message has been accepted. |
| 400 | Bad Request | Many possible reasons not specified by the other codes (e.g., missing required parameters or incorrect format). The response body may include a further exception code and text. HTTP 400 errors may be mapped to SVC0001 (general service error), SVC0002 (bad parameter), SVC2000 (general service error with details) or PO9003 (message content size exceeds the allowable limit). |
| 401 | Unauthorized | Authentication failed or was not provided. HTTP 401 errors may be mapped to POL0001 (general policy error) or POL2000 (general policy error with details). |
| 404 | Not Found | The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent. |
| 405 | Method Not Allowed | A request was made of a resource using a request method not supported by that resource (e.g., using PUT on a REST resource that only supports POST). |
| 409 | Locked | The request could not be completed due to a conflict with the current state of the resource. |
| 500 | Internal Server Error | The server encountered an internal error or timed out; please retry (general catch-all server-side error).HTTP 500 errors may be mapped to SVC1000 (no server resources). |
| 503 | Service Unavailable | The server is currently unable to handle the request due to a temporary overloading or maintenance of the server. The implication is that this is a temporary condition which will be alleviated after some delay. |
| 504 | Gateway Timeout | The server, while acting as a gateway or proxy, did not receive a timely response from the upstream process. |

### Sample Request and Response

#### Sample Request

|  |
| --- |
| POST /eventListener/v5/clientThrottlingState HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: nnn  accept: application/json  {  "eventThrottlingState": {  "eventThrottlingMode": "throttled",  "eventDomainThrottleSpecificationList": [  {  "eventDomain": "fault",  "suppressedFieldNames": [  "alarmInterfaceA",  "alarmAdditionalInformation"  ]  },  {  "eventDomain": "thresholdCrossingAlert",  "suppressedFieldNames": [  "associatedAlertIdList",  "possibleRootCause"  ],  "suppressedNvPairsList": [  {  "nvPairFieldName": "additionalParameters",  "suppressedNvPairNames": [  "someCounterName",  "someOtherCounterName"  ]  }  ]  }  ]  }  } |

#### Sample Success Response

|  |
| --- |
| HTTPS/1.1 204 No Content |

#### Sample Error Responses

##### Sample Policy Exception

|  |
| --- |
| HTTPS/1.1 400 Bad Request  content-type: application/json  content-length: 12345  Date: Thu, 04 Jun 2009 02:51:59 GMT  {  “requestError”: {  “policyException”: {  “messageId”: “POL9003”,  “text”: “Message content size exceeds the allowable limit”,  }  }  } |

##### Sample Service Exception

|  |
| --- |
| HTTPS/1.1 400 Bad Request  content-type: application/json  content-length: 12345  Date: Thu, 04 Jun 2009 02:51:59 GMT  {  “requestError”: {  “serviceException”: {  “messageId”: “SVC2000”,  “text”: “Missing Parameter: %1. Error code is %2”  “variables”: [  “severity”,  “400”  ]  }  }  } |

# Appendix: Historical Change Log

For the latest changes, see the Change Block just before the Table of Contents.

| Date | Revision | Description |
| --- | --- | --- |
| 5/22/2015 | 0.1 | Initial Release - Draft |
| 5/29/2015 | 0.2 | * Introduction: removed all system names and references to internal AT&T components * Security: changed ‘event publisher’ to ‘event source’ * Generic Event Format: updated the JSON schema per the below: * eventHeader: clarified the description of id, made sourceId a required field, changed the datatype of timestamps to timestamp [ ] * performanceFields: removed overflowFields * tmestamp: added a description of this datatype * Exceptions: fixed indentation of sections * Approvers: updated the list of approvers and added attuids |
| 6/3/2015 | 0.3 | * Updated the security section to use HTTP Basic Authentication per AT&T REST standards. Updated the input parameters and messaging examples to use the new security scheme. |
| 6/5/2015 | 0.4 | * Added otherFields sub section to the defined datatypes * Added locale field to the eventHeader. |
| 6/5/2015 | 0.5 | * Updated the embedded event format json schema to match the changes made in v0.4 |
| 6/10/2015 | 0.6 | * Updated the {ServerRoot} format to contain an optional routing path (for D2 service modules). |
| 7/7/2015 | 0.7 | Common Event Format updates:   * EventHeader: added ‘measurement’ to the ‘domain’ enumeration; changed ‘locale’ to ‘location’ and clarified in the description that this should be a clli code * Added a MeasurementFields datatype, which required the addition of the following datatypes: codecsInUse, cpuUsage, diskUsage, featuresInUse, memoryUsage |
| 7/15/2015 | 1.0 | * Changed sourceInstance in the eventHeader to be an array of name value pairs * Changed the performanceFields block to thresholdCrossingAlertFields. Updated the domain field of the eventHeader to match. |
| 7/23/2015 | v1.1 | Changes to eventHeader data format:   * moved sourceInstance to internalHeaderFields * moved serviceInstanceId to internalHeaderFields * moved productId to internalHeaderFields * moved subscriberId to internalHeaderFields * moved location to internalHeaderFields * added the following new fields in internalHeaderFields: policyType, policyName, correlationEventType, correlationType, correlationName, correlationRootEventId * Changes to faultFields data format: * moved the eventSourceDeviceDescription to internalFaultFields and renamed it equipmentVendorModel * moved eventSourceHostname to internalFaultFields * changed alarmObjectInterface to alarmInterfaceA * changed alarmRemoteObject to alarmRemoteObjectZ and moved it to internalFaultFields * changed alarmRemoteObjectInterface to alarmInterfaceZ and moved it to internalFaultFields * Changes to thresholdCrossingFields data format: * changed several references from the old ‘performanceFields’ block to the new ‘thresholdCrossingFields’ block * Other: * Fixed several comma and colon syntax errors in the JSON schema as detected by a JSON schema syntax checker. |
| 8/11/2015 | v1.2 | Timestamp format:   * Section 4.18: added a note in the datetime field of the Timestamp datatype specifying the (GMT) format required * Updated the JSON schema with the same information   Event Header Severity Enumeration:   * Section 4.8: modified the severity enumeration to remove the numbers in parentheses that followed the names. The names were not changed. * Updated the JSON schema with the same information. |
| 8/20/2015 | v1.3 | JSON Schema rev’d to v9:   * Alphabetized all fields in the JSON schema * Fixed the way arrays were specified (JSON schema syntax issue)   Sample Responses:   * 2.1.1.1: alphabetized fields, fixed timestamps array depiction, fixed severity enum value to conform to latest format * 6.2.6.1: alphabetized fields, fixed timestamps array depiction, fixed severity enum value to conform to latest format * 6.3.6.1: alphabetized fields, fixed timestamps array depiction, fixed severity enum value to conform to latest format * 6.4.6.1: alphabetized fields, fixed timestamps array depiction, fixed eventList array depection, fixed severity enum value to conform to latest format |
| 9/16/2015 | v1.4 | JSON Schema rev’d to v10:   * Fixed an error in the way that the top level “event” object was specified in the v9 json schema. This was discovered when validating examples against the schema using this site: <http://json-schema-validator.herokuapp.com/index.jsp>. * Changed the embedded json file in section 4   Sample Responses:   * Removed an extra comma after the timestamp brace in section 6.2.6 and 6.3.6. |
| 11/11/2015 | v1.5 | Section 4 was the only section changed: JSON Schema rev’d to v11 and Datatype tables were updated to match. Numerous data structure changes were made based on VNF vendor proof of concept feedback. Modified sample requests and responses to match. |
| 11/12/2015 | v1.6 | * The internalFaultFields were merged into the internalHeaderFields; then the internalFaultFields datatype was deleted. * Updated the JSON schema to v12. * Also corrected some background color issues in the sample requests and responses. |
| 1/18/2016 | v1.7 | * Section 2 changes: updated the sample request to conform with the changes below * Section 4 datatype changes: * Changed 'eventHeader' to 'commonEventHeader' * Moved 'eventSeverity' from the 'commonEventHeader' to 'faultFields' * Added 'priority' to 'commonEventHeader' * moved 'vFstatus' to 'faultFields' * removed 'firstDateTime' and 'lastDateTime' and changed 'firstEpoch' to 'startEpochMicrosec' and changed 'lastEpoch' to 'lastEpochMicrosec'. * Added 'functionalRole' to the commonEventHeader * In the commonEventHeader, changed the 'eventDomain' enumeration to remove 'measurements' and add 'measurementsForVfScaling'. * Changed the 'measurementFields' to 'measurementsForVfScalingFields' * In the commonEventHeader, changed the following fields: * 'eventDomain' to 'domain' * 'eventSequence' to 'sequence' * 'eventSourceId' to 'sourceId' * 'eventSounceName' to 'sourceName' * Updated the JSON schema to v13 * Section 6 changes: updated the input parameters and sample requests to conform to the changes above. * Section 7: changed the section from Approvers to Contributors. |
| 1/22/2016 | v1.8 | * Section 4: Added support for ‘mobileFlow’ in the commonEventHeader ‘domain’ enumeration. Added the mobileFlowFields datatype and the gtpPerFlowMetrics datatype referenced by that datatype. * Section 7: alphabetized the contributors |
| 2/11/2016 | v1.9 | * Added section 1.3: Naming Standard for Event Types |
| 2/12/2016 | v2.0 | * Updated request – response examples to reflect the naming standards for event types introduced in v1.9. * Added a paragraph on use of Avro as a transport in section 1.4 |
| 3/11/2016 | v2.1 | * Updated the embedded JSON schema to v15 to fix a typo in the required fields for the measurementsForVfScalingFields, namely, changed ‘configuredEntites’ to ‘configuredEntities’. Additionally, added an ‘Event Listener’ title block at the bottom of the file with a single required event object. |
| 3/15/2016 | v2.2 | * Added mobileFlowFields to the event datatype definition in section 4.7 and updated the embedded json schema at the top of section 4 to v16. |
| 4/26/2016 | v2.3 | * Generic Event Format updates: 1) made ‘priority’ lowercase in the Word doc table for commonEventHeader; 2) added ‘requestError’ data structure to the Word doc and JSON schema (which is now at v17) |
| 4/27/2016 | v2.4 | * JSON Schema: In the 'event' data structure, changed 'thresholdCrossingFields' to 'thresholdCrossingAlertFields' to product v18 of the schema. * 'codecsInUse' data structure: changed 'numberInUse' to 'codecUtilization’ |
| 5/26/2016 | v2.5 | * Changed responses from ‘204 No Content’ to ‘202 Accepted’ and added a body to the response that enable AT&T to throttle the events being sent and/or to request the current state of throttling at the event source. * Added new datatypes to support the above: eventDomainThrottleSpecification, eventDomainThrottleSpecificationList, eventThrottlingState, suppressedNvPairs * Modifed the commonEventFormat json schema to v19 * Note: for the VendorEventListener: added new licensing language on the back of the title page; added an “attCopyrightNotice” definition at the top of the commonEventFormat\_Vendors.json file; also removed all references to internalHeaderFields from this file and from the VendorEventListener spec. |
| 8/9/2016 | v2.6 | * commonHeader: added a note on the description of sourceId and sourceName in the commonHeader: "use reportingEntity for domains that provide more detailed source info" * commonHeader: deleted the capacity, measurementsForVfScaling and usage domains in the domain enumeration * commonHeader: added the following domains to the domain enumeration: licensingKci, scalingKpi, stateChange * event: removed references to capacityFields, measurementsForVfScalingFields and usageFields and added references to licensingKciFields, scalingKpiFields, stateChangeFields * licensingKciFields: added this section along with 'additionalMeasurements', which is an optional list of measurementGroup structures. Changed the name of kciFieldsVersion to licensingKciFieldsVersion. * scalingKpiFields: added this section but changed measurementFieldsVersion to scalingKpiFieldsVersion * stateChangeFields: added this section along with 'additionalFields', which is an optional list of name-value pairs. Other fields included newState and oldState which were enumerations of the following possible states: 'inService', 'maintenance', 'outOfService' * sysLogFields: added 'additionalFields', which is an optional list of name-value pairs * vNicUsage: added two required fields to the vNicUsage data structure: packetsIn and packetsOut |
| 8/10/2016 | v2.7 | * commonHeader: removed the note on the description of sourceId and sourceName in the commonHeader: "use reportingEntity for domains that provide more detailed source info" * commonHeader: added measurementsForVfScaling domain back and removed the licensingKci and scalingKpi domains * event: removed references to licensingKciFields and scalingKpiFields; added references to measurementsForVfScalingFields * measurementsForVfScalingFields: combined the kciDetail and kpiDetail structures into the measurementsForVfScalingFields structure; referenced the errors structure * errors: added a new structure to capture the receive and transmit errors for the measurements domain * removed the following structures: kci, kpi, scalingKpiFields and licensingKciFields * eventDomainThrottleSpecification: updated the reference to commonEventHeader domain field * faultFields: removed the numbers from the enumerated strings for eventSourceType * vNicUsage: made the broadcast, multicast and unicast fields optional * contributors: updated Alok’s organizational area |
| 8/12/2016 | v2.8 | * commonHeader: copied the descriptions of sourceId and sourceName from the JSON schema into the word document tables. * sample request examples: moved the reportingEntityId and reportingEntityNames to the same relative place in all sample requests in the document * Fixed the sample request shown for publishEventBatch to take an eventList as input. * Fixed the sample request shown for publishSpecificTopic to put the topic in the URL * errors: changed the receiveErrors and transmitErrors fields to be datatype number * codesInUse: changed 'codecUtilization' to 'numberinUse' * vNicUsage: updated the description of the fields |
| 8/27/2016 | v2.9 | * Added a note "(currently: 1.1)" in the descriptions of the following fields: commonEventHeader:version, faultFields:faultFieldsVersion, measurementsForVfScalingFields:measurementsForVfScalingFieldsVersion, stateChangeFields:stateChangeFieldsVersion, sysLogFields:syslogFieldsVersion, thresholdCrossingAlertFields:thresholdCrossingFieldsVersion * stateChangeFields: made stateInterface mandatory * changed 'enum' to 'enumeration' throughout section 4 of the document (note: this can't be done in the JSON schema). * measurementsForVfScalingFields: made the following fields optional: conurrentSessions, configuredEntitites, cpuUsageArray, fileSystemUsageArray, memoryConfigured, memoryUsed, requestRate, vNicUsageArray * measurementsForVfScalingFields: concurrentSessions and configuredEntities: changed the description to support both VMs and VNFs * measurementsFor VfScalingFields: clarified the descriptions of latencyDistribution, measurementInverval and requestRate * syslogFields: clarified the descriptions of syslogSData, syslogTag, syslogVer * thresholdCrossingAlertFields: made the following fields optional and clarified their descriptions: elementType, networkService * command and commandList: created a list of command structures to enable the event collector to request changes of event sources. Commands consist of a commandType along with optional fields (whose presence is indicated by the commandType). Three command types are currently supported: 'measurementIntevalChange', ‘provideThrottlingState’ and 'throttlingSpecification'. * eventDomainThrottleSpecificationList: removed this and replaced it with commandList. * Operations and Sample Requests: modified the operations and samples to support the new command and commandList structures. |
| 9/1/2016 | v2.10 | * measurementsForVfScaling block: made the following fields optional: latencyDistribution (which is an array of latencyBucketMeasure structures) and meanRequestLatency. Updated the JSON schemas (now v24) to match. |
| 9/16/2016 | v2.11 | * 1 Introduction: updated the introduction to clarify the usage of eventTypes and the possibility of support for other protocols. * 6.1 REST Operation Overview: added two new subsections (6.1.2 and 6.1.3) discussing Api Version and Commands Toward Event Source Clients. * 6.2 publishAnyEvent: fixed the sample to conform to the latest changes * 6.3 publishSpecificTopic: fixed the sample to conform to the latest changes * 6.4 publishEventBatch: fixed the sample to conform to the latest changes * 6.5 provideThrottlingState operation: added the Input Parameters section heading back and fixed the sample request to provide eventThrottlingState (instead of eventThrottlingClientState). * The remaining bullets describe changes made to section 4 datatypes in alphabetical order: * command datatype: referenced the new section 6.1.3 which provides an explanation of command state expectations and requirements for a given eventSource: * commonEventHeader datatype:   + made sourceId and reportingEntityId fields optional (although the internal Generic Event Listener spec indicates, in the field descriptions, that the AT&T enrichment process shall ensure that these fields are populated)   + domain enumeration: changed measurementsForVfScalingFields to measurementsForVfScaling * eventDomainThrottleSpecificationList: added this array of eventDomainThrottleSpecification stuctures back to the schema because it is used by the provideThrottlingState operation. * eventList: added eventList back to the vendor version of the commonEventFormat. This is used by the publishEventBatch operation. * faultFields datatype:   + eventSourceType: made this a string (and provided the previous enumerated values as examples) * filesystemUsage datatype:   + changed vmIdentifier to filesystemName * gtpPerFlowMetrics datatype:   + flowActivationTime: changed the format and description to be compliant with RFC 2822.   + flowDeactivationTime: changed the format and description to be compliant with RFC 2822. * internalHeaderFields datatype:   + Added the following optional fields: firstDateTime, lastDateTime compliant with RFC 2822. Noted in the description that these fields must be supplied for events in the following domains: fault, thresholdCrossingAlerts and measurementsForVfScaling.   + ticketingTimestamp: changed the format and description to be compliant with RFC 2822. * syslogFields datatype:   + eventSourceType: made this a string (and provided the previous enumerated values, without the numbers, as examples) * thresholdCrossingAlerts dataypte:   + collectionTimestamp: changed the format and description to be compliant with RFC 2822.   + eventStartTimestamp: changed the format and description to be compliant with RFC 2822.   + added the same eventSeverity field as from the faultFields and made it required |
| 9/23/2016 | v2.12 | * Section 4 Datatypes: commonEventHeader: made reportingEntityName a required field (note: the JSON schema already had this field as required) |
| 11/29/2016 | v3.0 | * Introduction:   + Introductory paragraph: changed '...Common Event Header Block followed by zero or more event domain blocks' to '...Common Event Header Block accompanied by zero or more event domain blocks' since the order of the blocks on the wire is not guaranteed.   + Added Section 1.5 Versioning * Section 4: codec processing:   + CommonEventFormat\_Vendors schema only: codesInUse: changed required field from "codecUtilization" which was removed previously to "numberInUse" which is the new field name.   + added ‘codecSelected’ datatype   + added ‘codecSelectedTranscoding’ datatype * Section 4 and section 6: command processing:   + Added commandListEntry which is an object that references the command object.   + commandList: changed commandList to contain an array of commandListEntry objects.   + Updated sample responses in section 6 where commands are used * Section 4: commonEventHeader:   + Incremented version to 1.2   + added two new values to the ‘domain’ enumeration: ‘serviceEvents’ and ‘signaling * Section 4: added endOfCallVqmSummaries datatype * Section 4: ‘event’: added two fields: ‘serviceEventsFields’ and ‘signalingFields’ * Section 4: added ‘eventInstanceIdentifier’datatype * Section 4: CommonEventListener only: internalHeaderFields:   + added ‘internalHeaderFieldsVersion’(initially set to 1.1)   + added ‘correlationFirstEpoch’   + added 'closedLoopControlName'   + added 'closedLoopFlag'   + added 'collectorTimeStamp'   + added 'eventTag'   + added ‘tenantName’   + changed 'operationalStatus' to 'inMaint'   + added required fields in the schema to match the word doc: 'equipmentNameCode', 'equipmentType', 'equipmentVendor', 'inMaint', 'provStatus' * Section 4: added ‘marker’datatype * Section 4: added ‘midCallRtcp’ datatype * Section 4: mobileFlowFields:   + added ‘mobileFlowFieldsVersion’(initially set to 1.1) * Section 4: added ‘serviceEventsFields’datatype * Section 4: added ‘signalingFields’ datatype * Section 4: syslogFields:   + Incremented syslogFieldsVersion to 1.2   + added 'syslogPri'   + added 'syslogSev'   + added ‘syslogSdId’ * Section 4: thresholdCrossingAlertFields:   + Incremented thresholdCrossingFieldsVersion to 1.2   + added 'additionalFields' which is an optional list of name value pairs. * Section 4: schema v26.0 embedded reflecting the above changes. * Section 6 and Section 2: changed all sample requests to use /v3 in the REST Resource URL. |
| 12/1/2016 | v3.1 | * Section 6: Updated the call flow diagrams to show ‘v3’ |
| 1/5/2017 | v4.0 | * Combined the Generic Event Listener and Vendor Event Listener into a single API service specification with version 4.0. * Changed the title to VES (Virtual Function Event Streaming) Listener. * Changed references to 'generic event' to 'common event' or 'VES event' (depending on the context) throughout the document. * Used the Legal Disclaimer from the Vendor Event Listener on the back of the title page. * Section 1: Introduction changes:   + modified wording to reference 'VES'   + removed the 'Audience' section, which described various AT&T groups the documented was intended for   + tweaked the naming standards for event types to clarify the purpose of the naming conventions * Section 3: Resource Structure: added a sentence describing the FQDN and port used in the resource URL. * Section 4: Common Event Format changes:   + renamed the section to 'Common Event Format' from 'Generic Event Format'   + reorganized the datatypes into separate sections; sections were defined for each of the domains as well as for common event, common event header and command list processing   + codecSelected datatype: removed this datatype   + codecSelectedTranscoding datatype: removed this datatype   + command datatype: added an enumerated value to commandType: 'heartbeatIntervalChange'   + commonEventHeader: added internalHeaderFields to the commonEventHeader, defined as "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."   + commonEventHeader: removed two enumerated values, 'serviceEvents' and 'signaling' from the domain enumeration   + commonEventHeader version: incremented the version to 2.0   + endOfCallVqmSummaries datatype: removed this datatype   + event: changed the description of the event datatype to: "fields which constitute the ‘root level’ of the common event format"   + event: removed 'serviceEventFields' and 'signalingFields' from the definition   + event: fixed a misspelling of ‘thresholdCrossingAlertFields’, which was only present in the Word document   + eventInstanceIdentifier datatype: removed this datatype   + internalHeaderFIelds datatype: defined this as follows: "The internalHeaderFields datatype is an undefined object which can contain arbitrarily complex JSON structures. It is intended to be defined separately by each provider implementing the VES Event Listener. The fields in internalHeaderFields are not provided by any event source but instead are added by the VES Event Listener service itself as part of an event enrichment process necessary for efficient internal processing of events received by the VES Event Listener"   + marker datatype: removed this datatype   + measurementsForVfScalingFields datatype: clarified that memoryConfigured and memoryUsed are measured in MB   + midCallRtcp datatype: removed this datatype   + mobileFlowFields datatype: added ‘additionalFields’   + mobileFlowFields datatype: incremented the version number for this field block to 1.2   + serviceEventsFields datatype: removed this datatype   + signalingFields datatype: removed this datatype   + syslogFields: added three fields to the schema that were previously described in the document but not incorporated into the schema: syslogPri, syslogSev, syslogSdId   + syslogFields version: incremented the version to 2.0 * Modified the Common Event Format JSON schema to v27.0 to incorporate the above changes. Also, added the AT&T Copyright Notice from the top of the retired CommonEventFormat\_Vendors schema. * Section 6 and 2: changed all sample requests to use /v4 in the REST Resource URL and call flow diagrams. * Section 6.1.3: added a row to the table in this section describing the ‘heartbeatIntervalChange’ command. * Section 6.1.4: added this new section describing expectations for buffering of events should all REST resource URL FQDNs be unreachable. * Section 6 Sample Requests: modified all sample requests showing the return of a commandList toward the event source to incorporate a heartbeatIntervalChange command; also corrected the spelling in the samples for the measurementIntervalChange command. * Section 7: Contributors: removed this section |
| 3/21/2017 | v4.1 | * JSON Schema changes to produce v27.2 (note: an earlier draft version of v27.1 had been distributed to a few individuals):   + To support use of the schema with event batches, removed the following statement near the end of the schema file:   “required”: [ “event” ]   * + Fixed the characters used in some of the quotes   + Fixed some typos in the descriptions.   + Removed the booleans, which were non-essential and which were causing problems across different implementations. * Section 4.5.7 measurementsForVfScalingFields:   + Fixed the spelling of measurementsForVfScalingFields in the Word document * Section 2 and 6 sample requests and responses:   + Removed quotes from numbers: sequence, and first/lastEpochMicrosec.   + Fixed all quote characters, some of which were using unusual symbols that wouldn’t validate with the json-schema Python package. * Section 6.2.6.1, 6.3.6.1, 6.4.6.1 sample requests:   + Added an alarmAdditionalInformation field array to the sample requests.   + Added missing commas. * Section 6.5.6.1 provideThrottlingState sample requests:   + Fixed the eventDomainThrottleSpecificationList to pass an array of anonymous eventDomainThrottleSpecification objects.   + Added missing quotes. * Fixed the suppressedNvPairsList to pass an array of anonymous suppressedNvPairs objects. |
| 4/14/2017 | v5.0 | * Section 1 Introduction:   + Clarified the Introduction (Section 1).   + Changed Section 1.1 title from ‘Terminology’ to 'Event Registration' and referenced the YAML event registration format, defined in a separate document.   + Clarified naming standards for eventName. * Section 3: updated the REST resource structure * Section 4.1 command list processing datatypes:   + Got rid of commandListEntry and returned commandList to a simple array of commands.   + Added heartbeatInterval to the command datatype.   + Changed the datatype of measurementInterval from number to integer. * Section 4.2 common event datatypes:   + event dataType: Added heartbeatFields, sipSignalingFields and voiceQualityFields to the event datatype as optional field blocks   + Added jsonObject which provides a json object schema, name and other meta-information along with one or more object instances.   + Added jsonObjectInstance which provides meta-information about an instance of a jsonObject along with the actual object instance   + Added the ‘key’ datatype   + Added the namedArrayOfFields datatype   + Added vendorVnfNameFields * Section 4.3 common event header fields:   + Add two new enumerations to domain: ‘sipSignaling’ and ‘voiceQuality’   + Renamed eventType to eventName. Note that the original usage of eventType was formally described in the Introduction back on 2/11/2016 with v1.9.   + Made eventName a required field   + Created a new field called eventType with a meaning that is different than the old eventType.   + Removed functionalRole, which was replaced by the following two fields.   + Added nfNamingCode   + Added nfcNamingCode   + Changed version to 3.0 (major version change) and made it a required field * Section 4.4: faultFields:   + added one optional field: eventCategory   + made faultFieldsVersion a required field   + changed faultFieldsVersion to 2.0 (major version change)   + fixed a typo on the spelling of alarmInterfaceA   + clarified field descriptions * Section 4.5: added heartbeatFields datatype which can be used to communicate heartbeatInterval. Note: this change was previously made in v4.2 * Section 4.6 measurements for vf scaling datatypes: changed the following datatypes from number to integer:   + In measurementsForVfScalingFields: concurrentSessions, configuredEntities, numberOfMediaPortsInUse, vnfcScalingMetric   + In codecsInUse: numberInUse   + In featuresInUse: featureUtilization * Section 4.6.2 modified cpuUsage * Section 4.6.3 added diskUsage * Section 4.6.7 measurementsForVfScalingFields:   + fixed the spelling of the measurementsForVfScalingFields in the Word document   + added additionalFields, which is an array of fields (i.e., name-value pairs)   + changed additionalMeasurements to reference the common datatype namedArrayOfFields (instead of referencing measurementGroup)   + added additionalObjects which is an array of jsonObjects described by name, keys and schema   + deleted aggregateCpuUsage   + added diskUsageArray   + deleted measurementGroup (which was replaced by the common datatype: namedArrayOfFields   + added memoryUsageArray   + deleted memoryConfigured and memoryUsed   + deleted errors and vNicUsageArray   + added vNicPerformanceArray   + changed the measurementsForVfScalingVersion to 2.0 (major version change) and made it a required field. Also changed the name of this version field in the Word document to match that in the JSON schema. * Section 4.6.8 added memoryUsage * Section 4.6.9 vNicPerformance: replaced vNicUsage and errors with vNicPerformance * Section 4.7 mobile flow fields changes:   + Made mobileFlowFieldsVersion a required field and changed the mobileFlowFieldsVersion to 2.0 (major version change).   + Changed the datatype of flowActivationTime and flowDeactivationTime in the Word doc to string.   + changed the following datatypes from number to integer: otherEndpointPort, reportingEndpointPort, samplingAlgorithm * Section 4.8: otherFields:   + Added otherFieldsVersion (set at 1.1)   + Added hashOfNameValuePairArrays   + Added jsonObjects   + Added nameValuePairs * Section 4.9: added sipSignaling domain datatypes with 4.8.1 sipSignalingFields. sipSignalingFieldsVersion is set at 1.0 * Section 4.10 stateChangeFields: made stateChangeFieldsVersion a required field and set it to 2.0 (major version change). * Section 4.11 syslogFields:   + Changed the following datatypes from number to integer: syslogFacility, syslogPri   + Changed additionalFields from a field [ ] to a string which takes name=value pairs delimited by a pipe symbol.   + Changed syslogFieldsVersion to 3.0 (major version change) and made it a required field   + Made syslogSev an enumerated string (previously just a string) * Section 4.12 thresholdCrossingAlertFields: made thresholdCrossingFieldsVersion a required field and set it to 2.0 (major version change). * Section 4.132: added voice quality domain datatypes with 4.13.1 endOfCallVqmSummaries and 4.13.2 voiceQualityFields. voiceQualityFieldsVersion is set at 1.0 * JSON Schema: changed the schema to v28.0 and incorporated all of the changes above. * Additional JSON Schema changes that are part of v28: Note: The following changes are provided relative to API Spec v4.0 (which embedded JSON schema v27.0), but they were also made in an interim release v4.1 (which embedded JSON schema v27.2):   + To support use of the schema with event batches, removed the following statement near the end of the schema file:   “required”: [ “event” ]   * + Fixed the characters used in some of the quotes   + Fixed some typos in the descriptions.   + Removed the booleans, which were non-essential and which were causing problems across different implementations. * Section 2 and 6 sample requests and responses (also incorporated in interim release 4.1):   + Removed quotes from numbers: sequence, and first/lastEpochMicrosec.   + Fixed all quote characters, some of which were using unusual symbols that wouldn’t validate with the json-schema Python package. * Section 2 and 6 sample requests and responses (only in v5.0):   + Changed the version numbers in the URL string.   + Added nfNamingCode and nfcNamingCode and removed functionalRole * Section 6 call flows: updated the version number (only in v5.0). * Section 6: removed the publishSpecificTopic operation * Section 6.1.4: Buffering: clarified event source expectations for buffering (only in v5.0). * Section 6.2.6.1, 6.3.6.1 sample requests (also incorporated in interim release 4.1):   + Added an alarmAdditionalInformation field array to the sample requests.   + Added missing commas. * Section 6.2.6.3, 6.3.6.3 commandList sample responses (only in v5.0):   + Fixed the commandList sample responses to pass an array of anonymous command objects (rather than an array of commandListEntry objects).   + Fixed the heartbeatIntervalChange commandType to pass a heartbeatInterval value instead of a measurementInterval value.   + Removed quotes from the measurementInterval and heartbeatInterval values since they are numbers. * Section 6.4.6.1 provideThrottlingState sample requests (also incorporated in interim release 4.1):   + Fixed the eventDomainThrottleSpecificationList to pass an array of anonymous eventDomainThrottleSpecification objects.   + Added missing quotes.   + Fixed the suppressedNvPairsList to pass an array of anonymous suppressedNvPairs objects (also incorporated in interim release 4.1). |
| 5/22/2017 | v5.1 | * Footers: removed proprietary markings and updated copyrights to 2017 * Section 4.2.3: field:   + Changed the API spec to make ‘name’ and ‘value’ start with lowercase letters. Note: this did not affect the schema, which already had them as lowercase. * JSON Schema:   + measurementGroup: deleted this object since it was replaced with ‘namedArrayOfFields’ in v28.0 and was no longer being used.   + namedArrayOfFields: Fixed an error in the specification of required fields: from ‘measurements’ to ‘arrayOfFields’. * Changed the version of the JSON schema to 28.1 |
| 6/14/2017 | v5.2 | * JSON Schema: created v28.2 by changing the field descriptions in the memoryUsage object to refer to ‘kibibytes’ instead of ‘kilobytes’. There were no changes to the 28.1 structure. * Word Document: measurementsForVfScaling Domain: memoryUsage object: changed the field descriptions in this object to refer to ‘kibibytes’ instead of ‘kilobytes’. There were no changes to the memoryUsage structure. * Reorganized the Word document to group the data structures in Section 4 into three broad categories to better align with the VNF Guidelines documentation that has been prepared for vendors:   + Common Event Datatypes:     - Command List Processing Datatypes     - Common Event Datatypes     - Common Event Header Datatypes   + Technology Independent Datatypes:     - ‘Fault Domain Datatypes     - ‘Heartbeat’ Domain Datatypes     - ‘Measurements For Vf Scaling’ Domain Datatypes     - ‘Other’ Domain Datatypes     - ‘State Change’ Domain Datatypes     - ‘Syslog’ Domain Datatypes     - ‘Threshold Crossing Alert’ Domain Datatypes   + Technology Specify Datatypes:     - ‘Mobile Flow’ Domain Datatypes     - ‘Sip Signaling’ Domain Datatypes     - ‘Voice Quality’ Domain Datatypes * Section 6.1.3: Commands Toward Event Source Clients: Added a statement: “Note: Vendors are not currently required to implement support for command processing; in addition, command processing may be supported by an App-C interface in future.” |

### 