AT&T Service Specification

Service: *Vendor Event Listener*

|  |  |
| --- | --- |
| Document Number | xxxx |
| Revision | 2.5 |
| Revision Date | 5/26/2016 |
| Author | Rich Erickson |

© 2015-2016, 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.

.

[1 Introduction 1](#_Toc453856926)

[1.1 Audience 1](#_Toc453856927)

[1.2 Terminology 1](#_Toc453856928)

[1.3 Naming Standards for Event Types 2](#_Toc453856929)

[1.4 Support for Protocols Other Than HTTPS 2](#_Toc453856930)

[2 Security 3](#_Toc453856931)

[2.1.1 Sample Request and Response 3](#_Toc453856932)

[2.1.1.1 Sample Request 3](#_Toc453856933)

[2.1.1.2 Sample Success Response 3](#_Toc453856934)

[3 Resource Structure 5](#_Toc453856935)

[4 Generic Event Format 6](#_Toc453856936)

[4.1 Datatype: capacityFields 6](#_Toc453856937)

[4.2 Datatype: codecsInUse 6](#_Toc453856938)

[4.3 Datatype: commonEventHeader 6](#_Toc453856939)

[4.4 Datatype: counter 7](#_Toc453856940)

[4.5 Datatype: cpuUsage 7](#_Toc453856941)

[4.6 Datatype: filesystemUsage 7](#_Toc453856942)

[4.7 Datatype: event 8](#_Toc453856943)

[4.8 Datatype: eventDomainThrottleSpecification 8](#_Toc453856944)

[4.9 Datatype: eventDomainThrottleSpecificationList 9](#_Toc453856945)

[4.10 Datatype: eventList 9](#_Toc453856946)

[4.11 eventThrottlingState 9](#_Toc453856947)

[4.12 Datatype: faultFields 9](#_Toc453856948)

[4.13 Datatype: featuresInUse 10](#_Toc453856949)

[4.14 Datatype: field 10](#_Toc453856950)

[4.15 Datatype: gtpPerFlowMetrics 10](#_Toc453856951)

[4.16 Datatype: latencyBucketMeasure 14](#_Toc453856952)

[4.17 Datatype: measurementsForVfScalingFields 14](#_Toc453856953)

[4.18 Datatype: measurementGroup 15](#_Toc453856954)

[4.19 Datatype: mobileFlowFields 15](#_Toc453856955)

[4.20 Datatype: otherFields 17](#_Toc453856956)

[4.21 Datatype: requestError 17](#_Toc453856957)

[4.22 Datatype: suppressedNvPairs 17](#_Toc453856958)

[4.23 Datatype: syslogFields 17](#_Toc453856959)

[4.24 Datatype: thresholdCrossingAlertFields 18](#_Toc453856960)

[4.25 Datatype: usageFields 19](#_Toc453856961)

[4.26 Datatype: vNicUsage 19](#_Toc453856962)

[5 Exceptions 21](#_Toc453856963)

[5.1 RESTful Web Services Exceptions 21](#_Toc453856964)

[5.2 Service Exceptions 21](#_Toc453856965)

[5.3 Policy Exceptions 22](#_Toc453856966)

[6 RESTful Web Services Definition 24](#_Toc453856967)

[6.1 REST Operation Summary 24](#_Toc453856968)

[6.2 Operation: publishAnyEvent 24](#_Toc453856969)

[6.2.1 Functional Behavior 24](#_Toc453856970)

[6.2.2 Call Flow 25](#_Toc453856971)

[6.2.3 Input Parameters 25](#_Toc453856972)

[6.2.4 Output Parameters 26](#_Toc453856973)

[6.2.5 HTTP Status Codes 27](#_Toc453856974)

[6.2.6 Sample Request and Response 27](#_Toc453856975)

[6.2.6.1 Sample Request 27](#_Toc453856976)

[6.2.6.1 Sample Success Response #1 28](#_Toc453856977)

[6.2.6.2 Sample Success Response #2 28](#_Toc453856978)

[6.2.6.1 Sample Success Response #3 29](#_Toc453856979)

[6.2.6.1 Sample Error Responses 29](#_Toc453856980)

[6.3 Operation: publishSpecificTopic 30](#_Toc453856981)

[6.3.1 Functional Behavior 30](#_Toc453856982)

[6.3.2 Call Flow 31](#_Toc453856983)

[6.3.3 Input Parameters 31](#_Toc453856984)

[6.3.4 Output Parameters 32](#_Toc453856985)

[6.3.5 HTTP Status Codes 33](#_Toc453856986)

[6.3.6 Sample Request and Response 34](#_Toc453856987)

[6.3.6.1 Sample Request 34](#_Toc453856988)

[6.3.6.2 Sample Success Response #1 34](#_Toc453856989)

[6.3.6.3 Sample Success Response #2 34](#_Toc453856990)

[6.3.6.4 Sample Success Response #3 35](#_Toc453856991)

[6.3.6.5 Sample Error Responses 35](#_Toc453856992)

[6.4 Operation: publishEventBatch 36](#_Toc453856993)

[6.4.1 Functional Behavior 36](#_Toc453856994)

[6.4.2 Call Flow 37](#_Toc453856995)

[6.4.3 Input Parameters 37](#_Toc453856996)

[6.4.4 Output Parameters 38](#_Toc453856997)

[6.4.5 HTTP Status Codes 39](#_Toc453856998)

[6.4.6 Sample Request and Response 39](#_Toc453856999)

[6.4.6.1 Sample Request 39](#_Toc453857000)

[6.4.6.2 Sample Success Response #1 40](#_Toc453857001)

[6.4.6.3 Sample Success Response #2 40](#_Toc453857002)

[6.4.6.4 Sample Success Response #3 41](#_Toc453857003)

[6.4.6.5 Sample Error Responses 41](#_Toc453857004)

[6.5 Operation: provideThrottlingState 42](#_Toc453857005)

[6.5.1 Functional Behavior 42](#_Toc453857006)

[6.5.2 Call Flow 43](#_Toc453857007)

[6.5.3 Output Parameters 44](#_Toc453857008)

[6.5.4 HTTP Status Codes 44](#_Toc453857009)

[6.5.5 Sample Request and Response 45](#_Toc453857010)

[6.5.5.1 Sample Request 45](#_Toc453857011)

[6.5.5.2 Sample Success Response 46](#_Toc453857012)

[6.5.5.3 Sample Error Responses 46](#_Toc453857013)

Figures

[Figure 1 – REST Resource Structure 5](#_Toc452044112)

[Figure 2 - publishAnyEvent Call Flow 24](#_Toc452044113)

[Figure 3 - publishSpecificTopic Call Flow 30](#_Toc452044114)

[Figure 4 – publishEventBatch Call Flow 36](#_Toc452044115)

[Figure 5 - provideClientThrottlingState Call Flow 42](#_Toc452044116)

Tables

[Table 1 - Service Exceptions 21](#_Toc452044117)

[Table 2 - Policy Exceptions 22](#_Toc452044118)

[Table 3 - REST Operation Summary 23](#_Toc452044119)

# Introduction

This document describes the AT&T-internal RESTful interface for AT&T’s Generic Event Listener. The Generic Event Listener is capable of receiving any event sent in AT&T’s Common Event Format. The Common Event Format is a JSON structure consisting of a required Common Event Header Block followed by zero or more event domain blocks. A JSON Schema of the Common Event Format is provided later in this document.

It should be understood that events are well structured packages of information, identified by an eventType, which are asynchronously communicated to subscribers who are interested in the eventType. Events can convey measurements, faults, CDRs, alerts, geolocation updates, and much more. Even something as unusual as SIP signaling messages could be captured by a system and published as a SIP Signaling Event to interested subscribers. 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 future, additional documents may describe other transports which make use of persistent TCP connections for high volumes of streaming events.

## Audience

General audience for API service specifications are:

* Security Architecture and Governance
* TD and D2 Architecture
* Testing Organizations
* Service Development Teams
* OSS/BSS Systems Teams
* AT&T Solution Providers
* AT&T VNF Providers

## Terminology

Topics, also known as Event Types, use the generic event format but may require that specific fields be present including specific name-value pairs in the extensible structures provided within the generic event structure.

Events are instances of topics.

## Naming Standards for Event Types

To prevent naming collisions, eventTypes sent as part of the **Error! Reference source not found.**, should follow the following convention:

{DomainAbbreviation}\_{ServiceOrResourceOrSystemName}\_{DescriptionOfInfoBeingConveyed}

Domain abbreviations are derived from the ‘domain’ field in the **Error! Reference source not found.**, as specified below:

* ‘Capacity’ for the capacity domain
* ‘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
* ‘Syslog’ for the syslog domain
* ‘Tca’ for the thresholdCrossingAlert domain
* ‘Usage’ for the usage domain

Examples of eventTypes following the naming standards are provided below:

* Fault\_MobileCallRecording\_PilotNumberPoolExhaustion
* Heartbeat\_vIsbcMmc
* Other\_Mso\_L3toHlsInstantiationStage1Complete
* Syslog\_vdbe
* Tca\_vdbe\_CpuThresholdExceeded

Any questions about eventType naming should be resolved as part of service and resource onboarding to ASDC (AT&T Service Design and Creation).

## Support for Protocols Other Than HTTPS

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

An alternative interface may be provided using Apache Avro which leverages a JSON schema (provided in this document) to support a compact binary data format over an RPC protocol to be defined.

# Security

Event sources must identify themselves to the Generic 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/v1 HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: 12345 {  "event": {  "commonEventHeader": {  “domain”: “heartbeat”,  "eventType": "Heartbeat\_vIsbcMmc",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546015",  "sequence": "0",  "priority": "Normal",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  “sourceName”: “EricssonECE”,  “functionalRole”: “SCF”,  “startEpochMicrosec”: “1413378172000000”,  “lastEpochMicrosec”: “1413378172000000”,  “reportingEntityId”: “de305d54-75b4-431b-adb2-eb6b9e546014”,  “reportingEntityName”: “EricssonECE”  }  }  } |

#### 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:

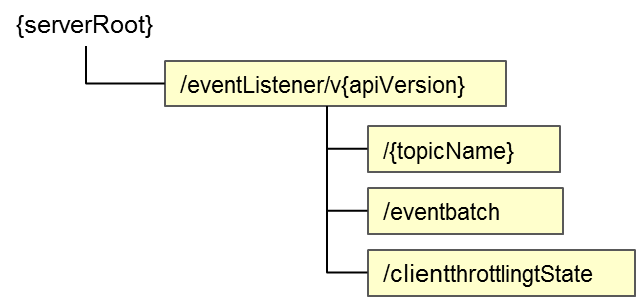


Figure – REST Resource Structure

# Generic Event Format

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



## Datatype: capacityFields

The capacityFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| capacityFieldsVersion | number | No | Version of the capacityFields block |
| additionalFields | field [ ] | No | Array of name-value pair fields |

## 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 |
| codecUtilization | Number | Yes | Number of such codecs in use |

## Datatype: commonEventHeader

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

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| version | number | No | Version of the event header |
| eventType | string | No | Unique event topic name |
| domain | string | Yes | Event domain enum: ‘capacity’, ‘fault’, ‘heartbeat’, ‘measurement’, ‘mobileFlow’, ‘other’, ‘syslog’, ‘thresholdCrossingAlert’, ‘usage’ |
| eventId | string | Yes | Event key that is unique to the event source |
| sourceId | string | Yes | uuid identifying the initial source (i.e., device) of the event |
| sourceName | string | Yes | Name of the device generating the event |
| functionalRole | string | Yes | Function of the event source e.g., eNodeB, MME, PCRF |
| reportingEntityId | string | Yes | UUID identifying the entity reporting the event, for example an EMS |
| reportingEntityName | string | Yes | Name of the entity reporting the event, for example, an EMS |
| priority | string | Yes | Processing priority enum: ‘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) |

## 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 | Enum: ‘CRIT’, ‘MAJ’ |

## Datatype: cpuUsage

The cpuUsage datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| cpuIdentifer | string | Yes | CPU Identifier |
| percentUsage | Number | Yes | CPU usage in percent |

## Datatype: filesystemUsage

The filesystemUsage datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| vmIdentifer | string | Yes | Virtual machine identifier |
| 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: event

The event datatype consists of the following fields which constitue the generic or common event format:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| commonEventHeader | commonEventHeader | Yes | Fields common to all events |
| capacityFields | capacityFields | No | Fields specific to capacity events |
| faultFields | faultFields | No | Fields specific to fault events |
| measurementsForVfScalingFields | measurementsForVfScalingFields | No | Fields specific to measurementsForVfScaling events |
| mobileFlowFields | mobileFlowFields | No | Fields specific to mobility flow events |
| otherFields | field [ ] | No | Fields specific to other types of events |
| syslogFields | syslogFields | No | Fields specific to syslog events |
| threholdCrossingAlertFields | thresholdCrossingAlertFields | No | Fields specific to threshold crossing alert events |
| usageFields | usageFields | No | Fields specific to usage events |

## 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 Event Listener Spec, currently: ‘capacity’, ‘fault’, ‘heartbeat’, ‘measurement’, ‘mobileFlow’, ‘other’, ‘syslog’, ‘thresholdCrossingAlert’, ‘usage’ |
| 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 is an array of eventDomainThrottleSpecifications; it consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| eventDomainThrottleSpeciciationList | eventDomainThrottleSpecification [ ] | Yes | List of fields to suppress within a given eventDomain |

## Datatype: eventList

The eventList datatype consists of the following fields:

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

## 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 | Enum: ‘normal’, ‘throttled’ |
| eventDomainThrottleSpecificationList | eventDomainThrottleSpecificationList | No | A list of eventDomainThrottleSpecifications currently in force at the event source, if the eventManagerMode is ‘throttled’ |

## Datatype: faultFields

The faultFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| faultFieldsVersion | number | No | Version of the faultFields block |
| eventSeverity | string | Yes | Event severity or priority enum: ‘CRITICAL’, ‘MAJOR’, ‘MINOR’, ‘WARNING’, ‘NORMAL’ |
| eventSourceType | string | Yes | Enum: ‘other(0)’, ‘router(1)’, ‘switch(2)’, ‘host(3)’, ‘card(4)’, ‘port(5)’, ‘slotThreshold(6)’, ‘portThreshold(7)’, ‘virtualMachine(8)’, ‘virtualNetworkFunction(9)’ |
| alarmCondition | string | Yes | Alarm condition reported by the device |
| specificProblem | string | Yes | Short description of the alarm or problem |
| vfStatus | string | Yes | Virtual function status enum: ‘Active’, ‘Idle’, ‘Preparing to terminate’, ‘Ready to terminate’, ‘Requesting Termination’ |
| alarmtInterfaceA | string | No | Card, port, channel or interface name of the device generating the alarm |
| alarmAdditional Information | field [ ] | No | Additional alarm information |

## 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 |
| feautureUtilization | Number | Yes | Number of times the identified feature was used |

## 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: 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 | Dateto,e | 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--should be like 'Wed, 15 Oct 2014 13:01:52 GMT' |
| flowDeactivatedBy | string | No | Endpoint deactivating the flow |
| flowDeactivationEpoch | number | Yes | Time for the start of the flow connection, in integer UTC epoch time aka UNIX time |
| flowDeactivationMicrosec | number | Yes | Integer microseconds for the start of the flow connection |
| flowDeactivationTime | datetime | Yes | Transmission time of the first packet in the flow connection being reported on with the following sample format: ‘Wed, 15 Oct 2014 13:01:52 GMT’ |
| 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: 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 measuremensForVfScalingtFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| measurementFieldsVersion | number | No | Version of the measurementFields block |
| aggregateCpuUsage | number | No | Aggregate CPU usage of the VM on which the VNFC reporting the event is running |
| codecUsageArray | codecsInUse [] | No | Array of codecs in use |
| concurrentSessions | number | Yes | Peak concurrent sessions for the VM over the measurementInterval |
| configuredEntities | number | Yes | Peak total number of users, subscribers, devices, adjacencies, etc., for the VM over the measurementInterval |
| cpuUsageArray | cpuUsage [] | Yes | Usage of an array of CPUs |
| featureUsageArray | featuresInUse [] | No | Array of features in use |
| filesystemUsageArray | filesystemUsage [] | Yes | Filesystem usage of the VM on which the VNFC reporting the event is running |
| latencyDistribution | latencyBucketMeasure [ ] | Yes | Array of integers representing counts of requests whose latency falls within per-VNF configured ranges |
| meanRequestLatency | number | Yes | 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 the usage measures are being reported |
| memoryConfigured | number | Yes | Memory configured in the VM on which the VNFC reporting the event is running |
| memoryUsed | number | Yes | Memory usage of the VM on which the VNFC reporting the event is running |
| numberOfMediaPortsInUse | number | No | Number of media ports in use |
| requestRate | number | Yes | Peak request rate per second, for the VM over the measurementInterval |
| vnfcScalingMetric | number | No | Represents busy-ness of the VNF from 0 to 100 as reported by the VNFC |
| vNicUsageArray | vNicUsage [ ] | Yes | Usage of an array of virtual network interface cards |
| additionalMeasurements | measurementGroup [ ] | No | Additional measurement fields |

## Datatype: measurementGroup

The measurementGroup datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| name | string | Yes | Name for the measurement Group |
| measurements | field [ ] | Yes | Name value pair measurements |

## Datatype: mobileFlowFields

The mobileFlowFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| 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 | string | Yes | IP Port for the reporting entity, as used for the flow being reported on |
| otherFunctionalRole | string | No | Functional role of the other endpoint for the flow being reported on e.g., MME, S-GW, P-GW, PCRF... |
| rac | string | No | Routing area code |
| radioAccessTechnology | string | No | Radio Access Technology e.g., 2G, 3G, LTE |
| reportingEndpointIpAddr | string | Yes | IP address for the reporting entity, as used for the flow being reported on |
| reportingEndpointPort | string | Yes | IP port for the reporting entity, as used for the flow being reported on |
| sac | string | No | Service area code |
| samplingAlgorithm | string | No | Integer identifier for the sampling algorithm or rule being applied in calculating the flow metrics if metrics are calculated based on a sample of packets, or 0 if no sampling is applied |
| tac | string | No | Transport area code |
| tunnelId | string | No | Tunnel identifier |
| vlanId | string | No | VLAN identifier used by this flow |

## Datatype: otherFields

The otherFields datatype is simply a field [ ]

## 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 **Error! Reference source not found.**) 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: 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) |

## Datatype: syslogFields

The syslogFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| eventSourceHost | string | No | Hostname of the device |
| eventSourceType | string | Yes | Type of event source enum: ‘other(0)’, ‘router(1)’, ‘switch(2)’, ‘host(3)’, ‘card(4)’, ‘port(5)’, ‘slotThreshold(6)’, ‘portThreshold(7)’, ‘virtualMachine(8)’, ‘virtualNetworkFunction(9)’ |
| syslogFacility | number | No | Numeric code from 0 to 23 for facility:  0 kernel messages  1 user-level messages  2 mail system  3 system daemons  4 security/authorization messages  5 messages generated internally by syslogd  6 line printer subsystem  7 network news subsystem  8 UUCP subsystem  9 clock daemon  10 security/authorization messages  11 FTP daemon  12 NTP subsystem  13 log audit  14 log alert  15 clock daemon (note 2)  16 local use 0 (local0)  17 local use 1 (local1)  18 local use 2 (local2)  19 local use 3 (local3)  20 local use 4 (local4)  21 local use 5 (local5)  22 local use 6 (local6)  23 local use 7 (local7 ) |
| syslogFieldsVersion | number | No | Version of the syslogFields block |
| syslogMsg | string | Yes | Syslog message |
| 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 |
| syslogVer | number | No | The version of the syslog protocol specification |
| syslogSData | string | No | Syslog structured data |
| syslogTag | string | Yes | Aka messageId identifies the type of message |

## Datatype: thresholdCrossingAlertFields

The thresholdCrossingAlertFields datatype consists of the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| additionalParameters | counter [ ] | Yes | Array of performance counters |
| alertAction | string | Yes | Enum: ‘SET’, ‘CONT’, ‘CLEAR’ |
| alertDescription | string | Yes | Unique short alert description (e.g., NE-CPUMEM) |
| alertType | string | Yes | Enum: ‘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 |
| dataCollector | string | No | Specific performance collector instance used |
| elementType | string | Yes | Type of network element |
| eventStartTimestamp | string | Yes | Time closest to when the measurement was made |
| interfaceName | string | No | Physical or logical port or card (if applicable) |
| networkService | string | Yes | Network name |
| possibleRootCause | string | No | Reserved for future use |
| thresholdCrossing FieldsVersion | number | No | Version of the thresholdCrossingAlertFields block |

## Datatype: usageFields

[omitted]

## Datatype: vNicUsage

The vNicUsage datatype consists of the following fields which describe the usage of an identified vNic in megabytes per second:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| broadcastPacketsIn | number | Yes | Per identified vNic in megabytes per second |
| broadcastPacketsOut | number | Yes | Per identified vNic in megabytes per second |
| bytesIn | number | Yes | Per identified vNic in megabytes per second |
| bytesOut | number | Yes | Per identified vNic in megabytes per second |
| multicastPacketsIn | number | Yes | Per identified vNic in megabytes per second |
| multicastPacketsOut | number | Yes | Per identified vNic in megabytes per second |
| unicastPacketsIn | number | Yes | Per identified vNic in megabytes per second |
| unicastPacketsOut | number | Yes | Per identified vNic in megabytes per second |
| vNicIdentifier | string | Yes | vNic identification |

# 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 Generic 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 Generic 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 Summary

| **Operation Action** | **HTTP**  **Verb** | **Resource URL relative to {ServerRoot}** |
| --- | --- | --- |
| publishAnyEvent | POST | /eventListener/v{apiVersion} |
| publishSpecificTopic | POST | /eventListener/v{apiVersion/{topicName} |
| publishEventBatch | POST | /eventListener/v{apiVersion}/eventBatch |
| provideClientThrottlingState | POST | /eventListener/v{apiVersion}/clientThrottlingState |

Table - REST Operation Summary

## Operation: publishAnyEvent

### Functional Behavior

Allows authorized clients to publish any single event to the generic 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

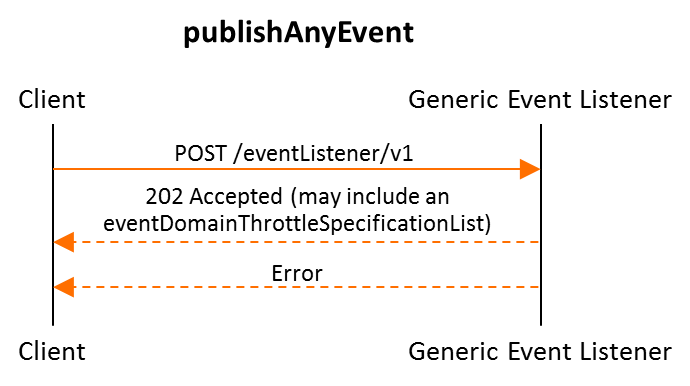


Figure - publishAnyEvent Call Flow

### 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 throttling specification or request for throttling state provided): no content is provided and the header fields are not required.

Body Fields (for success responses with a throttling specification provided):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| eventDomainThrottleSpecificationList | eventDomainThrottleSpecificationList | Yes (for throttling specs) | Array of specifications for what fields to suppress within the event domains |

Body Fields (for success responses with a request for throttling state—the client should subsequently provide the throttling state by calling the provideThrottlingState operation):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| provideThrottlingState | string | Yes (for requests for throttling state) | Enumeration with one value: ‘yes’ |

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/v1 HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: 12345 {  "event": {  "commonEventHeader": {  “domain”: “fault”,  "eventType": "Fault\_MobileCallRecording\_PilotNumberPoolExhaustion",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546015",  "sequence": "0",  "priority": "High",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  “sourceName”: “EricssonECE”,  “functionalRole”: “SCF”,  “startEpochMicrosec”: “1413378172000000”,  “lastEpochMicrosec”: “1413378172000000”,  “reportingEntityId”: “de305d54-75b4-431b-adb2-eb6b9e546014”,  “reportingEntityName”: “EricssonECE”  },  "faultFields": {  "alarmCondition": "PilotNumberPoolExhaustion",  "eventSourceType": "other(0)",  "specificProblem": "Calls cannot complete because pilot numbers are unavailable"  "eventSeverity": "CRITICAL",  “vfStatus”: “Active”  }  }  } |

#### Sample Success Response #1

For success responses without a throttling specification provided:

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

#### Sample Success Response #2

For success responses with a throttling specification provided:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {  “eventDomainThrottleSpecificationList”: [  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “fault”,  “suppressedFieldNames”: [  “alarmInterfaceA”,  “alarmAdditionalInformation”  ]  }  },  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “thresholdCrossingAlert”,  “suppressedFieldNames”: [  “associatedAlertIdList”,  “possibleRootCause”  ],  “suppressedNvPairsList”: [  {  “suppressedNvPairs” {  “nvPairFieldName”: additionalParameters”,  “suppressedNvPairNames”: [  “someCounterName”,  “someOtherCounterName”  ]  }  }  ]  }  }  ]  } |

#### Sample Success Response #3

For success responses that request the event source to report the state of event throttling by event domain that is currently in force in the event source.:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {  “provideThrottlingState”: “yes”  } |

#### 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: publishSpecificTopic

### Functional Behavior

Allows authorized clients to publish any single event to the generic 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

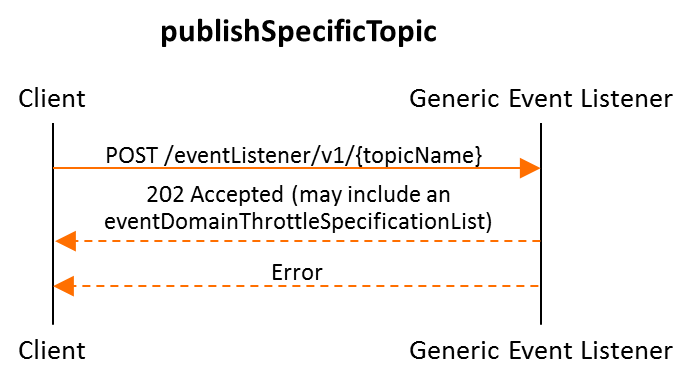


Figure - publishSpecificTopic Call Flow

### Input Parameters

Querystring parameters:

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| TopicName | string | Yes | Specifies the specific event topic which the event body must contain. |

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 throttling specification provided): no content is provided and the header fields are not required.

Body Fields (for success responses with a throttling specification provided):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| eventDomainThrottleSpecificationList | eventDomainThrottleSpecificationList | Yes (for throttling specs) | Array of specifications for what fields to suppress within the event domains |

Body Fields (for success responses with a request for throttling state—the client should subsequently provide the throttling state by calling the provideThrottlingState operation):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| provideThrottlingState | string | Yes (for requests for throttling state) | Enumeration with one value: ‘yes’ |

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/v1 HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: 12345 {  "event": {  "commonEventHeader": {  “domain”: “fault”,  "eventType": "Fault\_MobileCallRecording\_PilotNumberPoolExhaustion",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546015",  "sequence": "0",  "priority": "High",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  “sourceName”: “EricssonECE”,  “functionalRole”: “SCF”,  “startEpochMicrosec”: “1413378172000000”,  “lastEpochMicrosec”: “1413378172000000”,  “reportingEntityId”: “de305d54-75b4-431b-adb2-eb6b9e546014”,  “reportingEntityName”: “EricssonECE”  },  "faultFields": {  "alarmCondition": "PilotNumberPoolExhaustion",  "eventSourceType": "other(0)",  "specificProblem": "Calls cannot complete because pilot numbers are unavailable"  "eventSeverity": "CRITICAL",  “vfStatus”: “Active”  }  }  } |

#### Sample Success Response #1

For success responses without a throttling specification provided:

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

#### Sample Success Response #2

For success responses with a throttling specification provided:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {  “eventDomainThrottleSpecificationList”: [  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “fault”,  “suppressedFieldNames”: [  “alarmInterfaceA”,  “alarmAdditionalInformation”  ]  }  },  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “thresholdCrossingAlert”,  “suppressedFieldNames”: [  “associatedAlertIdList”,  “possibleRootCause”  ],  “suppressedNvPairsList”: [  {  “suppressedNvPairs” {  “nvPairFieldName”: additionalParameters”,  “suppressedNvPairNames”: [  “someCounterName”,  “someOtherCounterName”  ]  }  }  ]  }  }  ]  } |

#### Sample Success Response #3

For success responses that request the event source to report the state of event throttling by event domain that is currently in force in the event source.:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {  “provideThrottlingState”: “yes”  } |

#### 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 generic 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

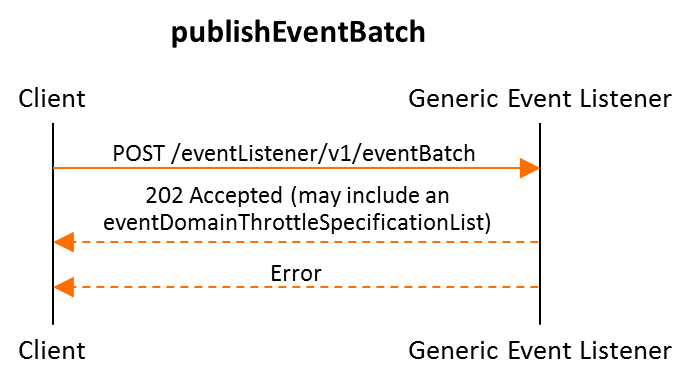


Figure – publishEventBatch Call Flow

### 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 throttling specification provided): no content is provided and the header fields are not required.

Body Fields (for success responses with a throttling specification provided):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| eventDomainThrottleSpecificationList | eventDomainThrottleSpecificationList | Yes (for throttling specs) | Array of specifications for what fields to suppress within the event domains |

Body Fields (for success responses with a request for throttling state—the client should subsequently provide the throttling state by calling the provideThrottlingState operation):

| **Parameter** | **Data Type** | **Required?** | **Brief description** |
| --- | --- | --- | --- |
| provideThrottlingState | string | Yes (for requests for throttling state) | Enumeration with one value: ‘yes’ |

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/v1 HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: 12345 {  "event": {  "commonEventHeader": {  “domain”: “fault”,  "eventType": "Fault\_MobileCallRecording\_PilotNumberPoolExhaustion",  "eventId": "ab305d54-85b4-a31b-7db2-fb6b9e546015",  "sequence": "0",  "priority": "High",  "sourceId": "de305d54-75b4-431b-adb2-eb6b9e546014",  “sourceName”: “EricssonECE”,  “functionalRole”: “SCF”,  “startEpochMicrosec”: “1413378172000000”,  “lastEpochMicrosec”: “1413378172000000”,  “reportingEntityId”: “de305d54-75b4-431b-adb2-eb6b9e546014”,  “reportingEntityName”: “EricssonECE”  },  "faultFields": {  "alarmCondition": "PilotNumberPoolExhaustion",  "eventSourceType": "other(0)",  "specificProblem": "Calls cannot complete because pilot numbers are unavailable"  "eventSeverity": "CRITICAL",  “vfStatus”: “Active”  }  }  } |

#### Sample Success Response #1

For success responses without a throttling specification provided:

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

#### Sample Success Response #2

For success responses with a throttling specification provided:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {  “eventDomainThrottleSpecificationList”: [  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “fault”,  “suppressedFieldNames”: [  “alarmInterfaceA”,  “alarmAdditionalInformation”  ]  }  },  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “thresholdCrossingAlert”,  “suppressedFieldNames”: [  “associatedAlertIdList”,  “possibleRootCause”  ],  “suppressedNvPairsList”: [  {  “suppressedNvPairs” {  “nvPairFieldName”: additionalParameters”,  “suppressedNvPairNames”: [  “someCounterName”,  “someOtherCounterName”  ]  }  }  ]  }  }  ]  } |

#### Sample Success Response #3

For success responses that request the event source to report the state of event throttling by event domain that is currently in force in the event source.:

|  |
| --- |
| HTTPS/1.1 202 Accepted  content-type: application/json  content-length: nnn  date: Sat, 04 Jul 2015 02:03:15 GMT  {  “provideThrottlingState”: “yes”  } |

#### 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

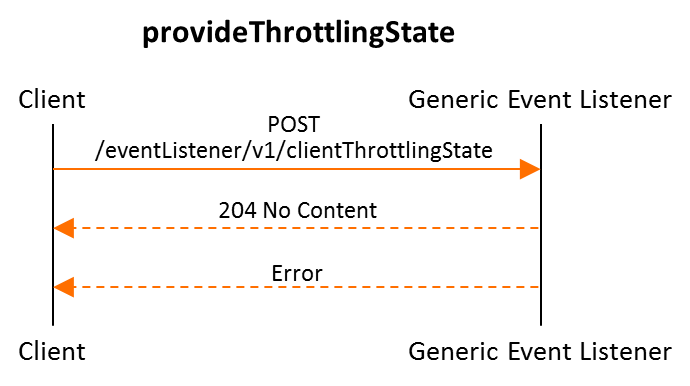


Figure - provideClientThrottlingState Call Flow

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 eventThrottlingrMode enumeration which can be ‘normal’ or ‘throttled’ followed by an optional array of eventDomainThrottlingSpecification 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/v1/clientThrottlingState HTTPS/1.1  Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==  content-type: application/json  content-length: nnn  accept: application/json  {  “eventThrottlingClientState”: {  “eventThrottlingMode”: “throttled”,  “eventDomainThrottleSpecificationList”: [  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “fault”,  “suppressedFieldNames”: [  “alarmInterfaceA”,  “alarmAdditionalInformation”  ]  }  },  {  “eventDomainThrottleSpecification”: {  “eventDomain”: “thresholdCrossingAlert”,  “suppressedFieldNames”: [  “associatedAlertIdList”,  “possibleRootCause”  ],  “suppressedNvPairsList”: [  {  “suppressedNvPairs” {  “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”  ]  }  }  } |