VNF Event Stream

Project Name:

- Proposed name for the project: VNF Event Stream (VES)
- Proposed name for the repository: ves

Project description:

Objective:

This project will develop OPNFV platform support for VNF event streams, in a common model and format intended for use by NFV Service Providers (SPs), e.g. in managing VNF health and lifecycle. The project's goal is to enable a significant reduction in the effort to develop and integrate VNF telemetry-related data into automated VNF management systems, by promoting convergence toward a common event stream format and collection system.

Powerpoint intro to the project: OPNVF VES.pptx

Background

In a typical Service Assurance (SA) Environment, the VNF Provider defines and provides the format and protocol for sending telemetry data to SPs responsible for managing VNF health and lifecycle. The standards for telemetry data come in many varieties and for each particular standard, the expected actions also differ from one VNF to another. As an example:

- Fault Events can be conveyed via SNMP, 3GPP Corba, MTOSI, OSSJ etc. For each standard the actions can differ from one VNF to another (e. g. Critical Severity can be marked as 1 in one implementation, 5 for another).
- For Measurement events, related to Key Performance Indicators (KPI)/Key Capacity Indicator (KCI), they can be in Comma Separated files (CSV) or XML. Within these file formats, events vary across implementations.

The result of a multitude of standards is that every time a new VNF or a new VNF release is introduced, implementer choice can require SPs to make changes to their SA support system that manages the VNFs. This results not only in added costs for SPs, but also in deployment delays.

Recommendations Summary:

We are recommending a Common Event Data Model:

- A VNF Event Stream (VES) Common Data Model consisting of a Header and an Event Specific Block (Domain is specified in the header).
- Additional name/value fields for extensibility.

Use of Restful API using JSON with Common Event Format for Telemetry Data (detailed definitions for Heartbeat. Fault, Measurements (KPI/KCI), Syslogs, Mobile Flow and with extensibility to others, i.e. Usage, Security, Signaling etc.).

Scope:

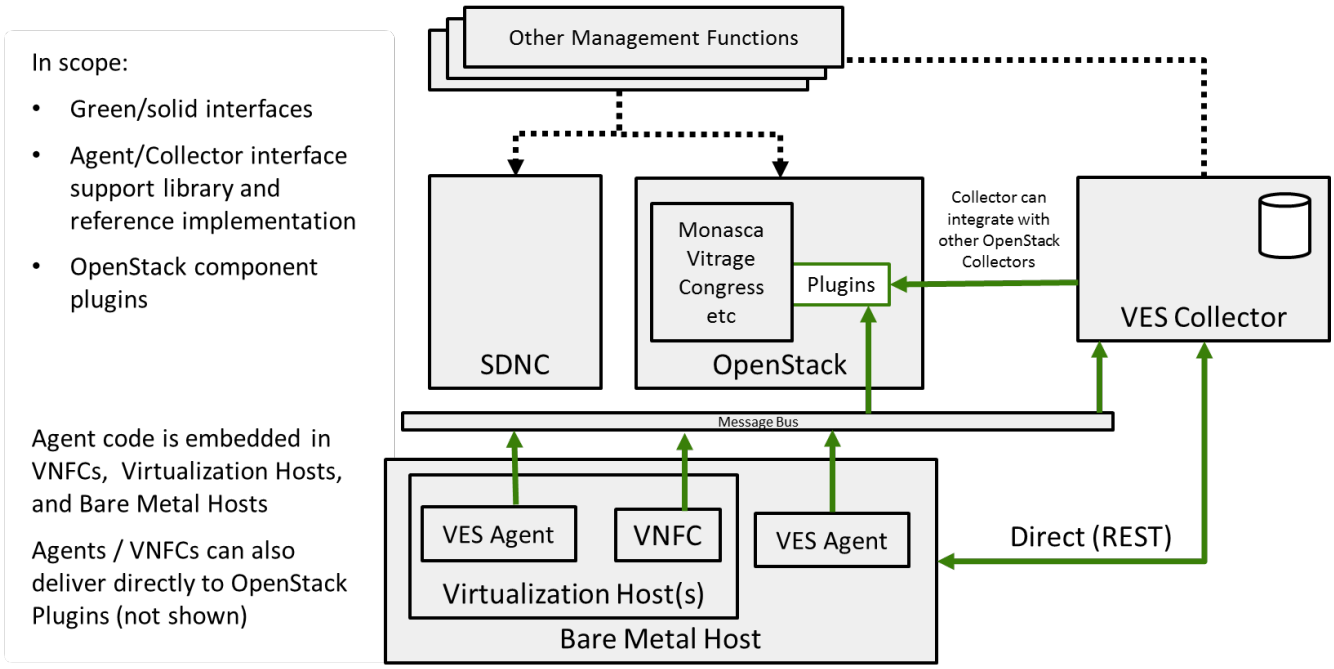
The project will develop and integrate into the OPNFV platform:

- A Common Event Data Model ("VNF Event Stream")
- OPNFV platform support the VNF Event Stream, including:
    - A VES Agent that can collect the VNF Event Stream data from the VNF and deliver it to the VES Collector
        - This includes telemetry aggregation from bare metal through to application KPIs - including host OS telemetry, and infrastructure
    - A VES Collector that can consume the VNF Event Stream data, and store the data in a database backend
    - VES plugins for integration with OpenStack infrastructure services such as Monasca and Vitrage
- A plan for upstreaming the project code

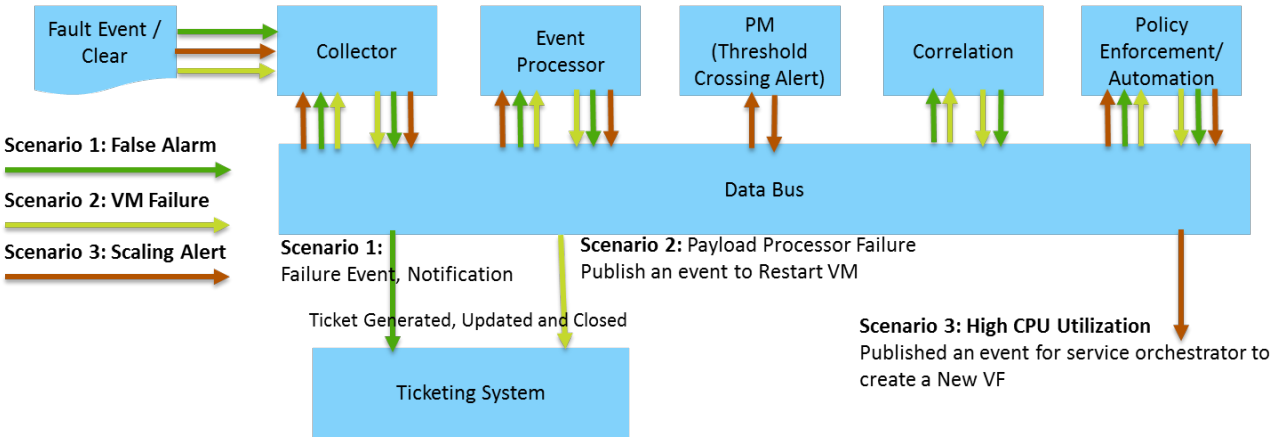
As input this project, AT&T will contribute a proposed specification and implementation of the VNF Event Stream framework developed for AT&T's ECOMP (Enhanced Control, Orchestration, Management and Policy) platform, and described as part of the "Analytic Framework" support in the ECOMP white paper available at http://att.com/ecomp.

The following diagram illustrates the concept for the VES architecture and scope:

VES Architectural View



In the diagram above, examples of Other Management Functions" (outside the scope of VES) are illustrated in the following use case diagram:



Testability:

The project will support testability through:

- VES collector support under all OPNFV base system installers, either as a base system deploy component or post-deploy installed components
- Test cases to be supported through Functest and DoveTail
- Reference VNF generating events, for lab testing per the event definitions

Documentation:

The project will produce:

- Detailed step by step documentation for developing to and using the code and event specification.
- Blueprint proposals for upstreaming the event library, collector, and specification.

Dependencies:

VES is a standardized event framework that can be used by OpenStack and OPNFV projects.

OPNFV projects that potentially benefit from the VES project:

- Fault Management (Doctor)
- Virtualized Infrastructure Deployment Policies (Copper)
- High Availability for OPNFV (Availability)
- Data Collection for Failure Prediction (Prediction)
- Audit (Inspector)
- Fault localization (RCA, Pinpoint)
- Service Function Chaining (sfc)
- Moon Security Management

OpenStack projects that potentially benefit from the VES project, or need to be considered in the VES design:

- Monasca: a monitoring-agent plugin for VES can provide a new/converged source for Monasca events
- Vitrage: a VES plugin can provide new data for the Graph DB of Vitrage
- Ceilometer and AODH: a VES plugin can populate the ceilometer DB
- Congress: Congress data source drivers can integrate additional data from Monasca, Vitrage, or directly
- Tacker: (From Sridhar, Tacker PTL) I'd like to explore the possibility of introducing a VES "event collector / listener" in the Tacker side to write policies ({ event-trigger-action}) based on the event-stream coming from the VNFs.
- Fault_Genes-WG: creation of an OpenStack failure mode taxonomy to be used from design to deployment of OpenStack

Other considerations noted during discussion of the proposal:

- There are efforts underway in OpenStack to do log normalisation, efforts to do telemetry aggregation for Neutron in Ceilometer (e.g. due to scaling issues), and there are also common needs across VMs, containers, and bare metal for host OS telemetry from something like collectd, and the "standard" ELK stack for log streams.
- This is an opportunity to ensure that whatever happens at the host and infrastructure layer is scalable in a way that fits the needs of VNFs, however if we focus on something as domain specific as a bus on top of the infrastructure with Yang models to define the object model for the messages, we may end up with something that will look strange to infrastructure management/OpenStack developers.
- A fault (say, a performance indicator going outside of acceptable bounds) could happen anywhere in the stack, in the guest OS or application code (where you would see it), on the host (if there's a noisy neighbour, for example), on the bare metal (some kind of hardware fault), or even on the network (if there is some network component involved), and that you would want to be aggregating and correlating data from all these sources to do accurate root cause analysis on the origin of the fault and the appropriate mitigating action.
- Re Fault_Genes-WG, this group is focused on "creation of an OpenStack failure mode taxonomy to be used from design to deployment of OpenStack". This would be important to consider in the model, certainly, similar to the other aspects mentioned re alignment with standards, OpenStack projects, etc.
- Re data model convergence ("multiple efforts to normalise logs using different technologies "), since we are modeling the data to be collected, we have the chance to evolve that model as the standards and open source converge. But for now it may be difficult to pick a single "gold standard", and that really isn't the intent anyway. The intent of this project is to enable all the relevant data, in whatever format, to be delivered from the hosts /VNFs using an easily-integrated set of agent and collector code artifacts.
- Re concern over creating another "event bus": this can mean various things. If the protocol/framework for eventing, Rabbit for example is a bus. Or the stream of events related to a particular purpose e.g. monitoring, which operates over a bus. VES is related to both, and is intended to address the very issues arising from having multiple ways to get analytics and fault info today: SNMP, CORBA, MTOSI (via JMS or REST), or the various agents/protocols developed/used by OpenStack as a VIM.
- As we consider dependencies and opportunities for convergence and "standardization" (which typically in an open source world occurs after convergence, or the excessive success of one approach), we need to consider these in their context and re various forums/projects related to them:
    - application-specific data
    - bare metal / virtualization host-related data
    - measurements
    - alerts
    - faults
    - syslogs

Committers and Contributors:

Committers:

- Alok Gupta, AT&T
- Bryan Sullivan, AT&T
- Feng Liu (feng.liu1@huawei.com)
- Li Yuanzhen(li.yuanzhen@zte.com.cn), ZTE

Contributors:

- Prakash Ramchandran, (Futurewei Technologies Inc.)
- Ifat Afek, Nokia
- (Contributor list is in development)

Planned deliverables:

The project will deliver specifications and code as described in the scope section, and supporting information:

- Reference VNF generating events, for lab testing per the event definitions.
- Detailed step by step documentation for developing to and using the code and event specification.
- Blueprint proposals for upstreaming the event library, collector, and specification.

Proposed Release Schedule:

VES is targeted for the 4th OPNFV release (“D” river). For the Colorado release, an initial reference implementation will be developed and made available for testing on the OPNFV Colorado release platform, as well as by VNF implementers.

Key Project Facts:

Project Name: VNF Event Stream (VES)
Repo name: VES
Lifecycle State: Incubation
Primary Contact: Alok Gupta, AT&T
Project Lead: Alok Gupta, AT&T
Jira Project Name: VNF Event Stream
Jira Project Prefix: VES
mailing list tag: VES
Committers:

- Alok Gupta, AT&T
- Bryan Sullivan, AT&T
- Feng Liu (feng.liu1@huawei.com)
- Li Yuanzhen(li.yuanzhen@zte.com.cn), ZTE