We live in a world evolving to require any machine-to-any machine communication. All parts of today’s digital business infrastructure need to be able to interact to facilitate such communication.

IT administrators need a framework for increased automation that makes them more agile and able to react and respond to events that arise in day-to-day business operations. With business infrastructure and information systems becoming more advanced and involving a complex interaction among various component sub-systems, agility is no longer optional—it is foundational to modern IT.

An automation framework offers more than just increased agility and productivity. In the world of IT security, where high exposure to security threats is a given, automation fulfills the now-crucial requirement to detect and respond quickly and continuously.

The solution to meeting these challenges for agility through automation and faster security response is to integrate a set of open, RESTful APIs into a network visibility infrastructure, enabling security and other network monitoring appliances to interact directly. By leveraging these open RESTful APIs, administrators can develop and use a programmable framework to enhance and extend the foundation of pervasive visibility. They can automate multiple functions in their infrastructure, including: dynamic response to detected threat patterns, adjustments to traffic mode configurations for in-line security tools, and additional IT operations-management functions and capabilities.

Visibility Fabric Backgrounder

Today’s data networks are teeming with devices, appliances, and applications. Network complexity is increasing dramatically on a daily basis, a trend that has been significantly enhanced by the ever-broadening adoption of virtual infrastructure, the need to leverage and monetize Big Data, and user device explosion. Just considering security architecture alone requires a variety of tools, including Next-generation Firewalls (NGFW), Intrusion Protection Systems (IPS), Web Application Firewalls (WAF), Intrusion Detection Systems (IDS), Security Information and Event Management Systems (SIEMs), and other inline or out-of-band security appliances. Still, all these protection systems are only as effective as the network traffic they see. In fact, the breadth of visibility to network traffic directly impacts the effectiveness of any security architecture.

For pervasive visibility, the network traffic should be acquired from as many of the devices and applications present in the data center and spanning physical, virtual and SDN/NFV environments, as well as private and public clouds. This distributed layer of high-performance nodes forms a Visibility Fabric™ that provides a view of the entire infrastructure to any operational tools that require inputs of network traffic or flow records derived from network traffic. This type of a unified visibility model eliminates blind spots and furnishes quick access to the whole network.

Software-Defined Visibility (SDV): What it is and Why it Matters

To understand what Software-Defined Visibility (SDV) is, let us begin by asking:

- What if the applications and other operational tools that receive traffic from a Visibility Fabric also had a way to respond dynamically to events they detect without waiting for administrative intervention?
- What if the security application that detected a threat pattern had the capability to auto-adjust traffic to react and respond to the threat?
- What is the best way to implement visibility with automation?

If you are a CISO, or stakeholder in IT operations, your response to the first two questions was likely, “That would be great!” The third question, however, might give you pause.

Pervasiveness of the Visibility Fabric is a necessary architectural foundation, but that alone is not sufficient to address the most salient of the current challenges: emergence of new blind spots requires dynamic changes to the visibility infrastructure to be done in order to first detect and then eliminate those blind spots. IT administrators need a framework for increased automation so that the visibility infrastructure can respond dynamically to events or situations that diminish network access. These capabilities are the building blocks for modern IT. How can they best be implemented?

A well-understood, general, and powerful approach is to integrate a Web-services framework based on RESTful Application Programming Interfaces (APIs) directly into the Visibility Fabric itself. This approach allows any device on the network to interact directly with the Visibility Fabric on an as-needed basis. APIs
exposed through a centralized policy controller provide the ability for external systems to interact with the Visibility Fabric in a programmatic fashion. These open RESTful APIs support programmability in the Visibility Fabric itself to ensure visibility that is pervasive, dynamic, active, and therefore very agile as well. We refer to this highly programmable and easy-to-automate framework as Software-Defined Visibility (SDV), a new paradigm for network security and IT operations management.

Using RESTful APIs to Implement SDV

Scalable Web services provide the fundamental architecture for the majority of activity on today's Web. One of the most common ways of implementing them is within the framework provided by the Representational State Transfer (REST) architectural approach. Based on a client-server model, the RESTful architecture has numerous characteristics that merit its power and popularity, including:

- **Performance**—REST API performance is bounded primarily by the actual interaction between system components, as opposed to extraneous factors like network latency
- **Scalability**—REST was designed specifically to accommodate large numbers of network components and interactions among them
- **Flexibility**—REST components are easy to modify as new needs arise
- **Visibility**—Communication among components is designed to be visible to service agents
- **Portability**—REST allows the movement of program code with data, making for exceptional portability
- **Reliability**—RESTful implementations are normally immune to failure at the system level even when problems are encountered at the individual component level

RESTful systems typically communicate using the ubiquitous HTTP protocol, including standard HTTP verbs like GET, POST, PUT, and DELETE, which are stateless, cacheable, and layered. The APIs operate on available application Resources, also known as URIs (Uniform Resource Identifiers), as shown in Figure 1.

**API-based Workflows within a Visibility Fabric**

The general REST workflow can be simply described as follows:

- A client application makes a request to the server to create, read, update, or delete a resource.
- The server responds to the request. The response typically contains a status code, indicating whether the request succeeded or failed. The response can also contain data in a structured format in addition to the status code. The application takes further actions based on the status code or processes the data.

In a Visibility Fabric, the “REST server” functionality is integrated into a centralized policy controller.

![Figure 2: Basic REST workflow](image)

When RESTful APIs are integrated into a Visibility Fabric, the most prominent of the newly enabled workflows include: managing nodes (i.e. devices), configuring ports, and configuring traffic maps. As such, these RESTful APIs provide an open interface to interact with the Visibility Fabric in a programmatic way.

**Managing nodes.** Functionality typical of node-management workflows includes adding or removing devices to or from the Visibility Fabric, and discovering and obtaining information about attached nodes or devices.

**Configuring ports.** After a node (device) has been added to the Visibility Fabric, the next step is to configure ports (i.e. interfaces) on the device. At a minimum, the application can use the APIs to create, show, update, and delete ports. Depending on the specifics of the Visibility Fabric, the application may also make it possible to group ports into logical bundles, allowing traffic to be distributed among multiple monitoring/management tools.

**Configuring traffic maps.** The ability to create traffic maps is normally a key capability supported by a Visibility Fabric. A traffic map is the instantiation of a policy within the Visibility Fabric that works by extracting a flow of interest from its source and reliably delivering it to the desired destination tool. When correctly implemented and integrated, the set of REST APIs should enable a client application to create, show, and update those maps in real-time, on an as-needed basis.

© 2015 Gigamon. All rights reserved.
Using RESTful APIs within a Visibility Fabric

These workflows in turn suggest three principal classes of use cases for handling nodes (devices), ports, and traffic maps. As noted above, specific use cases fall into three categories:

- Adding or removing a node, or getting information about it
- Getting information about a port, modifying its type, and enabling it
- Creating or modifying traffic maps

It is easy to imagine how capabilities of this type are useful for automating a large number of security and IT operations-management functions. For instance, if a security device that detects a pattern of traffic that deviates from the norm can configure the Visibility Fabric to perform a subsequent security operation like inspect or block.

Consider SSL traffic, which is typically expected on a standard TCP port number (443). If a large number of SSL interactions happen on a non-standard port, the application receiving the SSL traffic from the Visibility Fabric can configure the fabric to decrypt the SSL traffic before sending it to the recipient. This will enable the receiving security application to further analyze and inspect the now-decrypted traffic.

A second example is monitoring of a remote site. Typically, remote sites (such as branch office sites, huts containing sensors, etc.) are connected to headquarters/central locations via expensive WAN links. They also tend to be short on resources like monitoring and security tools as well as personnel. An efficient way to monitor such remote sites is to generate unsampled flow records from a visibility node that is physically located at a remote site and is yet part of the Visibility Fabric. When anomalous activities are detected, the remote node can be programmed to trigger and send full traffic flows to obtain further information for analysis.

SDV in the Software-Defined Networking (SDN) Context

Software-Defined Visibility is to a visibility infrastructure what Software-Defined Networking is to a network infrastructure. SDV combines the pervasive reach of visibility with an automation framework.

In an SDN infrastructure, network switches and routers form the physical network or the Layer 2-3 data plane. Virtual networks are abstracted from the underlying data plane using encapsulations such as VXLAN, MPLS, NVGRE etc. to allow multi-tenancy on a common infrastructure. The SDN controller supports the control and management planes that in turn provide control of the virtual and physical networks.

An SDN infrastructure is architecturally layered in a way that is similar to an SDN, but is optimized to provide intelligent visibility. Visibility Fabric nodes form the distributed elements of the Visibility Fabric and can be implemented as a physical (hardware) node, virtual (software) node, or visibility software running on a third-party network switch. These fabric nodes provide fabric services (such as Flow Mapping®, clustering, and many others), traffic intelligence (such as SSL decryption, application filtering, centralized flow record generation with rich metadata and many others), thereby reducing the amount of unwanted traffic sent to those security/operational tools connected to the Visibility Fabric. A centralized policy controller provides centralized management of the entire fabric including physical, software, and virtualized nodes. Third-party systems can get information from the fabric or modify behavior of the fabric through the integrated APIs.

Figure 3: The relationship between SDN and SDV
Principal SDV Use Cases

Adding RESTful APIs to a Visibility Fabric transforms it profoundly, from an essentially static layer for network monitoring, to an active, dynamic, and agile visibility layer. It is capable of responding in real-time to security threats, changes in node or device status, and changes in traffic patterns, including the emergence of new blind spots. RESTful APIs make the fabric programmable, and support client applications running on the full range of nodes, devices, appliances, and other monitoring/management tools.

In the use cases discussed below, the RESTful APIs interact directly with the Visibility Fabric via the policy controller, as depicted in the figures that accompany each example.

Use Case 1: Threat pattern detection and response

This use case provides an example of how threat pattern detection by a security appliance, combined with the ability to respond programmatically in real time, transforms devices and processes that are otherwise reactive to truly proactive.

In this use case, critical operations performed by the integrated APIs might include, for example:

1. Creating Flow Maps to evaluate traffic pattern irregularities or other suspicious behaviors
2. Decryption and sniffing of SSL-encrypted traffic seen on unexpected servers or at unusual volumes
3. On-demand capture of packets based on selective filtering criteria and other techniques

Use Case 2: Automatic adjustment of “monitor mode” for in-line security tools

This case provides an example of how an API integrated into the Visibility Fabric can save system administrators a lot of time and effort. Configuration of a security device often has to be fine-tuned before it can be placed inline with the network. Once a predetermined set of conditions is met, an administrator changes the security device placement from out-of-band operation to inline with network traffic flows. Rather than doing this manually, an administrator with the right access privileges to Visibility Fabric policy management can schedule the change programmatically using the Visibility Fabric APIs. Conversely, the inline tool may have to be taken off-line for maintenance purposes, a task that can also be automated via a similar process using the Visibility Fabric APIs.

Discrete operations executed by means of the APIs for this case include:

1. Changing the monitor mode from out-of-band to inline within the fabric
2. Updating the Flow Map in the Visibility Fabric accordingly
3. Dropping traffic based on a set of defined criteria
Use Case 3: Private cloud and virtual datacenter monitoring
This use case is yet another example of how automating processes through an active and agile Visibility Fabric can save time and effort. It further illustrates the importance of extending the fabric to the virtualized environment. In this case, the APIs will be executed by clients running on both application performance and network management monitoring devices or appliances, as well as on equivalent equipment performing security functions.

Discrete operations executed by APIs for this case might include, for example:

1. Deploying the virtual elements of the Visibility Fabric itself, i.e., in this case the software that will monitor all cloud-based virtual network and machine activities
2. Creating and deploying traffic and security policies
Use Case 4: Use of the automated Visibility Fabric for IT Ops management

This final case illustrates how the capabilities enabled in the Visibility Fabric by the APIs are used to transform critical, but previously mostly manual, IT operations management tasks into automated actions. As seen in Figure 7, we note two discrete sub-cases, namely, performing an inventory of every device on the network and responding to service request tickets with appropriate provisioning.

Some discrete operations executed by the APIs in the performance of these might include, for example:

- Getting device/node information for all network-attached items
- Collecting statistics from these devices as needed
- Configuring ports as required by provisioning requests
- Creating new traffic maps that incorporate the newly provisioned network elements

Authentication and Role-based Access Control (RBAC)

While automation enabled by APIs provides for an efficient means to manage and monitor systems, it is critical to make sure that only authenticated and authorized users are allowed to operate on the permissible resources. The Visibility Fabric provides for robust RBAC and authentication services via local database or commonly used protocols like RADIUS, TACACS+, and LDAP. Administrative roles can further be defined to limit administrative oversight to specific interfaces. SDV enabled RESTful APIs also adhere to the same set of RBAC oversight criteria.

Gigamon’s Unified Visibility Fabric: Harnessing the SDV Paradigm

In Gigamon’s portfolio, GigaVUE-FM (the Fabric Manager) performs the function of a policy controller for the Unified Visibility Fabric. The Fabric Manager incorporates a robust, general, and complete set of RESTful APIs. These APIs augment and complement the Unified Visibility Fabric in a way that creates a new dimension of capability, making it active, agile, and capable of dynamic response to changing conditions automatically, and in real time. In short, it represents a true paradigm shift, and introduces the new reality of Software-Defined Visibility.

Figure 7: Ways the APIs can help simplify IT operations management
The APIs are integrated into GigaVUE-FM, and thus work seamlessly with the other principal components of the fabric. These include GigaVUE-OS™, the underlying operating system software that powers the Visibility Fabric nodes and provides fabric services such as Flow Mapping, clustering, and inline bypass, and GigaSMART®, an industry-leading set of traffic intelligence applications that are implemented on a common high-performance compute platform that are built-in to the Visibility Fabric nodes. Examples of traffic intelligence applications include packet de-duplication, SSL decryption, packet slicing, packet masking, header stripping, tunneling, NetFlow generation, and Adaptive Packet Filtering functionality.

The RESTful APIs by which SDV is implemented provide an open interface for development partners and customers to use to convert their existing visibility apparatus into a fully automated system capable of responding immediately to network conditions and other infrastructure changes.

Gigamon Visibility App for Splunk, using these open RESTful APIs, extends the health and analytics of the Visibility Fabric for the IT Operations Management (ITOM) user. This application augments intelligence collected from the production network to help the SecOps and NetOps teams to trigger first-level troubleshooting within the ITOM realm.

To help enable user community adoption of the RESTful APIs and SDV, the Gigamon Customer Portal acts as a central hub for sample cookbooks and scripts for customers to consume and exchange ideas and use cases.

Example Output of RESTful APIs

If an IT operator wants to view the Visibility Fabric inventory, below is a sample request/response using APIs with JSON (JavaScript Object Notation) output:

Request: GET https://<GigaVUE-FM IP address>/api/v1/nodes/flat

Response:

```
{  
  "clusters" : [  
    {  
      "family" : "H",  
      "clusterId" : "10.115.152.50",  
      "members" : [  
        {  
          "deviceId" : "10.115.152.50",  
          "deviceIp" : "10.115.152.50",  
          "dnsName" : "hc2-c04-29.gigamon.com",  
          "hostname" : "HC2-C04-29",  
          ...  
          "clusterMode" : "Standalone",  
          "clusterId" : "10.115.152.50",  
          ...  
        }  
      ]  
    }  
  ]
}
```

Figure 8: Gigamon’s Unified Visibility Fabric
Conclusion
In this paper, we answered the question of how to best enhance a Visibility Fabric so that it is capable of enabling any machine-to-any machine communications. We have shown how this approach is transformative allowing for triggers from monitoring, management, and security appliances to change programmatically the behavior of the fabric in response to events or conditions as they occur. The new paradigm of Software-Defined Visibility (SDV), implemented with RESTful APIs, was proffered as an optimal approach to satisfy this requirement.

About Gigamon
Gigamon provides Active Visibility into physical and virtual network traffic, enabling stronger security and superior performance. Gigamon's Visibility Fabric™ and GigaSECURE®, the industry’s first Security Delivery Platform, deliver advanced intelligence so that security, network and application performance management solutions in enterprise, government and service provider networks operate more efficiently and effectively. For more information visit www.gigamon.com