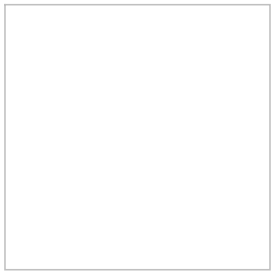
The API Is the New CLI

Lori MacVittie, 2009-04-11



Infrastructure 2.0, from a purely developmental standpoint, is about APIs. It's about offering up the functionality and capabilities of a wide variety of infrastructure – network, storage, and application network – to be externally controlled, integrated, and leveraged for whatever purpose a developer might dream up. It enables providers and enterprises alike to turn infrastructure functionality into services. Need compression? Caching? Routing? Load balancing? Via service-enabled management APIs these can become services, provisioned and released through the invocation of a service. When expanded to include the sharing of actionable data – performance statistics, status, availability of application services (context!) – this integration becomes the mechanism through which a dynamic infrastructure is created. One that reacts to events and conditions in the network, storage, application network, and application infrastructure in real-time.

But all of this functionality, the automation of functions, the codification of processes (orchestration) requires integration. Where previous generations of administrators evaluated the manageability of network and application network devices based on the CLI (command line interface) the next generation of administrators and the developers who will support integration efforts, will almost certainly look to APIs as a means to determine suitability of solutions within their architecture.

APIs are the new CLI

In the past network administrators would compare the CLI syntax and functionality of network and application network devices to Cisco's IOS. IOS became the de facto standard for command line interfaces and even today you'll find reviewers and discussions that mention how "IOS-like" any given CLI might be. But as infrastructure 2.0 and the need for dynamic infrastructures continues to drive administrators and developers toward APIs for integration and automation the CLI will wane in importance and the API will rise to take its place. That's because the APIs provided by network and application network devices will be the primary interface through which the device is configured, controlled, and managed.

Luckily network and application network vendors learned from the trials and travails of enterprise software and the first implementations of these APIs have been primarily standards (web-services, XML) based. Service-enabled APIs means both administrators and developers can take advantage of the functionality and do so in whatever language or environment they are most comfortable. This flexibility is key to adapting to the myriad possible environments and architectures in which such devices may be deployed.

The **danger** in this shift toward APIs is that it is infinitely more difficult to replace systems that are integrated via an API or library – any programmatic-based integration, really – than it is to replace those for which the CLI is the primary administrative route. Administrators comfortable with the APIs of a Cisco router or switch will be less inclined, for example, to replace those core networking devices with a Juniper or other networking solution because of the inherent difficulty and time involved in learning – and using – a new API. This is true across the infrastructure spectrum; the APIs that allow complete control and management over BIG-IP (iControl) are very different from those available for Citrix Netscaler, or Cisco ACE or any of the other API-enabled application delivery platforms.

It is quite possible that whomever can win the "API wars" for Infrastructure 2.0 will become the new de facto standard for that particular "tier" (for lack of a better term) in the infrastructure architecture. Eventually one API will be preferred over the other – either due to saturation and usage or specific demand and it will give the vendor an edge that will not easily be dulled.

EAI and Adapters

But it's not just network-facing IT that will help set the direction of APIs. Because part of the premise of infrastructure 2.0 and the APIs that are part of parcel of the standards and devices within its domain is *integration* there is a developer-focused component to the success of infrastructure APIs. While administrators are most likely to be closest to the APIs of network and application network devices, developers are most likely closest to the applications that drive orchestration and integration with business-focused systems.

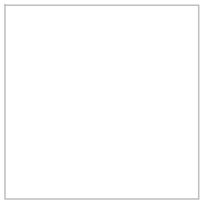
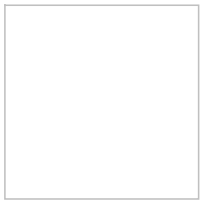
One of the ways software application vendors knew they'd "made it" was the inclusion of adapters in EAI (enterprise application integration) systems for their product. ODBC drivers for databases, message queuing adapters for MQ and JMS, and more recently "salesforce.com" and other SaaS offerings. The inclusion of adapters for specific solutions in EAI and development environments is tantamount to declaring that solution a "win" for the enterprise. Thus it will be important to vendors of network and application networking solutions to court management and orchestration system vendors to include at distribution an "adapter" or samples, at a minimum, as the means to integrate and include their particular solution.

This seems counterintuitive, as most APIs are service-enabled and thus the bulk of the integration work is implicit in the API. But the ease with which those APIs are used and integrated by developers is paramount to successful adoption of infrastructure 2.0 APIs. The inclusion as an "adapter" provides the ease of use, often via a GUI, necessary to garner use and support from developers and business-focused orchestration analysts because of the inherent differences in the *data plane*. Mapping of objects from one device to another, from one system to another, is required and it is this core requirement that is fulfilled by middleware systems such as EAI and ESB (enterprise service bus) implementations. The easy integration with these middle-tier applications will be increasingly important as we move from operational policies based purely on technical metrics toward data centers driven by both technical *and* business metrics.

APIs are the new basis for standards

The first generation of the Internet used protocols and structural definitions to engender interoperability and even portability. Infrastructure 2.0 heralds the coming of a second generation of the Internet just as Web 2.0 signaled the beginning of the second generation of the Web. This next generation of infrastructure interoperability and portability will certainly be driven by protocols, but those protocols will include APIs and encompass a broader set of functions at higher layers of the network stack. Many of the ongoing efforts in the standards arena today are based not on structural definitions of data but on the APIs that will enable integration across the infrastructure and the Internet, a la "InterCloud."

Both are necessary components to ensuring interoperability and portability, but until we see standardization of meta-data and component definitions, the emphasis will continue to be on the APIs.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com