efficient iP® *for*

Automation Through IT Abstraction Layer

The complexity of infrastructures is increasing with all the IT initiatives being run by organizations today, from digital transformation to data management and artificial intelligence. With technologies ever more complex to manipulate and integrate, an opportunity resides in using coding as a base principle for infrastructure management, normal operations, and incident management. But the multitude of products and solutions in the infrastructure that need to be more integrated into the rest of the IT ecosystem, requires API links in between. Using a middleware kind of approach, via an automation abstraction layer, is one way to simplify operations and interoperability. Within this layer, the role of DDI is key as it brings: rich IP data, automation of core network services, expandable ontology, and event-driven architecture.

Solution Benefits

SIMPLIFY INTEGRATION BETWEEN IT TOOLS	exposed service from the abstraction layer is easy and simple to use for infrastructure code
INCREASE DEVELOPMENT CYCLE VELOCITY	IPAM data is simple to use for all IT components through high-level services increasing code adoption and agility
IMPROVE QUALITY OF CODE & SERVICES	limit errors, bugs, and comprehension issues due to fewer models to handle, less code to manipulate, fewer APIs to learn
RAISE OPERATIONAL EFFICIENCY	with code easier to write and use, automate more operations, easily troubleshoot, and free up time to spend on complex tasks or other integrations

Business Challenges

Creating and maintaining software as part of an Infrastructure as Code initiative (IaC) is very complex. I&O teams are automating their operational tasks and exposing various APIs to the other IT teams and tools, but with infrastructure solutions always evolving this requires permanent adaptation and refactoring. It becomes even more challenging when there are a lot of components in the portfolio, from networking to storage, compute, or security, to list just a few.

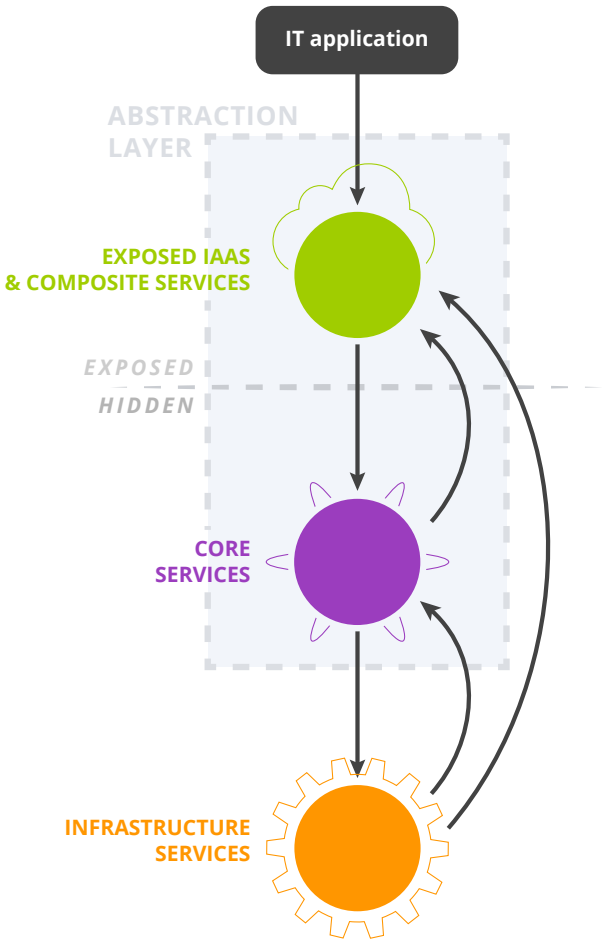
In order to ease Infrastructure as Code progression, using a coherent and standardized dataset and an evolved solution architecture are fundamental pillars

Abstraction Layer Overview

It is fairly evident that development and scripting in infrastructure as code are simplified by the usage of a DDI solution. The IPAM repository stores all the IP and network-related information, from the address plan to VLAN, VRF, and devices and their logical links and can be even more powerful with the use of metadata attached to these objects. In addition, DDI automation brings simplicity to the provisioning of the core network services that are DNS and DHCP.

To move a step forward in the IaC journey, it is necessary to architect and use proven methods from other software and application industries. A good approach to bring into the IaC world is the abstraction layer principle. By decoupling how things are done technically at a low level (near the infrastructure solutions in place) and the services exposed to upper layer tools, we automatically gain flexibility and obtain a nice way to change one half of the connection whenever adjustments are required.

The global architecture of the Infrastructure Service layer can follow this general topology:



The infrastructure macro services are exposed at the top (in green) to other IT applications requiring infrastructure (e.g. pipeline of continuous integration or testing). They are offered via very simple, rich, and high-level APIs. These APIs should stay as close as possible to functional requirements and be totally dissociated from the equipment or technical solution in place underneath. For example, when we talk about a network, we don't need to know that this is composed of VLAN, VRF, and IP subnets and would be running on a Cisco or Juniper architecture, we simply want a connectivity facility between some IT resources.

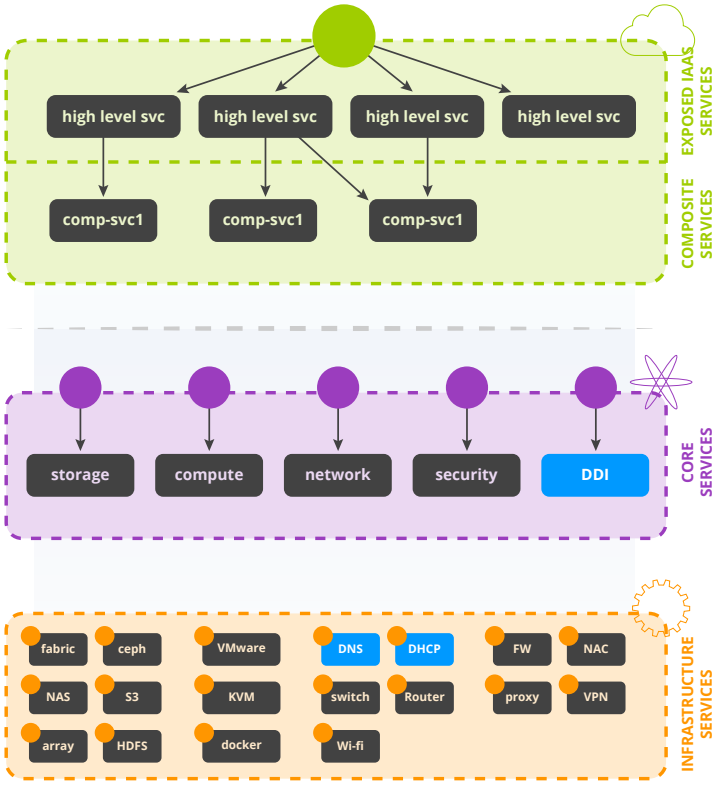
It may be necessary at this exposition layer to have 2 levels of API, the exposed one and some composite underneath that can mix multiple solutions and perform orchestration of multiple API calls.

Then we have the core services (in purple). These are arranged by infrastructure solutions, like storage, compute, network, or DDI. Each domain is able to expose some APIs that are directly related to the service they propose but the technical solution itself should be hidden. Here we're talking about VLAN, VRF, and IP subnets (rather than networks) but there's no need to expose whether they run on Cisco or Juniper devices.

The last layer (in orange) is directly connected to the solutions, the devices, and the equipment (or service providers and clouds). Here we need to know exactly how to create a VLAN on the network, how to create a disk volume on the storage, and a virtual server with regards to the solution chosen. In this case, we talk directly to the Cisco and Juniper components. All the complexity and variety of solutions to drive API should ideally remain at this layer and not be exposed to the upper layers of the architecture.

With such an approach, which we call the automation abstraction layer, we can work with a high level of independence. If we need to expose a new high-level functionality, it doesn't need to directly use a vendor-exposed API. It may be synchronous or not, and can use direct calls or flows depending on the job to perform. At the other end, if a new vendor needs to be integrated with the solution portfolio, via the proposed API and some mapping towards the upper level exposed services, migration would be eased and multiple usages made possible. Furthermore, each layer can have dedicated development cycles, as well as dedicated teams with various levels of understanding of the bits and bytes that run the solutions.

The complete picture could potentially look like this:



Main Features of SOLIDserver DDI

DDI automation of core network services: as explained in our solution overview dedicated to Infrastructure as Code, the role of the DDI is fundamental and is considered here in the architecture as a core service similar to computing or networking. The DNS and DHCP core services are more related to the infrastructure and this is where we can see the interest in using a fully automated DDI solution where the IPAM is tightly linked to service data in order to ease the configuration. The abstraction layer can propose direct access through core services of the whole service features, but can also for the DNS service offer directly high-level APIs, for example, to manage a single record or an alias (CNAME).

DDI and service overlay: it can be very helpful when DNS and DHCP are integrated into an overlay layer (managing other vendor solutions from the central DDI solution), this is especially useful when services are run in the cloud. SOLIDserver is able to manage multiple services from different vendors or cloud providers inside a SmartArchitecture that can be considered as a single and unique configuration entity. The abstraction layer can take advantage of this automation to reduce the complexity of internal code as well as to propose a rich API on the northbound endpoints.

IPAM as the repository of network information: any piece of network-related information can be stored in the IPAM, either because it is already managed by the DDI solution or as a repository (e.g. devices or applications). The abstraction layer can expose some functions to manipulate objects as well as to gather information like the IP address plan. With all the metadata that can be associated with any object in the IPAM, it becomes possible to expose services which are business-oriented, for example when grouping or filtering is needed by an entity, a region, or a business unit.

IPAM activities converted as events: with the event forwarding feature, the SOLIDserver can immediately inform the abstraction layer of any activity on the IPAM such as the addition of a subnet, or the destruction of a DNS record. Through this simple asynchronous mechanism, the abstraction layer can perform business tasks and inform other IT components about valuable information. An excellent opportunity is, therefore, provided for offering a publish/subscribe event bus in the architecture of the abstraction layer.

SOLIDserver proposes a rich set of APIs allowing any data manipulation and the orchestration of internal DDI automation. Hence, EfficientIP has an important role to play in the Infrastructure as Code initiative, particularly for those architectures which permit a very flexible abstraction layer.

Key Takeaways

The cloud has pushed the model of the abstraction layer up to a very high level. Infrastructure as Code can directly follow this architecture pattern to allow it to be easily maintainable by I&O teams, easy to refactor, and prevent lockage to any solution or vendor due to the complex use of its own API or configuration methods.

DDI already enables manipulation of abstracted objects and should probably be one of the first tiles to implement in an abstraction layer principle, providing northbound services with some very simple objects (IP addresses, subnets, DNS names), hiding the complexity of overlay management, and offering technical coherence checking via DDI automation.

efficient iP®

REV: C-210225

Americas
EfficientIP Inc.
1 South Church Street
West Chester, PA 19382-USA
+1 888-228-4655

Europe
EfficientIP SAS
90 Boulevard National
92250 La Garenne Colombes-FRANCE
+33 1 75 84 88 98

Asia
EfficientIP PTE Ltd
60 Paya Lebar Road #11-47
Paya Lebar Square SINGAPORE 409051
+65 6678 7752