Infrastructure Rules

Naming For Automation

Edmund J. Sutcliffe

File name: rules-1.2.doc

Table of Contents

1. Introduction

At the heart of all human learning is the necessity for patterns and the rules which drive them. Information Technology functions best when these rules are clearly communicated and the arguments for their existence clearly understood.

Embedding these rules into the day to day operation of the IT Landscape whether by automation or human processes assists in the management of risk and the measurement of what has been delivered.

One of science's greatest endeavours has been taxonomy and with information technology this is less often done well. The processes used in the delivery of the IT landscape should have taxonomy embedded into them, so simplifying its administration.

What follows is the development of a taxonomy for the provisioning for an IT Landscape. This Taxonomy is tested with example of how these capabilities and instances should be used.

1.1. What should these rules archive ?

For all services instantiated against these rules an consistent and reliable method for the management of the relationships between the physical and virtual components that provide the landscapes for the environments from which the development, deployment and support of these services occur.

For all these components, a consistent method for the collation of measurement of the environment, making best use of the rules and associated name-spaces

For all components the ability to recreate its configuration, software and user data bit for bit identically using the taxonomy laid out here and their associate data recovery methods

The resultant solution must be such that it can be mechanically compared with the range of Configuration Management resources and activity Monitoring platforms such that both the current access service name-spaces and the current monitoring of the delivery services can be converged towards a model defined target state in a known set of activities

2. Architectural Principles and Rules

When beginning any project one should lay out a set of principles for the project and a series of rules against which the delivery of the project should be delivered.

These principles and rules are placed here in an attempt to keep those using and updating this document honest in their delivery and consistent in their rulings.

2.1. Abstraction

Abstraction is the rendering of the general case from which an instance occurs. It is the process of removing detail to expose the essential features of a particular concept or object.

2.1.1. Separation of State

By careful installation of Applications, it is possible to abstract the application from the host Operating System, and so it becomes relatively easy to move them from host to host, so meeting the requirements of modularity. This also improves the uniformity with which an application can be installed and the efficiency under which it is managed.

These techniques also could allow the separation of historical transaction state and the configuration from the environment, so simplifying the problems of providing resilience in the environment and making most efficient usage of the backup windows and storage. You would end up with Application storage, which can be used by all applications making use of the same application, and capability storage which is unique to the capability

2.1.2. Name-spaces

Following the principles of simplicity and uniformity, it is crucial that we have good naming schemes. Additionally it is crucial that everything within the environment makes use of them in the appropriate way in the appropriate sub-domains.

2.1.3. Configuration

Resources within the environment be capable of centralised configuration, and make use of the Resource Naming to ensure that connection between the resources meets the principles of uniformity and resiliency.

For example, making use of DHCP it is possible to configure not just the IP addresses of a host, it is possible to configure access to the DNS, NTP and web proxy services. In turn having gained access to DNS individual devices, making use of the Resource Naming conventions to discover the IP address of the database server it should connect to.

Following this idea, it may also be possible making use of tools such as Microsoft’s Dfs or Unix Automounters to centrally control where file systems are made available and how they are mounted, so meeting our migration and availability requirements in a very agile manner.

This idea can be continued into the start-up procedures used by packages so that instead of having a static configuration they are dynamically constructed and then executed.

2.2. Principles

A principle is a guiding instinct; how things are to be approached. These are soft touch things but crucial to guiding the designs themselves

2.2.1. Simplicity

Design guidelines and conventions must be simple

Design, conventions and guidelines must be simple to understand and implement. Applying those guidelines must also be very simple. So simple, that even non-administrators have a clear picture of what is going on. So simple, that even non-administrators can execute the steps required to perform a certain operation with success

2.2.2. Uniformity

Common design for enterprise applications and infrastructure

Having a uniform environment for both end-users and system administrators will alleviate their work. If the same concepts apply in all circumstances, this will greatly enhance the overall quality and understanding of the environment, and concepts once learned can be applied everywhere

2.2.3. Agility

Ability to respond quickly to demands and opportunities

The ability to respond quickly to demands or opportunities within the environment, in a measured manner, without extending risk. Agility can be maintained by maintaining and adapting goods and services to meet customer demands, adjusting to the changes in a environment while maximising the advantage of appropriate automation and human resources.

2.2.4. Efficiency

Design must be efficient to use and manage

Besides being simple and uniform, the system must also be efficient to use and apply. The overall performance and manageability of the system is not allowed to suffer too much from the first two objectives. This means that in some cases exceptions have to be applied in order to ensure better performance or manageability

2.2.5. Proven

Design productive, adopted and successful elsewhere

The design should be established beyond doubt and have a significant amount of testing associated to confirm that it can be recreated and its behaviour understood in all conditions.

At the heart of this are clear lines of responsibility and interfaces which are understood and documented.

2.2.6. Supportability

Capable of being maintained and supported

The environment should be capable of being kept in an appropriate condition, operation, or force; kept unimpaired. In turn much of this will be with the acceptance of maintenance by third party or other operational agreements.

Driving this is the need for 'vanilla' deployments as close to that used by the vendor in their training material and as little modified as possible so allowing new members of the support team to become useful with the minimum amount of guidance. This also assists in any outsourcing or shared support agreement.

2.2.7. Resiliency

Recovery from effects of adversity

Resiliency is the ability to avoid, minimize, withstand, and recover from the effects of adversity, whether natural or man-made, under all circumstances of use. Resiliency applied to the infrastructure is trustworthiness under stress and spans high availability, continuous operations, and disaster recovery. Almost always Resiliency is implemented through redundancy, though this should not be assumed the only solution.

When discussing Resiliency we use two measures

- Recovery Point Objective (RPO), the point in time to which the systems and associated data must be restored to.
- Recovery Time Objective (RTO), the period of time in which environment must be restored to the RPO.

The Recovery Point Objective (RPO) is length of time we can be without data that is acceptable. If the RPO of a company is 2 hours and the time it takes to get the system back into production (RTO) is 5 hours, the RPO is still 2 hours.

It is additionally crucial to be able to separate the RPO objectives brought about by logical errors, compared to those brought about by failure. It is much easier to deal with failure within a well designed environment. However, logical errors, take periods to be discovered and then a strategy for resolution is required

2.3. Rules

It has been said that “Rules are not for breaking”. However, these rules are driven by the aforementioned principles and these may cause them to be bent.

The main thought behind the rules is to give a framework against which measurement of the quality of delivery can be done. For example, by encouraging modularity, and reproducibility we should deliver more manageable and scalable solutions probably at lesser cost.

2.3.1. Reproducible

Accurately replicate what has previously been performed

2.3.2. Modular

Standardised building blocks for easy assembly and repair

2.3.3. Manageable

Centralised change and control

2.3.4. Scalable

Ability to meet changing business demand

2.3.5. Available

Platform to support stable and predictable operations

2.3.6. Auditable

The ability to enquire about what has been provisioned

2.3.7. Secure

Protection against attack, corruption or loss

3. Domains

3.1. Security Domains

A Security Domain is the aggregation of resource for the classification that enclave within an IT Landscape. Security Domains are defined by a boundary which delimits the interaction of the classes of data and their traffic.

3.2. Environments

An Environment is an aggregation of resources for the provision of services to the business.

Separate environments exist to manage the risk and the underlying service quality for the business.

To this end we typically have a “production” environment where change control and so risk is tightly managed. This is augmented by a series of one or more “production like” environments where change control is less tightly controlled and is driven by the business needs, such as requirements for testing new services and their integration into existing environments. In turn these “production like” environments are supported by one or more “technical regression” environments in which technology and the associated processes can be developed.

Each of these environments requires a Security Domain separating them and this should be reflected in naming and authorisation structures.

3.3. DNS Domains and Kerberos Realms

A DNS domain name is an identification label that defines a realm of administrative autonomy or control and so reflects easily into the ideas of security domain.

The first and most important group of sub-domains are associated with environments.

The environment sub-domains are typically required for “production”, “production like” and “technical regression”, the majority of hosts and their associated services live in these domains.

Location based sub-domains identify devices and elements which resided within a physical location. Only in exceptional circumstances should system information be recorded in these sub-domains.

Typically location based sub-domains map directly to buildings and locations, but occasionally they are identifying logical concepts such as “virtual data centre” or location spread over multiple buildings.

The final sub-domain is the core-infrastructure sub-domain. This is typically where the core routers and network equipment reside as they provide services across many environments.

3.4. Functional Groupings

Functional groupings are defined by the data and the interaction with it.

Typically these are implemented as VLANs and have ACLs associated with them based around their layer 3 network numbers.

These groupings usually break into those within a data centre and those presented to clients.

Within the group associated with the data-centre there a further groupings associated with the storage and manipulation of data, presentation services, administrative access to servers and boot strap networking.

Within the group associated with the clients, one finds groups associated with the wireless access, un-trusted guest users, “secured data” users.

There are other groups which frequently exist beyond the data centre and client groups. There may be a media grouping, associated with telephony and video and the associated signalling. Additionally there is a grouping associated with core networking, network services (such as load balancing, NAT and firewalls) and finally external presentation (DMZs)

3.5. Layer 2 Domains

Layer 2 Domains are generally Ethernet VLANs though other Layer 2s are possible. Layer 2 Domain are the children of Functional Domains. They are needed for the implementation of networking. Interfaces attached to devices and elements are associated with them.

3.6. IP Address Usage

The breaking up of IP addresses into appropriate network and host portions is critical for the management of the network and its associated growth.

Careful allocation of these addresses and their reflection of their network portions into such things as VLAN ID assist in problem diagnosis.

Additionally putting the same functional groups within the network mask positions simplifies the creation of ACL rules and their number so easing management and reducing risk.

4. Business Services

Services are what Information Technology has to offer. These should map to processes the Business are aware of and have a need for.

A service may be a single system, built on an application, packaged to provide the results required for the business, or be more complex containing many systems.

Services must have an owner identity. This identity is often associated with a role in the business, e.g. Customer Manager, Finance Director. Services must have a group identity. This is used for associating the package identities with the service and restricting access to them.

Service must belong to an environment.

Services and all their component systems and packages have their DNS information stored within the appropriate environment DNS sub-domains.

Services and all their component systems and packages must have unique fully qualified domain name entries but may be repeated between sub-domains. These names may be re-used.

4.1. Software Applications

Applications are a set of binaries which are provided by a software vendor or developer, that are then *specialised* into a Package.

Applications must have a user identity associated with them. This user is required for the management of these binaries.

Applications should have a group identity associated with them. This group is to allow access to these binaries by packages which specialise them.

4.2. Packages

Packages are the unit of work within the IT landscape and as such require an identity to perform them.

Packages must have a user identity associated with to perform the work. This user is required for process separation and accounting purposes.

Packages must belong to the group associated with the application they have specialised, so enabling auditing of the occurrences of that application and allowing administrative access from that application to the package.

Packages must belong to the group associated with the service so enabling accounting for the occurrences of the packages and restriction of access to only that service if necessary

When access to these packages is made over the network a DNS CNAME shall exist which matches the package user name and point to the host name or VIP name on which the package is executing.

It is possible for packages to require multiple instances for the sake of performance. In these cases the same user identity should be used for the execution of the package however the associated DNS

CNAME should match the user identity followed by a hyphen and index sequence. These DNS CNAMES should point to the host or VIP on which the package is executing.

Packages must have a functional domain

Packages should have disk quotas associated with the user name associated with the package.

4.3. Systems

Systems are the aggregation of packages to perform a useful task as seen from the service perspective.

Systems must have a parent, the service to which it belongs or another system. This parent relationship must be recorded within repository. This parent child relationship allows the recording of dependencies for fault monitoring, also it acts as access control between systems and their component packages.

5. Identity

One of things we need to write so much more

5.1. Forests

A for

6. Hardware

At the heart of infrastructure are the physical devices from which the services are delivered.

Traditionally there has been a tight connection between the device on which the operating system has been instantiated and the device. In a virtualised environment this relationship is no longer with the device but with the Service which offers the management console.

It is usual to have some sort of database which records these relationships and access to this data is required as part of the day to day administration of the IT landscape.

DNS is a scalable read-mostly database which is already embedded within the OS instantiation and administration and its use minimises new infrastructure required in the landscape.

When making use of DNS it is crucial that simple examination of the form of the name used, conveys the maximum amount of information about type of component and its use.

6.1. Elements

An element is something which cannot function without the aid/power of a device.

An element must have a parent, the device to which it belongs or another element. This parent relationship must be recorded within repository. This parent child relationship is necessary for purposes of hardware maintenance.

An element should have a unique identifier which is never re-used, for example an asset tag. Where this identifier is presented to the DNS it is usually prefaced by a single character **e**.

Elements may be in DNS, though many may not be and must be recorded in another repository. For example, a SATA disk is an element and it must have a parent object, for example another element, the blade on which it resides. The SATA disk does not require a DNS entry but the blade may require such.

When an element requires a DNS entry it must be a DNS A record, it should point to the out of band access to this element, where this is not possible it should point to the out of band access for the parent device.

Elements have physical locations. Their DNS entries must live within the location or core infrastructure sub-domains.

6.2. Device

A device can perform work without any support and is capable of providing support for elements.

A device must have a unique identifier which is never re-used, for example an asset tag. When this identifier is presented to the DNS it is usually prefaced by a single character **t**.

Devices should have a DNS A record pointing to the out of band access for this device.

Devices have physical locations. Their DNS entries must live within the location or core infrastructure sub-domains

Examples of devices include blade chassis, standalone network devices and servers. It should be made clear that a blade is not a device, but an element as it cannot perform work without the support of the device chassis.

6.3. Hosts and Clusters

A host is an instance of an operating system.

A cluster is an aggregation of hosts grouped for the purposes of improving the availability or scalability of the execution of the packages presented to the operating system instances.

Examples of hosts would be a device executing RedHat 5.4 or a VMWare Guest executing Windows 2008 Enterprise Edition.

All hosts require a unique identifier. This unique identifier must be present within DNS as an A record presenting the primary interface as seen by the operating system. e.g.”eth0” on linux or “Local Area Connection”   on Windows hosts.

This unique identifier should be associated with the OS type and not the IT services offered from the OS instances, so allowing these IT services to be repositioned as required onto different hosts as required.

The host identifier may be re-used.

Some OS vendors require modified OS binaries for the host to partake of cluster membership. This modification in the binaries should be reflected within the naming.

There are often operational boundaries associated with the administration of different OS types and operations are performed by OS type.

Hosts should be associated with the environments sub-domains in which they are executing. However, hosts use for core- have physical locations, and so be presented in location sub-domains; such hosts are in the minority.

6.4. Virtual IPs (VIPs)

A Virtual IP or VIP is an additional logical IP address presented by a host or cluster.

All VIPs have a unique identifier which is associated with the host or cluster with which the VIP is associated. Usually this VIP takes the form of the host name or cluster name followed by a hyphen then a single 'v' and two zero filled digits.

The VIP should live in the same DNS sub-domain as the host or cluster.

The VIP name may be reused.

6.5. Management Interfaces

Hosts have a requirement for administrative access. This access is usually gained via the primary host network. However on host failure it is necessary to gain access to this via out-of-band.

Clear access to this management interface for a host by the population of a DNS CNAME containing the host name and appending an identifier.

In the case of hosts executing on physical devices, the DNS CNAME will point to the device entry on which the host is executing.

In the case of hosts executing inside a virtualisation system this will point to the service name of administrative or management service for virtualisation system which owns that host, e.g. vCenter

6.6. Additional Interfaces

There are occasions when it is necessary for hosts to have additional physical interfaces. Where these physical interfaces are visible at layer 3 of the network stack these should be clearly named. These interfaces should be named according to the interface name as presented to the host’s OS. These interface names should be appended to the existing host name prefaced by a hyphen

7. Storage

Let us begin with some definitions.

A RAID group is a collection of disks for the benefit of performance and reliability.

An Aggregate collates RAID groups into a single management unit. These units are typically associated with a single environment, or site locality.

A Plex grouping of two groups or more RAID groups used for mirroring storage from one physical location to another.

Volumes are carved out of an aggregate, and it is these which present data to Hosts.

The crucial thing here is to understand that there is mapping between storage as seen on the store and the presentation to the Hosts

7.1. Types of Storage

The classification of storage around function allows informed discussions about what can be thrown away and what must be always available.

7.1.1. Local

Unique storage tied to an individual host or capability

e.g. storage tied to the instantiation of a host such as a VMDK of a VMWARE Guest internal storage for an host or unique storage for a disk less environment. Typically , this includes things like certificates specific to a host, ssh keys and kerberos tickets.

7.1.2. Transient

Storage for temporary files, transient during operational activity

e.g. /usr/sap/tmp or /var/spool/lpd on Unix hosts

7.1.3. Shared

Storage which is in common between a number of capabilities/hosts.

e.g. binaries shared on a common file system such as used in disk-less environments
    users home directories,
    /usr/sap/trans

7.1.4. Exclusive

Storage which is exclusively available for a capability which can be made available via another host

e.g. capability data that is moved between hosts within a Highly Available Cluster.

7.2. Performance

this is in terms of access speeds gained by the application to provide the capability.

e.g. it may be possible to improve performance by caching access to read-only file systems with the OS

7.3. Resilience

has to meet the RPO & RTO classified and the associated retention policies

7.4. Utilisation

This is how much a single piece of storage can be used.

For example, it is possible to deploy block based storage containing the OS binaries once for every host that is used within the landscape. However, it is also possible to deploy OS binaries over file based storage in such a way that ever host used within the landscape makes use of them.

This gives greater utilisation of the disk and so a better return on the investment associated with the storage.

7.5. Presentation

The access method for the storage, this should be decided by meeting the other storage requirements.

There are a range of presentation methods available to storage. These include, block based via fibre channel or iSCSI and File based, via NFS, CIFS or a series VMDK files.

8. File System Layout

Classification of the usage of the data placed on the file systems is crucial to the management of the environment particularly when assessing performance and capacity. The file system layout must also reflect the relationships within the components used within the Landscape.

The driving thoughts behind this is to manage duplication and simplify backups. If we have only a few places to change things, and everything points to it, then we ease administration of the systems. If we set down clear rules as to where the data is, we know what to backup.

We also can decide to only backup data which is of value, and not data which can be derived or replicated or re-created.

For example, if we have the OS binaries in two different sites, with reasonable connectivity, it may be decided not to back up the OS binaries as they can always be recovered. Again we may choose not to replicate these binaries but recreate them as this confirms the quality of our scripts to build things from original media, so allowing fast migration into new sites.

The directories should act as a "chroot" into which the binaries are installed. This helps with the isolation of one application from another and assists in recording dependencies . Additionally it helps when the need arises for side by side execution of different versions.

The easiest way to arrange installations in this way is to put a prefix into the package installation command. For example with you can do

```
yum –installroot=/host/OS/kernel/proc/ver/svn-tag/ group install '*'
rpm –root=/application/appname/OS/kernel/ver/svn-tag/ libc.rpm
```

In addition to these "chroot" installations it is crucial that the scripts use to install the applications are recorded and the tests associated with the verification of successful build and instantiation are recorded. These would not usually be found on the file systems but in the version controlled repository.

8.1. OS binaries

/host/OS/kernel/proc/ver/svn-tag
e.g.
/host/CentOS/Linux/x86_64/5.5/build1
/host/Solaris/SunOS/sun4u/10/build23
/host/Windows/2008EE/x86/6/build23

Let us begin with those associated with the OS installations and the associated applications which execute upon them. Key factors here are that they are processor/hardware vendor specific and OS library specific.. Among UNIX operating systems, the ***uname*** command allows us to test this information.;'uname -m' returning the processor information and 'uname -s' returning the "kernel" name.

These binaries in the most part will be read-only. This simplifies backup as these can be replicated to another data-centre or site and these binaries are shared between all systems running them. A single change here effects all systems.

The OS installation should be as generic as possible i.e. not embed host and IP information. For the most part the OS installations will be read-only.

The packages executed on an host should be determined at start-up as far as is possible.

The package start-up scripts usually found on Unix systems in /etc/init.d/ should where possible determine their configuration at run time i.e. where possible calculate within the script what to do, and on shut down tidy up the temporary files used for configuration.

This either can be via the use of a lookup service, such as DNS MX and SRV records, or configuration files held under version control.

If any additions are to be added to the OS installation, it should be remembered that they will be visible to hosts. This should only be done as a last resort or when it is the intention for all systems to see them

8.2. Application binaries

```
/application/appname/OS/kernel/ver/svn-tag
e.g
/application/Oracle/Linux/x86_64/10.1.1g/build21/...
```

Application binaries are the base installation of a software package. They are somewhat complicated by the fact that some applications are embedded in OS installation. For example Apache is embedded many Unix builds including Solaris and Linux. In these cases if a version is required which does not match that supplied with the OS it should be separately installed in the application tree.

Application binaries must be installed via some form of automation, to ensure that they perform as expected, this scripted build must be stored within version control. Once installation is complete they will be read-only.

A series of test must be included to confirm the OS onto which the installation is to occur has met the pre-requisites for the application to operate. These test script form a contract between the OS installation and the application to document and confirm that the requirements have been met. Examples of these tests include confirmation of kernel parameters exist or verification of necessary user ids exist for the application.

The application itself, will never be started on any host, but a script must be created to create a “package” which can make use of these binaries. This script is responsible for creating the start and stop scripts used by the package. This script should live within the file system in the “chroot” init.d directory.

8.3. Packages

/packages/packagename/
e.g.
/packages/webctw-01/

Packages are the specialisation of the Application, and as such can share large parts with the application. This should be achieved by ensuring that the application binaries are symbolically lined to from the package directory. The files found in the package directory should be the configuration files and the associated read-write data files.

Packages act as though they are root directory into which all their software and configuration are installed.

The Package directories will require regular backup to ensure that no data is lost.

Packages are the placed where work is performed. This is also where the detailed configuration is held.

Packages cannot support side by side installation, as they are the point of instantiation of the system. If a different behaviour is required, this is instantiated in a separate package.

8.4. Service

/service/servicename
e.g
/service/webct

Services are really what the business care about. It is crucial that we embedded their names into the day to day work of the organisation. But it is also important to easily view the complexity of a service and so the associated risks.

The Services directory, contains symbolic links to the systems which construct the service.

8.5. Systems

/systems/systemname/svn-tag
e.g.
/systems/webctwww/10.1.1

Systems are where measurements occur that are relevant to the business. This is where availability is noticed and capacity has to change depending on need.

The Systems directory contains symbolic links pointing to the packages which executed for these systems.

8.6. Home Directories

Let us being with some properties of a home directory. There should be at most one home directory per environment per identity. It should be the same on all hosts regardless of OS or protocol of access.

Access to this home directory is controlled by the its associated owning identity and the Security Domains in which that identity resides.

It should have a disk and file quota imposed upon it.

For identities associated with Services, Systems,Applications and Packages, these will be to the directories in which they are installed.

For identities associated with users this will be to /home/

9. Measurement

Measurement is not possible without knowledge of what should be examined; this is the desired state of the environment.

A repository must be constructed containing the desired state for the IT Landscape. This repository is constructed via the deployment processes associated with the landscape.

After the fact discovery of dependencies and the behaviour of systems is rarely possible in any detail. However, discovery of what is currently happening and comparison against the desired state allows for problem diagnostics.

The repository of desired state, should match the relationships described in this document

The goal of a good management system is the ability to comprehensively describes the "goal state" of the system and have "viewers" which can see how a system differs from the goal state. "Controllers" would be responsible for moving a system to its goal state. This is exactly the same idea as model-based system management, just using different language

9.1. Relationships

The key relationships types required for measurement are those of parent-child and aggregation.

A Landscape is the aggregation of Environments.

An Environment is the aggregation of the Services within an Environment.

A DNS Domain has a parent that is an Environment.

A Kerberos Realm has a parent that is an Environment.

A Functional Group is an aggregation of Layer 2 Domains.

A Layer 2 Domain is an aggregation of interfaces and hosts.

A Service is the aggregation of Systems.

A System has a parent. This parent may be another System or Service. Failure of any of these child systems results in degradation in the delivery of the parent.

A System is an aggregation of packages.

A Package has a parent Application.

A Package has an aggregation of hosts or VIPs on which they execute.

A VIP has a parent. This parent may be either a host or a cluster.

An Interface has a parent. Interfaces must have a parent Layer 2 domain.

An Interface must have a parent host

A Host has a parent. This parent may be either a device or a service. If it is a service then the host is virtual.

An Element has a parent. This parent may be either a device, or another element. Failure of any of these system elements results in the degradation of the parent.

A Device is the aggregation of elements

A Host has a parent. This parent may either be a device, so the host is physical. If the parent is a service then the host is virtual and capable of moving between devices.

Hosts may have an aggregation of additional interfaces.

A Cluster is an aggregation of hosts and VIPs.

9.2. Data Collection

Ideally data collection should occur via SNMP polling and ICMP. The parsing of SYSLOG assists in problem diagnosis. The parsing of NetFLOW assists in the measurement of performance and connectivity of systems.

9.2.1. Packages

Packages are the unit of work. This unit of work executes upon hosts.

In UNIX, a package is a process tree which has the user name of the package Processes in this tree may change their effective id to another user, but they are all started by an init script or daemon which commences running as this username of the package.

In Windows, a package is an aggregation of processes executing as the username of that package.

It is possible to have two identical packages running on a single host; however, they can be separated as they have unique process trees.

On some other hosts, such as routers, the successful execution of packages is confirmed by routing information or network transits. Another example is the execution of packages on a storage device, which can be confirmed by the existence of their exports or shares and their accessibility.

For each Package instance, CPU, Memory and IO usage, as well as the host or cluster on which it is currently executing are reported on. Also it is useful to report on the package owner’s disk quota usage.

All of this information can be collected via SNMP.

Logs produced by the packages should be submitted to the measurement service over the SYSLOG service, marking the submissions clearly with their package name and host, where possible.

9.2.2. Applications

Applications are never executed; however, reporting the number of packages per application allows accounting of software costs. Additionally reporting which hosts the Applications have executed on may affect software costs.

Applications are the binaries and so come with storage costs.

9.2.3. Systems

Availability and outages associated with systems are of interest for the availability. The hierarchy of packages allows rapid evaluation of root cause of the failure of a system

9.2.4. Services

These are what the business cares about. For each environment, there should be a dashboard to report on these and their availability and outages.

9.2.5. Devices

Ultimately, hosts end up being executed on devices. Monitoring of devices can be done via SNMP to the out of band management interfaces of the device. This should report device related information such as uptime of the device, availability of the underlying elements.

Logs produced from devices should be forwarded to the measurement service over SYSLOG

9.2.6. Hosts

Hosts have some overheads associated with their existence. For each host, reporting is performed for the amount of CPU, memory, and IO utilisation. Additionally it may be useful to report on particularly system disk partition usage such as /tmp and /var or c:\ to ensure these are not going to be the cause of failures. Hosts should report on their availability

Logs produced from hosts should be forwarded to the measurement services over SYSLOG

9.2.7. Clusters

For Clusters additional information should be collected about the availability of the hosts. If the number of hosts drops below a predefined threshold of quorum then the cluster should be viewed as failed.

9.3. Faults

Faults may be reported by any of the collection points, however to be useful we to classify them appropriately. Below is listed a suggest classification scheme for these events.

9.3.1. Device

Examples of these messages would be Network errors, Device reboot, Temperature, power usage.

9.3.2. Host

Examples of these messages are Paging Rate, CPU and Memory utilisation, process creation difficulties, allocation of swap.

9.3.3. Module

These are associated with executed code and runtime: Examples of this are un-handled exceptions,malloc failures, unable to write to a file.

9.3.4. Data

These are associated with the execution of the package and its configuration. Examples of this include null account, divide by zero, unable to find find file

9.3.5. Session

These are associated with component interactions. Examples include unable to open ODBC, Session dropped.

9.3.6. Transaction

These are associated Logical Units of work. It is difficult to separate these from session issues, though these are more often of monetary value and regulatory interest.

9.3.7. Security

These are exceptional events, seen across all collections

9.4. Configuration

Ideally all configuration changes will be under version control and their provision automated. Any change that has occurred from a different route, should be spotted and treated as a security event. This is relatively easy as the changes within version control are recorded.

Any configuration change, should cause a message to occur in any related event log, rising to the service level associated.

9.5. Accounting

A simple Net-SNMP script can be deployed to all hosts, which return the summation of CPU, IO and Memory Usage for all Packages running on that host.

It is not necessary to connect to VIP offered by Clusters, as the executing processes will be upon a host which returns all packages executing on that host.

This package information can easily be aggregated over hosts to show resources used by that packages. Additionally it is easy by group aggregation to see the resources used by a particular service.

By careful package deployment it is often possible to determine further resource usage such as storage since the home directory of the user used for the execution of the package can be examined for disk space usage.

9.6. Performance

A simplistic approach to the evaluation of the performance of a package can be determined by examine the time to complete a particular transaction. NetFlow information allows us to return information about which IP addresses exchanged information and how long that exchange took. This provides an elementary base line for the performance of the package.

9.7. Security

TBD

10. Worked Example

This is far from a fully worked example. The intention here is to show a flavour of use of the name spaces within the environment and how they should be populated and used.

10.1. AD Forests

An AD Forest rdp.baml.com with domains inside prod.rdp.baml.com and dev.prod.rdp.baml.com

This AD forest and domains is purely for the reference data platform. The only user and group ids present within this are for the packages, software applications and business services which are provided by the reference data platform.

There are trust relationships with other AD forests within the organisation. These are generally based around the line of business used within the organisation.

Identity in these 3rd party Forests, rests with the line of business. They are also responsible for the attestation of the identities into groups within their own Forests.

It is these groups which are placed into a trust relationship with a group within the Reference Data AD Domain. These groups in turn are used to permission access to the package which is being executed.

10.2. DNS Domain names

prod.rdp.baml.com for the OS instances (hosts)
site.uk.rdp.baml.com for the Out-of-band access to physical devices
site.us.rdp.baml.com
site.jp.rdp.baml.com
prod.loc.rdp.baml.com for the mapping of OS instances (hosts) to physical devices

10.2.1. Nota Bene

It would be possible to do much what is achieved with multiple DNS domains within fewer DNS domains. However not all OS's support DNS LOC or SRV records which would allow this.

The presence of multiple DNS domains representing environments allows a host with its configuration almost untouched to be moved from one environment to another using a technique like XCOPY.

Any entries found in the country code sub-domains should map to asset tags which map to the out of band access to the elements or devices.

10.3. Host

Take the OS identifier and the sequence number and add this onto the environment name.

r540n0010.prod.rdp.baml.com is a DNS A Record

10.3.1. Nota Bena

Host name cannot have a zero index thus host names ending with 00 are not allowed.

Hosts started with DHCP often fail to manage multiple host names over multiple instances. Thus it is important for all DHCP started interfaces that the host name offered via DHCP is identical

10.4. Clusters

A host name in which the 4 character is a 'c' is part of a cluster. In addition to a host name, you require a cluster name. The cluster name is determined by converting the last two digits of the host name to zeros. Thus a host name r540c0011 and r540c0010 are both members of the cluster r540c0000.

10.5. VIPs

All VIPs are DNS A Records. Thus, if a host or cluster requires a VIP, then this VIP is stored as a DNS A record in a name that takes the host or cluster name and appends a '**-V**' and a series of zero filled sequence numbers. Thus one might get

r540n0010-v010.prod.rdp.baml.com
r540c0000-v001.prod.rdp.baml.com

10.6. Location

Take the hostname, and the environment name and look this up in the loc.rdp.baml.com

r540n0010.prod.loc.rdp.baml.com

If the host is physical it will return a DNS CNAME to an A record in the DNS Domain
facility.iso-country-code.rdp.baml.com

e50123112.canary-wharf.uk.rdp.baml.com

The same pattern applies for virtualised hosts. Thus for a hostname w280n0010.prod.rdp.baml.com, a lookup for w280n0010.prod.loc.rdp.baml.com might return
vcentre.vmfarm01.uk.rdp.baml.com

Another example might be that r550n0012.prod.rdp.baml.com, following the same pattern a looked up for r550n0012.prod.loc.rdp.baml.com might return

ami1231231.amazon.west.rdp.baml.com

10.6.1. Virtual Locations

There are various cloud providers who have little concept of country. For example they describe the facilities in terms of region. In these situations they typically describe themselves in terms of emea or east or west. These should be reflected into the ISO-Country code arena but will require special handling.

10.7. What to start up?

So do a DNS enumeration for CNAMES which point the host-name or cluster-name.

For each of these CNAMES confirm that a username exists . At that point, find a start-up script within that users home directory, and execute this start-up script as that user.

Thus a DNS CNAME iis01.prod.rdp.baml.com pointing to DNS A record r540n0010.prod.rdp.baml.com would require verification of the existence of a username ii01@prod.rdp.baml.com. Assuming that this user exists a start-up script ~iis01/etc/init.d/start would be executed as that user.

10.8. Business Services.

Each Business Services gains its own sub-domain. For example there might be sub-domains

copper.rdp.baml.com
cpw.rdp.baml.com

All entries in these domains are DNS CNAMES which point to entries which already exist as CNAMES in production domains. This additional indirection is to allow each Business Service to evaluate in real time what resources are being used and move the services around a number of providers without comparative ease