HARVARD
UNIVERSITY

Information Technology

# Integer Distribution Model

Scale, Reliability, Security and Data Resiliency

Wednesday, February 26, 2014

## Table of Contents

# The Harvard Integrated Management System - Project Integer[1] Distribution Model

| Version | Date | Description | Contributors |
|---|---|---|---|
| 1.0 | Feb 24, 2014 | Initial version. | Dave Taylor, Jon Saperia |
| 1.1 | Feb 25, 2014 | Added assumption about port for MariaDB Clustering | Dave Taylor, Jon Saperia |
| 1.2 | Feb 26, 2014 | Integrated Comments from Scott Bradner | Dave Taylor, Jon Saperia |

## Introduction

Integer is designed to be deployed in a number of environments from the traditional data center to a variety of cloud environments. It is also anticipated that it will have to be deployed in hybrid environments with some system elements in one or more cloud environments and others in a more traditional data center.

Scale, security, high-availability and data resilience are also key requirements regardless of the deployment environment(s) used for an Integer installation. Because it is not possible to anticipate every possible configuration in advance, the approach Integer employs is to provide a flexible deployment framework that can be customized to the requirements of the environment.  It is through the configuration of these components that operations staff tailor the system with regard to these requirements. This document describes the elements whose configuration and deployment determine these properties.

## Document Purpose

This document is intended to:

1. Illustrate for core engineers working on the Integer project, anticipated deployment options so that as code is developed it can be structured to accommodate these deployment variations.
2. Provide information to operations personnel about different ways the system may be deployed to support a number of scale, security, reliability and data resiliency requirements.
3. Identify the expected ‘default’ configuration for the Phase 1 system so engineering resources can focus on these capabilities while still understanding the strategy for the topics described in this document.

[1] The project, Integer, is an attempt to create a unified whole from the separate protocols, data elements and software systems we use to operate our increasingly complex computing environments. See: http://www.thefreedictionary.com/integer. Also see:  http://en.wiktionary.org/wiki/integer#Latin

4. Allow operational people to prepare for both the FCS version as well as later versions of Integer.
5. Provide general background to non-HUIT/non-Harvard people what is planned for the system.

## Security

Many factors influence the security characteristics of a system. How a system is distributed and the different components accessed influence how vulnerable the system is to various threats. Some factors are outside the basic architecture of a system but are dependent on how it is deployed and configured in a given environment. The following sub-sections describe the features Integer uses to secure the system and its data given the distributed architecture.

### Access - Authentication and Authorization

The system employs a model where access is controlled across several axes:

1. Authentication to access the system
2. Authorization to access functions and data in the system.

Additional details on this portion of the system are found in the Integer Access Control Model document. The operationally important details are that system operators can control:

1. The required external authentication and attribute service systems on user and role basis.
2. Whether to use external systems or not - the default is that users will an external authentication system. Support for which systems are supported will be detailed in each release. The initial target is the Central Authentication Service (CAS).
3. Which users have direct access to the system - that is, do not use an external authentication service. This is provided primarily as a failsafe should any of the authentication/authorization infrastructure fail or not be useable by Integer. This type of access can also be configured to also grant the user 'cli' access to the system in the same way that one would ssh directly to a host.

#### Default for Phase 1

For the Initial release of the Integer system, direct login to Integer will be supported. If time permits, integration with a CAS service and LDAP attribute service will be added.

#### Authorization and Role-based Control

After a user authenticates, the system will determine the role(s) that apply at login-time. Based on these roles, user access to information is controlled in the following ways:

1. Which functions for which service elements the user is granted. These are:

    1. Read
    2. Write
    3. Create
2. Specific locations associated with an organization.
3. Criticality of the system.
4. Type of service element.
5. Certain technologies (e.g., access control, but not routing)
6. Specific instances in a service element (e.g., interfaces 1 and 3 but not 2).
7. Authentication type.

This discussion of access control is relevant in the context of security and distribution because it, rather than compartmentalization of data by location, function or other facility is used. Integer is designed to create a whole view of information so that better operational decisions can be made. While the system provides great flexibility in distributing the operational elements and data they contain, relevant data for these distributed elements is collected by the 'core' system for effective analysis.

**Default for Phase 1**
The default for all users in the system is that no access to data or functions is permitted until they have been associated with one or more roles.

### User to System Communications
All user to system communications are via HTTPS. It is a matter of location configuration as to whether two factor authentication is required for users when they authenticate. In addition, the system may be deployed so that direct user access requires a VPN - this is a matter of network configuration. Integer can also be configured so that access can be restricted to specific IP addresses for the direct access mode described above.

## System Model - Functional Perspective
In its simplest deployment, all of Integer's functions and data may be deployed on a single system. In this system, even the front end service for client communications is executed on a single platform and is illustrated in Figure 1:

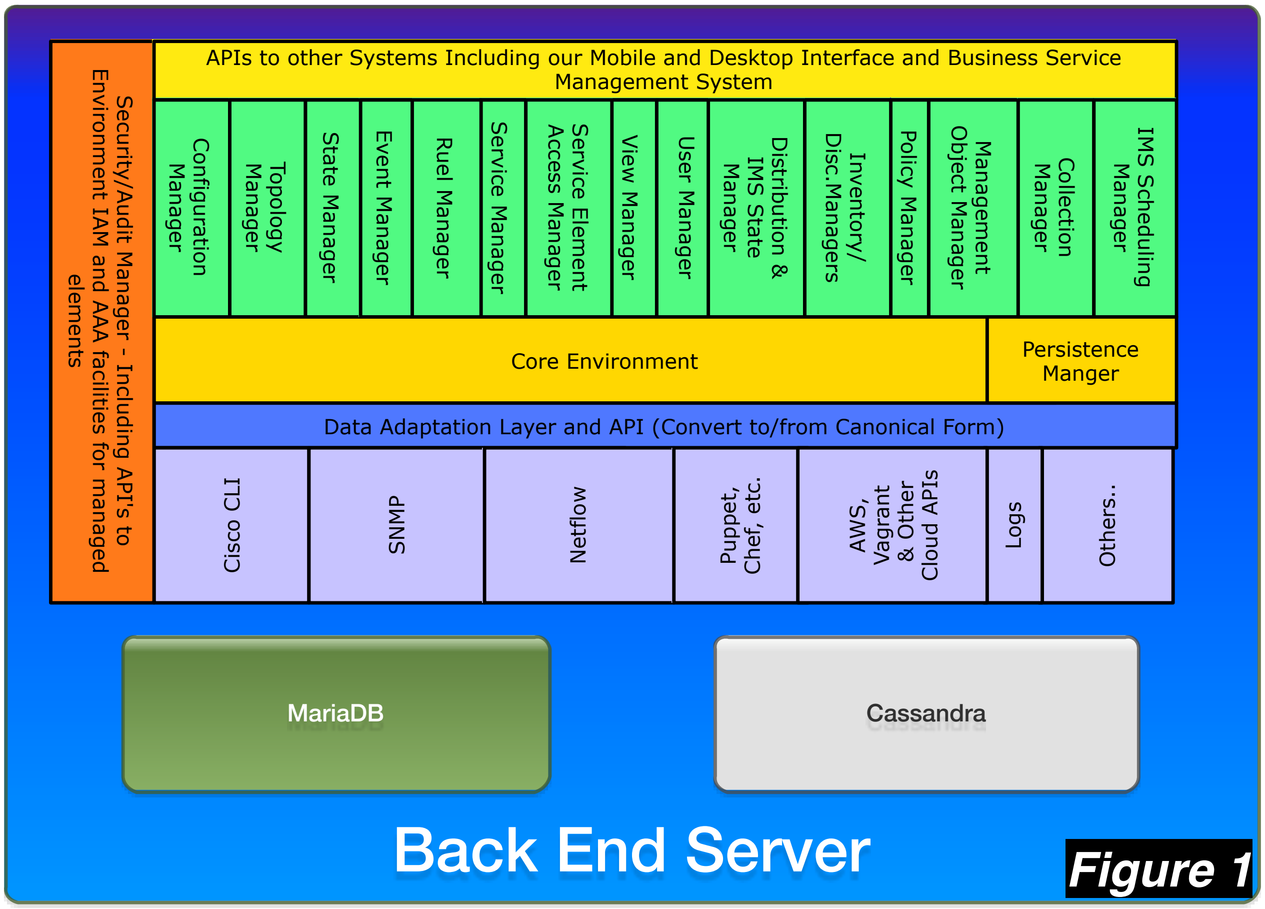

APIs to other Systems Including our Mobile and Desktop Interface and Business Service Management System
Security/Audit Manager - Including API's to Environment IAM and AAA facilities for managed elements
Configuration Manager
Topology Manager
State Manager
Event Manager
Ruel Manager
Service Manager
Service Element Access Manager
View Manager
User Manager
Distribution & IMS State Manager
Inventory/ Disc.Managers
Policy Manager
Management Object Manager
Collection Manager
IMS Scheduling Manager
Core Environment
Persistence Manger
Data Adaptation Layer and API (Convert to/from Canonical Form)
Cisco CLI
SNMP
Netflow
Puppet, Chef, etc.
AWS, Vagrant & Other Cloud APIs
Logs
Others..
MariaDB
Cassandra
Back End Server

*Figure 1*

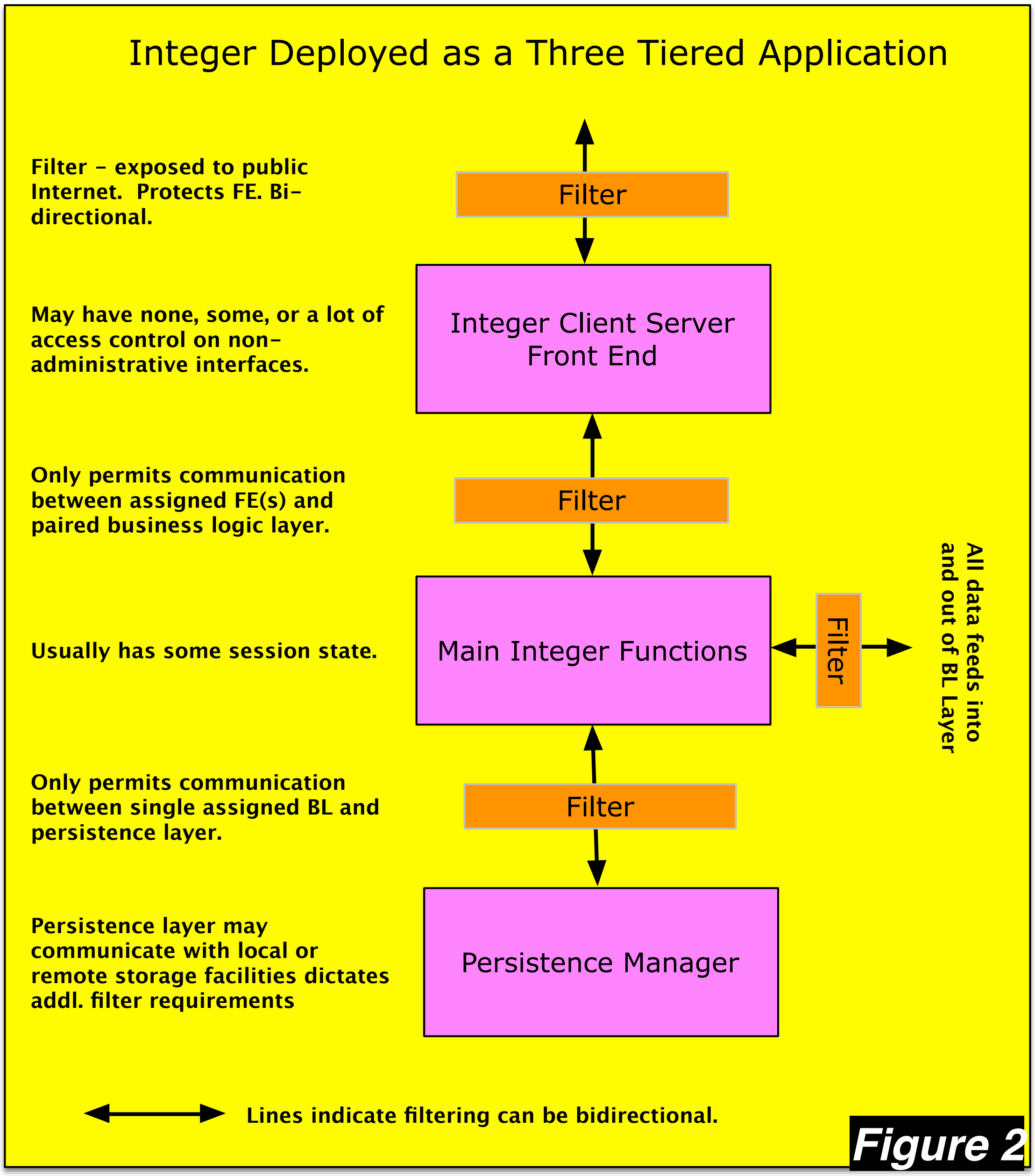
While simple, this deployment approach will not meet the scale, security and other characteristics identified above that some environments require. For this reason, it is possible to separate components in a number of ways.

## Basic Separation of Functions and Deployment

The diagram above could be simplified into three simple blocks that would resemble how modern web applications can be deployed to achieve the same properties of scale, security, availability and data resiliency required for Integer. It would look like Figure 2 on the following page.

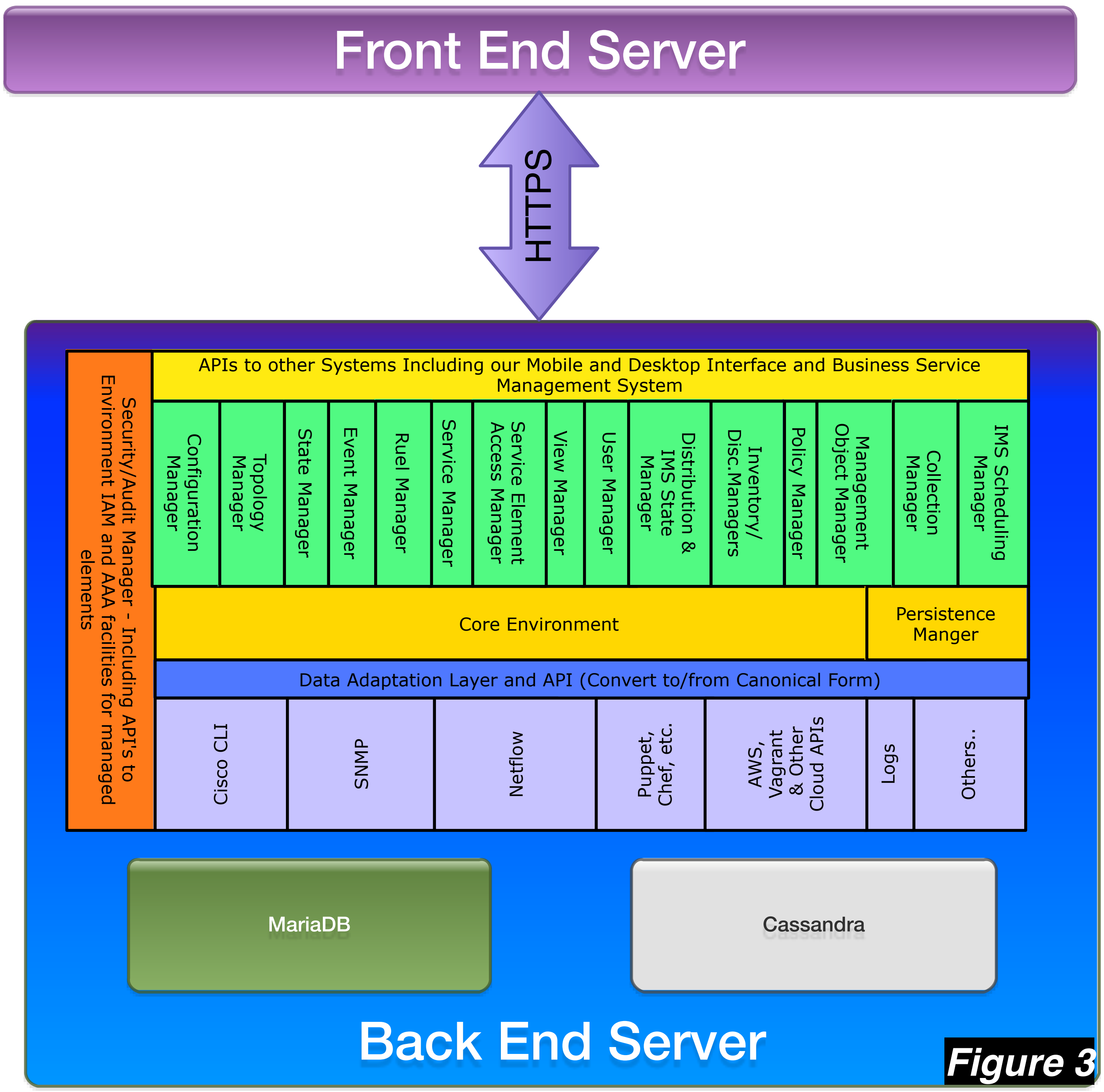
Note that the filters in the diagram could be thought of as hardware or software firewalls. In a cloud-based deployment mechanisms such as an Amazon Web Services[2] security group could be used to achieve the same function. Also note that the Integer software makes no assumption about the presence or absence of these filters. It is left to the operational personnel to determine how/if they are to be set up.

[2] See section on Amazon Web Services-specific Features and issues.


Integer Deployed as a Three Tiered Application
Filter - exposed to public Internet. Protects FE. Bi-directional.
Filter
May have none, some, or a lot of access control on non-administrative interfaces.
Integer Client Server Front End
Only permits communication between assigned FE(s) and paired business logic layer.
Filter
Usually has some session state.
Main Integer Functions
Filter
All data feeds into and out of BL Layer
Only permits communication between single assigned BL and persistence layer.
Filter
Persistence layer may communicate with local or remote storage facilities dictates addl. filter requirements
Persistence Manager
Lines indicate filtering can be bidirectional.
**Figure 2**

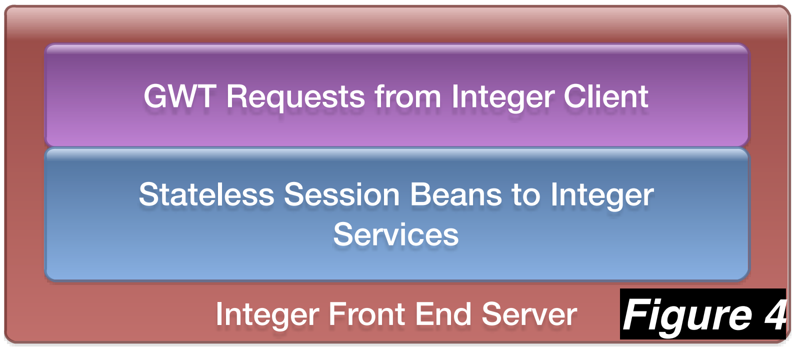
## Phase 1 Default Deployment Model

A simpler variant of this, and the default deployment model, is one where the persistence and main Integer functions are housed on a single system as illustrated in Figure 3, below:


Front End Server
HTTPS
Security/Audit Manager - Including API's to Environment IAM and AAA facilities for managed elements
APIs to other Systems Including our Mobile and Desktop Interface and Business Service Management System
Configuration Manager
Topology Manager
State Manager
Event Manager
Ruel Manager
Service Manager
Service Element Access Manager
View Manager
User Manager
Distribution & IMS State Manager
Inventory/ Disc.Managers
Policy Manager
Management Object Manager
Collection Manager
IMS Scheduling Manager
Core Environment
Persistence Manger
Data Adaptation Layer and API (Convert to/from Canonical Form)
Cisco CLI
SNMP
Netflow
Puppet, Chef, etc.
AWS, Vagrant & Other Cloud APIs
Logs
Others..
MariaDB
Cassandra
Back End Server
Figure 3

## Front End Server
Integer front end servers exist primarily to isolate those systems which must be publicly accessible from the back end systems that process and contain potentially sensitive information.

The function provided by the front end server is to take requests from Integer's GWT-based clients and transform them into stateless session beans that are forwarded to the appropriate Integer services. This function is illustrated in Figure 4:


GWT Requests from Integer Client
Stateless Session Beans to Integer Services
Integer Front End Server
Figure 4

## Deployment and Separation of Components

Integer is designed to provide flexibility in the way it can be deployed to meet performance, availability, security and data resilience requirements.  The design objectives for the system are[3]:

1. Hundreds of concurrently logged in users.
2. Up to 25 active users at one time, for example making configuration changes, producing ad hoc reports, etc.
3. 100,000 Systems and all the sub-elements they contain whether in physical or virtual form.
4. Continuous operation - when deployed in a high-availability form, the system will will continue operation in the event of a failure of an Integer node or one or more of the communication facilities between systems.
5. Durability - No lost data - when deployed in a high availability configuration, the destruction/unavailability of data on one system will not impact operation.
6. Isolation of systems that have sensitive information from those that are publicly accessible.

The following sections describe the approach to realizing these objectives.

### Distribution Models

To achieve the scale, availability, data durability and security objectives for the system, Integer may be deployed in a number of progressively more complex patterns.

### Cassandra Functions

Systems like Integer often have scale problems, most often in the area of serialization of data in and out of the database. Integer has removed the high volume/frequency

[3] These are end state goals, it will take many releases to achieve them all.

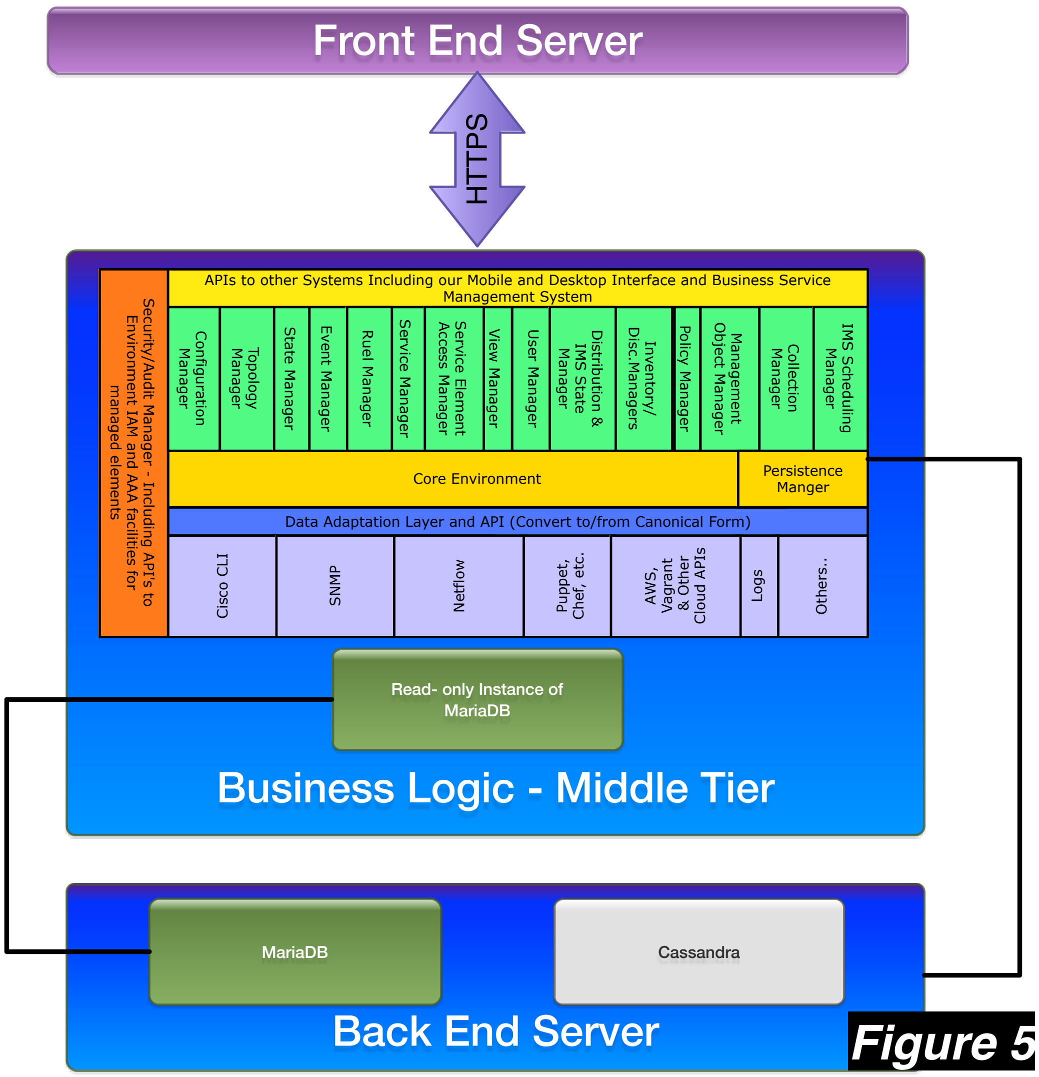
data to Cassandra which will significantly reduce the load on the main DB and give better overall performance.  The data elements moved to this system are:

- High frequency event/state data.
- State history.
- Statistics

**Cassandra Default Configuration**
The default for Cassandra is that it exists on the same server as the Database. While future configurations allow for its operation on a separate server or cluster operation, that facility is not provided in the Phase 1 product.

## Basic Configuration

In Figure 5, the system has been deployed so that all business logic components are running on a single server with the exception of  separate front end and persistence servers. The front end server is used by Integer clients requesting Integer functions.


Front End Server
HTTPS
APIs to other Systems Including our Mobile and Desktop Interface and Business Service Management System
Security/Audit Manager - Including API's to Environment IAM and AAA facilities for managed elements
Configuration Manager
Topology Manager
State Manager
Event Manager
Ruel Manager
Service Manager
Service Element Access Manager
View Manager
User Manager
Distribution & IMS State Manager
Inventory/ Disc.Managers
Policy Manager
Management Object Manager
Collection Manager
IMS Scheduling Manager
Core Environment
Persistence Manger
Data Adaptation Layer and API (Convert to/from Canonical Form)
Cisco CLI
SNMP
Netflow
Puppet, Chef, etc.
AWS, Vagrant & Other Cloud APIs
Logs
Others..
Read- only Instance of MariaDB
Business Logic - Middle Tier
MariaDB
Cassandra
Back End Server
***Figure 5***

## Persistence Functions Separation

The MariaDB and Cassandra, the Integer persistence services, have been moved to a separate system. Note the Read-only instance of MariaDB at the business logic layer. This is provided as another way to improve performance and while it may not be needed in this configuration, it is the design pattern that will be used for larger scale deployments using clustering and geographic/functional distribution.

The reason that persistence functions are moved to a separate system as the first in a series of higher performance alternatives is that experience has shown, these functions have historically been the bottleneck in an otherwise well constructed system. This leaves the the WildFly portion of the system that contains all of the other functions (this can be thought of as the business logic layer) in Figure 5 on a dedicated system. In this case, the communications between the WildFly system and the back end system that houses the persistence functions (MariaDB and Cassandra) are secured via SSL.

The read-only MariaDB instance that always in exists in this deployment configuration, and all subsequent models presented at the BL layer, serves the dual function of reducing load on the read-write instance and improving performance for task on the BL system that require frequent data reads. The Cassandra system's basic performance characteristics should not require the installation of a parallel system at the BL layer.

The changes described above are the first in a series of increasingly more sophisticated deployment approaches that improve performance.

## Separate Cassandra and MariaDB Servers and Clustering

Significant increases in performance are obtained by:

- Separation of the Cassandra and MariaDB servers onto separate systems.
- Using Cassandra, MariaDB and WildFly clustering facilities (while it is possible to have a cluster for the MariaDB read-only server in the diagram that follows, it is not anticipated that this will be needed.
- Adding additional front end servers.

### MariaDB Clustering via Galera

MariaDB uses Galera[4] to provide its clustering capabilities, references in this document to MariaDB clustering are intended to incorporate Galera Facilities. These facilities are further enhanced by the ability to be deployed in cloud environments, in particular, AWS,-though it is not required for even very robust deployments as illustrated in Figure 6:

[4] For more information see https://mariadb.com/kb/en/what-is-mariadb-galera-cluster/

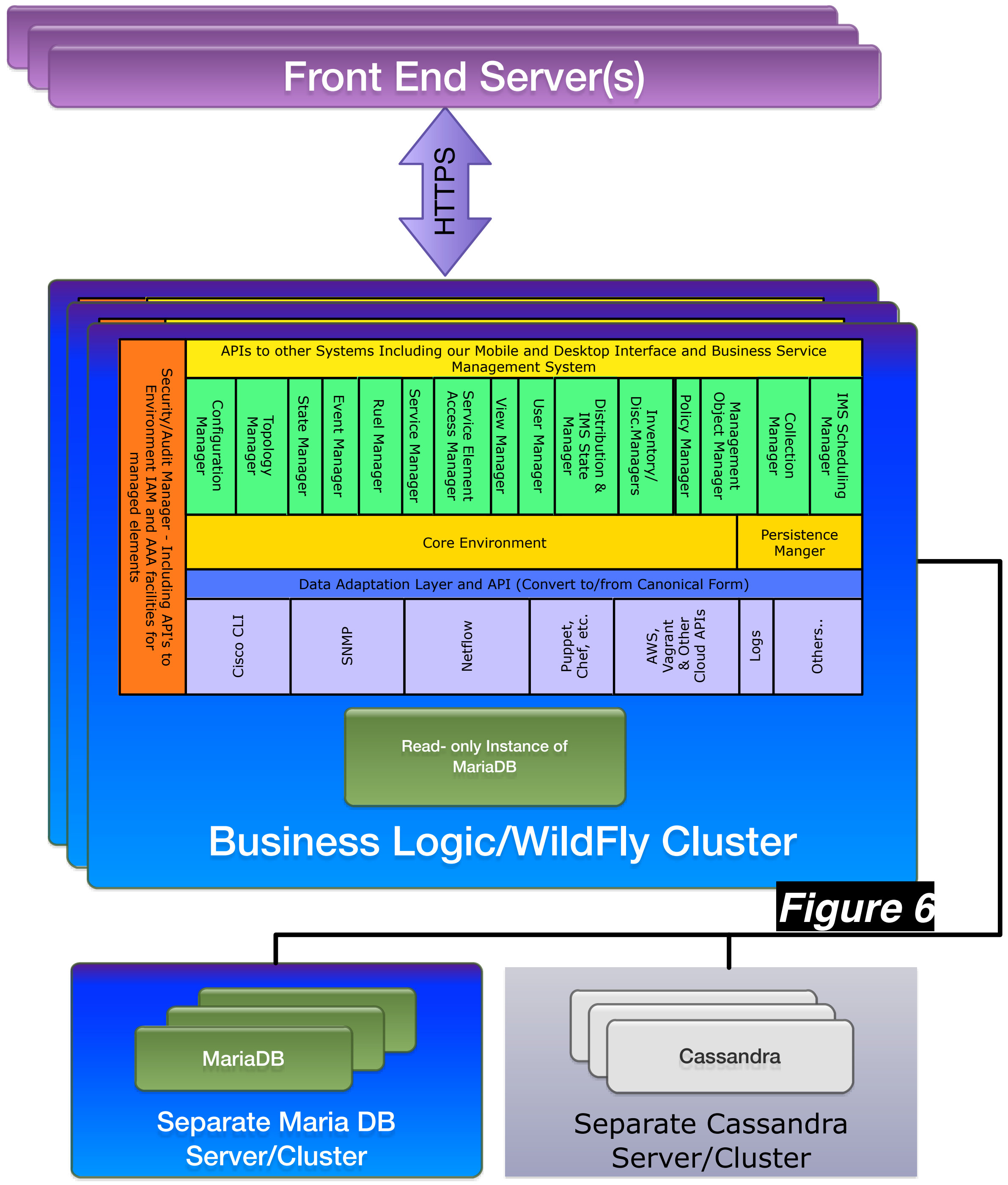

Front End Server(s)
HTTPS
APIs to other Systems Including our Mobile and Desktop Interface and Business Service Management System
Security/Audit Manager - Including API's to Environment IAM and AAA facilities for managed elements
Configuration Manager
Topology Manager
State Manager
Event Manager
Ruel Manager
Service Manager
Service Element Access Manager
View Manager
User Manager
Distribution & IMS State Manager
Inventory/ Disc.Managers
Policy Manager
Management Object Manager
Collection Manager
IMS Scheduling Manager
Core Environment
Persistence Manger
Data Adaptation Layer and API (Convert to/from Canonical Form)
Cisco CLI
SNMP
Netflow
Puppet, Chef, etc.
AWS, Vagrant & Other Cloud APIs
Logs
Others..
Read- only Instance of MariaDB
Business Logic/WildFly Cluster
MariaDB
Separate Maria DB Server/Cluster
Cassandra
Separate Cassandra Server/Cluster

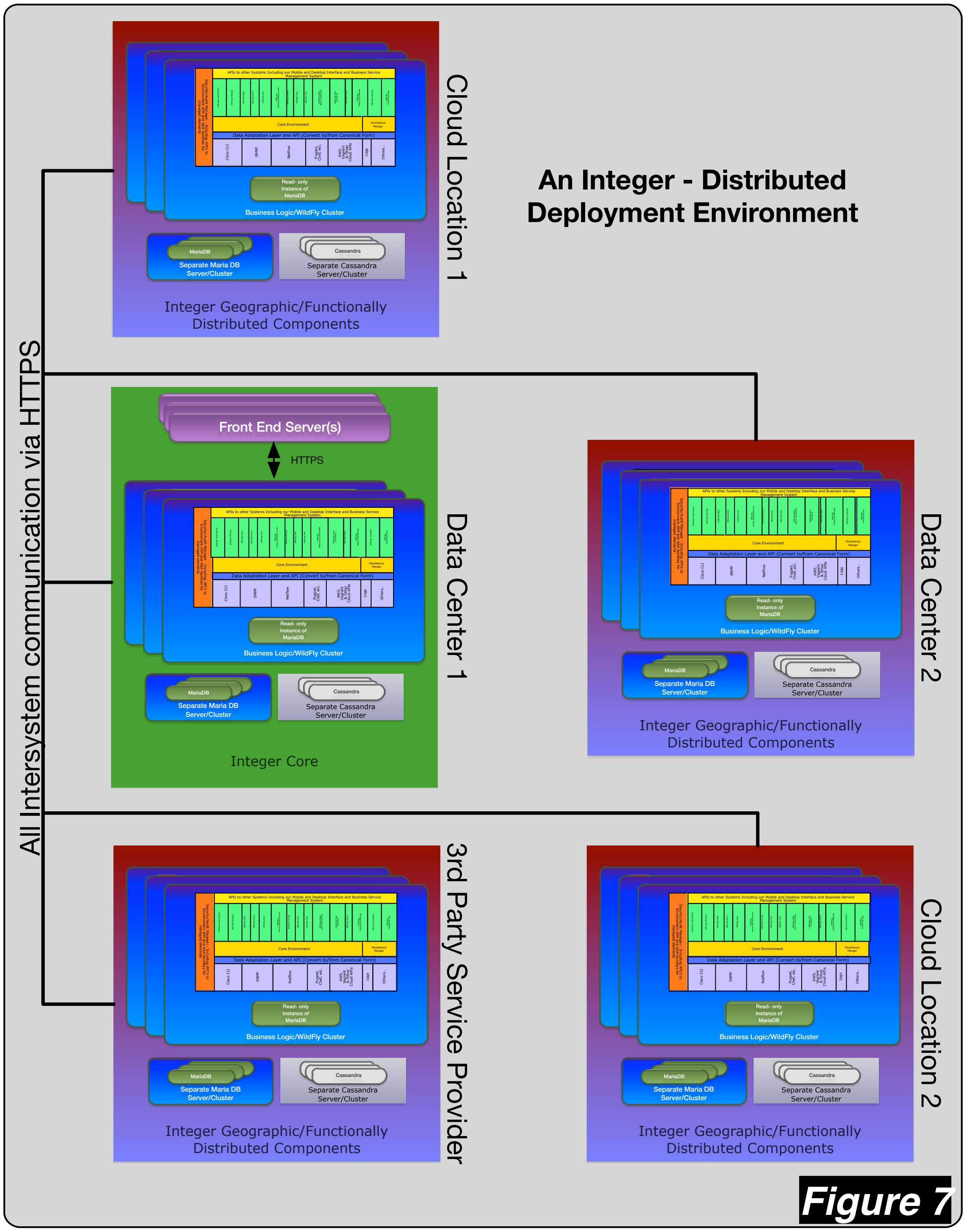
Figure 6

## Improvements in Availability and Data Resiliency

In the preceding discussion, the focus has been on performance. Use of clustering technologies for Cassandra and MariaDB along with a read-only instance at the business logic layer also result in a higher availability system, even when deployed within a single data center. Greater availability and resiliency may be achieved if the clusters are distributed across data centers, or the cloud.

## Geographic or Functional Distribution

With the exception of the front-end servers that can only connect to the ‘core installation’, see Figure 7 that follows, all the other variations in distribution can be


An Integer - Distributed Deployment Environment
All Intersystem communication via HTTPS
Cloud Location 1
Data Center 1
Data Center 2
3rd Party Service Provider
Cloud Location 2
Front End Server(s)
HTTPS
Integer Core
Integer Geographic/Functionally Distributed Components
Read- only Instance of MariaDB
Business Logic/WildFly Cluster
Separate Maria DB Server/Cluster
Separate Cassandra Server/Cluster
MariaDB
Cassandra
Figure 7

deployed in various locations across the environment.

This feature allows for increased scale, and improved reliability.

## Integer Core

In the preceding diagram, Integer Core (shown in green - not the internal yellow strip in every diagram) refers to the system or set of systems that contain the front end servers for clients and the consolidated MariaDB and Cassandra databases. The basic scale provisions of Integer will make a configuration with only an Integer Core suitable in a wide variety of cases.

### Using an Integer Core with Geographic/Functionality Distributed Components.

In addition to the obvious scale advantages associated with deploying additional servers there are two additional significant advantages that come from this deployment approach:

- Ability to distribute integer functions. For example, if the system is required to produce frequent reports, then it may make sense to offload reporting to another server/cluster so that these functions will not impact the ongoing operation of the Integer system or the user experience.
- Simplified network configuration. Many networks impose limits on the types or protocols that they permit through firewalls and other infrastructure elements. This is one of the factors that influenced the decision to use WildFly as it multiplexes protocols over port 8080. From time to time, operations personnel will access servers directly using conventional SSH.

## Distributing Integer Functions

This approach also means that distributed elements that do discovery, monitoring, configuration and other functions my be placed where they protocols used for these functions need not be carries across network boundaries that restrict them. For example, some environments restrict management and configuration protocols like SNMP or Puppet from traversing certain network boundaries. With this approach a distributed Integer system may be deployed to the isolated network (or cloud environment) and configured to perform functions only on those elements inside that segmented environment. From the Integer perspective, this is a matter of configuration. It will perform management functions on Service Elements in the networks/environments it has been assigned. It will perform appropriate aggregation and other functions to reduce unnecessary traffic between Integer distributed components.

## General Architecture for Supporting Distributed Functions

Each of the services in the integer system have architectural elements that support this distribution function. These services have appeared in previous diagrams in green. Examples include:

- Configuration manager
- State manager
- Event manager
- Policy manager
- etc.

The following diagram illustrates the general structure for these services:

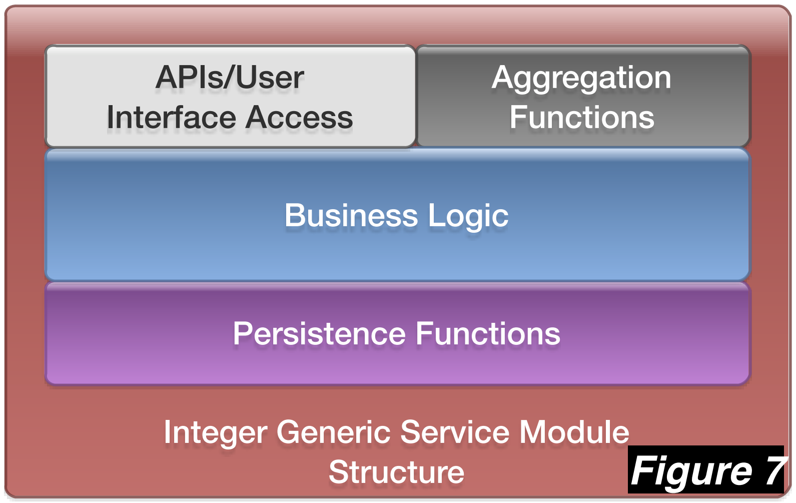

APIs/User Interface Access
Aggregation Functions
Business Logic
Persistence Functions
Integer Generic Service Module Structure

Figure 7

The element in the above diagram most relevant to this discussion is the one that performs aggregation functions:

- Based on configuration it knows if it is on an Integer Core installation, if It is, it will know from which other systems performing the same service at itself it will be receiving or sending information.
- If the system is not a Core Installation, it will know about the Core system and the ‘rules’ for aggregating information to forward to the Core and what will be sent from the core to it. A future capability might be to deploy systems in a hierarchical function, but that it not anticipated at this time.
- In the event of a communication failure between the Core system and one or more of the distributed systems or clusters, the distributed systems continue to function and cache the data until communications are restored. When communications have been restored the cached data is sent to the core system.  The other advantage of this approach is that the system can still be used to support the rest of the environment not impacted the the communication failure.

## Same Integer Image for Distributed Configurations

In the previous sections a number of deployment configurations have been described. In each of these deployment models, the Integer software is the same, it is not a separate build.  Configuration of the Integer installation controls the scope of the provided functions.

### High Availability and Data Resilience

The previously described distributed and clustering configurations all add to a system with very high availability.

### Data Security - Intra System Communication

In addition using HTTPS for Client front end communications and 8080 for inter WildFly communications. Data are also secured both in transit and at rest in the Cassandra and MariaDB clusters.

#### Amazon Web Services-specific Features and Issues

To the greatest extent possible, Integer is being designed to avoid dependence on proprietary technologies. This includes Amazon's AWS.  That does not mean that Integer could not be deployed in a way that would take advantage of AWS facilities like autoscale, availability zone and regions to enhance scale, reliability and data resilience.

## Implementation Order of Scale/Availability Functions

All of the features for distribution, scale, availability and security will take time to implement. The design of the system incorporates all the features needed to support the identified capabilities, however, the will be implemented in phases to allow the system to be deployed in less complex configurations that will inform future development.  The planned sequence is:

1. Deployment with all elements on one server and a single front end server.
2. Separation of persistence functions with MariaDB and Cassandra moved to a separate server and a read only MariaDB in the business logic server.
3. Deployments with Wildfly, MariaDB and Cassandra Clustering.
4. Ability to distribute the system across multiple locations with aggregation back to a central point.

## Assumptions

This section will be updated as needed.

1. The HTTPS protocol is permitted between Integer components.
2. Access protocols used by Integer to access managed elements are native and will be constrained based on the network configuration.  For example, if SNMP is not permitted across certain network boundaries, and distributed Integer components have not been set up on the opposite sides of the boundary, the systems without a distributed Integer component will not be manageable by integer.
3. It is believed that the core technologies used to implement and distribute Integer will 'play' well in a wide range of environments. These technologies are:
    1. GWT

    2. MariaDB
    3. Cassandra
    4. WildFly
4. Documentation suggests the Galera clustering technology will work across subnets. We have yet to deploy it in that fashion to identify issues.
5. Galera clustering technology requires ports 3306 and 4567 to be open.  It is assumed that clusters will be deployed in environments that can support cluster elements access to each other over these port numbers.