**NIST Big Data**

**Technology Roadmap**

**Version 1.0**

**Input Listing: M0087**

**Technology Roadmap Subgroup**

**NIST Big Data Working Group (NBD-WG)**

**September, 2013**

[Executive Summary 4](#_Toc367419217)

[1 Purpose, Background, and Vision 4](#_Toc367419218)

[1.1 NIST Technology Roadmap Purpose 4](#_Toc367419219)

[1.2 Big Data Background 4](#_Toc367419220)

[1.3 NIST Big Data Technology Roadmap Stakeholders 4](#_Toc367419221)

[1.4 Guiding Principles for Developing the NIST Big Data Technology Roadmap 5](#_Toc367419222)

[2 NIST Big Data Definitions and Taxonomies (from Def. & Tax. Subgroup) 5](#_Toc367419223)

[3 Big Data Requirements (from Requirements & SecNPrivacy Subgroups) 6](#_Toc367419224)

[4 Big Data Reference Architecture (from RA Subgroup) 8](#_Toc367419225)

[5 Big Data Security and Privacy (from SecNPrivacy Subgroup) 9](#_Toc367419226)

[6 Features and Technology Readiness 9](#_Toc367419227)

[6.1 Technology Readiness 9](#_Toc367419228)

[6.1.1 Types of Readiness 9](#_Toc367419229)

[6.1.2 Scale of Technological Readiness 9](#_Toc367419230)

[6.2 Organizational Readiness and Adoption 10](#_Toc367419231)

[6.2.1 Types of Readiness 10](#_Toc367419232)

[6.2.2 Scale of Organizational Readiness 11](#_Toc367419233)

[6.2.3 Scale of Organizational Adoption 11](#_Toc367419234)

[6.3 Features Summary 12](#_Toc367419235)

[6.4 Feature 1: Storage Framework 16](#_Toc367419236)

[6.4.1 Physical Storage Frameworks 17](#_Toc367419237)

[6.4.2 Logical Data Distribution 19](#_Toc367419238)

[6.5 Feature 2: Processing Framework 29](#_Toc367419239)

[6.6 Feature 3: Resource Managers Framework 29](#_Toc367419240)

[6.7 Feature 4: Infrastructure Framework 30](#_Toc367419241)

[6.8 Feature 5: Information Framework 30](#_Toc367419242)

[6.9 Feature 6: Standards Integration Framework 30](#_Toc367419243)

[6.10 Feature 7: Application Framework 30](#_Toc367419244)

[6.11 Feature 8: Business Operations 31](#_Toc367419245)

[6.12 Feature 9: Business Intelligence 31](#_Toc367419246)

[7 Big Data Related Multi-stakeholder Collaborative Initiatives 31](#_Toc367419247)

[7.1.1 Characteristics supported by standards 32](#_Toc367419248)

[7.1.2 Information and Communications Technologies (IT) Standards Life Cycle 33](#_Toc367419249)

[7.2 Data Service Abstraction 33](#_Toc367419250)

[7.2.1 Data Provider Registry and Location service 33](#_Toc367419251)

[7.2.2 Data Provider Interfaces 35](#_Toc367419252)

[7.2.3 Data Sources 35](#_Toc367419253)

[7.3 Usage Service Abstraction 38](#_Toc367419254)

[7.4 Capability Service Abstraction 38](#_Toc367419255)

[7.4.1 Security and Privacy Management 38](#_Toc367419256)

[7.4.2 System Management 38](#_Toc367419257)

[7.5 Standards Summary 39](#_Toc367419258)

[8 Big Data Strategies 39](#_Toc367419259)

[8.1 Strategy of Adoption 39](#_Toc367419260)

[8.1.1 Identify and include stakeholders 39](#_Toc367419261)

[8.1.2 Identify potential road blocks 39](#_Toc367419262)

[8.1.3 Define Achievable Goals 40](#_Toc367419263)

[8.1.4 Define “Finished” and “Success” at the beginning of the project 40](#_Toc367419264)

[8.2 Strategy of Implementation 41](#_Toc367419265)

[8.3 Resourcing 43](#_Toc367419266)

[9 Concerns and Assumptions Statement 44](#_Toc367419267)

[Appendix A: Industry Information 44](#_Toc367419268)

# Executive Summary

Provide executive level overview of the Technology Roadmap, introduce the vision of the document.

Author: Carl

[Content Goes Here]

# Purpose, Background, and Vision

## NIST Technology Roadmap Purpose

What are we trying to accomplish with this document. *From Charter*: The focus of the NIST Big Data Working Group (NBD-WG) is to form a community of interest from industry, academia, and government, with the goal of developing a consensus definitions, taxonomies, reference architectures, and technology roadmap. The focus of the NBD-WG Technology Roadmap Subgroup is to form a community of interest from industry, academia, and government, with the goal of developing a consensus vision with recommendations on how Big Data should move forward by performing a good gap analysis through the materials gathered from all other NBD subgroups. This includes setting standardization and adoption priorities through an understanding of what standards are available or under development as part of the recommendations.

Author: Carl

[Content Goes Here]

## Big Data Background

An introduction to the state of Big Data in terms of capabilities and features, not focused on products or individual technologies. This could be where we include other initiatives that are going on within industry, Government, and Academic realms. 2-3 Paragraphs will be fine

Author: Dave

[Content Goes Here]

## NIST Big Data Technology Roadmap Stakeholders

Who should read this Tech Roadmap, what should they plan to takeaway from reading this document. Define stakeholders and include a stakeholder matrix that relates to the remaining sections of this document. This should likely also include a RACI matrix (RACI == Dan’s section)

Author: Carl

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Executive Stakeholders | Technical Architects and Managers | Quantitative Roles | Application Development | Systems Operation  And Administration |
| Organizational Adoption and Business Strategy | R | A | C | C | I |
| Infrastructure and Architecture | I | R | C | A | A |
| Complex analytics, reporting, and business intelligence | C | A | R | A | I |
| Programming paradigms and information management | I | A | C | R | A |
| Deployment, administration, and maintenance | I | A | C | A | R |

## Guiding Principles for Developing the NIST Big Data Technology Roadmap

Author: Carl

This document was developed based on the following guiding principles.

* Technologically Agnostic
* Audience of Industry, Government, and Academia

# NIST Big Data Definitions and Taxonomies (from Def. & Tax. Subgroup)

Author: Get from subgroups

[Content Goes Here]

# Big Data Requirements (from Requirements & SecNPrivacy Subgroups)

Author: Get from subgroups

<Need Intro Paragraph>

**Government Operation**

1. Census 2010 and 2000 – Title 13 Big Data; Vivek Navale & Quyen Nguyen, NARA
2. National Archives and Records Administration Accession NARA, Search, Retrieve, Preservation; Vivek Navale & Quyen Nguyen, NARA

**Commercial**

1. Cloud Eco-System, for Financial Industries (Banking, Securities & Investments, Insurance) transacting business within the United States; Pw Carey, Compliance Partners, LLC
2. Mendeley – An International Network of Research; William Gunn , Mendeley
3. Netflix Movie Service; Geoffrey Fox, Indiana University
4. Web Search; Geoffrey Fox, Indiana University
5. IaaS (Infrastructure as a Service) Big Data Business Continuity & Disaster Recovery (BC/DR) Within A Cloud Eco-System; Pw Carey, Compliance Partners, LLC
6. Cargo Shipping; William Miller, MaCT USA
7. Materials Data for Manufacturing; John Rumble, R&R Data Services
8. Simulation driven Materials Genomics; David Skinner, LBNL

**Healthcare and Life Sciences**

1. Electronic Medical Record (EMR) Data; Shaun Grannis, Indiana University
2. Pathology Imaging/digital pathology; Fusheng Wang, Emory University
3. Computational Bioimaging; David Skinner, Joaquin Correa, Daniela Ushizima, Joerg Meyer, LBNL
4. Genomic Measurements; Justin Zook, NIST
5. Comparative analysis for metagenomes and genomes; Ernest Szeto, LBNL (Joint Genome Institute)
6. Individualized Diabetes Management; Ying Ding , Indiana University
7. Statistical Relational Artificial Intelligence for Health Care; Sriraam Natarajan, Indiana University
8. World Population Scale Epidemiological Study; Madhav Marathe, Stephen Eubank or Chris Barrett, Virginia Tech
9. Social Contagion Modeling for Planning, Public Health and Disaster Management; Madhav Marathe or Chris Kuhlman, Virginia Tech
10. Biodiversity and LifeWatch; Wouter Los, Yuri Demchenko, University of Amsterdam

**Deep Learning and Social Media**

1. Large-scale Deep Learning; Adam Coates , Stanford University
2. Organizing large-scale, unstructured collections of consumer photos; David Crandall, Indiana University
3. Truthy: Information diffusion research from Twitter Data; Filippo Menczer, Alessandro Flammini, Emilio Ferrara, Indiana University
4. CINET: Cyberinfrastructure for Network (Graph) Science and Analytics; Madhav Marathe or Keith Bisset, Virginia Tech
5. NIST Information Access Division analytic technology performance measurement, evaluations, and standards; John Garofolo, NIST

**The Ecosystem for Research**

1. DataNet Federation Consortium DFC; Reagan Moore, University of North Carolina at Chapel Hill
2. The ‘Discinnet process’, metadata <-> big data global experiment; P. Journeau, Discinnet Labs
3. Semantic Graph-search on Scientific Chemical and Text-based Data; Talapady Bhat, NIST
4. Light source beamlines; Eli Dart, LBNL

**Astronomy and Physics**

1. Catalina Real-Time Transient Survey (CRTS): a digital, panoramic, synoptic sky survey; S. G. Djorgovski, Caltech
2. DOE Extreme Data from Cosmological Sky Survey and Simulations; Salman Habib, Argonne National Laboratory; Andrew Connolly, University of Washington
3. Particle Physics: Analysis of LHC Large Hadron Collider Data: Discovery of Higgs particle; Geoffrey Fox, Indiana University; Eli Dart, LBNL

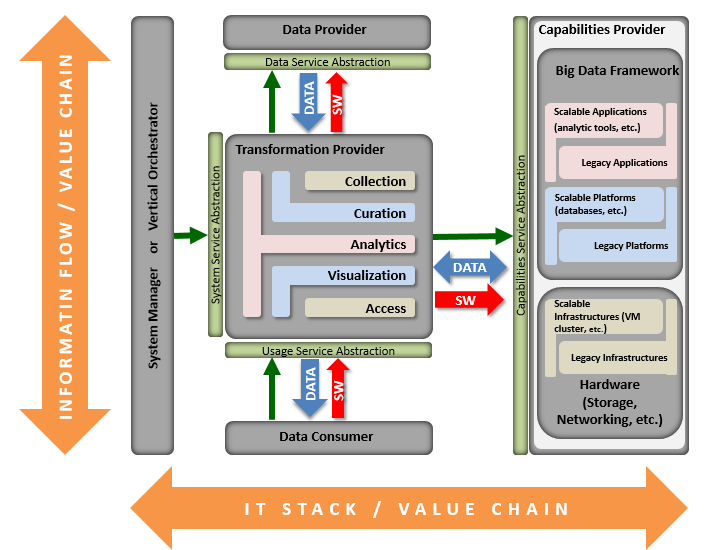
**Earth, Environmental and Polar Science**

1. EISCAT 3D incoherent scatter radar system; Yin Chen, Cardiff University; Ingemar Häggström, Ingrid Mann, Craig Heinselman, EISCAT Science Association
2. ENVRI, Common Operations of Environmental Research Infrastructure; Yin Chen, Cardiff University
3. Radar Data Analysis for CReSIS Remote Sensing of Ice Sheets; Geoffrey Fox, Indiana University
4. UAVSAR Data Processing, Data Product Delivery, and Data Services; Andrea Donnellan and Jay Parker, NASA JPL
5. NASA LARC/GSFC iRODS Federation Testbed; Brandi Quam, NASA Langley Research Center
6. MERRA Analytic Services MERRA/AS; John L. Schnase & Daniel Q. Duffy , NASA Goddard Space Flight Center
7. Atmospheric Turbulence - Event Discovery and Predictive Analytics; Michael Seablom, NASA HQ
8. Climate Studies using the Community Earth System Model at DOE’s NERSC center; Warren Washington, NCAR
9. DOE-BER Subsurface Biogeochemistry Scientific Focus Area; Deb Agarwal, LBNL
10. DOE-BER AmeriFlux and FLUXNET Networks; Deb Agarwal, LBNL

# Big Data Reference Architecture (from RA Subgroup)

Author: Get from subgroups

<Need Intro Paragraph>



# Big Data Security and Privacy (from SecNPrivacy Subgroup)

Author: Get from subgroups

[Content Goes Here]

# Features and Technology Readiness

## Technology Readiness

Author: Dan

The technological readiness for Big Data serves as metric useful in assessing both the overall maturity of a technology across all implementers as well as the readiness of a technology for broad use within an organization. Technology readiness evaluates readiness types in a manner similar to that of technology readiness in Service-Oriented Architectures (SOA). However, the scale of readiness is adapted to better mimic the growth of open source technologies, notably those which follow models similar to the Apache Software Foundation (ASF). Figure 1 provides a superimposition of the readiness scale on a widely recognized "hype curve." This ensures that organizations which have successfully evaluated and adopted aspects of SOA can apply similar processes to assessing and deploying Big Data technologies.

### Types of Readiness

* **Architecture**: Capabilities concerning the overall architecture of the technology and some parts of the underlying infrastructure
* **Deployment**: Capabilities concerning the architecture realization infrastructure deployment, and tools
* **Information**: Capabilities concerning information management: data models, message formats, master data management, etc.
* **Operations, Administration and Management**: Capabilities concerning post-deployment management and administration of the technology

### Scale of Technological Readiness

1. **Emerging**

* Technology is largely still in research and development
* Access is limited to the developers of the technology
* Research is largely being conducted within academic or commercial laboratories
* Scalability of the technology is not assessed

1. **Incubating**

* Technology is functional outside laboratory environments
* Builds may be unstable
* Release cycles are rapid
* Documentation is sparse or rapidly evolving
* Scalability of the technology is demonstrated but not widely applied

1. **Reference Implementation**

* One or more reference implementations are available
* Reference implementations are usable at scale
* The technology may have limited adoption outside of its core development community
* Documentation is available and mainly accurate

1. **Emerging Adoption**

* Wider adoption beyond the core community of developers
* Proven in a range of applications and environments
* Significant training and documentation is available

1. **Evolving**

* Enhancement-specific implementations may be available
* Tool suites are available to ease interaction with the technology
* The technology competes with others for market share

1. **Standardized**

* Draft standards are in place
* Mature processes exist for implementation
* Best practices are defined

## Organizational Readiness and Adoption

Technological readiness is useful for assessing the maturity of the technology components which make up Big Data implementations. However, successful utilization of Big Data technologies within an organization strongly benefits from an assessment of both the readiness of the organization and its level of adoption with respect to Big Data technologies. As with the domains and measures for the Technology Readiness scale, we choose definitions similar to those used for SOA.

### Types of Readiness

#### Organizational Readiness Domains

* **Business and Strategy:** Capabilities that provide organizational constructs necessary for Big Data initiatives to succeed. These include a clear and compelling business motivation for adopting Big Data technologies, expected benefits, funding models etc.
* **Governance:** The readiness of governance policies and processes to be applied to the technologies adopted as part of a Big Data initiative. Additionally, readiness of governance policies and processes for application to the data managed and operated on as part of a Big Data initiative.
* **Projects, Portfolios, and Services:** Readiness with respect to the planning and implementation of Big Data efforts. Readiness extends to quality and integration of data, as well as readiness for planning and usage of Big Data technology solutions.
* **Organization:** Competence and skills development within an organization regarding the use and management of Big Data technologies. This includes, but is not limited to, readiness within IT departments (e.g., service delivery, security, and infrastructure) and analyst groups (e.g. methodologies, integration strategies, etc.).

### Scale of Organizational Readiness

1. **No Big Data**

* No awareness or efforts around Big Data exist in the organization

1. **Ad Hoc**

* Awareness of Big Data exists
* Some groups are building solutions
* No Big Data plan is being followed

1. **Opportunistic**

* An approach to building Big Data solutions is being determined
* The approach is opportunistically applied, but is not widely accepted or adopted within the organization

1. **Systematic**

* The organizational approach to Big Data has been reviewed and accepted by multiple affected parties.
* The approach is repeatable throughout the organization and nearly-always followed.

1. **Managed**

* Metrics have been defined and are routinely collected for Big Data projects
* Defined metrics are routinely assessed and provide insight into the effectiveness of Big Data projects

1. **Optimized**

* Metrics are always gathered and assessed to incrementally improve Big Data capabilities within the organization.
* Guidelines and assets are maintained to ensure relevancy and correctness

### Scale of Organizational Adoption

1. **No Adoption**

* No current adoption of Big Data technologies within the organization

1. **Project**

* Individual projects implement Big Data technologies as they are appropriate

1. **Program**

* A small group of projects share an implementation of Big Data technologies
* The group of projects share a single management structure and are smaller than a business unit

1. **Divisional**

* Big Data technologies are implemented consistently across a business unit

1. **Cross-Divisional**

* Big Data technologies are consistently implemented by multiple divisions with a common approach
* Big Data technologies across divisions are at an organizational readiness level of Systematic or higher

1. **Enterprise**

* Big Data technologies are implemented consistently across the enterprise
* Organizational readiness is at level of Systematic or higher

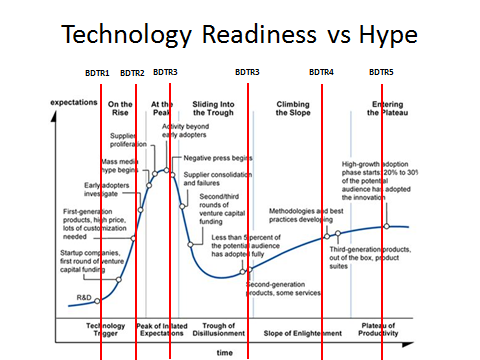


Figure 1 Technology Readiness levels visualized along Gartner's "hype curve."

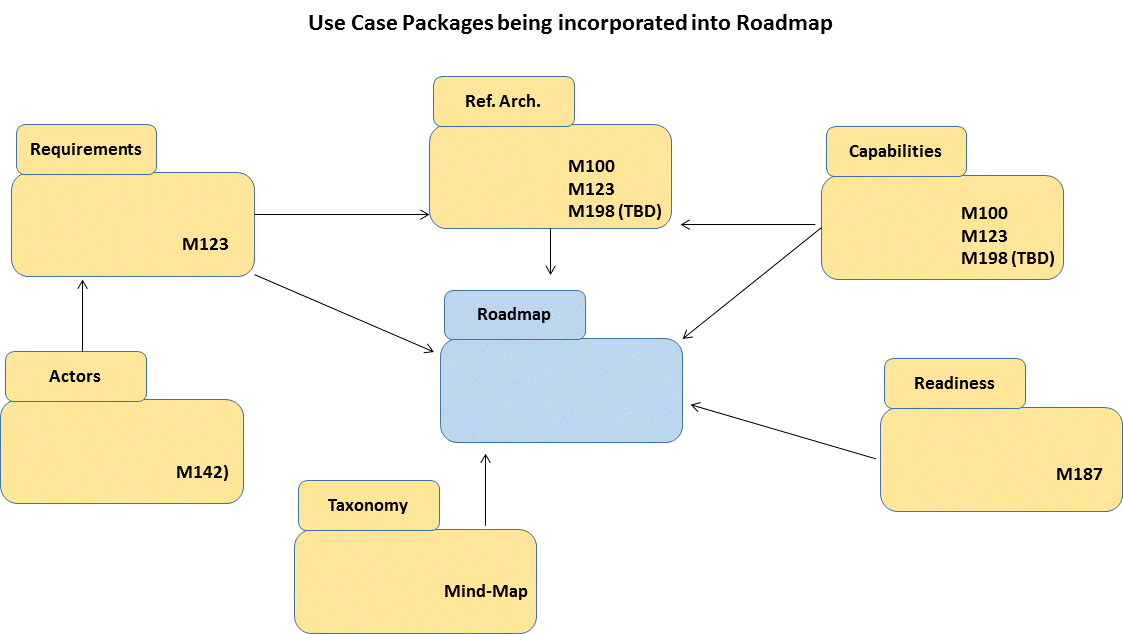
## Features Summary

Author: Bruno

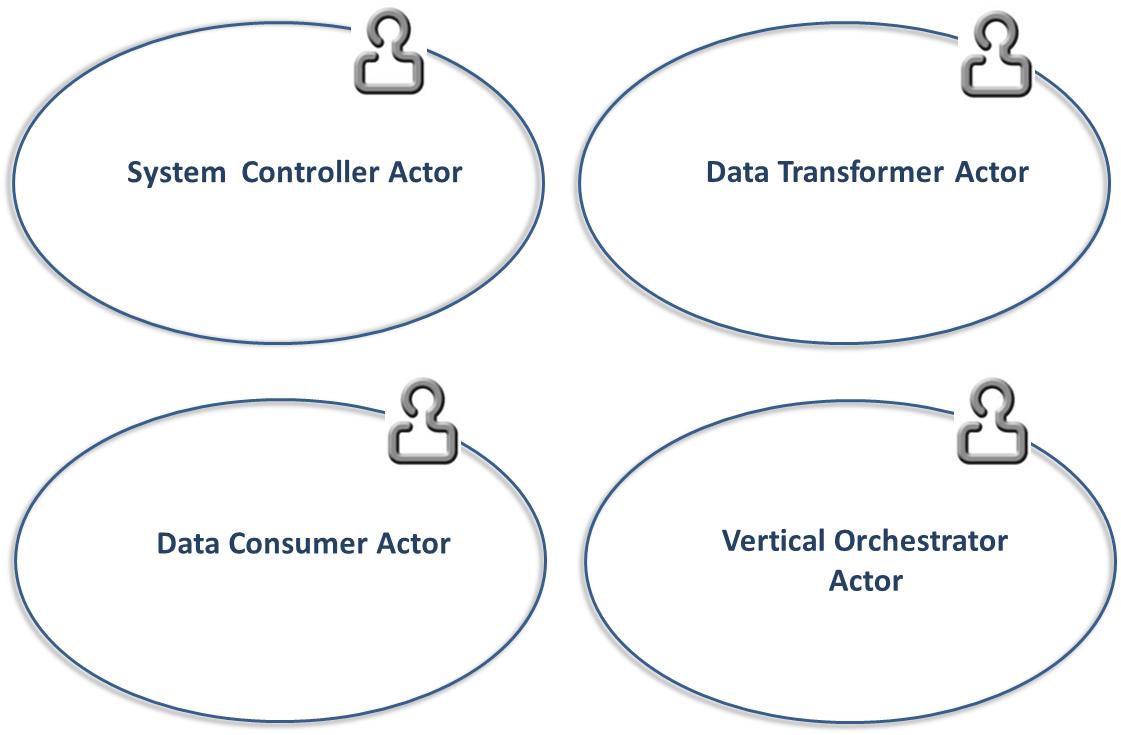
The NIST Roadmap is composed of features that outline the current and future state of Big Data. These features consists of four primary categories: 1) Data Services, 2) Usage Services, 3) Capabilities, and, 4) Vertical Orchestrator. These four categories also mirror the respective NIST Big Data Reference Architecture and Taxonomy. Within these two categories are the Top Nine Features.

What is the importance of having the Roadmap portray features? The ability for technical and business stakeholders to view the current and future state of features enables such individuals to better make decisions in using Big Data. The three ‘dials’ of technology, problems, and, end users is that all three are in constant change. In addition, they all interplay with one another. When one moves – the other are effected. Leaders must then pivot. The NIST Big Data Roadmap provides the best available snapshot in time of this moving target.

The reasoning of how the NIST Working Group arrived at these four categories and nine features is primarily due to the rationalizing of the Big Data landscape. The four categories were arrived at by aligning to the Reference Architecture. The nine Roadmap features were an output from the mind-share of the Reference Architecture, Requirements [Use Cases], [architectural] Capabilities, Actors, Taxonomy, and, [Big Data] Readiness. The below diagram displays the inputs and relationships between the NIST artifacts and the Roadmap.



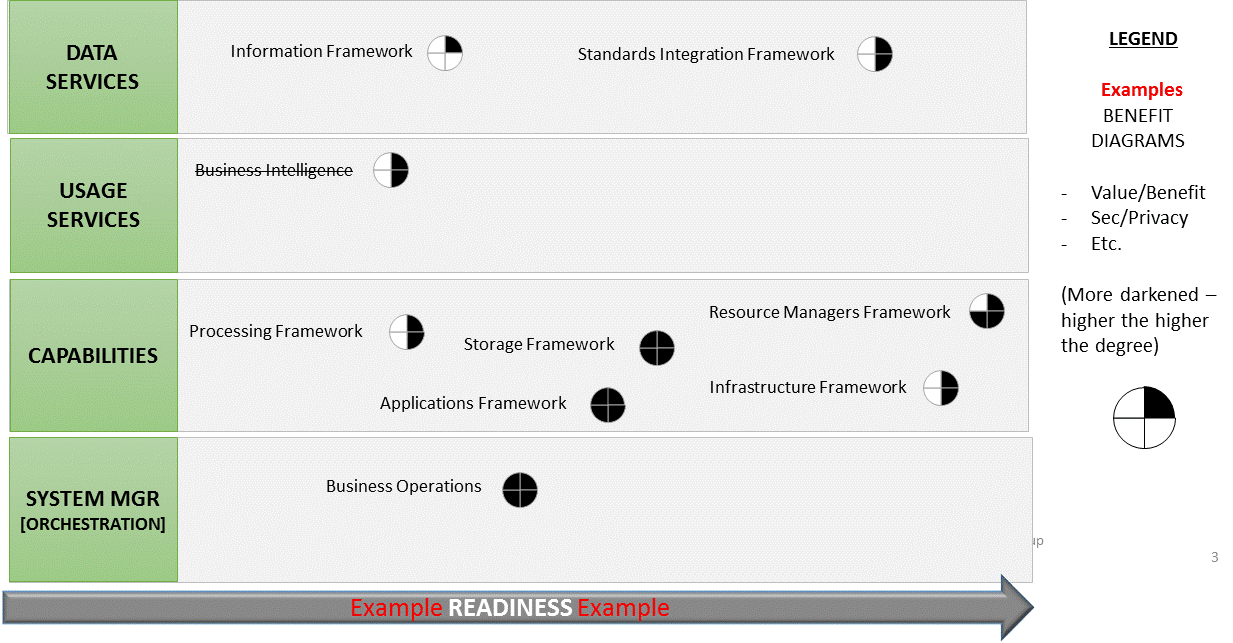
Realizing the centricity of the end-user is critical, the Roadmap has inherited the four actor groups from the NIST Big Data artifacts. These four abstract actors are: 1) System Controller, 2) Data Transformation, 3) Data Consumer, and, 4) Vertical Orchestrator (see diagram below). Very similar to Unified Modeling Language (UML) standards, the Actors are abstract and can be not only individuals but also systems. Actors can also fulfill more than just one role.



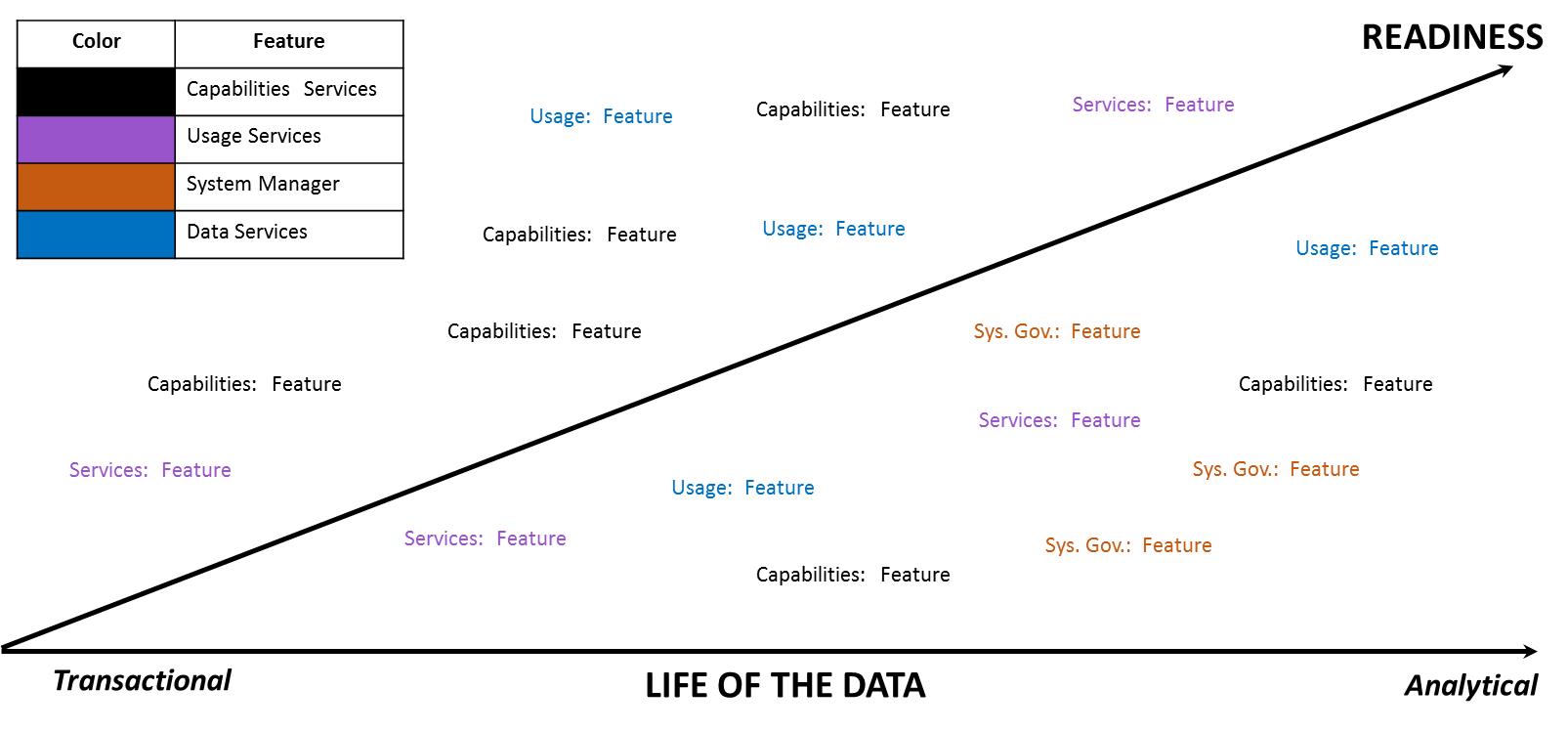
The below tables outlines the value statements for each of the Big Data features. In addition, the NIST Working Group has mapped each feature to technology and organizational readiness.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Value Statement | Roles | Readiness | RA Mapping |
| Storage Framework | Storage Framework defines how big data is logically organized, distributed, and stored. The volume and velocity of big data frequently means that traditional (file systems, RDBMS) solutions will not hold up to one or both of these attributes. | TBD | TBD | Capabilities |
| Processing Framework | Processing Frameworks defines how data is operated on in the big data environment. The volume or velocity of big data often means that analysis of the data requires more resources (memory, cpu cycles) than are available on a single compute node and that the processing of the data must be distributed and coordinated across many nodes. | TBD | TBD | Capabilities |
| Resource Managers Framework | Because many big data storage and processing Frameworks are distributed and no single storage and processing solution may meet the needs of the end user, resource management solutions are required to manage and allocate resources across disparate frameworks. | TBD | TBD | Capabilities |
| Infrastructure Architecture | Big Data requires the ability to operate with sufficient network and infrastructure backbone. For Big Data to deploy, it is critical that the Infrastructure Framework has been right-sized. | TBD | TBD | Capabilities |
| Information Architecture | Prior to any Big Data decision, the data itself needs to be reviewed for its informational value. | TBD | TBD | Data Services |
| Standards Integration Framework | Integration with appropriate standards (de jure, consortia, reference implementation, open source implementation, etc.) can assist both in cross-product integration and cross-product knowledge. Identifying which standards efforts address architectural requirements and which requirements are not currently being addressed provides input for future standards efforts. | TBD | TBD | Data Services |
| Applications Framework | The building blocks of applications are data. The Application Lifecycle Management needs to take into consideration how applications will interact with a Big Data solution. | TBD | TBD | Capabilities |
| Business Operations | Big Data is more than just technology, but also a cultural and organizational transformation. Business Operations need to be able to strategize, deploy, adopt, and, operate Big Data solutions. | TBD | TBD | Vertical Orchestrator |
| Business Intelligence | The presenting of data into information, intelligence, and, insight is typically the end value of Big Data. | TBD | TBD | [Will be moved under ‘Applications’] |

The top features of the Roadmap can also be viewed visually. Two examples of displaying the NIST Big Data Roadmap and relevant decision making criteria are provided below. The goal of these diagrams is to enable decision-makers to see as best ‘around-the-corner’ as possible for their Big Data discussions. Alignment with the other NIST Big Data artifacts have taken place, providing the reader with the ability to toggle to other documents for reference and discussion drill-down. The below diagram groups Roadmap features into the four category ‘swimlanes’, with examples of the features spanning readiness. The pie charts are to provide either the level of value, security, or, privacy.



The second diagram displaying the Roadmap visually (below) is a scatter diagram with an X and Y axis. It provides an ‘X’ axis showing the life of the data. The ‘Y’ axis provides the level of readiness. The Roadmap features are shown by the color of the one of the four categories. They are placed in the diagram as to where they sit in terms of the longitudinal use of the data – transactional to analytical – and the level of their organizational and architectural readiness for adoption. The below diagram lands these features below as purely as examples.



In conclusion, the Roadmap outlines the future of Big Data with the best available for-sight as possible. It captures the relation of features to decision-making criteria, such as: readiness, value, security, and longitudinal use of data. Additional value-add is the incorporation of the Roadmap with other NIST Big Data artifacts.

## Feature 1: Storage Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Dave

*Storage Framework defines how big data is logically organized, distributed, and stored. The volume and velocity of big data frequently means that traditional (file systems, RDBMS) solutions will not hold up to one or both of these attributes.*

Storage Frameworks have and continue to undergo dramatic changes as the need to capture, store, process, and archive ever-larger amounts of data. The nature of this increase covers both the total volume of data artifacts - 9100 Tweets per second (Twitter Statistics, 2013) to the extreme size of individual artifacts – 110 megapixel images from the Vigilant Stare Wide Area Persistent Surveillance (WAPS) Platform (Persistence On Patrol, 2013). In order to support the processing of data across this continuum unique and novel approaches are frequently required to support not just the storage but the indexing for access/processing but also the preservation/backup and transfer of the data.

Typically storage frameworks consist of two interdependent aspects, the physical organization of the data on the media, and the logical organization of the data within the physical layout. For the purposes of this roadmap we are specifically focusing on persistent storage approaches but it should be noted that most of these approaches could be implemented in non-persistent storage (e.g. RAM disk). In addition, we are not going to delve into the physical media itself. While today, the primary media are traditional spinning magnetic disks and Solid State Disks built on flash memory technology new storage media technologies such as holographic, quantum, and nano-bubble storage are under development and maturing rapidly. In addition the density of both current prevalent data storage technologies and emerging technologies continue to increase roughly along Moore’s Law. From a big data perspective this generally means that more data can be stored in a smaller footprint and that in general (especially where mechanical processes like disk head seeks are involved) access times will continue to decrease and throughput increase. As these new media types mature and their performance characteristics are understood they should provide drop in replacements for existing storage media.

### Physical Storage Frameworks

The physical organization of storage generally follows the continuum from local to distributed as shown in figure 6.4-1 below with their associated technology readiness indicated by the number in the circle.



There are two aspects of these technologies that directly influence their suitability for Big Data solutions. First there is capacity (dealing with the volume of the data). Local disks/filesystems are specifically limited by the size of the available media. HW/SW Raid solutions (in this case local to a processing node) help that scaling by allowing multiple pieces of media to be treated as a single device. That approach however is limited by the physical dimension of the media and the number of devices the node can accept. SAN and NAS implementations (often known as shared disk solutions) remove that limit by consolidating storage into a storage specific device. However, they start to run into the second influencing factor which is transfer bandwidth. While both network and I/O interfaces are getting faster and many implementations support multiple transfer channels I/O bandwidth can be a limiting factor. In addition, despite the redundancies provided by RAID, Hot Spares, multiple power supplies, and multiple controllers these boxes can often become I/O bottle necks or single points of failure in an enterprise. Distributed Filesystems (also known as cluster file systems) seek to overcome these issues by combining I/O throughput through multiple devices(spindles) on each node with redundancy and failover by mirroring/replicating data at the block level across multiple nodes. This specifically, is designed to allow the use of heterogeneous commodity hardware across the big data cluster. Thus, if a single drive or an entire node should fail no data is lost because it is replicated on other nodes and throughput is only minimally impacted as that processing can be moved to those other nodes. In addition, replication allows for high levels of concurrency for reading data and for initial writes. Updates and transaction style changes tend to be an issue for many distributed file systems because latency in creating replicated blocks will create consistency issues (e.g. a block is changed but another node reads the old data before it is replicated).

Unlike the other technologies described here which implement a traditional filesystem approach, Distributed Object Stores (sometimes called Global Object Stores) present a flat name space with a global id (GUID) for any given chunk of data. Data in the store is located generally through a query against a meta-data catalog that returns the associated GUID(s). The underlying implementation of the software generally knows from the GUID where that particular chunk of data is stored. These object stores are being developed and marketed for storage of very large data objects from complete data sets (which Amazon’s S3 exposes) to large individual objects (high resolution images in the 10s of GB size range). The biggest limitation of these stores for big data problems tends to be network throughput that can be a limiting factor since many require the object to be accessed in total. Future trends however point to the concept of being able to send the computation/application to the data versus needing to bring the data to the application.

From a maturity perspective the two key areas where distributed file systems would be expected to improve are on random write I/O performance and consistency and the generation of at least IETF RFC level standards such as are available today for NFS. Distributed object stores while currently available and operational (Amazon S3) and part of the roadmap for large organizations such as the National Geopspatial Intelligence Agency (NGA) currently are essentially stove pipe/proprietary implementations. In order for them to become prevalent within Big Data ecosystems there will need to be some level of interoperability available (through standardized APIs), standards based approaches for data discovery, and probably most importantly standards based approaches that allow the application to be transferred over the grid and run local to the data versus the need for the data to be transferred to the application.

### Logical Data Distribution

In most aspects, the logical distribution/organization of data in big data storage frameworks mirrors what is common for most legacy systems. Figure 6.4.2 below shows a brief overview of data organizations approaches for big data.

As mentioned above many Big Data logical storage organizations leverage the common filesystem concept (where chunks of data are organized into a hierarchical namespace of directories) as their base and then implement various indexing methods within the individual files. This allows many of these approaches to be run both on simple local storage file systems for testing purposes or on fully distributed file systems for scale.

#### File Systems

Many big data processing frameworks and applications are content to access their data directly from an underlying file systems. In almost all cases, the file systems implement some level of the POSIX standards for permissions and the associated file operations. This allows other higher level frameworks for Indexing or Processing to operate relatively transparent to whether the underlying file system is local or fully distributed. Within, files systems there is really nothing new or novel in the storage of data. It can be text or binary data, fixed length records, or some sort of delimited structure (e.g. comma separated values, XML). Several of these file system implementations also support data compression and encyrption at various levels. The one major caveat to this this is that for distributed block based file systems the compression/encryption must be splittable and allow any given block to be decompressed/decrypted out of sequence and without access to the other blocks. For record oriented storage (delimited or fixed length) this generally is not a problem unless individual records can exceed a block size. Some distributed file system implementations provide compression at the volume or directory level and implement it below the logical block level (e.g. when a block is read from the file system it is decompressed/decrypted before being returned). Because of it’s simplicity, familiarity, and portability delimited files are frequently the default storage format in many big data implementations. The tradeoff for this is IO efficiency. While individual blocks in a distributed file system might be accessed in parallel, each block still needs to be read in sequence. In the case of a delimited file if you are interested in only the last field of certain records with maybe 100s of fields that is a lot of wasted IO and processing bandwidth.

Binary formats tend to be application or implementation specific. While they can offer much more efficient access both due to smaller data sizes (integers are 2-4 bytes in binary while they are 1byte per digit in ASCII) they offer limited portability between different implementations. At least one popular distributed file system provides its own standard binary format which at least allows data to be portable between multiple applications without additional software. That said the bulk of the indexed data organization approaches discussed below leverage binary formats for efficiency.

#### Indexed Storage Organization

The very nature of big data (volume and velocity primarily) practically drive requirements to some form of indexing structure. The volume requires that specific elements of data can be locate quickly without scanning across the entire dataset. The velocity, also requires that data can be located quickly either for matching (e.g. does any incoming data match something in my existing data set) or to know where to write/update new data.

The choice of a particular indexing method or methods depends mostly on your data and the nature of the application you are trying to implement. For example, graph data (vertexes, edges, and properties) can be easily represented in flat text files as vertex,edge pairs, edge, vertex,vertex triples, or vertex, edge list records. However, processing this data efficiently would require potentially loading the entire data set into memory or being able to distribute the application and data set across multiple nodes so a portion of the graph is in memory on each node. That then of course would require the nodes to communicate when graph sections have vertexes that connect with vertexes on other processing nodes. This is perfectly acceptable for some graph applications such as shortest path especially when the graph is static. And in fact some graph processing frameworks operate using this exact model. However, if the graph is dynamic or you need to quickly search or match to a portion of the graph then this approach becomes infeasible for large scale graphs requiring a specialized graph storage framework.

The indexing approaches described below tend to be classified by the features provided typically in the implementation specifically - the complexity of the data structures that can be stored, how well they can process links between data, and how easily they support multiple access patterns as shown in Figure 6.2.2.2-1 below. Since any of these features can be implemented in custom application code the values portrayed represent approximant norms. Key-Value stores for example work really real for data that is only accessed through a single key, whose values can be expressed in a single flat structure and where multiple records do not need to be related. Document stores, can support very complex structures of arbitrary width and tend to be indexed for access via multiple document properties but do not tend to support inter-record relationships well. In reality, the specific implementations for each storage approach vary significantly enough that all of the values for the features represented are really ranges. For example, relational data storage implementations are supporting increasingly complex data structures and there is work going on to add more flexible access patterns natively in bigtable columnar implementations. Within big data the performance of each of these features tends to drive the scalability of that approach depending on the problem being solved. For example, if my problem is to locate a single piece of data for a unique key then Key-value stores will scale really well. If on the other hand my problem requires general navigation of the relationships between multiple data records then a graph storage model will likely provide the best performance.

The following paragraphs will describe several indexing approaches for storing big data and the advantages and issues commonly found for each one.

##### Relational Storage Models

This model is perhaps the most familiar to folks as the basic concept has existed since the 1950s and the Structured Query Language (SQL) is a mature standard for manipulating (search, insert, update, delete) relational data. In the relational model, data is stored as rows with each field representing a column organized into Table based on the logical data organization. The problem with relational storage models and big data is the join between one or more tables. While the size of 2 or more tables of data individually might be small the join (or relational matches) between those tables will generate exponentially more records. The appeal of this model for organizations just adopting big data is its familiarity. The pit falls are some of the limitations and more importantly the tendency to adopt standard RDBMS practices (high normalization, detailed and specific indexes) and performance expectations

Big data implementations of relational storage models are relatively mature and have been adopted by a number of organizations. They are also maturing very rapidly with new implementations focusing on improved response time. Many big data implementations take a brute force approach to scaling relational queries. Essentially, queries are broken into stages but more importantly processing of the input tables is distributed across multiple nodes (often as a map reduce job). The actual storage of the data can be flat files (delimited or fixed length) where each record/line in the file represents a row in a table. Increasingly however these implementations are adopting binary storage formats optimized for distributed file systems. These formats will often use block level indexes and column oriented organization of the data to allow individual fields to be accessed in records without needing to read the entire record. Despite this most Big Data Relational storage models are still “batch oriented” systems designed for very complex queries which generate very large intermediate cross-product matrices from joins so even the simplest query can required 10s of seconds to complete. There is significant work going on and emerging implementations that are seeking to provide a more interactive response and interface.

Early implementations only provided limited data types and little or no support for indexes. However, most current implementations have support for complex data structures and basic indexes. However, while the query planners/optimizers for most modern RDBMS systems are very mature and implement cost-based optimization through statistics on the data the query planners/optimizers in many big data implementations remain fairly simple and rule-based in nature. While for batch oriented systems this generally acceptable (since the scale of processing the big data in general can be orders of magnitude more an impact) any attempt to provide interactive response will need very advanced optimizations so that (at least for queries) only the most likely data to be returned is actually searched. This of course leads to the single most serious draw back with many of these implementations. Since distributed processing and storage are essential for achieving scalability these implementations are directly limited by the CAP (Consistency, Availability, and Partition Tolerance) theorem. Many in fact provide what is generally referred to a t-eventual consistency which means that barring any updates to a piece of data all nodes in the distributed system will eventually return the most recent value. This level of consistency is typically fine for Data Warehousing applications where data is infrequently updated and updates are generally done in bulk. However, transaction oriented databases typically require some level of ACID (Atomicity, Consistency, Isolation, Durability) compliance to insure that all transactions are handled reliably and conflicts are resolved in a consistent manner. There are a number of both industry and open source initiatives looking to bring this type of capability to Big Data relational storage frameworks. One approach is to essentially layer a traditional RDBMS on top of an existing distributed file system implementation. While vendors claim that this approach means that the overall technology is mature a great deal of research and implementation experience is needed before the complete performance characteristics of these implementations are known.

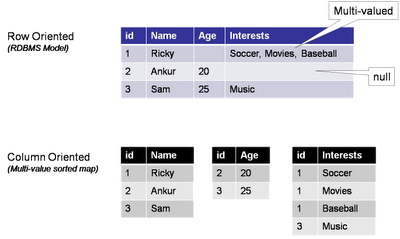
##### Key-Value Storage Models

Key-value stores are one of the oldest and mature data indexing models. In fact, the principles of key value stores underpin all the other storage and indexing models. From a Big Data perspective, these stores effectively represent random access memory models. While the data stored in the values can be arbitrarily complex in structure all the handling of that complexity must be provided by the application with the storage implementation often providing back just a pointer to a block of data. Key-Value stores also tend to work best for 1-1 relationships (e.g. each key relates to a single value) but can also be effective for keys mapping to lists of homogeneous values. When keys map multiple values of heterogeneous types/structures or when values from one key need to be joined against values for a different or the same key then custom application logic is required. It is the requirement for this custom logic that often prevents Key-Value stores from scaling effectively for certain problems. However, depending on the problem certain processing architectures can make effective use of distributed key-value stores. Key-value stores generally deal well with updates when the mapping is one to one and the size/length of the value data does not change. The ability of key-value stores to handle inserts is generally dependent on the underlying implementation. Key-value stores also generally require significant effort (either manual or computational) to deal with changes to the underlying data structure of the values.

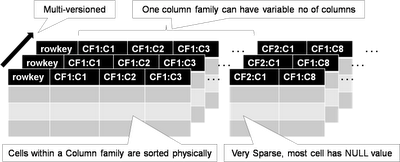
Distributed key-value stores are the most frequent implementation utilized in big data applications. One problem that must always be addressed (but is not unique to key-value implementations) is the distribution of keys across over the space of possible key values. Specifically, keys must be chosen carefully to avoid skew in the distribution of the data across the cluster. When data is heavily skewed to a small range it can result in computation hot spots across the cluster if the implementation is attempting to optimize data locality. If the data is dynamic (new keys being added) for such an implementation then it is likely that at some point the data will require rebalancing across the cluster. Non-locality optimizing implementations employ various sorts of hashing, random, or round-robin approaches to data distribution and don’t tend to suffer from skew and hot spots. However, they perform especially poorly on problems requiring aggregation across the data set.

##### Columnar Storage Models

Much of the hype associated with Big Data came with the publication of the Big Table paper by Google in 2006 (Chang, et al., 2006) but column oriented storage models like Bigtable are not new to even big data and have been stalwarts of the data warehousing domain for many years. Unlike traditional relational data that store data by rows of related values, columnar stores organize data in groups of like values. The difference here is subtle but in relational databases an entire group of columns are tied to some primary-key (frequently one or more of the columns) to create a record. In columnar, the value of every column is a key and like column values point to the associated rows. The simpliest instance of a columnar store is little more than a key-value store with the key and value roles reversed. In many ways, columnar data stores look very similar to indexes in relational databases. Figure 6.4.2.2.2-1 below shows the basic differences between row oriented and column oriented stores.



In addition, implementations of columnar stores that follow the Google Bigtable model introduce an additional level of segmentation beyond the table, row, column model of the relational model. That is called the column family. In those implementations rows have a fixed set of column families but within a column family each row can have a variable set of columns. This is illustrated in figure 6.4.2.2.2-2 below.



The key distinction in the implementation of columnar store over relational stores is that data is high de-normalized for column stores and that while for relational stores every record contains some value (perhaps NULL) for each column, in columnar store the column is only present if there is data for one or more rows. This is why many column oriented stores are referred to as sparse storage models. Data for each column family is physically stored together on disk sorted by rowed, column name and timestamp. The last (timestamp) is there because the Bigtable model also includes the concept of versioning. Every, RowKey, Column Family, Column triple is stored with either a system generated or user provided Timestamp. This allows users to quickly retrieve the most recent value for a column (the default), the specific value for a column by timestamp, or all values for a column. The last is most useful because it permits very rapid temporal analysis on data in a column.

Because data for a given column is stored together two key benefits are achieved. First aggregation of the data in that column requires only the values for that column to be read. Conversely in a relational system the entire row (at least up to the column) needs to be read (which if the row is long and the column at the end it could be lots of data). Secondly, updates to a single column do not require the data for the rest of the row to be read/written. Also, because all the data in a column is uniform, data can be compressed much more efficiently. Often only a single copy of the value for a column is stored followed by the rowkeys where that value exists. And while deletes of an entire column is very efficient, deletes of an entire record are extremely expensive. This is why historically column oriented stores have been applied to OLAP style applications while relational stores were applied to OLTP requirements.

Recently, security has been a major focus of existing column implementations primarily due to the release by the National Security Agency (NSA) of it’s Bigtable implementation to the open source community. A key advantage of the NSA implementation and other recently announced implementations is the availability of security controls at the individual cell level. With these implementations a given user might have access to only certain cells in group based potentially based on the value of those or other cells.

There are several very mature distributed column oriented implementations available today from both open source groups and commercial foundations. These have been implemented and operational across a wide range of businesses and government organizations. Emerging are hybrid capabilities that implement relational access methods (e.g. SQL) on top of BigTable/Columnar storage models. Also, relational implementations are adopting columnar oriented physical storage models to provide more efficient access for big data OLAP like aggregations and analytics.

##### Document

Document storage approaches have been around for some time and popularized by the need to quickly search large amounts of unstructured data. Modern, document stores have evolved to include extensive search and indexing capabilities for structured data and metadata and why they are often referred to as semi-structured data stores. Within a document-oriented data store each “document” encapsulates and encodes the metadata, fields, and any other representations of that record. While somewhat analogous to a row in a relational table one-reason document stores evolved and have gained in popularity is that most implementations do not enforce a fixed or constant schema. While best practices hold that groups of documents should be logically related and contain similar data there is no requirement that they be alike or that any two documents even contain the same fields. That is one reason that document stores are frequently popular for data sets which have sparsely populated fields since there is far less overhead normally than traditional RDBMS systems where null value columns in records are actually stored. Groups of documents within these types of stores are generally referred to as collections and like key-value stores some sort of unique key references each document.

In modern implementations documents can be built of arbitrarily nested structures and can include variable length arrays and in some cases executable scripts/code (which has significant security and privacy implications). Most document-store implementations also support additional indexes on other fields or properties within each document with many implementing specialized index types for sparse data, geospatial data, and text.

When modeling data into document-stores the preferred approach is to de-normalize the data as much as possible and embed all one-to-one and most one-to-many relationships within a singled document. This allows for updates to documents to be atomic operations which keeps referential integrity between the documents. The most common case where references between documents should be use is when there are data elements that occur frequently across sets of documents and whose relationship to those documents is static. As an example, the publisher of a given book edition does not change and there are far fewer publishers than there are books. It would not make sense to embed all the publisher information into each book document. Rather the book document would contain a reference to the unique key for the publisher. Since for that edition of the book the reference will never change and so there is no danger of loss of referential integrity. Thus information about the publisher (address for example) can be updated in a single atomic operation the same as the book. Where this information embedded it would need to be updated in every book document with that publisher.

In the big data realm document stores scale horizontally through the use of partitioning or sharding to distribute portions of the collection across multiple nodes. This partitioning can be round-robin based insuring an even distribution of data or content/key based so that data locality is maintained for similar data. Depending on the application required the choice of partitioning key like with any data base can have significant impacts on performance especially where aggregation functions are concerned.

There are no standard query languages for document store implementations with most using a language derived from their internal document representation (e.g. JSON, XML).

##### Graph

While social networking sites like Facebook and LinkedIn have certainly driven the visibility of and evolution of graph stores (and processing as discussed below), graph stores have been a critical part of many problem domains from military intelligence and counter terrorism to route planning/navigation and the semantic web for years. Graph stores represent data as a series of nodes, edges, and properties on those. Analytics against graph stores include very basic shortest path and page ranking to entity disambiguation and graph matching.

Graph databases typically store two types of objects nodes and relationships as show in figure 6.4.2.2.4-1 below. Nodes represents objects in the problem domain that are being analyzed be they people, places, organizations, accounts, etc. Relationships describe those objects in the domain relate to each other. Relationships can be non-directional/bi-directional but are typically expressed as uni-directional in order to provide more richness and expressiveness to the relationships. Hence, between two people nodes where they are father and son there would be two relationships. One “is father of” going from the father node to the son node, and the other from the son to the father of “is son of”. In addition, nodes and relationships can have properties or attributes. This is typically descriptive data about the element. For people it might be name, birthdate, etc. For locations it might be an address or geospatial coordinate. For a relationship like a phone call it could be the date, time of the call, the duration of the call. Within graphs relationships are not always equal or have the same strength. Thus relationship often have one or more weight, cost, or confidence attributes. A strong relationship between people might have a high weight because they have known each other for years and communicate every day. A relationship where two people just met would have a low weight. The distance between nodes (be it a physical distance or a difficulty) is often expressed as a cost attribute on a relation in order to allow computation of true shortest paths across a graph. In military intelligence applications, relationships between nodes in a terrorist or command and control network might only be suspected or have not been completely verified so those relationships would have confidence attributes. Also, properties on nodes may also have confidence factors associated with them although in those cases the property can be decomposed into it’s own node and tied with a relationship. Graph storage approaches can actually be viewed as a specialized implementation of a document storage scheme with two types of documents (nodes and relationships). In addition, one of the most critical elements in analyzing graph data is locating the node or edge in the graph where you want to begin the analysis. To accomplish this most graph databases implement indexes on the node or edge properties. Unlike, relational and other data storage approaches most graph databases tend to use artificial/pseudo keys or guids to uniquely identify nodes and edges. This allows attributes/properties to be easily changed due to both actual changes in the data (someone changed their name) or as more information is found out (e.g. we get a better location for some item or event) without needing to change the pointers two/from relationships.



The problem with graphs in the Big Data realm is that they grow to be too big to fit into memory on a single node and by their typically chaotic nature (few real world graphs follow well defined patterns) makes their partitioning for a distributed implementation problematic. While distance between or closeness of nodes would seem like a straight forward partitioning approach there are multiple issues which must be addressed. First, would be balancing of data. Graphs often tend to have large clusters of data very dense in a given area thus leading to essentially imbalances and hot spots in processing. Second, no matter how you distribute the graph there are connections (edges) that will cross the boundaries. That typically requires that nodes know about or how to access the data on other nodes and requires inter-node data transfer or communication. This makes the choice of processing architectures for graph data especially critical. Architectures that do not have inter-node communication/messaging tend not to work well for most graph problems. Typically, distributed architectures for processing graphs assign chunks of the graph to nodes then the nodes use messaging approaches to communicate changes in the graph or the value of certain calculations along a path.

Even small graphs quickly elevate into the realm of big data when one is looking for patterns or distances across more than one or two degrees of separation between nodes. Depending on the density of the graph, this can quickly cause a combinatorial explosion in the number of conditions/patterns that need to be tested.

A specialized implementation of a graph store known as the Resource Description Framework (RDF) is part of a family of specifications from the World Wide Web Consortium (W3C) that is often directly associated with Semantic Web and associated concepts. RDF triples as they are known consist of a Subject (Mr. X), a predicate (lives at), and an object (Mockingbird Lane). Thus a collection of RDF triples represents and directed labeled graph. The contents of RDF stores are frequently described using formal ontology languages like OWL or the RDF Schema (RDFS) language, which establish the semantic meanings and models of the underlying data. To support better horizontal integration (Smith, Malyuta, Mandirck, Fu, Parent, & Patel, 2012) of heterogeneous data sets extensions to the RDF concept such as the Data Description Framework (DDF) (Yoakum-Stover & Malyuta, 2008) have been proposed which add additional types to better support semantic interoperability and analysis.

Graph data stores currently lack any form of standardized APIs or query languages. However, the W3C has developed the SPARQL query language for RDF which is currently in a recommendation status and there are several frameworks such as Sesame which are gaining popularity for working with RDF and other graph oriented data stores.

## Feature 2: Processing Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Dave

[Content Goes Here]

## Feature 3: Resource Managers Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Dave

[Content Goes Here]

## Feature 4: Infrastructure Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: PW

[Content Goes Here]

## Feature 5: Information Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: PW

[Content Goes Here]

## Feature 6: Standards Integration Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Keith

[Content Goes Here]

## Feature 7: Application Framework

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Carl

[Content Goes Here]

## Feature 8: Business Operations

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Bruno

[Content Goes Here]

## Feature 9: Business Intelligence

We need to build off of the value statement described in the table above in the “Features Summary” section to develop a case for the feature. Why it is important in the Big Data Ecosystem? What readiness level do we need it to get to and/or what should this feature be like in 5 years to meet the collective needs of Government / Academia / Industry? What needs to happen and how do we get there?

Author: Carl – Combine with Feature 10: Application Framework

[Content Goes Here]

# Big Data Related Multi-stakeholder Collaborative Initiatives

Author: Keith

Big Data has generated interest in a wide variety of organizations, including the de jure standards process, industry consortiums, and open source organizations. Each of these organizations operates differently and focuses on different aspects, but with a common thread that they are multi-stakeholder collaborative initiatives.

Integration with appropriate collaborative initiatives can assist both in cross-product integration and cross-product knowledge. Identifying which collaborative initiative efforts address architectural requirements and which requirements are not currently being addressed provides input for future multi-stakeholder collaborative initiative efforts.

“Collaborative initiatives” include:

* Subcommittees and working groups of Accredited Standards Development Organizations (the de jure standards process)
* Industry Consortia
* Reference implementations
* Open Source implementations

Open source implementations (such as those from the Apache Software Foundation) are providing useful new technology that is being used either directly or as the basis for commercially supported products. These open source implementations are not just individual products. One needs to integrate an eco-system or products to accomplish ones goals. Because of the ecosystem complexity, and because of the difficulty of fairly and exhaustively reviewing open source implementations, such implementations are not included in this section.

The following sections describe work currently completed, in planning and in progress in the organizations:

* INCITS and ISO – de jure standards process
* IEEE – de jure standards process
* W3C – Industry consortium
* OGC – Industry consortium

Organizations working in this area that are not included in this section are omitted through oversight. It is anticipated that as this document is more widely distributed, standards efforts addressing additional segments of the Big Data mosaic will be identified.

This work is mapped onto the Big Data Reference Architecture abstraction layers:

* Data Service Abstraction
* Capability Service Abstraction
* Usage Service Abstraction
* Usage Service Abstraction

In all cases, the standards identified cover aspects of the Big Data requirements identified in this document. They contain starting points for additional work but are not complete answers today. The value of identifying these incomplete efforts in this document is to avoid the complete recreation of work that has already been accomplished.

### Characteristics supported by standards

Standards efforts support aspects of the following characteristics:

* Interoperability
* Portability
* Reusability
* Extensibility

Evaluating the effectiveness of particular standards efforts with respect to these characteristics is complicated because of the complexity of the Big Data Architecture and the current patchwork nature of relevant standards.

### Information and Communications Technologies (IT) Standards Life Cycle

Different collaborative initiatives have different processes and different end goals, so the life cycle varies. The following is a broad generalization of the steps in a multi-stakeholder collaborative initiative life cycle:

* No standard
* Under development
* Approved
* Reference implementation
* Testing and certification
* Products/services
* Market acceptance
* Sunset

## Data Service Abstraction

The data service abstraction layer needs to support the ability to:

* Identify and locate data providers with relevant information
* Access the data source through some sort of query and/or analysis tool.

The following sections describe the standards related to:

* Data Provider Registry and Location Services
* Data Provider Interfaces
* Data Stores

### Data Provider Registry and Location service

A Data Provider Registry and Location service allows a data provider to:

* Create metadata describing the data source(s), usage policies/access rights, and other relevant attributes
* Publish the availability of the information and the means to access it
* Make the data accessible by other RA components using suitable programmable interface.

While ISO/IEC JTC1 SC32 WG2 has a variety of standards in the areas of registering metadata, the pieces do not completely specify the support needed to create a registry of the content and location of data providers. The related standards are:

| **Data Provider Interface** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| Metadata | INCITS DM32.8 & ISO/IEC JTC1 SC32 WG2 | The ISO/IEC 11179 series of standards provides specifications for the structure of a metadata registry and the procedures for the operation of such a registry. These standards address the semantics of data (both terminological and computational), the representation of data, and the registration of the descriptions of that data. It is through these descriptions that an accurate understanding of the semantics and a useful depiction of the data are found. These standards promote:   * Standard description of data * Common understanding of data across organizational elements and between organizations * Re-use and standardization of data over time, space, and applications * Harmonization and standardization of data within an organization and across organizations * Management of the components of data * Re-use of the components of data |
|  | INCITS DM32.8 & ISO/IEC JTC1 SC32 WG2 | The ISO/IEC 19763 series of standards provides specifications for a metamodel framework for interoperability. In this context interoperability should be interpreted in its broadest sense: the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units (ISO/IEC 2382-1:1993). ISO/IEC 19763 will eventually cover:   * A core model to provide common facilities * A basic mapping model to allow for the common semantics of two models to be registered * A metamodel for the registration of ontologies * A metamodel for the registration of information models * A metamodel for the registration of process models * A metamodel for the registration of models of services, principally web services * A metamodel for the registration of roles and goals associated with processes and services * A metamodel for the registration of form designs |

The missing pieces are:

* A data model for the registry
* Mechanisms (such as a call interface or query language) to register a data source
* Mechanisms (such as a call interface or query language) to retrieve information about a data source
* Mechanisms to broker the connection to the data source
* Integration of security and privacy requirements

### Data Provider Interfaces

Once a data consumer has identified a data source with the require characteristics, a communication link must be established. This communications link needs to be able to transfer:

* Queries and Requests to the data provider where the request could include some sort of analytical function.
* Result sets to the requester where the result set could be as simple as a series of documents or as complex as the output training from a machine learning engine.

The following data source interfaces are frequently used:

| **Data Source Interface** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| SQL/CLI | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-9:2008 Information technology – Database languages – SQL – Part 9: Management of External Data (SQL/MED) supports mapping external files underneath an SQL interface. |
| JDBC™ | Java Community ProcessSM | JDBC™ 4.0 API Specification |

These interfaces are primarily designed for row-style data. They need to be enhanced, or new interface languages constructed, to support more complex datatypes such as images, video, trained machine learning engines, etc.

### Data Sources

The Data Service Abstraction layer needs to support a variety of data retrieval mechanisms including (but not limited to):

* Flat Files with known structure
* Flat files with free text
* XML documents
* SQL Databases
* Audio, Picture, Multimedia, and Hypermedia
* Spatial Data
* Sensor network data
* Streaming data – Video
* Steaming data – Textual
* NoSQL Data Stores

| **Data Source** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| Flat Files with known structure | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-9:2008 Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED) supports mapping external files underneath an SQL interface. |
| Text | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-2 SQL/MM Part 2: Full Text provides full information retrieval capabilities and complement SQL and SQL/XML. SQL/XML provides facilities to manage XML structured data while MM Part 2 provides contents based retrieval. |
| XML documents | W3C XQuery Working Group | XQuery 3.0: An XML Query Language — uses the structure of XML to express queries across all these kinds of data, whether physically stored in XML or viewed as XML via middleware. |
| INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | ISO/IEC 9075-14:2011 Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML) supports the storage and retrieval of XML documents in SQL databases |
| SQL Databases | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG3 | The SQL Database Language is defined by the ISO/IEC 9075 family of standards:   * ISO/IEC 9075-1:2011 Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework) * ISO/IEC 9075-2:2011 Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation) * ISO/IEC 9075-3:2008 Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI) * ISO/IEC 9075-4:2011 Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM) * ISO/IEC 9075-9:2008 Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED) * ISO/IEC 9075-10:2008 Information technology — Database languages — SQL — Part 10: Object Language Bindings (SQL/OLB) * ISO/IEC 9075-11:2011 Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata) * ISO/IEC 9075-13:2008 Information technology — Database languages — SQL — Part 13: SQL Routines and Types Using the Java TM Programming Language (SQL/JRT) * ISO/IEC 9075-14:2011 Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML) |
| Audio, Picture, Multimedia, and Hypermedia | INCITS L3 & ISO/IEC JTC 1 SC29 | ISO/IEC 9281:1990 Information technology — Picture coding methods  ISO/IEC 10918:1994 Information technology — Digital compression and coding of continuous-tone still images  ISO/IEC 11172:1993 Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s  ISO/IEC 13818:2013 Information technology — Generic coding of moving pictures and associated audio information  ISO/IEC 14496:2010 Information technology — Coding of audio-visual objects  ISO/IEC 15444:2011 Information technology — JPEG 2000 image coding system  ISO/IEC 21000:2003 Information technology — Multimedia framework (MPEG-21) |
| INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-5 Part 5: Still Image provides basic functionalities for Image data management within SQL databases. |
| Spatial Data | INCITS L1 - Geographical Information Systems & ISO/TC 211 – Geographic information/Geomatics | ISO 6709:2008 Standard representation of geographic point location by coordinates  The ISO 191nn suite of geospatial standards |
| Open Geospatial Consortium | The OGC suite of geospatial standards and Abstract Specifications |
| INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 13249-3 Part 3: Spatial provides support for the functionalities required by geospatial applications. This work is carefully coordinated with ISO TC 211 and the Open Geospatial Consortium |
| Sensor network data | IEEE | ISO IEEE 21451 series of sensor standards and standards projects e.g. ISO IEEE 21451-2 Information technology — Smart transducer interface for sensors and actuators — Part 2: Transducer to microprocessor communication protocols and Transducer Electronic Data Sheet (TEDS) formats  ISO IEEE 21451-7 Standard for Information Technology - Smart Transducer Interface for Sensors and Actuators - Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats |
| Streaming data – Video | IEEE | IEEE 2200-2012 Standard Protocol for Stream Management in Media Client Devices |
| Streaming Data — Textual | INCITS DM32.2 & ISO/IEC JTC1 SC32 WG4 | ISO/IEC 9075-2:201x Information technology —Database languages — SQL — Part 2: Foundation (SQL/Foundation) supports queries using regular expressions across series of rows, but does not (yet) support operating on data streams |
| NoSQL Data Stores | Various | A large number of “open source” products exist but currently have no common interface language |

Note that while standards such as ISO/IEC 15444:2011 “Information technology — JPEG 2000 image coding system”, provide standard encodings for images, additional information is needed so that the images can be appropriately transformed for analysis. The transformation needed for images of fingerprints is different from those of photos of text, people, aerial photos, astronomical photos, etc. Many of these transforms already exist in various environments. Additional work is needed to identify the transforms (as well as related standards) and describe how the appropriate transforms can be represented in the Data Provider Registry and Location Service.

## Usage Service Abstraction

It is possible that a data consumer in the Usage Service Abstraction could make some transformation available as a data provider in the Data Service Abstraction. Therefore, the standards discussed in the Data Service Abstraction section are potentially relevant here.

## Capability Service Abstraction

Need some introductory words…

### Security and Privacy Management

Security and Privacy topics have been the subject of multiple standards efforts.

| **Requirement** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| Infrastructure Security | INCITS CS1 & ISO/IEC JTC 1/SC 27 | ISO/IEC 15408-2009 Information technology — Security techniques — Evaluation criteria for IT security  ISO/IEC 27010:2012 Information technology — Security techniques — Information security management for inter-sector and inter-organizational communications  ISO/IEC 27033-1:2009 Information technology — Security techniques — Network security  ISO/IEC TR 14516:2002 Information technology — Security techniques — Guidelines for the use and management of Trusted Third Party services |
| Data Privacy |  | ISO/IEC 29100:2011 Information technology — Security techniques — Privacy framework |
| Data Management, Securing data stores, Key management, and ownership of data |  | ISO/IEC 9798:2010 Information technology — Security techniques — Entity authentication  ISO/IEC 11770:2010 Information technology — Security techniques — Key management |
| Integrity and Reactive Security |  | ISO/IEC 27035:2011 Information technology — Security techniques — Information security incident management  ISO/IEC 27037:2012 Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence |

These standards effort need to be reviewed and mechanisms identified to apply the capabilities to the data service abstraction layer.

### System Management

| **Requirement** | **Standards Group** | **Related Standards** |
| --- | --- | --- |
| System Management and Scalable Infrastructure | Distributed Management Task Force | ISO/IEC 13187: 2011 “Information Technology – Server Management Command Line Protocol (SM CLP) Specification”  ISO/IEC 17203 2011 “Open Virtualization Format” |

## Standards Summary

While there are many standards efforts that touch pieces of the elements identified in the Big Data Reference Architecture, significant gaps exist. This section identifies many standards efforts, the pieces they do support, and areas needed to fully support the Big Data Reference Architecture described in this document.

# Big Data Strategies

## Strategy of Adoption

Author: Keith

### Identify and include stakeholders

Who are the stakeholders in this project?

* Marketing?
* Operations?
* Information Technology?
* Others?

It is important to include the critical stakeholders.

### Identify potential road blocks

In any project, there are a variety of potential people and situations that can derail progress.

**People**

For a Big Data project the people can be:

* Systems/network/security people who are too overwhelmed with day-to-day requirements to spend time working on a new project
* People who are wedded to the status quo
* ?

Where ever possible, co-op these people into the project.

**Data**

An additional potential issue is incomplete and/or poor quality data. In general, data that is write-only (Write Once, Read Never – WORN) will have data quality issues. These data risks must be identified as early as possible.

As data issues are identified, their potential impact must be reviewed to assess the impact. The possible impacts are:

* Issue is annoying but will have only minor impact on the results
* Issue will affect the validity of the results
* Issue is a complete show stopper

Whenever possible, these types of impediments should be documented and bypassed.

### Define Achievable Goals

Much of the current hype around Big Data takes the form of “By accumulating all of this data, we will be able to understand the universe, end hunger, and achieve world peace.” Most Big Data projects will achieve much more limited results, so it is critical to set realistic expectations up front. These expectations might look like:

* We expect to identify types of items that customers frequently purchase together so that we can organize the shelves in a way that increases year-over-year per-store sales.
* We are looking for patterns and correlations that will help identify potential maintenance issues before the problems actually occur, thereby reducing production down time.
* We are looking for places where we can slow down storm runoff into the sewage system in order to keep peak volumes within the sewage processing plant capacity and prevent untreated sewage discharges into the river.
* We don’t know yet what we can do with the data, but if we don’t accumulate it now, we will not have the opportunity to investigate.

It usually makes sense to include both a realistic estimate of the possibility of success as well as the potential benefit if the project does actually achieve something useful. This could take the form of:

* From what we know today, there is about a 10% chance that we will achieve something useful, but if we can identify changes that improve efficiency by 1%, it could save the company ten million dollars a year.
* ???

### Define “Finished” and “Success” at the beginning of the project

Two decades ago, I worked with a project manager who claimed there are two critical criteria that should be defined at the start of a project:

* How do you know when the project is done?
* How do you know if the project is successful?

This can also be categorized as correctly setting expectations at the beginning of the project but it also provides dispassionate criteria for evaluating a project’s status.

## Strategy of Implementation

Author: Bruno

The NIST Roadmap Working Group cannot provide the answer for every organization on how to decide on which Big Data solution to use. The Working Group can provide general direction for stakeholders on how to incorporate Roadmap artifacts into decision making. A Big Data Framework has been designed to enable stakeholders to make decisions based on an agnostic approach.

To assist in deciding on how to decide on Big Data for an organization, templates, business conversations, and, dependencies have been outlined by the NIST Working Group.

A set of templates have been created to provide both a technology and methodology agnostic approach in making a decision on Big Data. The following describes the upcoming templates to be provided:

1. Internal workshops: This is a daily agenda format providing an outline for a team to collaborate on Big Data strategizing.
2. Readiness Self-Assessment: This template provides an approach to defining if the organization and its technology is prepared for Big Data.
3. Questionnaire: This template provides example Big Data questions a team should ask themselves.
4. Vendor Management: This template explains how a team can use its findings and incorporate them into an RFI, RFQ, or, RFP.

For the above templates, business conversations will typically drive the Big Data course-of-action. Five types of business conversations are:

1. Optimize: This conversation revolves around how Big Data will improve the efficiency of the business, to include processes, CAPEX and OPEX.
2. Agility: This conversation revolves around Big Data assisting in the ability to pivot to the demands of the market, customers, and, any other dependencies.
3. Innovate: This conversation revolves around Big Data assisting the business to create new ways to operate.
4. Compliance: This conversation revolves around Big Data supporting audit capabilities for industry and government standards as: SOX, HIPAA, SEC, etc.
5. Green: This conversation revolves around Big Data supporting Green initiatives for the business.

The templates and business conversations have dependencies. These dependencies have two groups: 1) Business, and, 2) Technology.

The following are the Business dependencies which feed into the templates and business conversations:

1. Culture
2. Organizational (Structure, Silos, Processes)
3. Governance
4. Fiscal Planning
5. Mergers & Acquisitions

The following are the Technology dependencies which feed into the templates and business conversations:

1. As-Is Architecture
2. IT Roadmap
3. IT Team (Skills and Aptitude)
4. IT Services Catalogue
5. Bill-of-IT
6. Vendor Strategy

Below is an output example of how a template can assist stakeholders in articulating their Big Data needs:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Business Conversation | Value | Big Data  Feature | Readiness Level | Use Cases | Actors |
| Agility:  IT needs to provide Marketing the ability respond real-time to acquiring on-line customers | **Value Statement:**  Lower cost of acquiring new customers by ‘X’ percent by October 1st. | **Roadmap Feature**:  Business Intelligence  (Real-Time BI)  [Reference Architecture capabilities can also be outlined here as well.] | **Technology**:  Reference Implementation  -One or more reference implementations are available  -Reference implementations are usable at scale  **Organization**:  Ad Hoc  -Awareness of Big Data exists  -Some groups are building solutions  -No Big Data plan is being followed | **Use Cases:**  #1, 3,6 | **Management**:  -On-line Marketing Officer  -Revenue Officer  **Analyst**:  -Online Marketing Leads (5)  **Technical**:  Network SME  Datacenter SME  Infrastructure SME  CRM Data SME  Storage SME (TBD)  **End Consumer:**  -Online Customers  -Retail Stores (TBD) |
|  |  |  |  |  |  |

## Resourcing

What are the types of skills, how many types of people, where should an organization start with building their Big Data team? If we can provide the types of skills, and an estimate of how many (obviously dependent on organization size), we could give organizations a starting point for building their Big Data team. Alignment with readiness indicators.

Author: Dan

[Content Goes Here]

# Concerns and Assumptions Statement

Include our master file we are developing to integrate across subgroups.

Author: Carl

[Content Goes Here]

# Appendix A: Industry Information

There was some discussion as it relates to discussing the industry as a whole, what products are out there, what architectures are in place. Need to decide if we think this is valid. I think this section could be dangerous in meeting our product agnostic approach.

Author: Nancy / Carl (Bob’s 12 page input)

[Content Goes Here]