Veterans Health Administration

Common Services

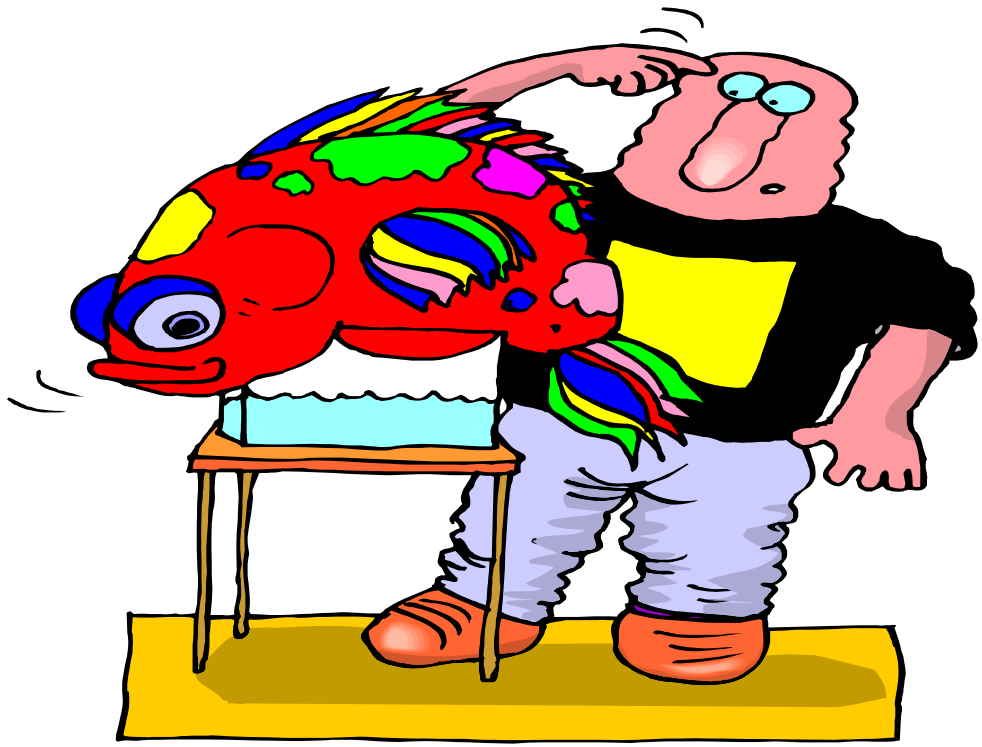
October 27,2005

Keith Cox

Acting Associate Deputy Director

Health Systems Design & Development

“A technical overview in non-technical terms…”



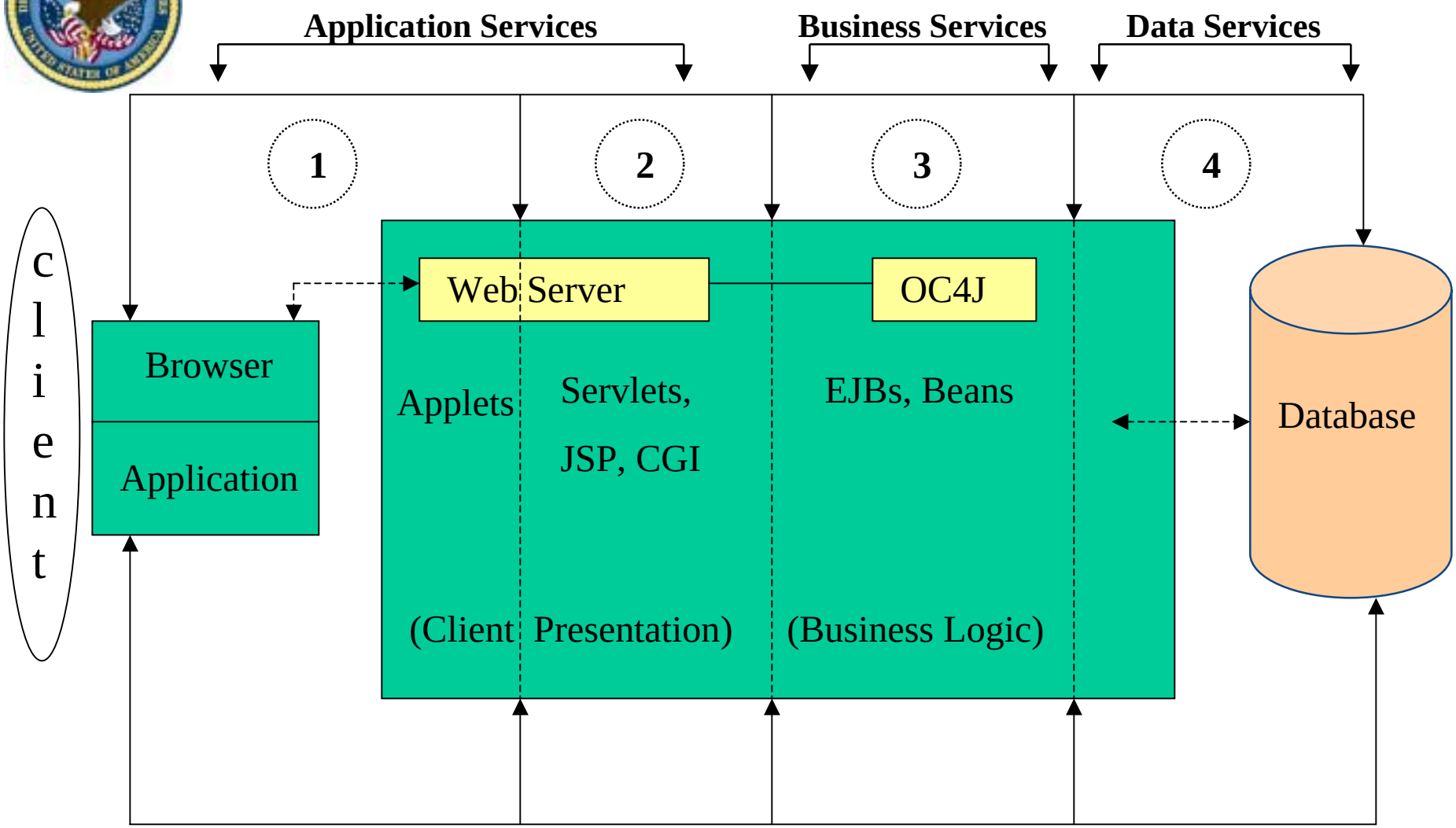
The Motivation

- Legislative and Departmental Mandates
  - The Clinger-Cohen Act
  - E-Government
  - One VA
- Improve level of interaction with private sector information systems.
- Enhance ability to utilize commercially produced software and IT products.
- Inadequacy of current technology and architecture
  - Incompatible technology
  - “Silo” application design
  - Difficult to maintain

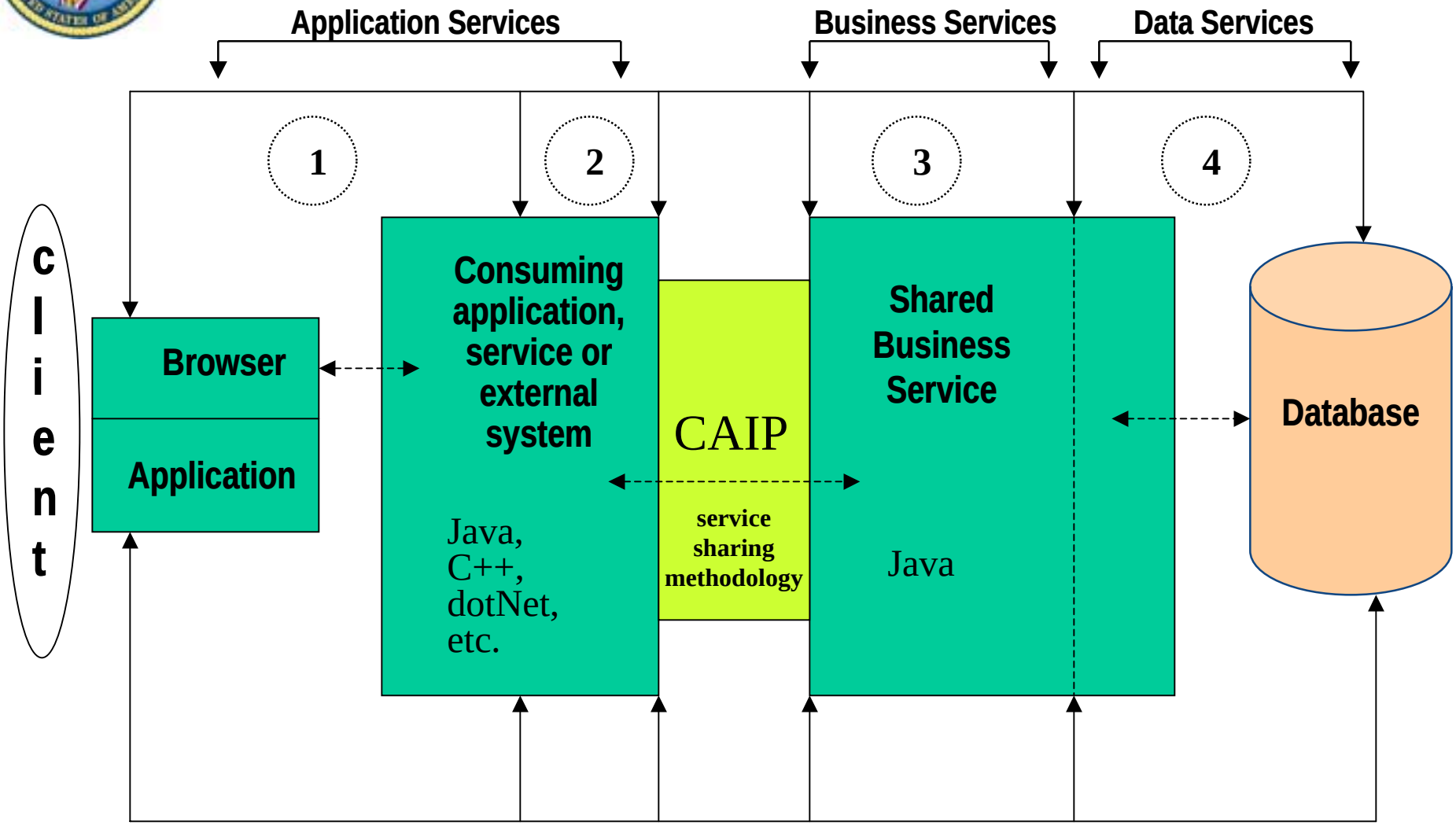
Enterprise Architecture Implementation

- N-tier software design
- Modern technology
- Service oriented architecture
  - Decomposition of application capabilities
  - Creation of service components
  - Methodology for sharing services
- Centralized/distributed deployment
  - Virtual single record view
  - Synchronization across systems of interest
  - Support for continuity of operations
- Standards / Information Model based

N-tier Architecture



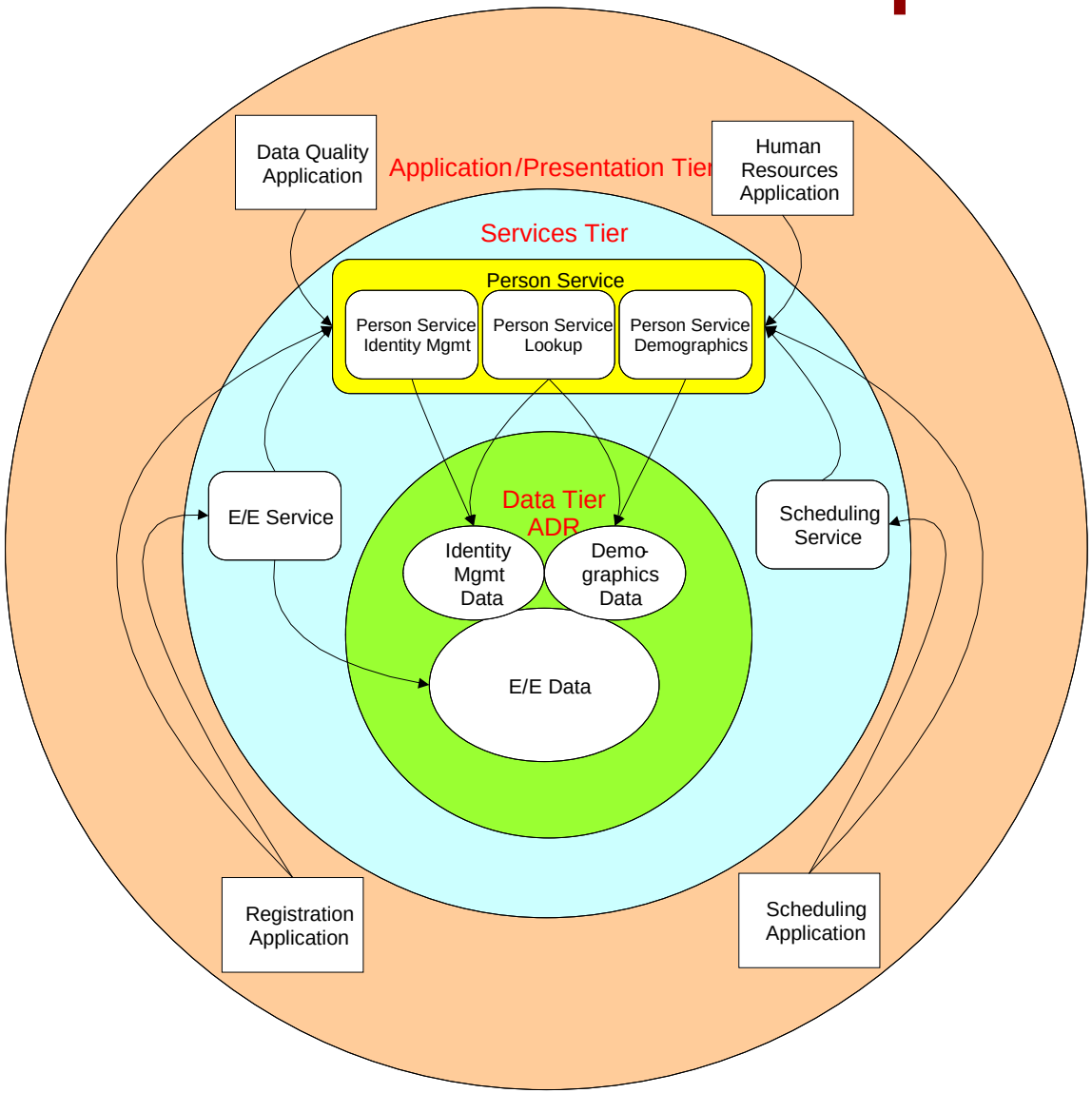
Shared Services Architecture



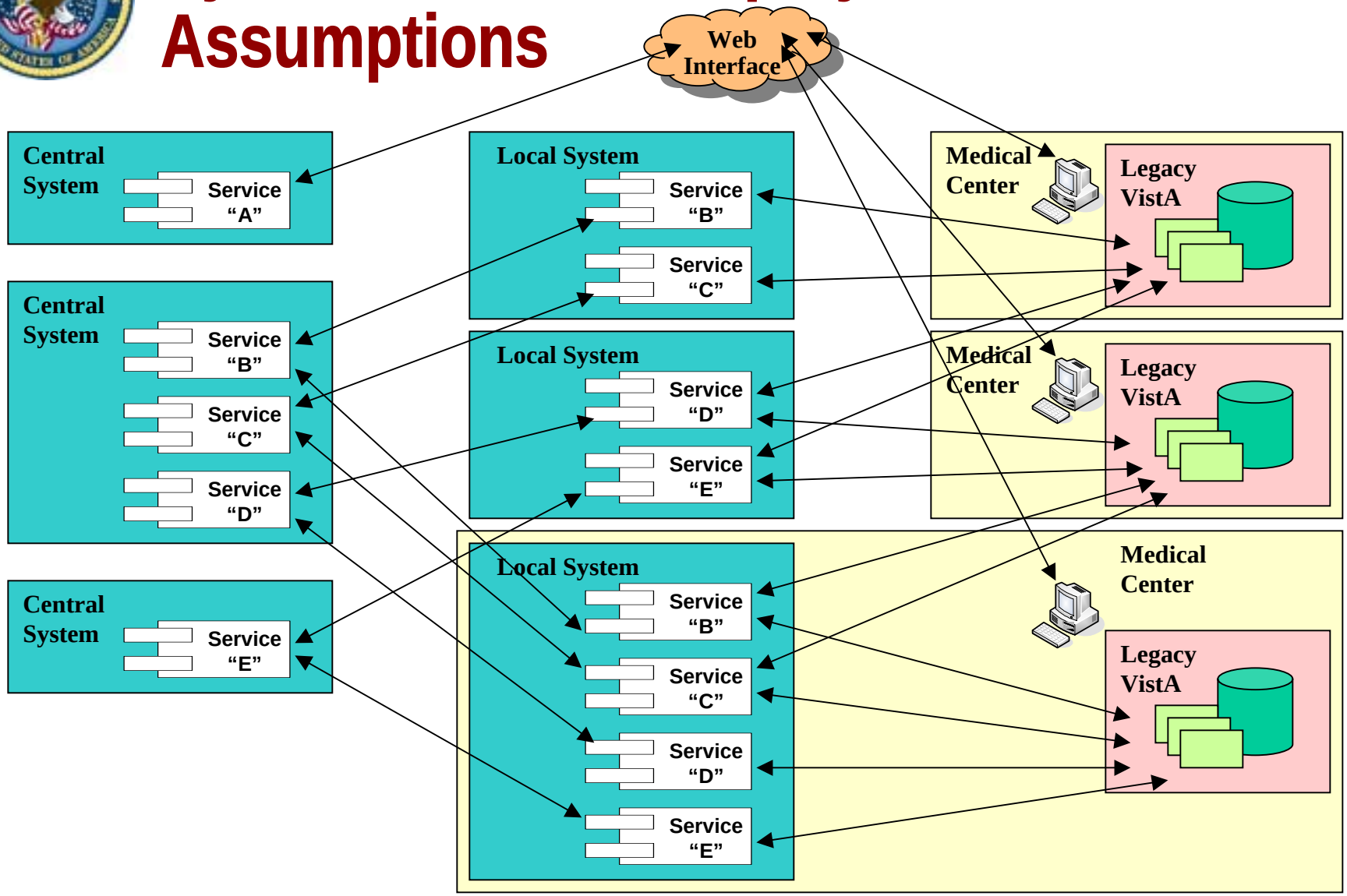
What is CAIP?

- “Cross Application Interface Protocol” (CAIP)
- CAIP defines a framework for establishing a consistent foundation for integration across applications
- A layer of abstraction between shared business services and the applications that subscribe to them
- CAIP can serve as a technology adapter between business services created in Java and applications or systems based in other technologies

N-tier Shared Services Example

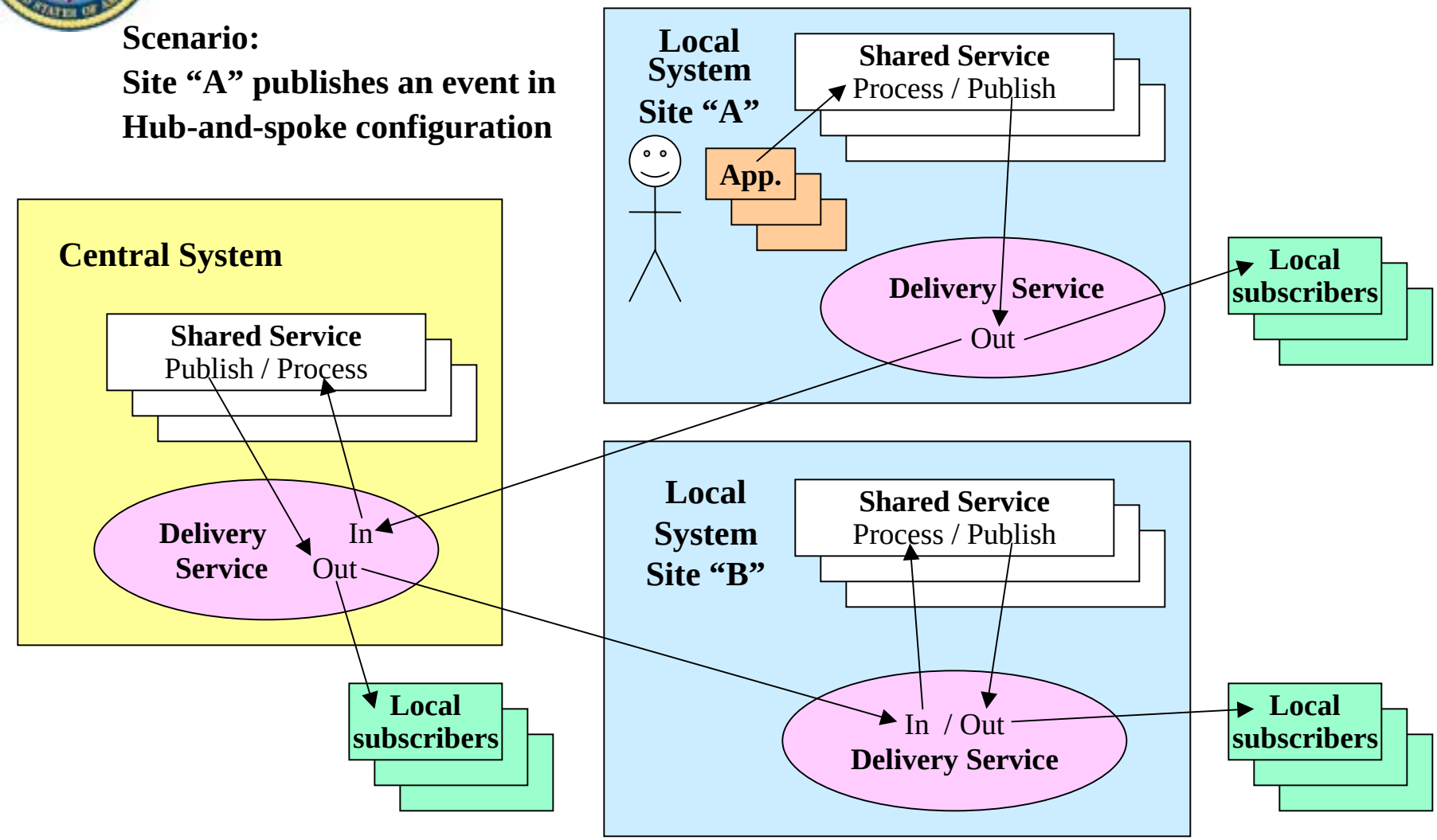


System / Service Deployment Assumptions

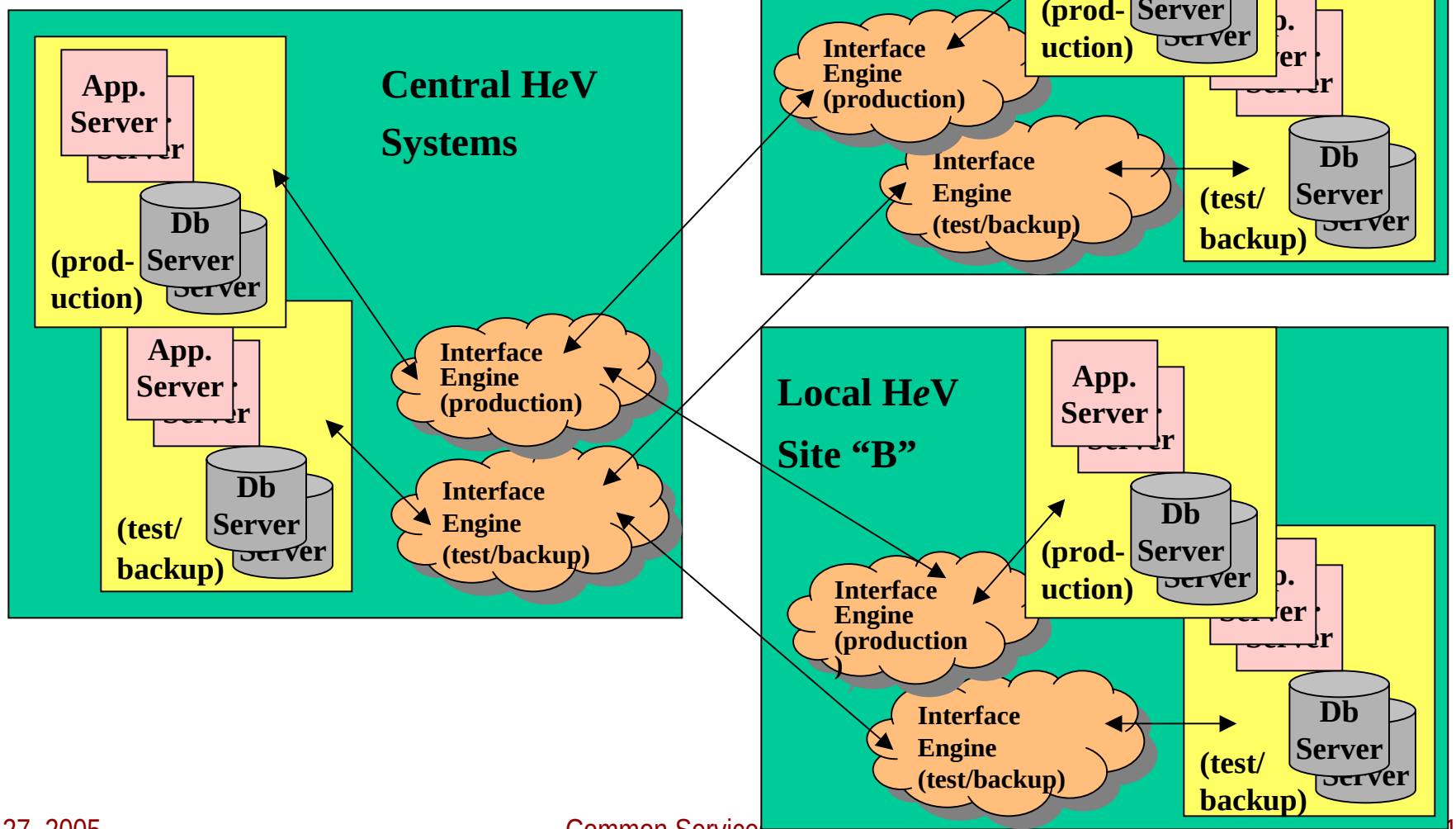


Publish & Subscribe

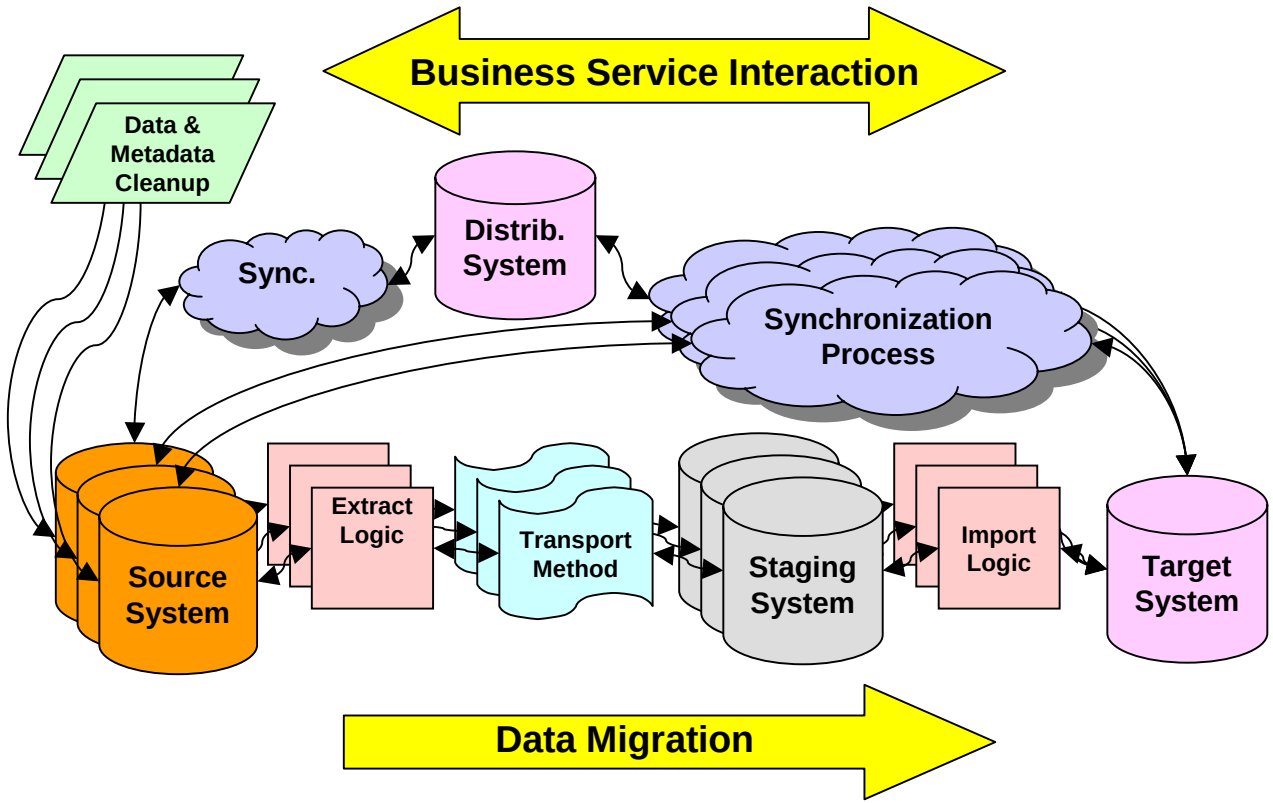
Scenario:
Site "A" publishes an event in Hub-and-spoke configuration



Redundancy for Test / Failover?



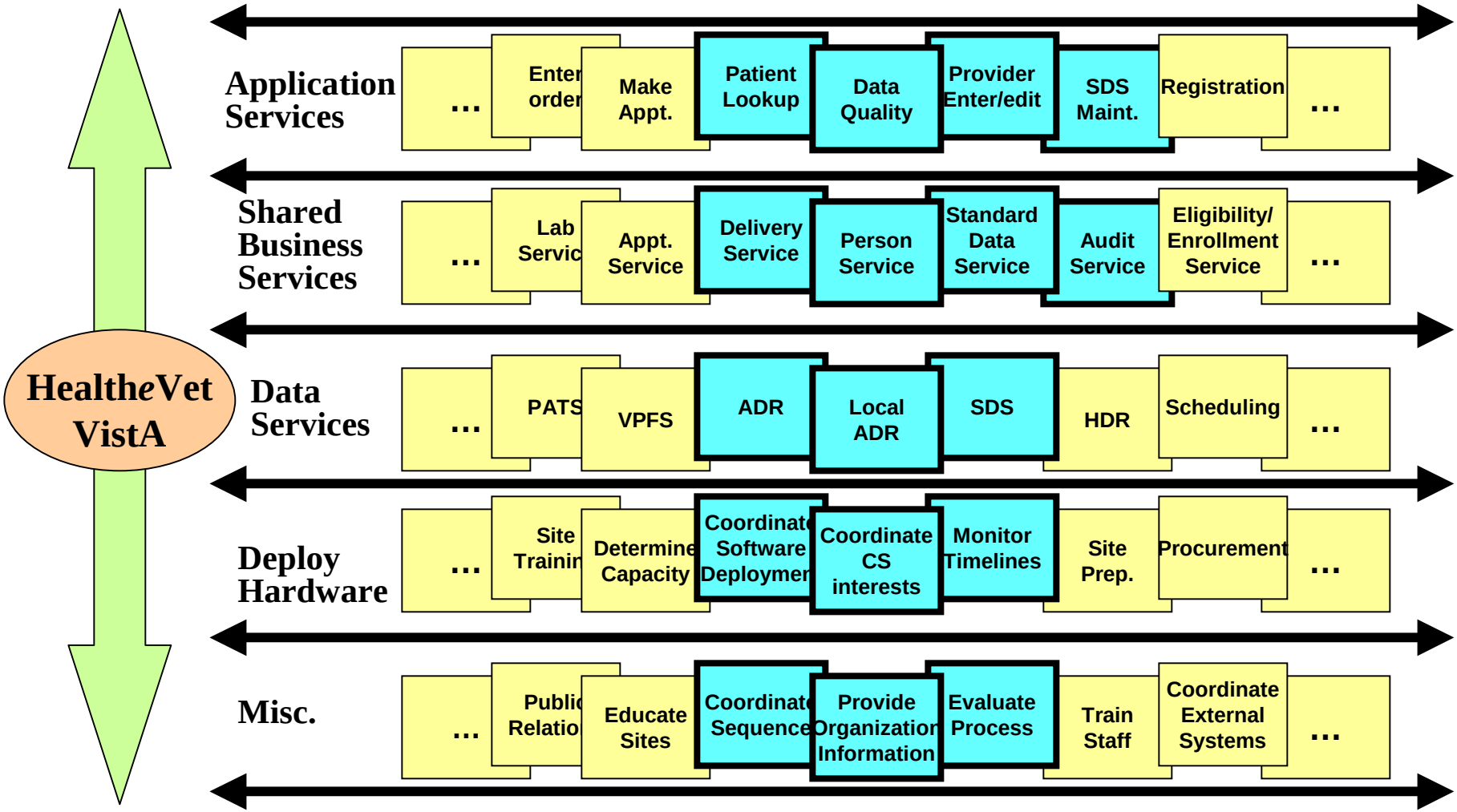
Data Migration Concepts



Common Services Focus

- Provision/coordination of core functionality
  - Business support functionality including Person Service and Standard Data Service
  - Middleware services that support the sharing of services and data between applications and systems
  - Infrastructure services such as security, auditing, archiving, etc.
  - Hardware deployment necessary to support application modernization.
- Maturation of organizational support structures
- Proof of concept for SOA implementation

Scope:
Common Services vs. common services



Common Services Development Perspectives

Use Case View – Business case, functionality scenarios

Logical View – Object models, classes, relationships

Process View – Control flows, intercommunication

Implementation View – modular structure (Service Oriented Arch.)

Deployment View – Allocation of software/hardware (Centralized/ Distributed deployment)

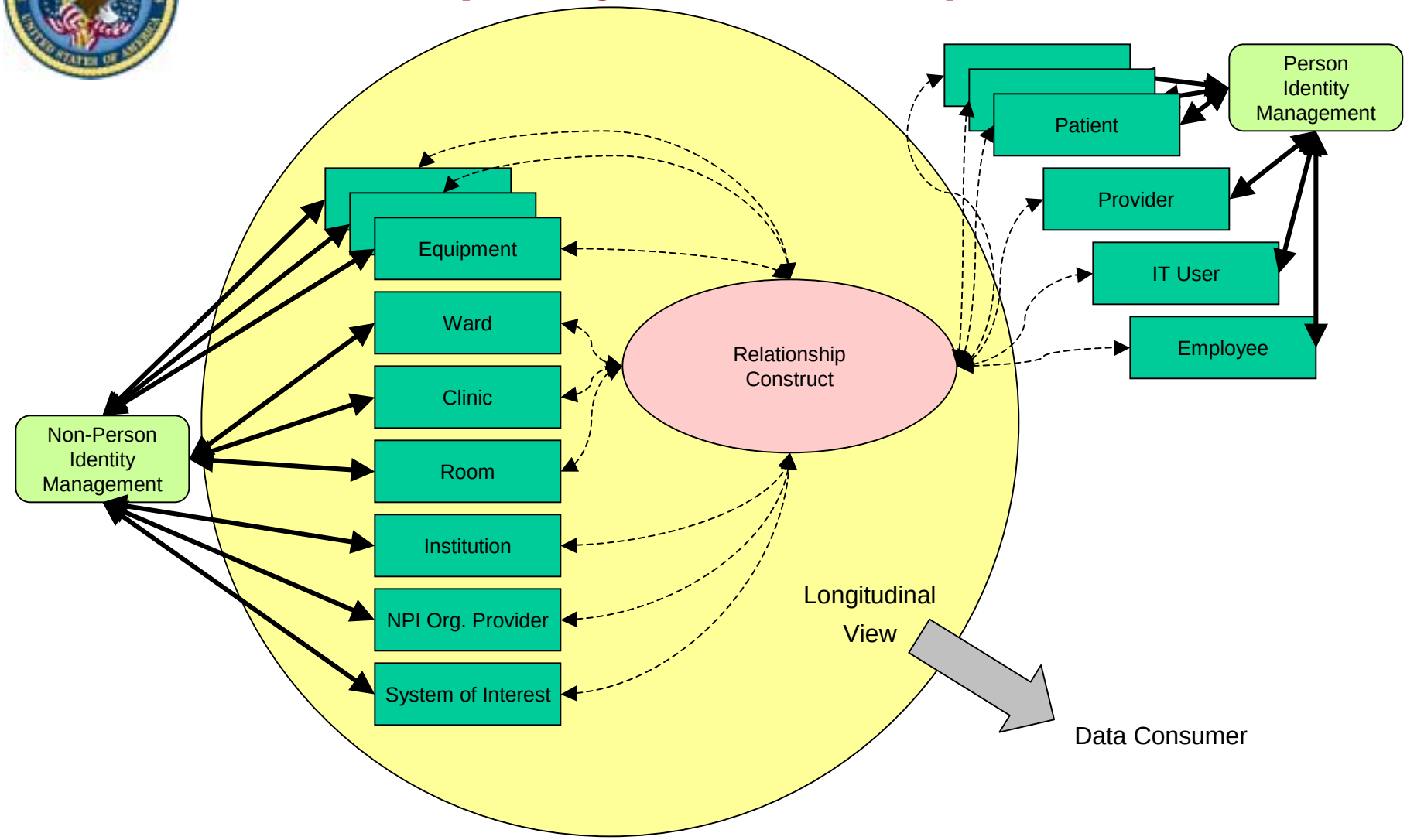
Common Services Efforts

- Person Service
  - Lookup
  - Demographics
  - Identity Management
- Standard Data Service
  - Table factory
  - Replication
  - Authoring
  - Maintenance
- Service Oriented Implementation
  - CAIP
  - Delivery Service
  - Subscription Management Service
  - Messaging Service
  - Dynamic Routing Service
  - Naming / Directory Service
  - HL7 Segment Service

Common Services Efforts (cont.)

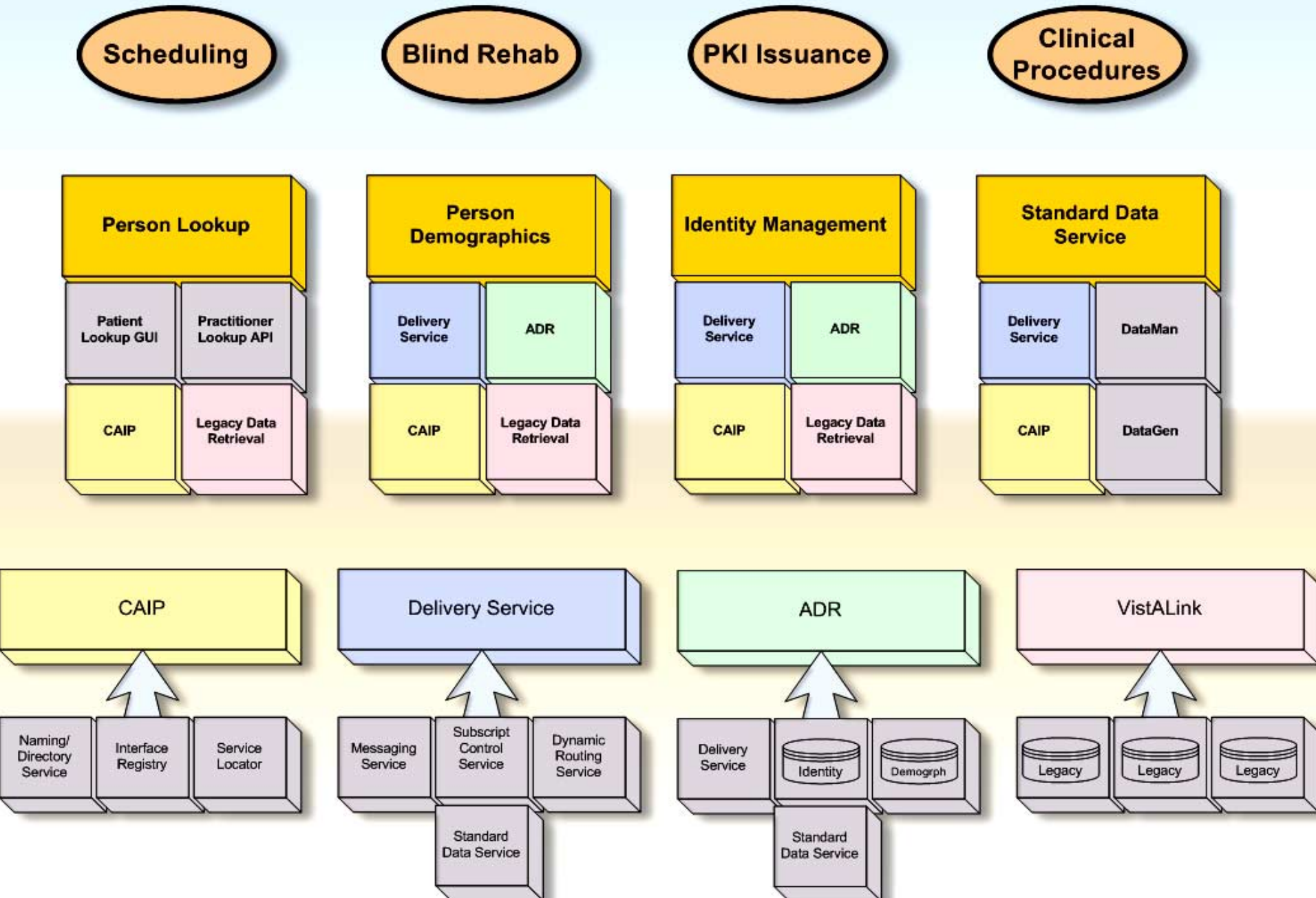
- Application Preference Service
- Audit Service
- Archive Service
- Enterprise Exception Log
- Administrative Data Repository (ADR)
- ADR Data Migration
- Institution file re-design
- Data Standardization
- National Provider Identifier
- Metadata Service
- Cross Service and Extended Transaction Manager
- Non-Person Service

Relationship/Longitudinal Concepts



Note: This is a representation of concepts, not technical structure of files, etc.

Common Services Building Blocks
9/30/04 Deliverables and Users



Issues – A share of the wealth…

- Data
  - Standardization
  - Cleanup
  - Metadata uniformity
  - Synchronization
  - Migration
  - Ill formed data
  - History
  - Checksums

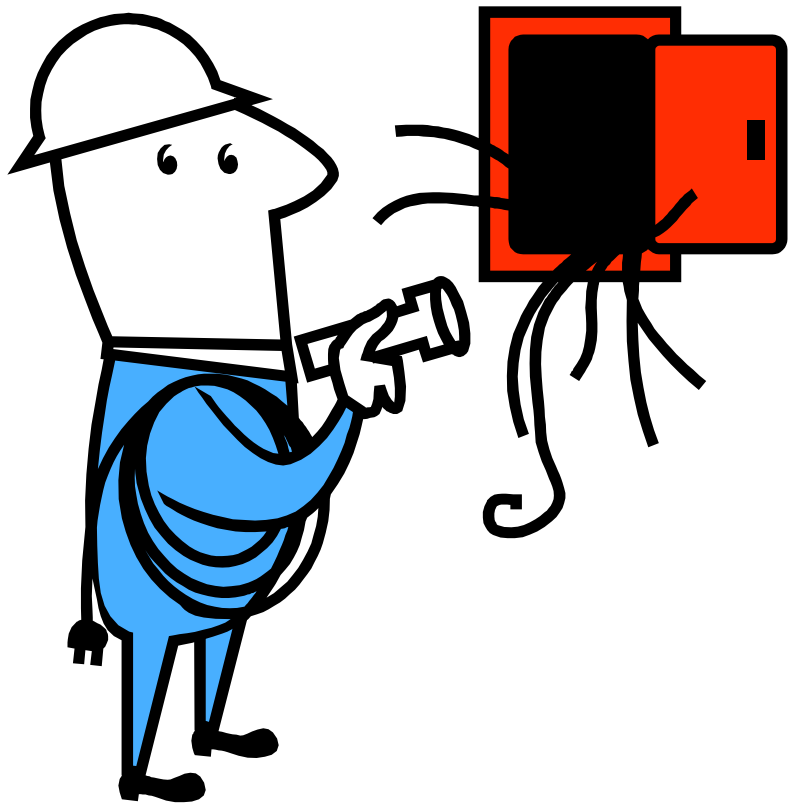
More Issues…

- Software Performance Engineering (SPE)
  - Invalid assumptions
  - Baseline
  - Usability vs. performance
  - Tolerance/threshold
  - Modeling/prediction
  - Benchmarking
  - Expectation management
  - Test environment

Yes, more issues…

- Versioning
  - Software
  - Metadata
  - Table content
- Overarching Requirements
- N-tier Requirements Management
- System interdependency

Questions?



Backup slides

Software Versioning Characteristics

- Co-existence of multiple component versions
- Abstraction of versioning consumption specifics
- Discoverable at development time and run time
- Centrally and locally manageable configuration
- Controlled consumption
  - Arbitrary – maintain FDA device relationships
  - Dynamic – upgrade per compatibility
  - Selected – honor consumer's preference
  - Forced – upgrade by mandate

Software Versioning Motivations

- Limited exposure in production environments
- Reduced complexity in implementing software upgrades
- Maximize functional flexibility
- Coordinated software rollout and activation
- Effective software configuration management
- Shift impacts from hard coded application logic to application runtime configuration.

Factors Limiting Versioning Effectiveness

- Backwards and forwards compatibility
- Shared computing environment
- Software update types
- Deployment architecture
- N-tier development characteristics

Factors: Compatibility Considerations

- Backward compatible – new versions must be compatible with impact of older versions on the shared environment
- Forward compatible – old versions must be compatible with impact of newer versions on the shared environment
- Compatibility types
  - Syntactic – the structure of API signatures, message formats, etc.
  - Semantic – the meaning of data, business rules, GUI navigation logic, etc.

Factors: Shared Computing Environment

- Shared transactions – both semantic and syntactic concerns (e.g. message flow across systems)
- Shared data – both semantic and syntactic concerns
  - Shared data types
    - Primary data (patient records, etc.)
    - Reference data
  - Shared data semantic and syntactic elements
    - Conceptual content
    - Domain (e.g. elements in a pick list)
    - Metadata (field length, etc.)

Factors: Software Update Types

- Defect repair – largely transparent from a syntactic/semantic standpoint
- Enhancement – potentially syntactic or semantic (or both)
- Business Process Re-engineering – likely to have significant semantic changes and potentially syntactic changes as well

Factors: Deployment Architecture

- Central/Web based – Voluntary
- Central/Web/Integrated – PATS, VPFS, Blind Rehab
- Central Database/Distributed processing – Scheduling Replacement
- Centralized/Distributed Database & processing – Common Services
- Decentralized/Integrated – Imaging

Co-existent Version Deployment

Opportunity	Limitation
No data, local data	Shared, distributed data
Local transactions	Cross-system transactions
Forward/backward compatible	Forward/backward incompatible
Application Service updates	Database, Business Service updates
Decentralized data and processing	Centralized/distributed data or processing
Defect repairs	Enhancements, Business Process Re-engineering

Conclusions: Versioned vs. Incremental release

- Governed by factors of environment and software update characteristics
- Constrained by limiting factors that exist
- Forward/backward compatibility is not consistently achievable
- Changes to shared data and cross-system transactions cannot be effectively partitioned
- Versioning rigor is needed to support both co-existent versioned and incremental (big bang) release approaches

Conclusions: Reference Data

- Temporal model vs. versioned data
  - Temporal – provides longitudinal view of single version, preferable for frequent content changes
  - Versioned – necessary for multiple views of a single table to co-exist
- Recommendation – Versioned/temporal

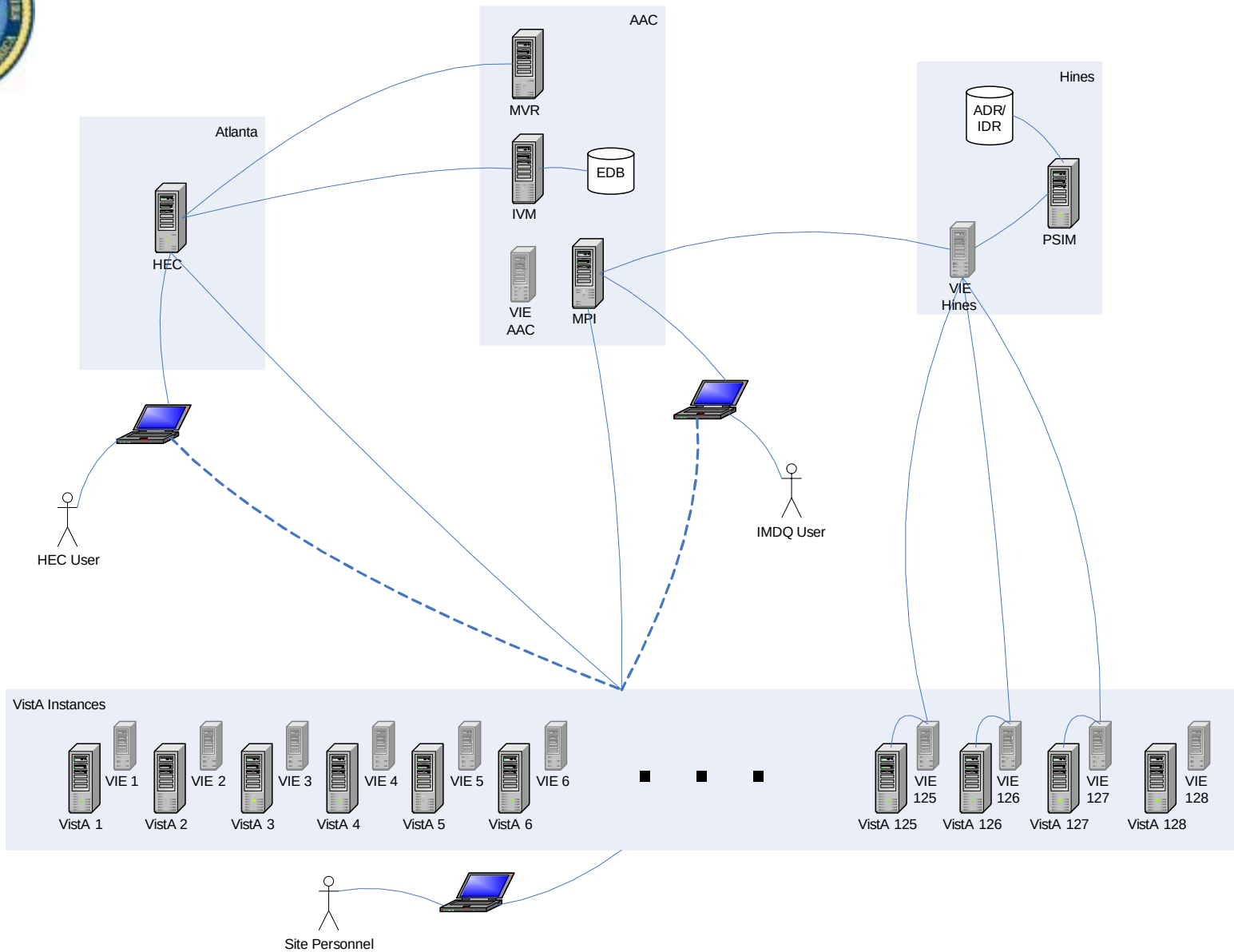
Conclusions: Versioning Implementation

- Implement for software components, business rules, XML schemas, software distributions, API integrations and reference data content
- Implement as needed for database schemas and metadata
- Make versioned vs. incremental release decisions in Release and Configuration Mgt. (not development)
- Create tools to manage runtime version consumption
- Establish test process for both versioned and incremental release paths

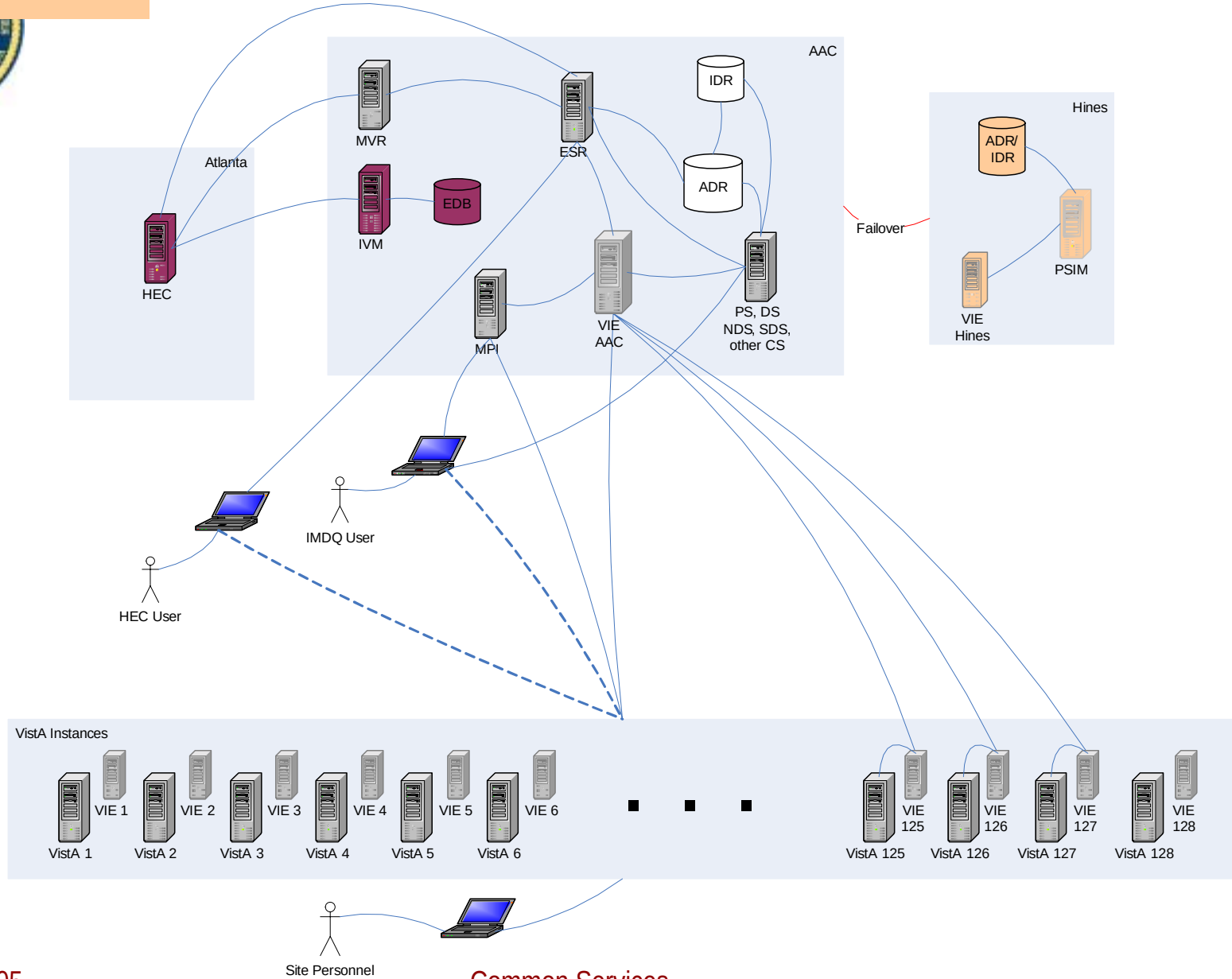
Next Steps

- Establish organizational consensus
- Define versioning standard or policy development teams must follow
- Define software test/release process
- Create development cookbook for meeting versioning standard
- Develop runtime version consumption mapping tools

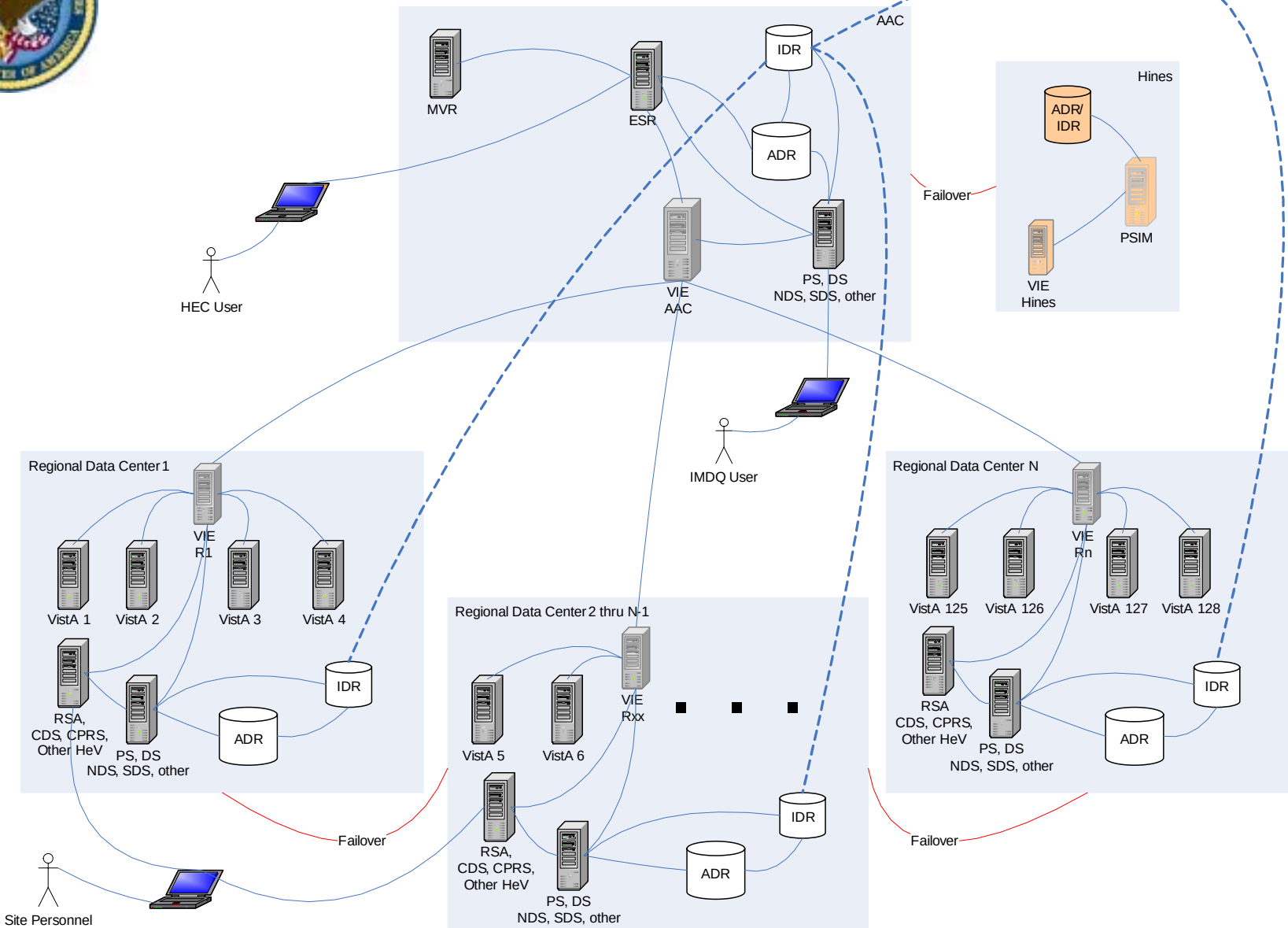
TODAY


AAC
MVR
EDB
IVM
VIE
AAC
MPI
Atlanta
HEC
Hines
ADR/
IDR
PSIM
VIE
Hines
HEC User
IMDQ User
VistA Instances
VIE 1
VIE 2
VIE 3
VIE 4
VIE 5
VIE 6
VistA 1
VistA 2
VistA 3
VistA 4
VistA 5
VistA 6
VIE
125
VIE
126
VIE
127
VIE
128
VistA 125
VistA 126
VistA 127
VistA 128
Site Personnel

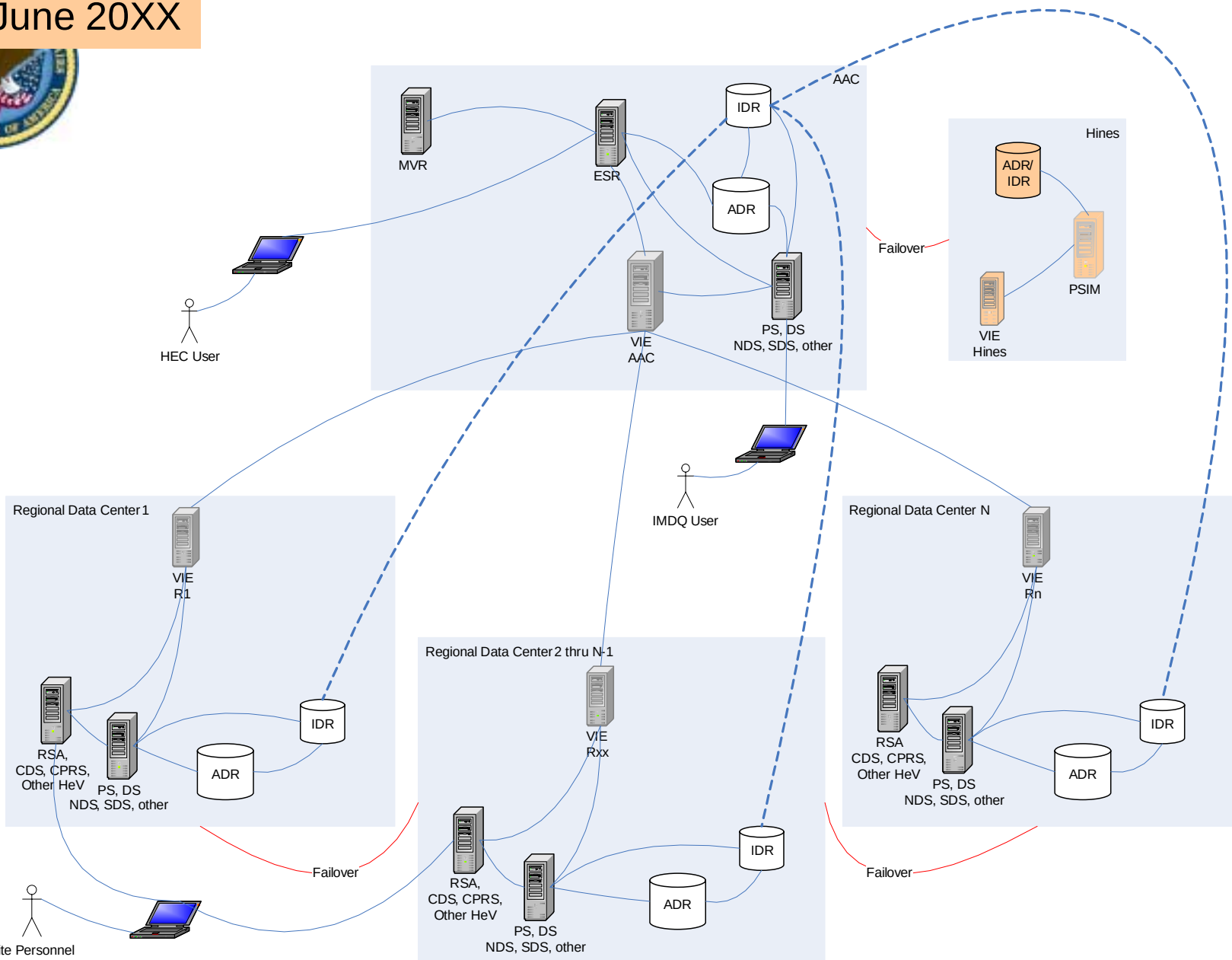
June 2006


AAC
Atlanta
Hines
MVR
ESR
IDR
ADR
EDB
IVM
HEC
MPI
VIE
AAC
PS, DS
NDS, SDS,
other CS
Failover
ADR/
IDR
PSIM
VIE
Hines
IMDQ User
HEC User
VistA Instances
VIE 1
VIE 2
VIE 3
VIE 4
VIE 5
VIE 6
VistA 1
VistA 2
VistA 3
VistA 4
VistA 5
VistA 6
VIE
125
VIE
126
VIE
127
VIE
128
VistA 125
VistA 126
VistA 127
VistA 128
Site Personnel

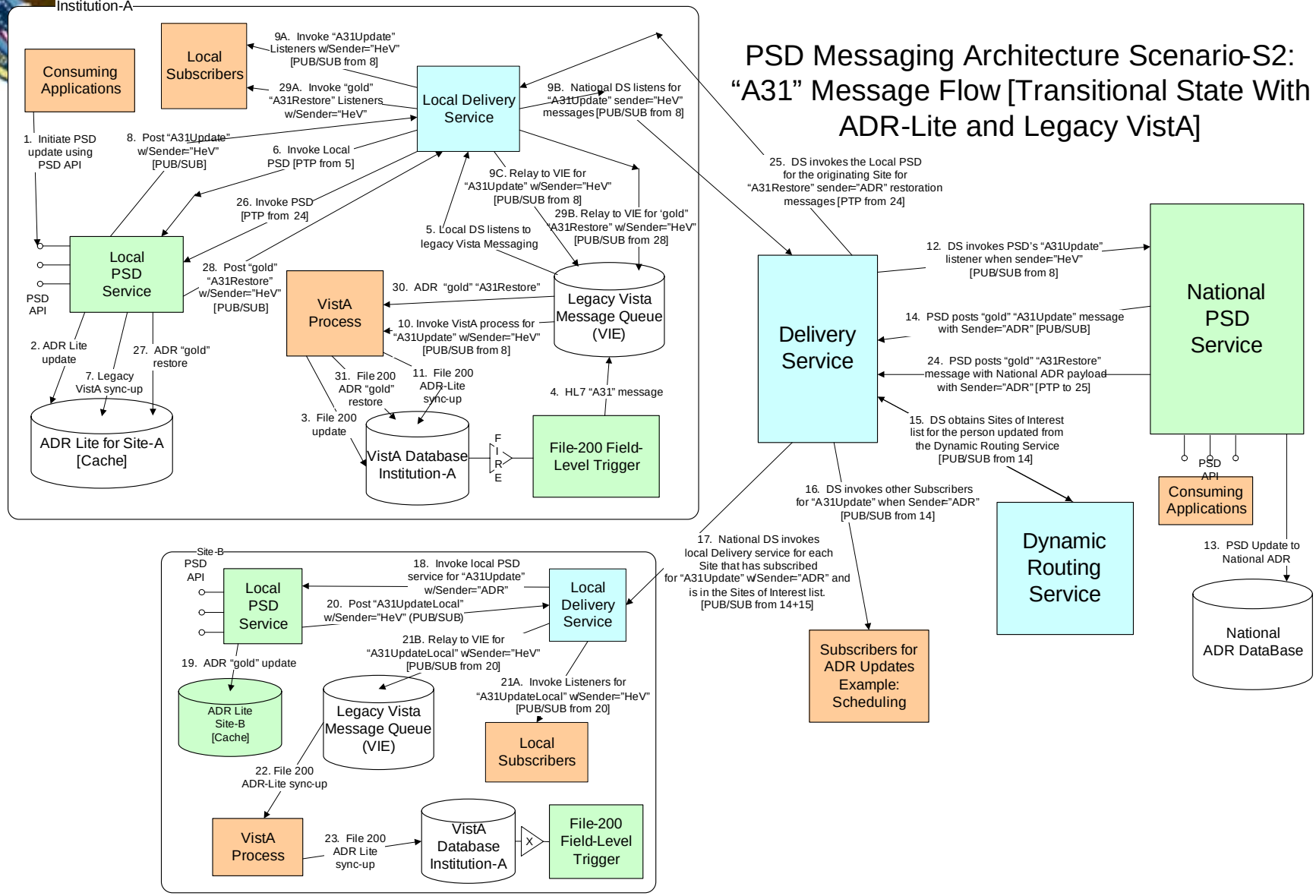
June 2008

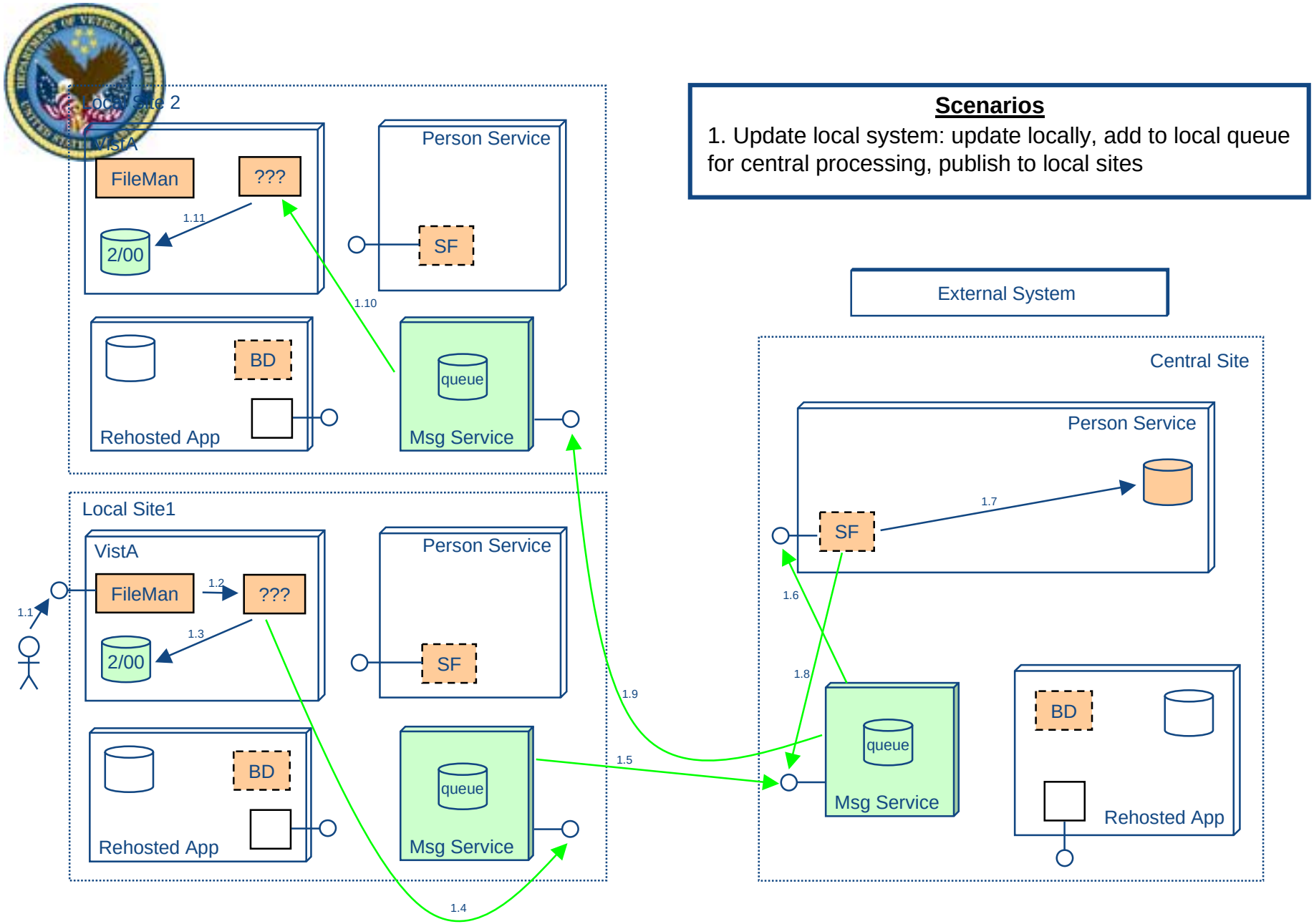

AAC
MVR
ESR
IDR
ADR
VIE
AAC
PS, DS
NDS, SDS, other
HEC User
Failover
Hines
ADR/
IDR
PSIM
VIE
Hines
IMDQ User
Regional Data Center 1
VIE
R1
VistA 1
VistA 2
VistA 3
VistA 4
RSA,
CDS, CPRS,
Other HeV
PS, DS
NDS, SDS, other
ADR
IDR
Failover
Site Personnel
Regional Data Center 2 thru N-1
VIE
Rxx
VistA 5
VistA 6
RSA,
CDS, CPRS,
Other HeV
PS, DS
NDS, SDS, other
ADR
IDR
Regional Data Center N
VIE
Rn
VistA 125
VistA 126
VistA 127
VistA 128
RSA
CDS, CPRS,
Other HeV
PS, DS
NDS, SDS, other
ADR
IDR
Failover

June 20XX


AAC
MVR
ESR
IDR
ADR
HEC User
VIE
AAC
PS, DS
NDS, SDS, other
Failover
Hines
ADR/
IDR
PSIM
VIE
Hines
IMDQ User
Regional Data Center 1
VIE
R1
RSA,
CDS, CPRS,
Other HeV
PS, DS
NDS, SDS, other
ADR
IDR
Regional Data Center 2 thru N-1
VIE
Rxx
RSA,
CDS, CPRS,
Other HeV
PS, DS
NDS, SDS, other
ADR
IDR
Regional Data Center N
VIE
Rn
RSA
CDS, CPRS,
Other HeV
PS, DS
NDS, SDS, other
ADR
IDR
Failover
Failover
Site Personnel

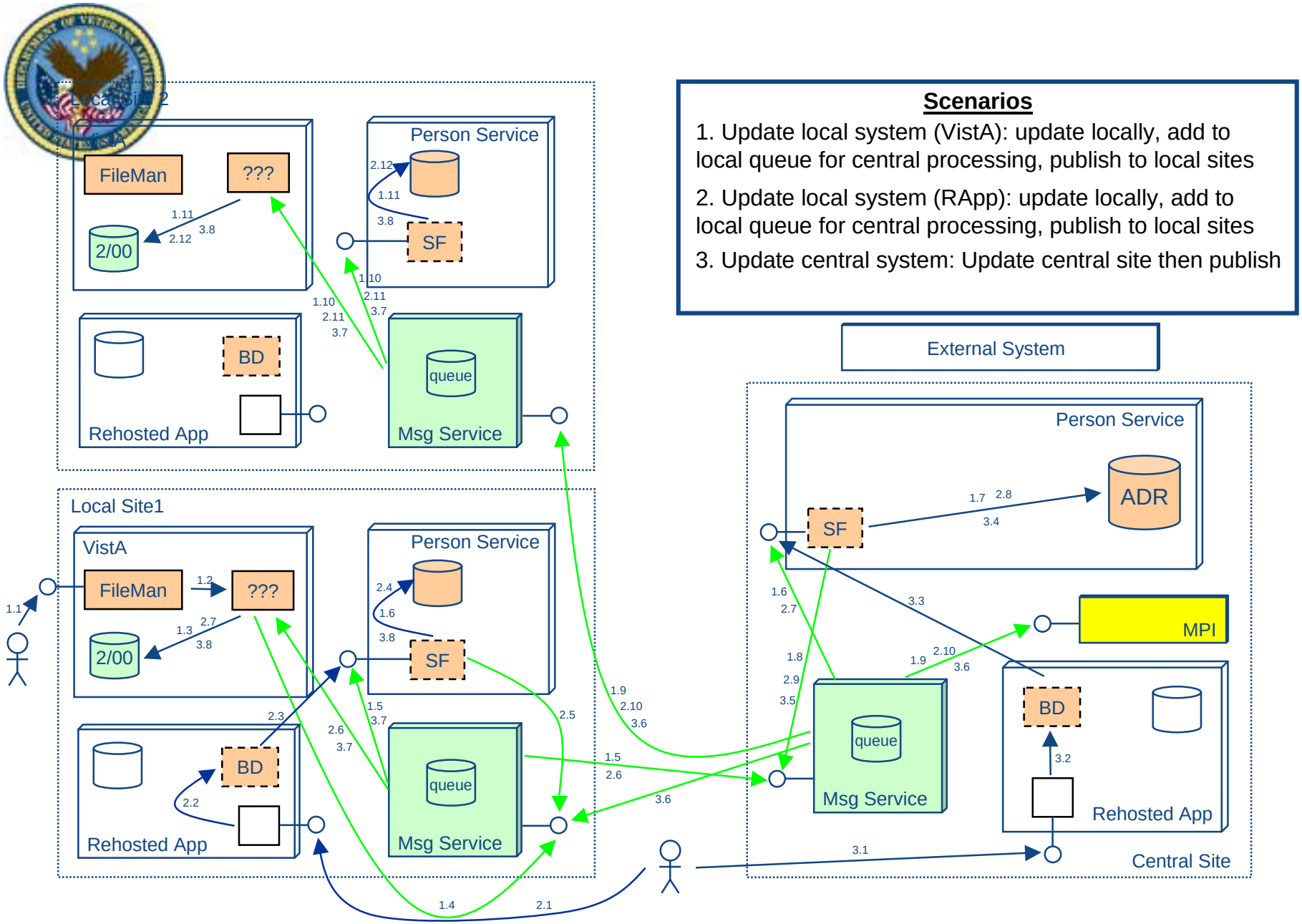
PSD Messaging Architecture Scenario-S2: "A31" Message Flow [Transitional State With ADR-Lite and Legacy VistA]





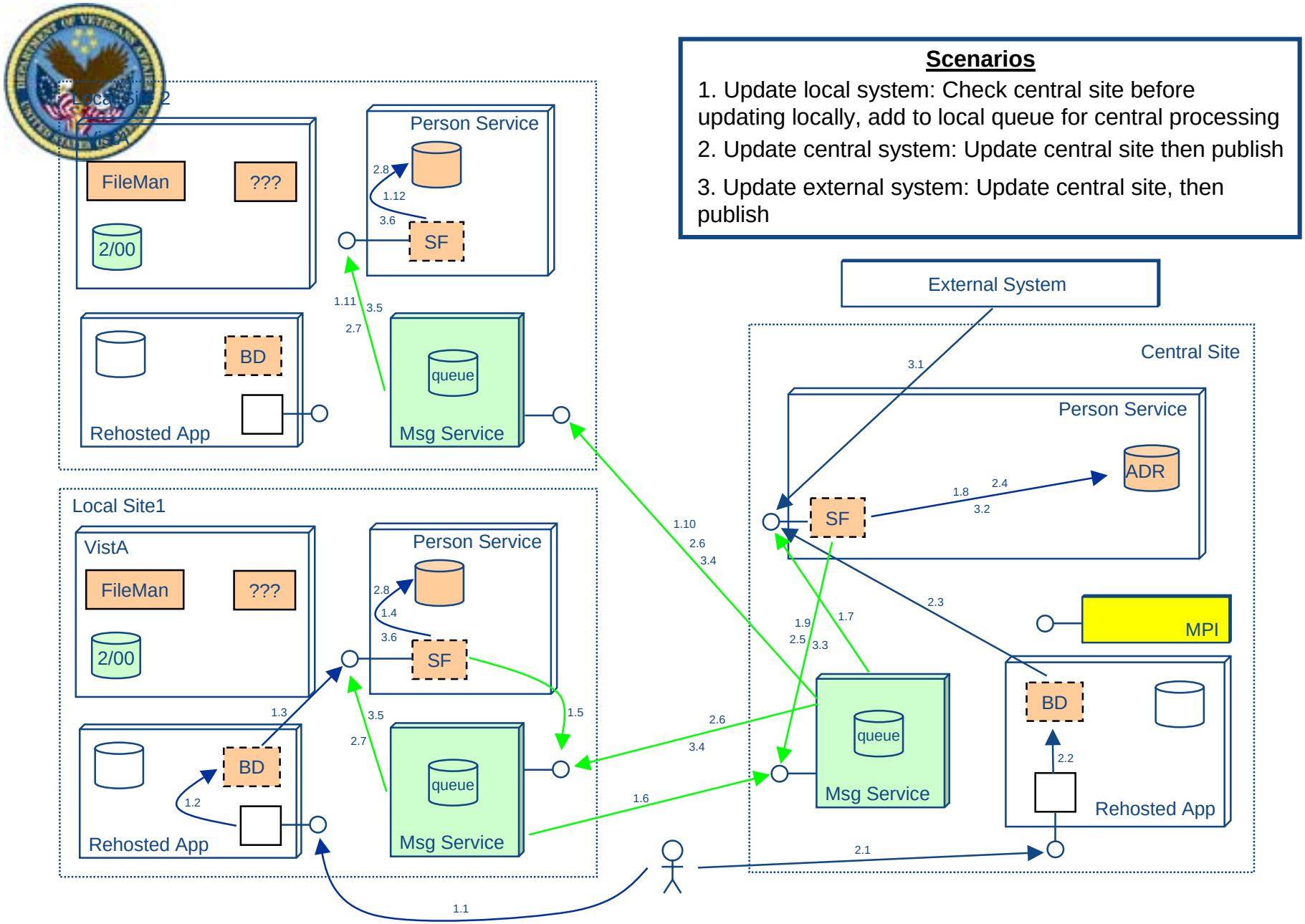
Scenarios

1. Update local system: update locally, add to local queue for central processing, publish to local sites



Scenarios

1. Update local system (VistA): update locally, add to local queue for central processing, publish to local sites
2. Update local system (RApp): update locally, add to local queue for central processing, publish to local sites
3. Update central system: Update central site then publish


Scenarios
1. Update local system: Check central site before updating locally, add to local queue for central processing
2. Update central system: Update central site then publish
3. Update external system: Update central site, then publish
Local Site2
FileMan
???
2/00
Person Service
SF
BD
Rehosted App
queue
Msg Service
Local Site1
VistA
FileMan
???
2/00
Person Service
SF
BD
Rehosted App
queue
Msg Service
External System
Central Site
Person Service
ADR
SF
MPI
BD
Rehosted App
queue
Msg Service
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
1.10
1.11
1.12
2.1
2.2
2.3
2.4
2.5
2.6
2.7
2.8
3.1
3.2
3.3
3.4
3.5
3.6