

# NIH Enterprise Architecture



## Application Integration — Technology Architecture

Version 1.0

26 July 2004



## Table of Contents

<b>1.0</b>	<b>Executive Summary</b> .....	<b>1</b>
<b>2.0</b>	<b>Introduction</b> .....	<b>1</b>
2.1	Application Integration Domain Team .....	3
2.2	Scope.....	3
2.3	Application Integration Domain in the NIH EA Framework.....	4
2.4	Principles .....	5
2.5	Related Topics .....	7
2.6	Benefits of the Application Integration Architecture.....	8
2.7	Summary of Key Findings .....	9
<b>3.0</b>	<b>Current State of Application Integration</b> .....	<b>10</b>
3.1	Extension Systems .....	10
3.2	Integration of Enterprise Systems .....	12
3.3	Grants Management .....	13
3.4	Grants and Financial Applications.....	14
3.5	Personnel and Related Applications .....	16
3.6	Clinical Center (Broker).....	18
3.7	Integration Mechanisms .....	19
3.8	Import/Export Matrix.....	19
3.9	Integration Issues.....	21
<b>4.0</b>	<b>Current Initiatives</b> .....	<b>23</b>
4.1	Summary of Initiatives.....	23
4.2	Clinical Center (CRIS Project).....	24
4.3	GSA Integrated Acquisition Environment .....	26
<b>5.0</b>	<b>Enterprise API Design Patterns (Future State)</b> .....	<b>27</b>
5.1	Pattern: Service-Oriented Architecture (SOA).....	28
5.2	Pattern: Web Services Architecture (WSA).....	30
5.3	Pattern: Application Program Interface (API) .....	31
5.4	General Guidelines for Integration/API Design .....	33
<b>6.0</b>	<b>Integration Design Patterns (Future State)</b> .....	<b>34</b>
6.1	Pattern: Basic Communication .....	34
6.2	Basic Integration Patterns (Introduction).....	37
6.3	Pattern: Data Consistency .....	39
6.4	Pattern: Multi-Step Process .....	40
6.5	Pattern: Composite Applications .....	41

6.6	Pattern: Large-Scale Integration .....	43
6.7	Pattern: Broker/Operational Data Store/Warehouse .....	44
<b>7.0</b>	<b>Application Integration Bricks.....</b>	<b>46</b>
7.1	Definitions and Taxonomy.....	47
7.2	Brick: Data Management Middleware .....	49
7.3	Brick: Communication Middleware.....	51
7.4	Brick: File Transfer Middleware.....	53
7.5	Brick: Integration Broker Suites (IBS).....	54
7.6	Brick: Integration Adapters.....	57
7.7	Brick: Business Process Managers (BPM Tools).....	59
7.8	Brick: Integration Middleware — Gateways .....	61
7.9	Brick: Web Services .....	62
<b>8.0</b>	<b>Gap Analysis .....</b>	<b>66</b>
8.1	Enterprise API Design and Implementation .....	66
8.2	Integration Patterns and Related Middleware .....	66
<b>9.0</b>	<b>Next Actions.....</b>	<b>67</b>
<b>10.0</b>	<b>Change History/Document Revisions.....</b>	<b>68</b>
<b>11.0</b>	<b>Appendix A — Glossary of Terms/Acronym Key .....</b>	<b>69</b>
<b>12.0</b>	<b>Appendix B — Current State Survey Results .....</b>	<b>77</b>
<b>13.0</b>	<b>Appendix C — Introduction to Web Services .....</b>	<b>86</b>
	What is a Web Service? .....	87
	What is SOAP?.....	87
	What is WSDL? .....	88
	What is UDDI?.....	88
<b>14.0</b>	<b>Appendix D — Introduction to Service-Oriented Architecture ..</b>	<b>90</b>
	Services and SOA .....	91
	Service Implementation .....	92
	Service Invocation .....	93
	SOA and Web Services .....	94
	When To Use SOA .....	95
	Benefits of SOA: Reality vs. Hype .....	96

## Table of Contents

(Continued)

### List of Figures

Figure 1.	NIH Enterprise Architecture Framework .....	4
Figure 2.	Extension Systems .....	10
Figure 3.	IMPAC II Interface Design .....	11
Figure 4.	Flows Between Enterprise Applications.....	12
Figure 5.	Grants Cluster Applications .....	13
Figure 6.	Flow Diagram: Grant, Financial and Related Data .....	15
Figure 7.	Flow Diagram: Personnel and Related Data.....	17
Figure 8.	Current Clinical Care Infrastructure .....	18
Figure 9.	Planned CRIS Clinical Care Infrastructure.....	25
Figure 10.	GSA Integrated Acquisition Environment.....	26
Figure 11.	Layered Service Architecture.....	27
Figure 12.	Logical Design Pattern: SOA .....	28
Figure 13.	Logical Pattern: Web Services Architecture (WSA) .....	30
Figure 14.	Logical Design Pattern: Application Program Interface (API) .....	31
Figure 15.	Web Services and XML Provide Loose Coupling (Example).....	32
Figure 16.	Logical Design Pattern: Basic Communication Models .....	35
Figure 17.	Three Integration Problems .....	37
Figure 18.	Integration Solution Characteristics .....	38
Figure 19.	Logical Design Pattern: Data Consistency.....	39
Figure 20.	Logical Design Pattern: Straight-Through Processing (STP).....	40
Figure 21.	Logical Design Pattern: Composite Application .....	41
Figure 22.	Logical Design Pattern: Large Scale Integration Hub-and-Spoke Solution.....	43
Figure 23.	Logical Design Pattern: Broker/Operational Data Store/Warehouse .....	44
Figure 24.	Technology Planning “Brick” .....	46
Figure 25.	Middleware Taxonomy.....	47
Figure 26.	Data Management Middleware .....	49
Figure 27.	Communication Middleware .....	51
Figure 28.	Integration Brokers .....	54
Figure 29.	Integration Broker Suite .....	55
Figure 30.	Integration Adapters .....	57
Figure 31.	Business Process Managers .....	59
Figure 32.	Gateways.....	61
Figure 33.	Components of Web Services .....	62
Figure 34.	Emerging Web Services Standards .....	65
Figure 35.	SOAP.....	87
Figure 36.	WSDL .....	88
Figure 37.	UDDI.....	89
Figure 38.	Service Scope vs. Granularity .....	91
Figure 39.	SOA — The Architecture of Interfaces .....	92
Figure 40.	SOA — What Happens Behind the Interface.....	93
Figure 41.	SOA — What Happens On the Wire.....	94

## List of Tables

Table 1.	NIH Enterprise Architecture Matrix .....	5
Table 2.	Application Integration Alignment With the NIH Enterprise Architecture Matrix.....	5
Table 3.	Application Integration Principles.....	6
Table 4.	Endorsed Principles.....	7
Table 5.	Import/Export Matrix (Partial).....	20
Table 6.	Guidelines for SOA Design.....	29
Table 7.	Data Management Middleware Brick.....	50
Table 8.	Communication Middleware Brick .....	52
Table 9.	File Transfer Middleware Brick .....	53
Table 10.	Integration Brokers Brick .....	56
Table 11.	Adapters Brick .....	58
Table 12.	Pure-Play vs. Integrated BPM .....	60
Table 13.	Business Process Managers Brick .....	60
Table 14.	Gateways Brick.....	61
Table 15.	Web Services Brick .....	64
Table 16.	When To Use SOA .....	95

## 1.0 Executive Summary

National Institutes of Health (NIH) has established an enterprise architecture program to develop a coordinated direction for designing and implementing infrastructure and technology solutions at NIH. The business architecture addresses processes and is the driving context for the information architecture, which addresses the application systems and information at NIH. The technology architecture focuses on the supporting technical components, both hardware and software. Application integration technology is a domain in the technology architecture that addresses the technical infrastructure for enabling applications to communicate with each other. The information architecture also contains an integration architecture that will specify how those interfaces need to be implemented from a business functionality standpoint.

This report specifies the technical architecture for integration of and with enterprise applications at the NIH. It lists principles that should be applied to future application integration initiatives, and highlights the endorsement of existing principles from other domain teams. It continues to note the business and technical benefits that application integration can provide, and the summary of key findings from the domain team workshops.

The primary function of the domain team workshops was to discuss the application integration design patterns, which are listed in this report, as well as to define the technical standards, which are presented as bricks. Implicit in these principles, patterns, and bricks is a more comprehensive and flexible approach to supporting integration at NIH. Analysis of the current state found primitive levels of integration implemented in an ad hoc manner that increased costs and reduced efficiency. Integration should be more intentional, with interfaces accommodating other applications and the needs of other organizations. The technologies underlying integration should be more sophisticated and more broadly leveraged so that all NIH applications can more easily interoperate with the enterprise applications.

Throughout the ten-week effort, technologists representing many ICs also developed some recommendations and next steps that NIH should complete to move the current state of application integration to the future state.

## 2.0 Introduction

Application integration is a major effort at NIH and other large enterprises. Gartner Research estimates that integration typically accounts for about 25 to 35 percent of the total cost of application design, development and maintenance<sup>1</sup>. While there is currently no way to break out this cost for NIH, all indications indicate that it is typical — or even higher, given the large number of “extension systems.”

The IT industry has been in a period of trial and error, trying to find the best designs and technologies to meet the challenge of integration. There is a consensus that integration is vital to most modern business strategies, but there has been confusion about how to

---

<sup>1</sup> This is usually a hidden cost, not separately broken out.

do it. At NIH, integration up to now has been accomplished via a wide variety of ad-hoc methods and mechanisms.

But now many of the core design principles and technologies for improving integration are better understood. At many enterprises, large-scale application integration is accomplished by using specialized integration tools and middleware. Standards such as Extensible Markup Language (XML), Java, .Net, and Web services are helping reduce the cost and time required to integrate applications with each other.

Section 3.0 discusses the current state of application integration at NIH. The main focus of this report is to set the future state direction for integration technology. However, the diagrams in this chapter provide additional information about the existing information flows between NIH applications.

Section 4.0 provides target state design patterns for the application program interfaces (APIs) that enterprise applications will provide to extension systems.

Section 5.0 provides target state design patterns for large-scale integration. Both application-to-application (A2A) and business-to-business<sup>1</sup> (B2B) integration can benefit from these patterns.

Section 6.0 includes the “bricks” which describe individual technology elements. Current (baseline) state information and target state plans are provided.

Several appendices are also included in this report, which provides some technical context and background for many of the concepts addressed, by the patterns and bricks.

---

<sup>1</sup> This is a common industry term. In this document, “business-to-business” also encompasses “government-to-business” and “government-to-government.”

## 2.1 Application Integration Domain Team

This report comprises the compilation of findings and recommendations derived from the joint NIH-Gartner enterprise architecture project team. A team of nine subject matter experts from various institutes and centers (ICs) and Center for Information Technology (CIT) worked together for 12 weeks to develop the application integration architecture patterns and bricks that are presented in this report. The following ICs and their representatives contributed to this effort:

- Jean Babb, NCRR
- Steve Bergstrom, CC
- Belinda Seach, NHLBI
- Toni Calzone, NIAAA
- Ken Molenda, NIAID
- Debbie Bucci, CIT
- Vivek Kamath, NINDS
- Shanthi Himachalapathy, OD
- Tracy Soto, OD
- Steve Hughes, OD.

## 2.2 Scope

### 2.2.1 Definition: Application Integration

The definition<sup>1</sup> of application integration is “making independently designed application systems work together”. This report focuses on setting the direction for the technology that supports application integration, not the listing of all the required interfaces or their specifications.

### 2.2.2 Subject Areas

There are three basic subjects/topics that are in scope:

1. Integration of Extension Systems with NIH enterprise applications
2. Integration between two or more NIH enterprise applications
3. Integration of NIH applications with applications outside NIH.

*Extension System* is an NIH term that is used to describe any application add-ons that extend the capabilities of a core application.

*Enterprise Applications* are major computer applications that have “enterprise scope” as defined in the Application Architecture report<sup>2</sup>.

### 2.2.3 Overlap with Application Development Technology

Application development (AppDev) technology is not in scope. However, there is some overlap between AppDev and the Enterprise API design patterns: application integration uses certain technologies and techniques to connect independent application systems, and AppDev uses those same mechanisms to connect components and services within an application system.

---

<sup>1</sup> Source: Gartner Research

<sup>2</sup> Separately published and available on the EA Portal, September 2003.



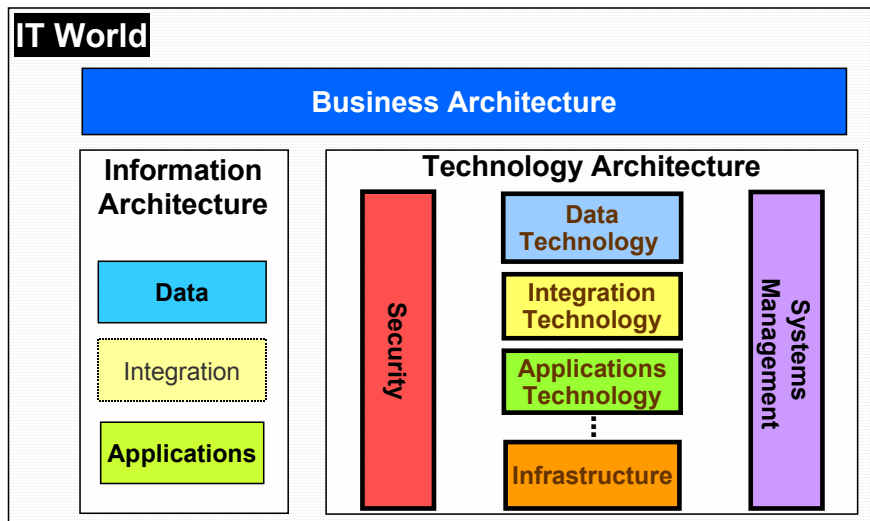
## 2.3 Application Integration Domain in the NIH EA Framework

The *NIH Enterprise Architecture Framework* and *NIH Enterprise Architecture Matrix* are based on the Federal Enterprise Architecture Framework (FEAF) and the FEAF Matrix<sup>1</sup>.

The NIH EA Framework recognizes three distinct component architectures: the business architecture, information architecture and technical architecture. The NIH EA Framework is illustrated in Figure 1.

The NIH Enterprise Architecture Matrix provides five potential *perspectives* or views of the architecture, at increasing levels of detail. The NIH EA Matrix is shown in Table 1.

Figure 1. NIH Enterprise Architecture Framework



The application integration domain is part of the technology architecture within the NIH EA framework. There is a related domain within the information architecture that deals with how the various application systems and data assets are integrated; while that is a separate matter, this report does shed some light on the current state.

<sup>1</sup> Level IV of the FEAF, derived from the Zachman Framework

**Table 1. NIH Enterprise Architecture Matrix**

	<b>Data Architecture</b>	<b>Application Architecture</b>	<b>Technology Architecture</b>
Planner Perspective	List of enterprise business objects	List of business processes and multi-enterprise processes.	List of business locations and business partners
Owner Perspective	Semantic Model	Business process models (including multi-enterprise)	Business logistics system and multi-enterprise logistics
Designer Perspective	Logical design patterns; use enterprise business objects	Logical design patterns, by style	Integration technology for enterprise systems
Builder Perspective	Physical design patterns; use shared database if applicable	Logical design patterns, by style	Physical design patterns; use bricks from TRM or request a waiver. TRM includes security, NIH network and other infrastructure
Subcontractor Perspective	Project scope	Use common services or APIs, if defined	

This architecture report focuses on the Technology Architecture column and the Planner, Owner, Designer and Builder perspectives.

**Table 2. Application Integration Alignment With the NIH Enterprise Architecture Matrix**

	<b>Data</b>	<b>Applications</b>	<b>Technology</b>
Planner View	N/A	N/A	Entire report. Chapter 3, Current State, should be especially useful to planners.
Owner View	N/A	N/A	Multi-enterprise logistics are covered to a limited extent in Chapter 4, Current Initiatives, and Section 6.6, Large Scale Integration.
Designer View	N/A	N/A	See the target state design patterns in Chapters 5 and 6.
Builder View	N/A	N/A	Chapter 7 specifies the integration technology “bricks.”

## 2.4 Principles

The principles defined below are high-level statements of the fundamental values that guide application integration at NIH. The team used a consensus approach in developing the principles.

When the integration domain team found that a candidate principle was similar to an existing one (from another domain) the usual response was to “endorse” that principle rather than restating it here. These endorsed principles are reiterated here because the impact of not following them is particularly detrimental to effective application integration.

## 2.4.1 Application Integration Principles

**Table 3. Application Integration Principles**

Principle	Rationale
<p><b>1. Plan for Integration</b> — Application integration is required in the early project planning process for enterprise applications. It must be included in the project plan and key deliverables such as requirements, analysis and design.</p>	<p>In the past, application integration has often been an afterthought, resulting in missing or inferior interfaces built quickly on a tight budget.</p>
<p><b>2. Loosely Coupled Interfaces</b> — Interfaces will be <i>loosely coupled</i><sup>1</sup>, backward compatible, self-describing, and offer a low impact to the enterprise if changed.  (An interface service is <i>tightly coupled</i> to the application of which it is a part.)</p>	<p>Loosely coupled interfaces are preferred, because when interfaces between independently designed applications are tightly coupled, they are (1) less general and (2) are more likely to result in undesired side effects when changed.</p>
<p><b>3. Publish Integration Points</b> — “Public” inputs and outputs of an application must be known, published and understood to promote open data exchange and interfaces for enterprise application integration. Diagrams of connections and data syntax and semantics should be published to promote re-use.</p>	<p>Lack of understanding results in ad-hoc integration that is inconsistent and inefficient. “Private” interfaces should not be published or used, as their stability is not guaranteed.</p>
<p><b>4. Platform Independent, Open Standards</b> — Open standards and industry standards are preferred for enterprise application integration solutions (mandatory for integration outside NIH); mechanisms should be language- and platform-independent.</p>	<p>This principle will allow for easier integration with the heterogeneous platforms and programming languages that are the norm at NIH today.</p>
<p><b>5. Reusable, Shared Services</b> based on a service-oriented architecture (SOA), and other forms of APIs, are preferred to direct data access.</p>	<p>This approach will minimize direct access to data; thus lowering the risk of bypassing the business logic or compromising data integrity.</p>
<p><b>6. Integration Change Management</b> — Software Configuration and Change Management (SCCM) is mandatory for application integration interfaces.</p>	<p>SCCM for interfaces is especially important, because multiple applications may be impacted.</p>
<p><b>7. Minimize Application Impact</b> — Enterprise application integration mechanisms used should be non-invasive to the applications as much as possible. For instance, data transformation should be done externally from the applications involved.</p>	<p>Adding application integration code to existing applications (1) delays integration projects and (2) increases the application maintenance burden.</p>

<sup>1</sup> *Loose coupling* means that services (e.g., enterprise APIs) are designed with no affinity to any particular service consumer. Inside the service, nothing is assumed as to the nature of the consumer. Thus, a service is fully de-coupled from a service consumer. However, the service consumer is dependent on the service (that is, it embeds literal references to service interfaces). The service is also responsible for exception handling. The result is a semi-coupled (or *loosely coupled*) architecture.

## 2.4.2 Endorsed Principles

**Table 4. Endorsed Principles**

Endorsed Domain, #	Endorsed Principle	Comment or Endorsement Rationale
Application Principle #7	<b>Meet Broad Needs</b> — Enterprise Applications must meet broad needs. The requirements and design must be published prior to development, and all IC's must have the opportunity to comment.	Integration Principles #1, <i>Plan for Integration</i> and #6, <i>Integration Change Management</i> , reference this related principle.
Application Principle #8	<b>SCCM Processes Are Required</b> — The Software Configuration and Change Management (SCCM) process must be documented, and all parties must adhere to it.	Integration Principle #6, <i>Integration Change Management</i> , references this related principle. SCCM for interfaces is especially important because multiple applications may be impacted.
Data Principle #2	<b>Data Creation</b> — All enterprise data should be captured once at the point of its creation.	Application integration attempts to reduce the amount of duplicate data entry.
Data Principle #9	<b>Primary Data Source</b> — All enterprise data will have an authoritative, official, primary data source that is the location for all Create, Update and Delete actions.	Application integration should recognize the primary data source.
Data Principle #5	<b>Data Ownership</b> — All enterprise data will have an identified business owner and a technical owner.	Having accurate and consistent data points will make application integration easier.
Data Principle #6	<b>Standardization of Shared Data</b> — Enterprise data standards should be identified when the value of interoperability with other information systems exceeds the value of uniqueness.	Shared data is often used in application integration — having this standardized will aid in application integration efforts.
Data Principle #7	<b>Standardization of Common Data</b> Enterprise data standards should be identified when the value of commonality across NIH exceeds the value of uniqueness.	Standardization of common data may reduce the duplication of effort and provide improved application integration.
Enterprise Principle #2	<b>Business Priority</b> — Information systems exist to support the needs of the business. Therefore, the NIH Enterprise Architecture must support the enterprise vision, business strategies and plans.	Requirements for application integration should be based on business needs, and may vary from application to application.

## 2.5 Related Topics

### 2.5.1 Application Architecture

The Application Architecture report<sup>1</sup> (a part of the information architecture) and the data gathered by that domain team's surveys were used extensively by this team to

<sup>1</sup> Version 1, September 2003

understand the current state. As this document includes updates to that original data, this document supersedes some portions of the Application Architecture document.

## **2.5.2 Identity Management**

Identity management and application access/permissions are important to application integration for multiple reasons, including:

- If a service provided by one application is shared by another application, the user of the “composite” application needs to be logged into two applications at once, transparently.
- When two applications are integrated behind the scenes, using messages or a similar mechanism, no “user” in the normal sense is involved.

Identity management is beyond the scope of this document. However, when future integration solutions are implemented, the implementation teams should work with CIT’s NIH Login experts to provide a seamless solution.

## **2.6 Benefits of the Application Integration Architecture**

In addition to the general benefits of enterprise architecture, this application integration technical architecture, when implemented, provides the following specific benefits.

### **2.6.1 Business Benefits**

- Reduce latency for data
- Provide real-time data capture and analysis
- Improve business efficiencies
- Improve responsiveness, delivery, or service (effectiveness)
- Reduce costs of integration efforts (integration typically consumes 40 percent of the cost of application development and maintenance)
- Ensure quality, currency and source of data
- Provide process streamlining and accountability
- Obtain agreement from multiple systems or business units on the facts (modern enterprises generally have redundant versions of data regarding customers, products, employees and other entities used by various application systems)
- Focus on zero-latency application integration.

### **2.6.2 Technical Benefits**

- Reduction in the need for duplicate data entry
- Reduction in data inconsistencies
- As driven by application requirements, data will be current, not several days out of date, due to near real-time updates

- Reduction of integration development and maintenance costs due to hub-and-spoke (rather than point-to-point) interfaces, a consistent architecture and specialized tools for building interfaces
- Reduction in the cost of developing extension systems due to consistent, superior interfaces
- Technical benefits of zero-latency integration.

## 2.7 Summary of Key Findings

The following are the key decisions regarding technology for Application Integration

- Web services are the preferred approach to the deployment of APIs for services
  - These provide a standard approach for the implementation of SOA.
  - Application development technologies selected for use should support the generation of Web services interfaces to application components.
  - Standards for Web Services including Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Universal Description, Discover and Integration (UDDI), and Web Services Description Language (WSDL) are considered strategic to the NIH architecture.
- An integration broker suite (IBS) will be selected to facilitate the development of a hub-and-spoke topology of application integrations where appropriate.
  - This will decrease the number of point-to-point integrations between applications and reduce the overall complexity of the application infrastructure.
- Selection of a strategic business process manager toolset will be influenced by the IBS selected. Most IBS technologies include integrated business process management (BPM) capabilities.
  - Handysoft Bizflow will continue to be used as a pure-play BPM tool in the near-term.
- Direct integration between databases will be accomplished through the use of database gateways over the near-term.
  - These include Oracle Transparent Gateway and Linked-Server Database Gateway for SQL Server databases.
- A variety of file transfer tools will continue to be used for the point-to-point batch integrations.
- Supported tools will include Sterling Commerce Connect:Direct, Digital Imaging and Communications in Medicine (DICOM) transfer, and the File Transfer Protocol (FTP) clients and servers provided by standard operating systems.

## 3.0 Current State of Application Integration

### 3.1 Extension Systems

The distinction between *extension systems* and *integrated applications* is not always clear.

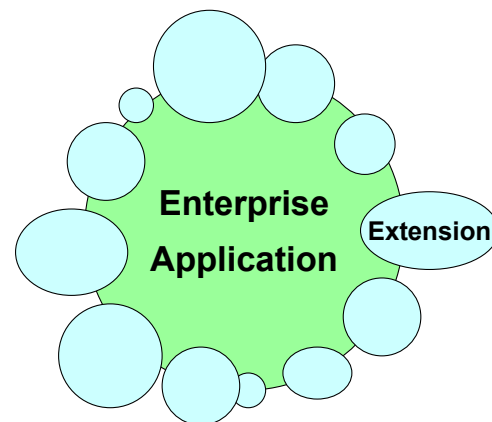
**Conceptually**, extension systems can be pictured as shown in the diagram on the right. There is an enterprise application in the middle (large bubble), used by all or most ICs. But many of the ICs add on functionality that reflects their own special needs (the smaller bubbles).

Some of the extension systems may be large or important in their own right. In the Application Domain Team survey<sup>1</sup>, ten of the extension systems were rated as “enterprise class” applications themselves! Over thirty percent of the applications surveyed were categorized as extension systems.

**Physically**, the methods used to “extend” the central application vary greatly. Extensions might be accomplished via bi-directional transfer of data, direct access to a database, a designated application programming interface (API), etc. The NIH will benefit from standardization of methods in this area.

Extension systems are application add-ons that extend the capabilities of a core application. This might be done for two reasons: (1) because the core application is missing generally needed functionality or (2) to meet needs that are unique to an IC.

Figure 2. Extension Systems



Observations from the Application Domain report<sup>2</sup>:

- Eighty-nine of the applications surveyed — *over one third* — were extension systems. Extension systems such as VSOF and QVR were counted only once, even though they are used by multiple ICs.
- Over half (53) of the 89 extension systems reported extended IMPAC II (eRA) directly, or extend TABS, which is itself an extension of IMPAC II.
- Some extension systems interface with multiple NIH applications. For instance, NCI's TFS extends HRDB, DW, FPS II (ADB), ISB and Commissioned Corps.
- Some extensions systems augment DHHS applications.

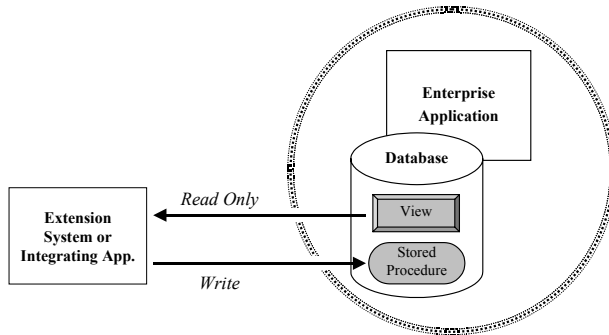
<sup>1</sup> See the Application Architecture Report, September 2003

<sup>2</sup> As of September 2003. By May 2004, over 270 extension systems were identified.

Each enterprise application (which can be extended) provides some sort of mechanism for this, formal or informal, documented or otherwise. Custom-developed NIH enterprise applications rarely provide stable, well-engineered application programming interfaces (APIs)<sup>1</sup>. The problem is not as severe with COTS applications because the market has demanded good APIs.

IMPAC II, the most-extended enterprise application at NIH, provides controlled access to its database through database views. Updates are handled through an application programming interface (API) written in Oracle's proprietary Programming Language for SQL (PL/SQL), stored in the database as stored procedures. The two mechanisms are depicted in the figure below.

**Figure 3. IMPAC II Interface Design**



**Notes:**

1. The database is read through database views.
2. The database is updated by invoking a proprietary stored procedure.
3. Most operations/actions are performed one record at a time.
4. Transaction control (commit/rollback) is handled by the extension system.

Source: Gartner

<sup>1</sup> This is a common problem at most large enterprises.



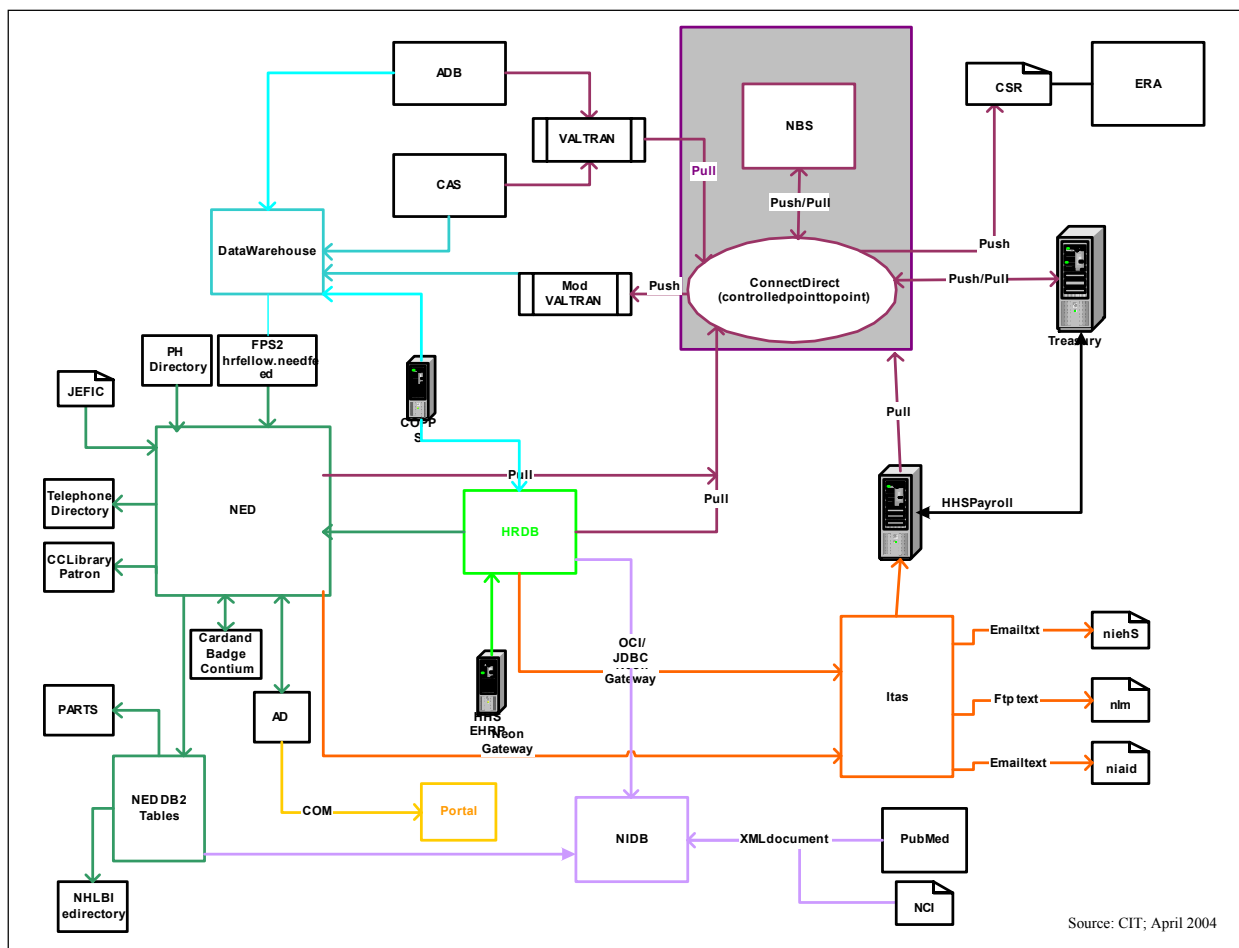
## 3.2 Integration of Enterprise Systems

NIH applications often exchange data with other NIH, DHHS and “outside world” applications.

### 3.2.1 Internal to NIH

NIH applications frequently exchange data with each other. Major applications are currently integrated through a series of point-to-point interfaces; a variety of mechanisms and protocols are used. The next figure is a flow diagram for enterprise applications, prepared by D. Bucci, the CIT team member, and updated and endorsed by the rest of the domain team.

Figure 4. Flows Between Enterprise Applications<sup>1</sup>



### 3.2.2 With the Outside World

NIH applications exchange data with DHHS, other agencies (e.g., EPA), contractors, research laboratories, universities, journal publishers, foreign government institutions (e.g., Swedish Riskline), commercial database repositories, public libraries and the

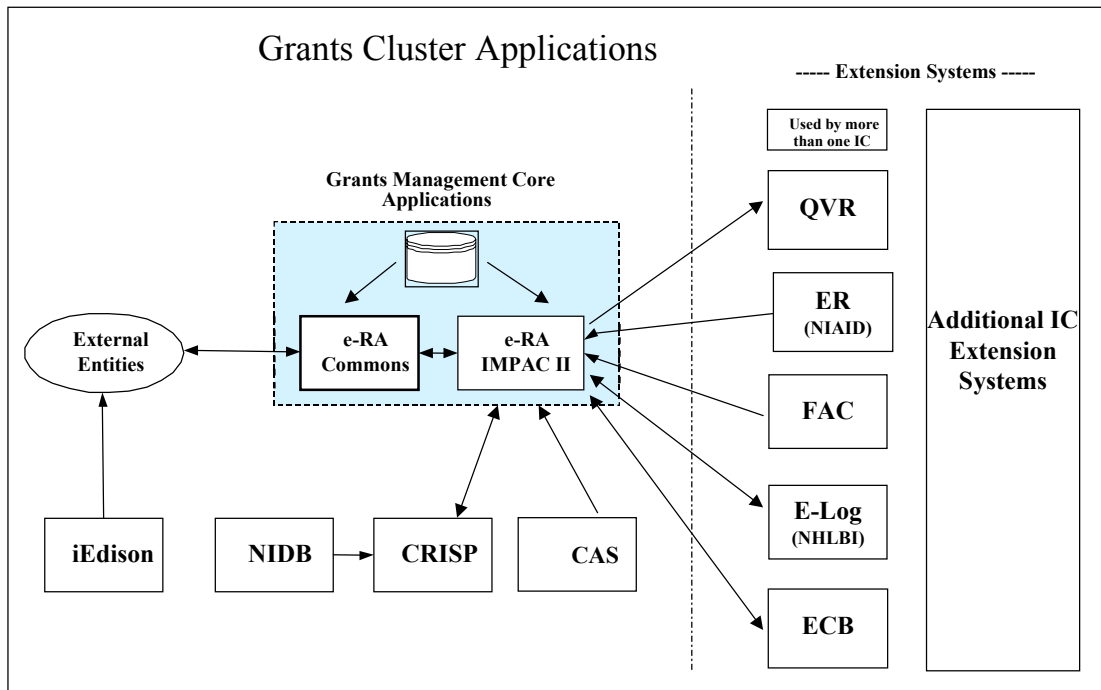
<sup>1</sup> As provided during current state workshop (05/04)

public at large. The number of direct and indirect outside users of NIH applications and data is huge.

### 3.3 Grants Management

The figure below shows the Grants Management core application along with closely related applications.

Figure 5. Grants Cluster Applications<sup>1</sup>



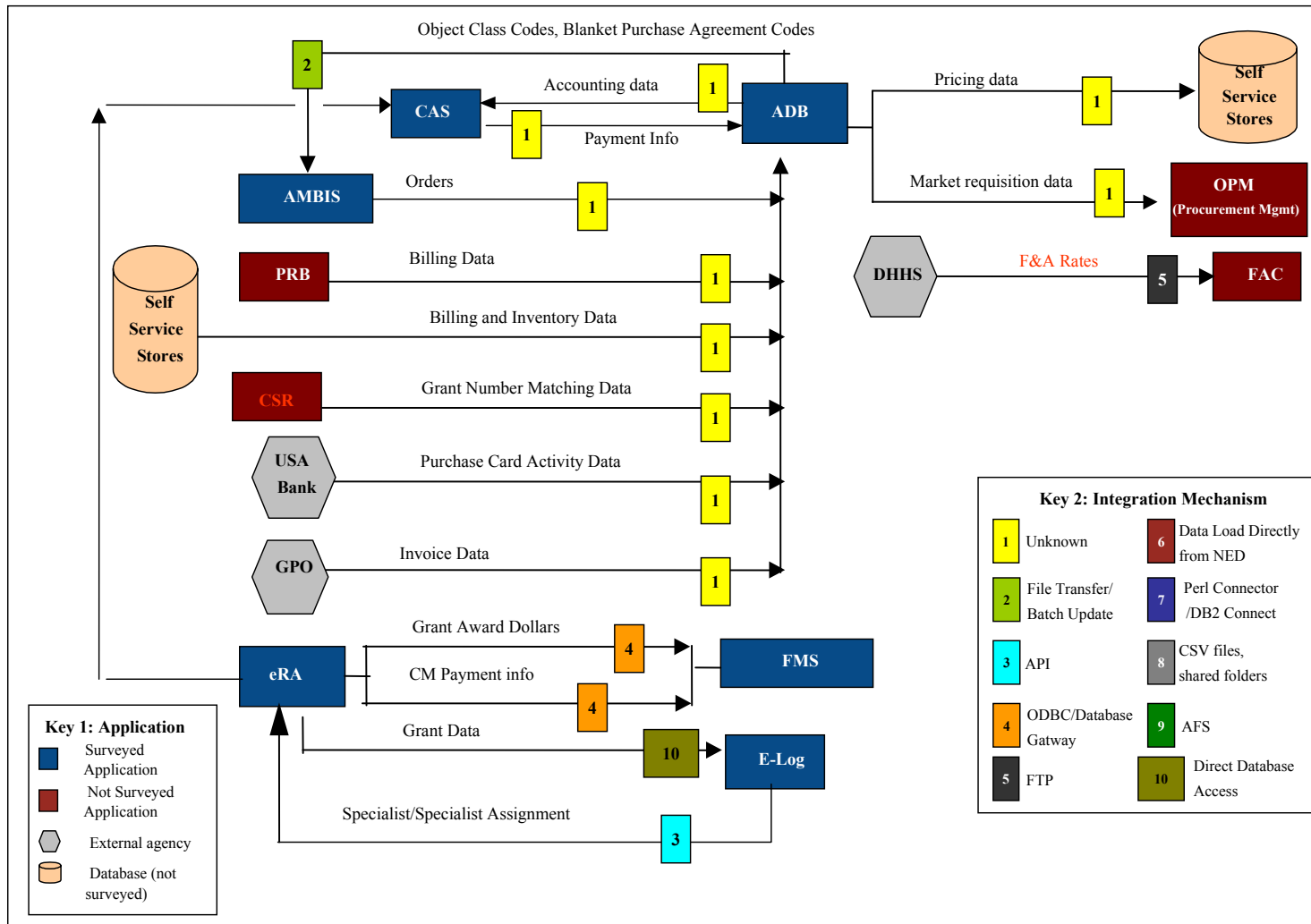
Source: NIH

<sup>1</sup> As provided during current state workshop (05/04)

### **3.4 Grants and Financial Applications**

The next figure shows the flows between grants and related financial applications, including some outside parties.

**Figure 6. Flow Diagram: Grant, Financial and Related Data<sup>1</sup>**

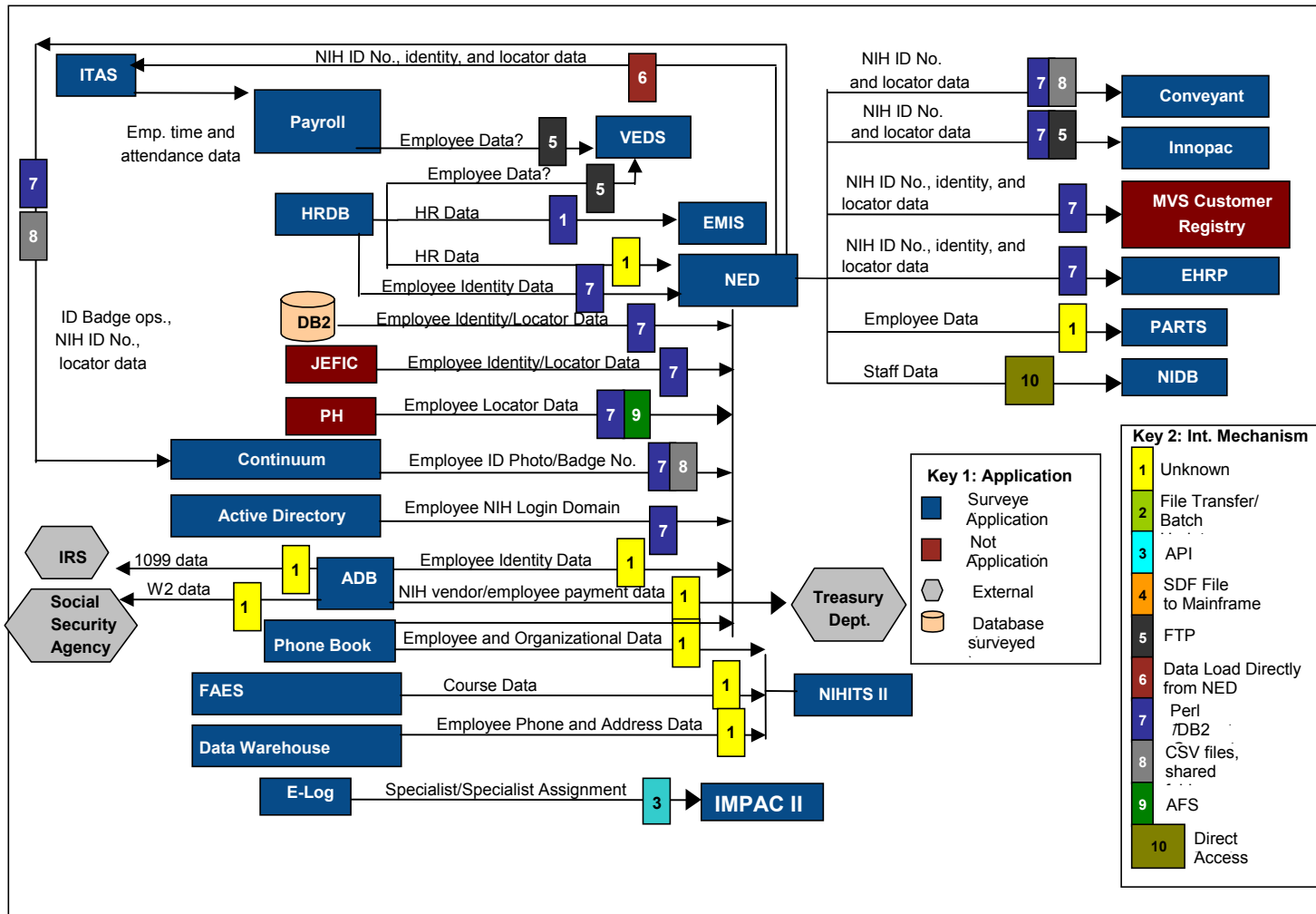


<sup>1</sup> As provided during current state workshop (05/04)

### **3.5 Personnel and Related Applications**

The personnel data environment is complicated by the fact that there is a complex mix of government employees, contract staff and outside scientists. The figure below depicts the current situation.

**Figure 7. Flow Diagram: Personnel and Related Data<sup>1</sup>**

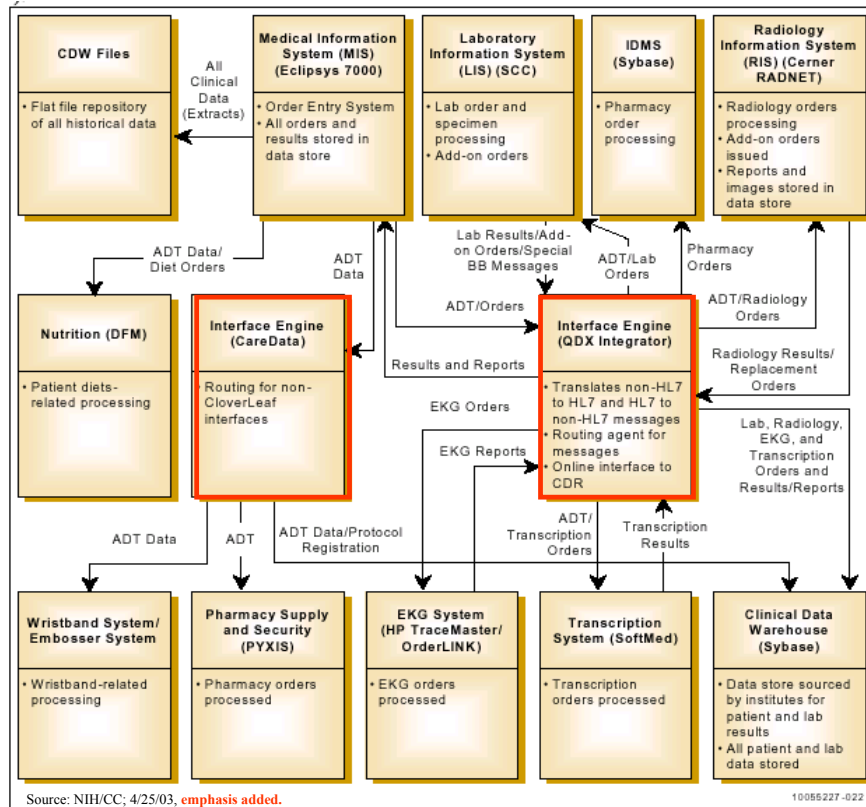


<sup>1</sup> As provided during current state workshop (05/04)

### 3.6 Clinical Center (Broker)

The Clinical Center is the only known place that an integration broker (QDX) is in use at NIH.

**Figure 8. Current Clinical Care Infrastructure**



Clinical Patient Care is supported through MIS, MAC/MIS and Web/MIS, multiple ancillary systems (radiology information system, laboratory information system, transcription system, EKG system), an interface engine and a Clinical Data Warehouse. MAC/MIS and WEB/MIS are used by Clinical Staff to place orders, view reports, enter results and enter documentation into MIS. Ancillary systems are used to track the order and the result of the order and to transfer the result back to MIS. The interface engine is used to transfer orders, result status and results between MIS and the ancillary systems. The Clinical Data Warehouse is used to store all historical clinical data for the purpose of providing data for clinical research.

### 3.7 Integration Mechanisms

Known integration mechanisms include:

- Direct database access
- XML formatted files
- Weekly/daily FTP
- Batch database read using stored procedure
- Put to public FTP server for licensed access
- PERL connector (DB2 connect)
- CSV files/shared folders/Andrew File System (AFS)
- Kermit
- Transfer through SCP (secure copy)
- Secure FTP
- MS DTS
- Oracle Advanced Queuing
- Batch update
- Excel spreadsheet
- SDF file to mainframe
- CD-ROM/Tape
- OCI calls
- ODBC/JDBC
- Cold Fusion API
- QDX Integrator
- SQL View API
- SSH tunnel
- Internet transfer
- BizFlow.

Diversity of integration mechanisms leads to greater complexity accompanied by lower reliability, lapses in data integrity and higher operational costs.

Recent advances in application integration technology, along with a systematic approach to the integration problem, can provide substantial relief. Chapters 5-7 provide an advanced integration architecture/approach.

### 3.8 Import/Export Matrix

The Application Architecture domain team constructed an import/export matrix (based on questionnaire answers) that shows mechanisms used for imports and exports; it should be regarded as preliminary until an analyst does further clarification and verification.

A portion of the matrix is shown below to illustrate the structure of the matrix and the diversity of integration mechanisms in place; the full matrix is available from the Office of the Chief Architect. Note that some of the applications named in the matrix did not appear in the survey; not much is known about them<sup>1</sup>.

---

<sup>1</sup> Additional data collection is expected in next refresh of the application architecture. Complete import/export matrix can be found in the Application Architecture report.



**Table 5. Import/Export Matrix (Partial)**

Application	Entity or Entity Type	Comment	M	EX	From Appl / Source	Import Mechanism	To Appl / Destination	Export Mechanism
ADB	1099 data	1099 reporting for: . Non-employee compensation . Participation in Clinical Center trial . Professional Services contract . Scientific Review & Evaluation Awards . Loan Repayment and Scholarship Program participants Data is collected from the ADB and CAS to report payments.		x			Internal Revenue Service	Statements to individuals annually in January. Information to SSA annually in February.
ADB	Accounting data	Accounting transactions generated by ADB actions are processed in the CAS		x			CAS	Daily
ADB	Billing and inventory data	Self Service Stores transmit a file of purchase and return activity to the ADB. The ADB uses this information to bill stores' customers and adjust inventory balances to reflect the day's activity.	x		Self Service Stores	Daily		
ADB	Billing data	PRB transmits print counts from the Copitrack Copy Center, Production Shop, and manual entries to the ADB for purposes of billing PRB customers.	x	or	PRB	Daily		
ADB	Data for reporting purposes	Data for reporting purposes.		x			DW	Daily
ADB	Grant number Matching data	Center for Scientific Review (CSR) creates a crosswalk file that allows the ADB to match Grant tracking numbers from the PRB (Printing and Reproduction Branch) Copy Center with Grant numbers.	x		CSR (Center for Scientific Review)	Daily		
ADB	Inventory data	ADB creates three files of activity for into RIMS. These files contain daily customer orders from the NIH inventories, items added to the NIH inventories and replenishment purchase orders pending delivery to NIH inventories.		x			RIMS (Robocom Inventory Management System)	Daily
ADB	Invoice data	GPO transmits a file of invoices to cover printing jobs performed in the previous month. These invoices are loaded in the ADB and matched against Service and Supply Fund Work Requests for billing to the ICs.	x		Government Printing Offices (GPO)	Monthly		

### 3.9 Integration Issues

The following table shows how different integration issues<sup>1</sup> have been manifested at NIH. The NIH Related Details column captures the negative impact of NIH’s current integration approach and, thereby, the drivers for the architecture recommendations in this document.

Issue Type	Issue Details	NIH Related Details
<b>Data Integrity</b>	Duplicate data entry	NHLBI currently has to enter contracts data twice in DCIS and the NHLBI contract tracking system — duplicate data entry has been mentioned as an issue in other cases as well.
	Lack of synchronization/maintenance Issues	ICs incur extra effort to synchronize with eRA: <ul style="list-style-type: none"> <li>■ eRA schema changes cause downloads to malfunction</li> <li>■ ICs manipulate local copy of data and if corrections to enterprise data aren’t made, there is an impact on enterprise reporting (since “good” data is not accessible outside of IC).</li> </ul> HRDB updates NED only every two weeks. The general rule is that the authoritative source for a person's legal name and home information is HRDB, which is updated by the AOs or the person in Employee Express. Business information, such as office phone and address, can be updated in NED with the self-service.
	Duplicate data	NIAID faces potential problems of faulty data replication (redundancy) in its central database server when using Oracle_Download.
	Inability to move data	NIAAA faces problems with exporting CAPS data into their in-house database — currently no way to export the data directly or for dual entry into both systems.
<b>Integration Planning</b>	Resource/budget issues	Teams should consider funding and resource planning (development and maintenance schedules) when planning application integration within NIH and DHSS systems.
	Lack of documentation of changes to interfaces	As there continues to be evolution of interfaces to other applications such as POPTRACK, the pain experienced to date is related to undefined or late changes to standard interfaces with timely documentation.

<sup>1</sup> Additional details around NIH integration issues can be found in the Appendix B — Current State Results

Issue Type	Issue Details	NIH Related Details
<p><b>Data Processing</b></p>	<p>Manual transfer of data</p>	<p>Common problem. Examples include:</p> <ul style="list-style-type: none"> <li>■ Until CRIS is implemented, CC is faced with some paper and pencil transfer of data amongst ancillary systems.</li> <li>■ NHLBI currently has a manual interface with the Fogarty tracking system that includes printing, manual lookup, and manual delivery.</li> </ul>
	<p>Real-time status of transactions</p>	<p>Integration with Central Accounting System (CAS) doesn't meet business needs. Real-time obligation is preferable to assuming transaction will complete within a day. Feedback loop for errors is not reliable.</p>
<p><b>Formats</b></p>	<p>Multiple formats within IC</p>	<p>NIAAA users have applications that use different file formats — utility programs for converting file format or direct ODBC to the other application is required for data sharing.</p> <p>There may be some value in working on standard formats (for date, for example) to alleviate data conversion efforts.</p>
	<p>Usage of standards</p>	<p>The lack of internal standards creates reliability issues and causes unnecessary work.</p> <ul style="list-style-type: none"> <li>■ The use of a message-based interchange could de-couple the information exchange from schema changes, up to a point. The notion of a stable API or stable message XML schema that changes only when the payload of the message changes but not when the eRA schema changes may be implemented.</li> <li>■ A desired area of growth related to integration points would be to increase the usage of XML at the points where we currently rely on flat file exchanges. This, of course, is constrained by the capability of our users (cooperative groups, et al.) and funding resources/competing priorities.</li> </ul>
<p><b>Platform Related</b></p>	<p>Web browser dependency</p>	<p>NCCR has integration pain points with supporting various Web browsers — no specifics provided.</p>
	<p>Platform dependency</p>	<p>NCCR has mentioned legacy data as an integration pain point.</p> <p>Integration with CAS is difficult due to residing on the mainframe.</p>

## 4.0 Current Initiatives

The domain team also asked ICs about existing plans, for the near future.

### 4.1 Summary of Initiatives

The initiatives mentioned during the current-state discussions are listed below. In some cases, multiple ICs had similar initiatives.

Some of the initiatives listed below have also been listed as part of the future-state application integration architecture at NIH, and can be found at the design-pattern/bricks level.

- Make secure file transfer mechanisms a standard, and phase out insecure file transfer.
- Investigate using Connect:Direct, a managed file transfer tool, as the transfer mechanism of choice.
- Use a hub-and-spoke architecture to integrate applications (i.e., CRIS, eRA), and evaluate tools accordingly.
- Automate and standardize the use of APIs.
- Automate and standardize how data is published between applications.
- Promote standards-based exchange, which may include expanded use of XML and ebXML.
- Phase out client server technologies and convert to Web-based (Web services) when possible.
- Phase out Microsoft Data Transformation Services (DTS) when used as an integration mechanism, and replace with C# technology.
- Investigate using enterprise-level (application-to-application) workflow tools for application integration.
- Use government off-the-shelf (GOTS) products like the IAE portal for application integration (see section 4.3).
- Roll out NIH Login, the single sign on functionality, to assist application integration.
- Merge LDAP structures (directory information) and standardize the definitions of roles to centralize user creation. This will help the NIH login effort.

## 4.2 Clinical Center (CRIS Project)

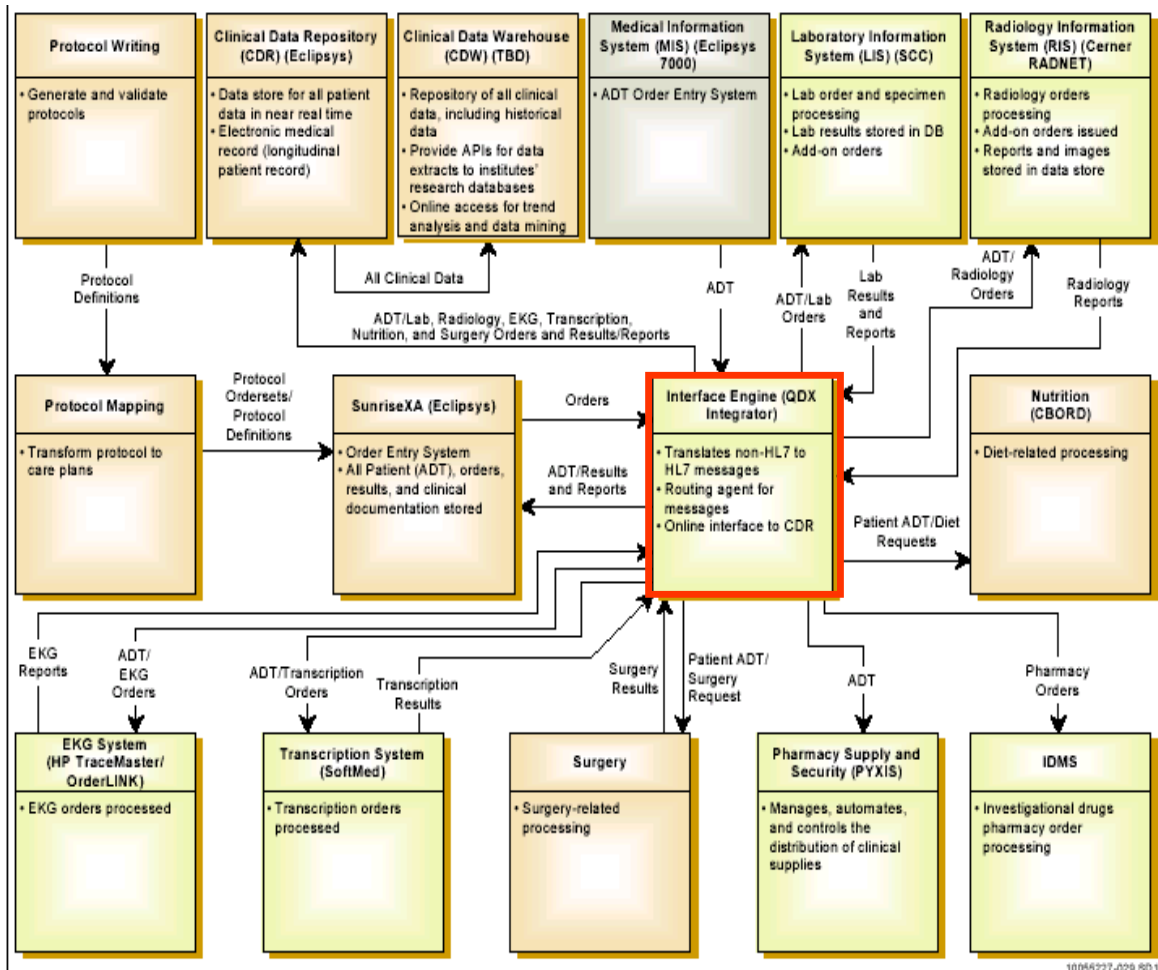
*The NIH Clinical Center provided information about CRIS, which is currently under development and expected to achieve production status in 2004.*

To address limitations of the present system and to fully automate clinical care, the NIH has embarked on the Clinical Research Information System (CRIS) project. Specific areas in which the current system fails to perform to the needs of the Clinical Center and Institutes include:

- Compliance with the Health Insurance Portability And Accountability Act of 1996 (HIPAA) and the Privacy Act regulations
- Interfacing to ancillary systems to provide integrated data. This will eliminate paper and pencil transfer of data between systems
- Reduction of potential medical errors through the implementation of a Pharmacy and Surgical Scheduling, Management and Documentation system
- Management and display of radiologic, anatomic, pathologic, ultrasound images and other image-based data
- Interfacing to Institute research databases
- Support for standardized medical vocabularies
- Support for analyzable electronic documentation (i.e. physician notes)
- Support for protocol-based provision of care
- Provision of management information for resource allocation and cost attribution
- Provision of longitudinal patient data
- Provision of historical patient data for research analysis
- Comprehensive support for patient appointing
- Support for bed management
- Support for nurse acuity assessments.

The next figure shows the planned CRIS Clinical Care Infrastructure.

**Figure 9. Planned CRIS Clinical Care Infrastructure**



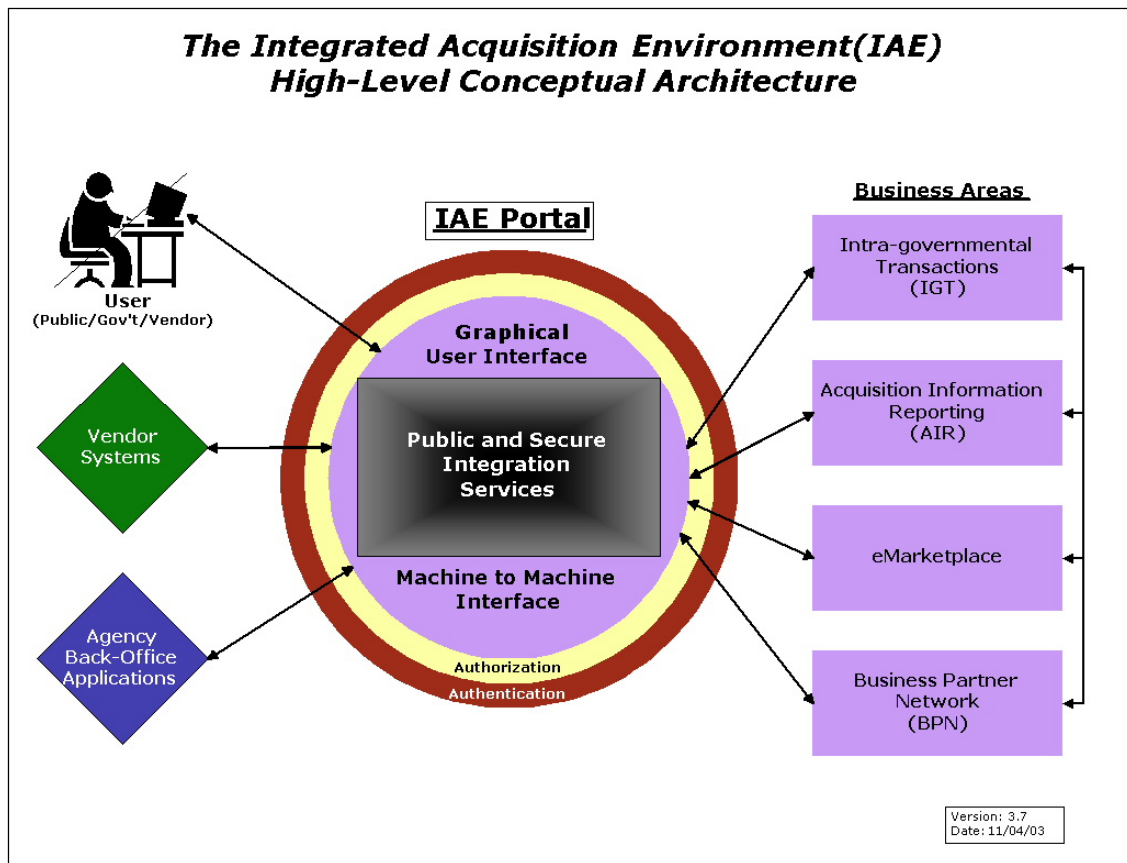
Source: NIH/CC; 4/25/03, **emphasis added**

### 4.3 GSA Integrated Acquisition Environment

Beyond NIH's own agenda, governmentwide integration initiatives are currently being planned or are under construction. The figure below shows one such e-government effort, sponsored by OMB and led by GSA.

The black rectangle in the center of the GSA diagram is of particular interest. GSA is reportedly planning to procure integration middleware, including an integration broker.

Figure 10. GSA Integrated Acquisition Environment



Source: NIH

## 5.0 Enterprise API Design Patterns (Future State)

Design patterns may be *logical* or *physical*. Logical design patterns do not specify specific technology platforms, products or brand names. A logical design pattern may be implemented by one or more related physical design patterns. Patterns provide design guidance to implementation teams and can occur in one domain or span multiple domains.

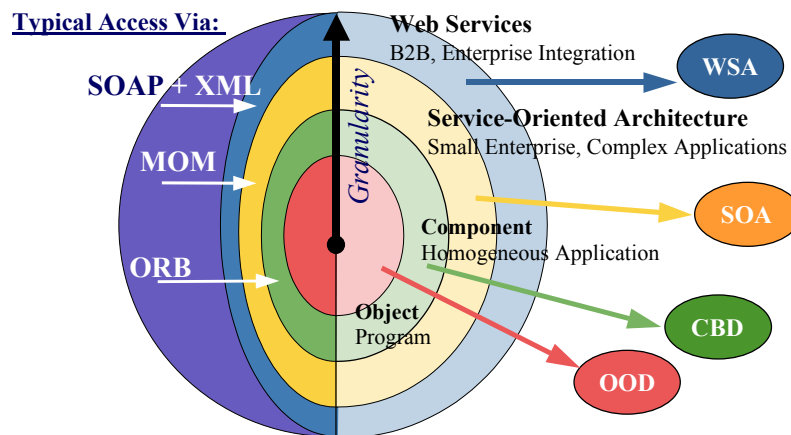
The future-state architecture addresses two related problem areas. This chapter provides an API solution to improve integration of *extension systems* with enterprise applications. The next chapter addresses *integration of enterprise applications* with each other.

The API solution builds on three related logical patterns for use by enterprise solution architects designing interfaces:

- **Service-Oriented Architecture (SOA):** Fundamental encapsulation of interface services.<sup>1</sup>
- **Web Services Architecture (WSA):** A service that is constructed and accessed according to Web services standards.
- **Application Program Interface (API):** The characteristics of the interface itself.

An SOA is useful for building reusable services, in general. But SOA and WSA also go beyond application development into the realm of integration.

Figure 11. Layered Service Architecture



Source: Gartner

<sup>1</sup> See Appendix D, Introduction to Service-Oriented Architecture, for more on both SOA and WSA.



## 5.1 Pattern: Service-Oriented Architecture (SOA)

### 5.1.1 Description

A fundamental characteristic of a service is the way it is *partitioned*. The user interface — if any — is completely separated from the application logic and data logic.

### 5.1.2 SOA Solution

The figure shows six common ways of partitioning program logic. Interfaces should be constructed using one of the three SOA patterns (bottom row).

Figure 12. Logical Design Pattern: SOA

1. Legacy/Terminal	2. Client-Server (a)	3. Client-Server (b)
All of the code runs on a single machine. Multiple terminals connect to the application.	“Fat client.” All of the code runs on a PC. The database is on a separate server.	“Plump Client.” Same as client-server (a) except the data logic, and perhaps some application logic, is coded in stored procedures.
4. SOA (1)	5. SOA (2)	6. SOA (3)
Service-oriented architecture. All app. and data logic is on a server.	Same as SOA (1), but with partitioning so multiple servers can be used.	Ultra-thin, partitioned SOA. Most of the presentation logic is really on a server.

Note that the resulting service may be usable both for external access (as an integration interface) and internal purposes (by adding presentation logic).

### 5.1.3 Guidelines for SOA Design

Today, NIH applications are heterogeneous; they are written in a variety of languages and execute on different computing platforms. While the internal construction is not in scope for this domain, many NIH applications are based on object-oriented design (OOD) or component-based design (CBD) principles, and many are already SOA as well<sup>1</sup>; thus the transition to SOA for APIs should not be inordinately difficult. The following table shows Gartner’s guidelines for effective SOA design.

**Table 6. Guidelines for SOA Design**

Guideline	Rationale
<ol style="list-style-type: none"> <li>1. Generated service wrappers typically make poor services.</li> <li>2. Services must be designed in their own right; objects and components are not services.</li> <li>3. Beware of too many small services.</li> <li>4. Beware of services that are too large.</li> <li>5. Beware of Web services extremism:                             <ul style="list-style-type: none"> <li>❑ Not all software should be service-oriented.</li> <li>❑ Not all services should be Web services.</li> <li>❑ Not all messages should be in XML.</li> </ul> </li> </ol> <p style="text-align: right;">Source: Gartner Research</p>	<p>The typical early approach to SOA was to simply wrap pre-existing components as services. Vendors offer tools that wrap C++ classes, Java classes, Enterprise JavaBeans, CORBA IDL and other transaction programs as Web services. Although this approach is natural and the quickest way into SOA, it can also lead to disasters.</p> <p>The problem is the mismatch of granularity. Properly designed services are relatively large, allowing for significant processing on behalf of a single request. These services incorporate dozens of components and hundreds of object classes. But if each component or object class is a service, communication traffic can overwhelm the network and render the entire application unusable. Unfortunately, this problem becomes apparent only after much work has been completed.</p> <p>The right granularity of services is an important design decision. Too small services can clog the network; too big services may deny the project most of the benefits of SOA.</p>

### 5.1.4 Benefits

- This pattern is a building block for the next two patterns, which is where the benefits for integration occur.
- An important secondary benefit, from an application technology architecture perspective, is the construction of reusable services.

### 5.1.5 Limitations

- A service developed in one programming language may be difficult to call from an application written in a different language. However, the use of interoperable Web services (next pattern) solves this limitation.

When the application development architecture is developed, SOA should be considered there as well.

<sup>1</sup> The Application Architecture survey indicated that SOA is already common at NIH (75 percent of the responses were SOA) and that Web services were often on the agenda.

## 5.2 Pattern: Web Services Architecture (WSA)

### 5.2.1 Description

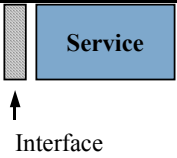
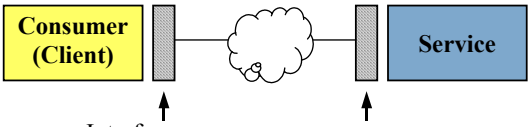
When a service program is invoked as a Web service, it is usable by other programs even if they are written in a different language and run on a different computing platform.

Web services may also be invoked or supplied by an integration broker.

### 5.2.2 Web Services Architecture

The definitions shown below build on the definition of application integration given in Chapter 2.

Figure 13. Logical Pattern: Web Services Architecture (WSA)

Definition	Conceptual Graphic
Integration Service — A software program that is a business-complete logical unit of work, accessible programmatically from independently designed contexts via a direct, openly documented interface.	
SOA for Integration — An application software topology consisting of services and service consumers (clients) in loosely coupled, one-to-many-consumers-of-each-service relationship.	
Web Service for Integration — The service interface is encoded using WSDL.	

Source: Gartner

### 5.2.3 Benefits

- Based on modern, widely accepted standards.
- Clients and services can be written in a wide variety of programming languages.

### 5.2.4 Limitations

- There may be performance limitations in extremely high-volume situations.
- Standards compliance/interoperability should be tested.
- Standards for transaction management across multiple Web services are still evolving.

## 5.3 Pattern: Application Program Interface (API)

### 5.3.1 Description

Interfaces to enterprise systems should be built according to this API design pattern. It builds on the two previous patterns.

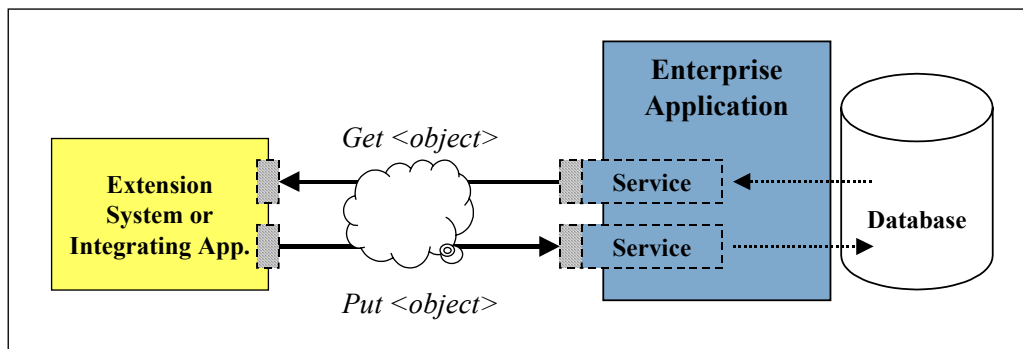
### 5.3.2 API Solution

The logical architecture for enterprise APIs is shown below. The cloud in the diagram represents both middleware and the network.

Notes about the pattern:

1. Data is exchanged via services (procedural code).
2. The services shown are Web services.
3. The application database is private (hidden) for this purpose.
4. Most operations involve complete business objects, which may be complex.
5. XML is used to encode the data in most cases.
6. Commit/rollback is typically controlled by the service-providing application.
7. Error and exception handling is handled by the enterprise application service, with the appropriate return code being passed back to the client.

Figure 14. Logical Design Pattern: Application Program Interface (API)



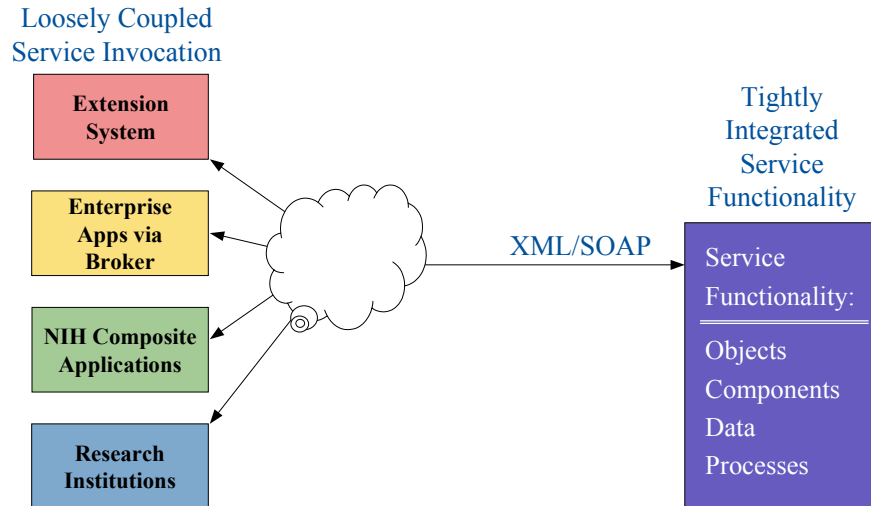
Source: Gartner

### 5.3.3 Benefits

- An important secondary benefit, from an application technology architecture perspective, is the construction of reusable services.
- Based on modern, widely accepted standards.
- Clients and services can be written in a wide variety of programming languages.
- Having a standard approach for APIs will be of great benefit to extension system developers.

- Transport of whole “business objects” (e.g., entire purchase orders or invoices), rather than line items, will optimize network usage.
- This approach complies with the “loose coupling” integration principle, as illustrated in the next figure. *This is an illustration only.*

**Figure 15. Web Services and XML Provide Loose Coupling (Example)**



Source: Gartner

### 5.3.4 Limitations

Limitations are few if the SOA design guidelines are followed.

- For extremely high-volume cases, benchmarking in advance is desirable.
- Early developers of enterprise APIs should verify interoperability (develop several test drivers written in other languages, etc.).

## 5.4 General Guidelines for Integration/API Design

The following guidance is provided to enterprise application architects and designers who are planning for integration.

### Guidelines for API Design:

- Use XML for loose coupling.
- Use a Web services architecture.
- Consider broad needs; try to make the API general enough to be used by many.
- The API/enterprise application is responsible for data integrity and transaction control.
- The API should provide informative status codes and error messages.
- APIs are documented, and source code is available for review (not modification). Inputs and outputs are clearly defined, including data syntax and semantics.

### Additional Integration Guidance:

- Edits, including complex edits (business rules), should always be enforced by the enterprise application.
- The enterprise application should provide valid field values to other point-of-entry applications.
- Complex edits (business rules) may be redundantly enforced by other, point-of-entry applications. Designers may consider a variety of mechanisms to allow this, including provision of a shared validation service.
- Security should be in place — know who is allowed to invoke an enterprise API.

## 6.0 Integration Design Patterns (Future State)

Design patterns may be *logical* or *physical*. Logical design patterns do not specify specific technology platforms, products or brand names. A logical design pattern may be implemented by one or more related physical design patterns. Patterns provide design guidance to implementation teams and can occur in one domain or span multiple domains.

This chapter is organized as follows:

- The chapter starts by providing patterns for basic program-to-program *communications*. These are fundamental building blocks used for many solutions, including integration.
- Next the focus turns to *integration*. The three basic integration problems are discussed, and three patterns to solve them are presented.
- Finally, patterns for *large-scale integration* are presented.

### 6.1 Pattern: Basic Communication

#### 6.1.1 Description

The five basic communication models below occur frequently in industry; they are used for application integration and other purposes. The first four are supported by MOM (Message Oriented Middleware<sup>1</sup>). The last model, file transfer, is frequently used today at NIH; it is included here for completeness and because it will continue to be useful in the future.

The new MOM models can be used to:

1. Understand how message-oriented middleware can be used for point-to-point integration on a (near) real-time basis, reliably.
2. Better understand how MOM works when it is used with a broker that supports hub-and-spoke integration.

#### 6.1.2 Five Communication Models

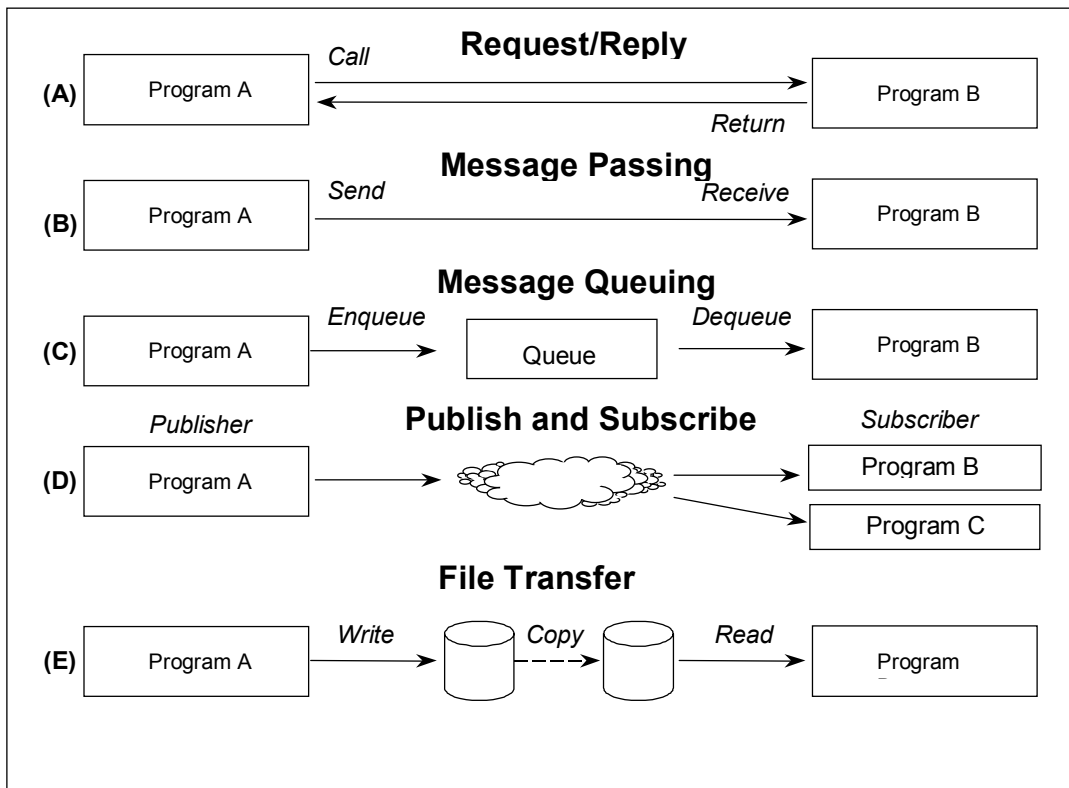
A communication model basically describes the relationship between the sending and receiving program in terms of relevant communication flow characteristics (e.g., one-way vs. two-way, one-to-one vs. one-to-many, synchronous vs. asynchronous, etc.). If the business issues are well understood, and the application design phase is carefully tackled, it should be simple to determine the most appropriate communication model. Non-trivial applications may need to use more than one communication model.

The five logical models shown in the next figure, while similar, have somewhat different characteristics.

---

<sup>1</sup> Message Oriented Middleware will be referred to as MOM, which is different than MoM (Monitor of monitors) which is used in the Enterprise Storage Management report

**Figure 16. Logical Design Pattern: Basic Communication Models**



Source: Gartner Research, 2004

- A. Request/Reply (synchronous) — Here MOM is being used to provide a synchronous interaction between two applications. This is similar to a normal subroutine call.
- B. Message Passing (synchronous) — This is a “fire and forget” operation. All that Program A knows is that Program B received the message. (Program B could, of course, send a confirmation message as a separate step.)
- C. Message Queuing (asynchronous) — This message passing also has an added queuing mechanism. Messages go into the queue for delivery later — perhaps milliseconds later or even days later. Queuing adds *resiliency* because the receiving program or the network can be down when the message is sent, but it still works out in the end.
- D. Publish and Subscribe (asynchronous) — Program A publishes a message to the middleware engine<sup>1</sup>, represented by a cloud in the diagram. The message is then routed to multiple subscribers. Program A does not itself know who the subscribers are. (The pattern is named “publish and subscribe” because the logical model is similar to magazine distribution.)

<sup>1</sup> The middleware may also provide queuing, like sub-pattern D.



E. File Transfer (asynchronous) — Program A puts messages in a file. Sometime later the file is copied to the machine on which Program B runs, and Program B reads them. This is essentially message passing on a delayed batch basis.

*Managed* file transfer — when it meets the needs of the business and point-to-point integration is acceptable — will continue to be an important integration mechanism.

### 6.1.3 Benefits

The use of MOM, rather than file transfer, provides the following benefits, assuming that the choice of the model (A, B, C or D) is appropriate:

- Information can be exchanged on a real-time or near real-time basis.
- Queuing provides resiliency.
- The publish-and-subscribe model is easy to program using MOM.
- Publish and subscribe provides loose coupling because the middleware knows who the subscribers are, not the sending program.
- MOM often improves network performance because small messages are sent continuously rather than all at once, in a batch file.

### 6.1.4 Limitations

There aren't many inherent limitations.

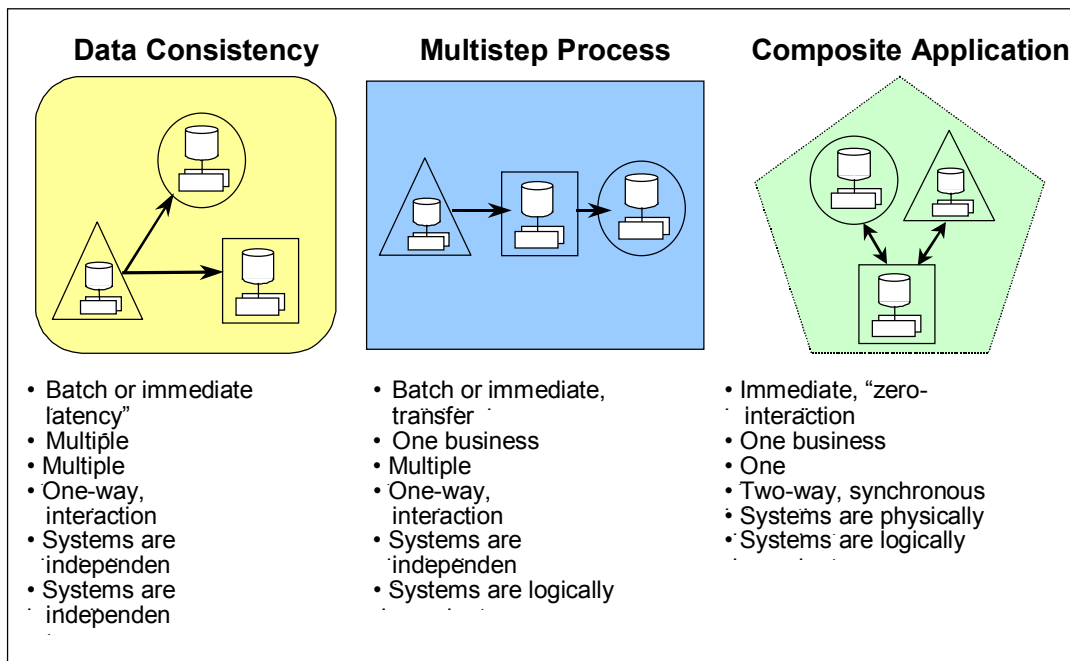
- File transfer — common today — has limitations in (a) timeliness and (b) ability to confirm receipt/acceptance of transactions. The MOM patterns eliminate these limitations.
- While MOM is commonly used to handle very large transaction volumes for major enterprises there could still be limitations. It is sometimes wise to benchmark the messages/second rate before committing to a design.
- Additional limitations cannot be determined until the NIH chooses a product.

Once a product has been chosen, further limitations should be documented. Areas to examine include: performance, transaction control, message routing details, hardware platforms supported, etc.

## 6.2 Basic Integration Patterns (Introduction)

Research has shown that there are three basic integration “problems” that consistently turn up at all large enterprises. The problems can be solved in a variety of ways, as demonstrated today at NIH. This section introduces the problems. The next three sections provide design patterns for solving them.

Figure 17. Three Integration Problems



Source: Gartner, 2004

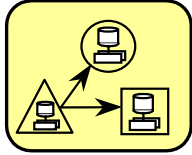
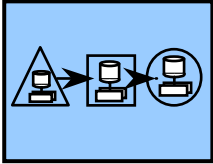
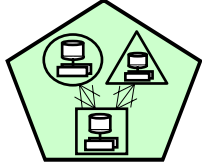
The goal of *data consistency* is to get multiple systems or departments to agree on the facts. The most common method of achieving data consistency is to create a batch file in the application that first captures the new information and then transfers those updates to the next application system in a nightly batch process.

A *multi-step process* involves a sequence of steps, each conducted by an application system or person. Each instance of a business process, such as each purchase order or insurance claim, has a life cycle that may span seconds, minutes, hours or days.

A *composite application* may service a human client, perhaps using a new Web application, to transparently invoke business logic in other applications, such as one or more mainframe transactions or calls to packaged Unix or Windows NT applications. To the user, this looks like a new application that provides the functionality previously available in multiple applications. Interactive, composite applications represent the most closely knit and hardest-to-implement integration pattern.

Each problem has its own characteristics, as shown in the next figure.

**Figure 18. Integration Solution Characteristics**

Basic Integration Patterns			
Solution Requirement	Data Consistency 	Multi-step (STP) 	Composite 
<i>Prevailing Interaction</i>	One-Way Asynchronous	One-Way Asynchronous	Two-Way Synchronous
<i>Communication</i>	File transfer/MOM	File transfer/MOM	Platform Middleware <sup>1</sup> , MOM, Gateways
<i>Transformation</i>	Advanced	Advanced	Basic (Mapping)
<i>Flow Control</i>	Intelligent Routing	BPM/Workflow	Application Code
<i>Unit of Work</i>	Message	Process	Composite Transaction
<i>Performance</i>	Messages/Sec.	Elapsed Time, Transactions/Sec.	Transactions/Sec.

Source: Gartner

Large-scale integration problems are typically solved by developers who are familiar with all three basic integration patterns and use a mix of one or more of them, depending on the nature of the applications.

Most integration infrastructures in operation have been designed to support data consistency and multi-step process integration scenarios. Data consistency is the fundamental integration requirement for most enterprises. Multi-step process integration can be supported by adding to the infrastructure supporting data consistency a BPM layer. Both patterns are essentially asynchronous, fire-and-forget oriented and require a MOM-based communication layer.

Composite applications need to be supported by an infrastructure with completely different characteristics. Composite applications require synchronous, request/reply interactions between applications, typically provided through remote procedure call (RPC) or object request broker (ORB) protocols. Messages flowing between the elements of the composite application are typically simple; hence, advanced message transformation is not required. Service composition can be implemented through application code or “light” forms of BPM technology. The “unit of work,” in which some degree of all-or-nothing atomicity must be supported, is a composite, logical transaction spanning multiple “physical” components. In terms of performance, users express their requirements in traditional metrics, such as transactions per second, and not in terms of messages per second or concurrent processes.

<sup>1</sup> RPC, ORB, TPM etc. Platform middleware is out of scope for this document.

## 6.3 Pattern: Data Consistency

### 6.3.1 Description

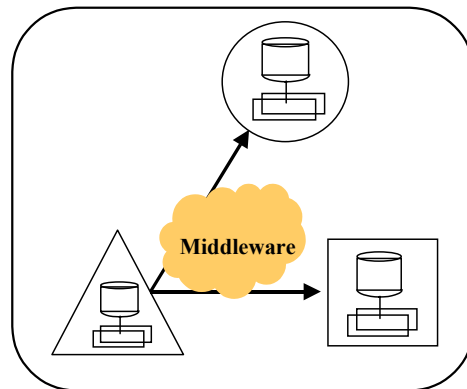
Modern enterprises generally have redundant versions of data regarding customers, products, orders, employees and other entities; NIH is no exception. The most common method of reconciling data at NIH is to create a batch file in one application and then transfer it to other interested applications in a nightly, point-to-point batch process.

In an advanced, zero-latency enterprise (ZLE) strategy, by contrast, each update is transferred as an individual transaction as soon as data is updated.

### 6.3.2 Data Consistency Solution

Instead of file transfers, ZLE data consistency solutions may use a publish-and-subscribe, MOM product, and extraction, transformation and loading (ETL) layer, or an integration broker to transform the files, documents or messages to reconcile syntactic and semantic differences among the sender and one or more receivers. Modern data consistency solutions often encode the data in XML to partially standardize the communication and transformation processes.

**Figure 19. Logical Design Pattern: Data Consistency**



Source: Gartner

### 6.3.3 Benefits

- Provides consistent data faster and better than file transfer.

### 6.3.4 Limitations

- File transfer should still be used where it meets business needs, because it is easier to implement due to its simplicity and less expensive.
- Should avoid usage when dealing with applications outside of NIH; but may be possible in a few cases (e.g., DHHS, GSA).

## 6.4 Pattern: Multi-Step Process

### 6.4.1 Description

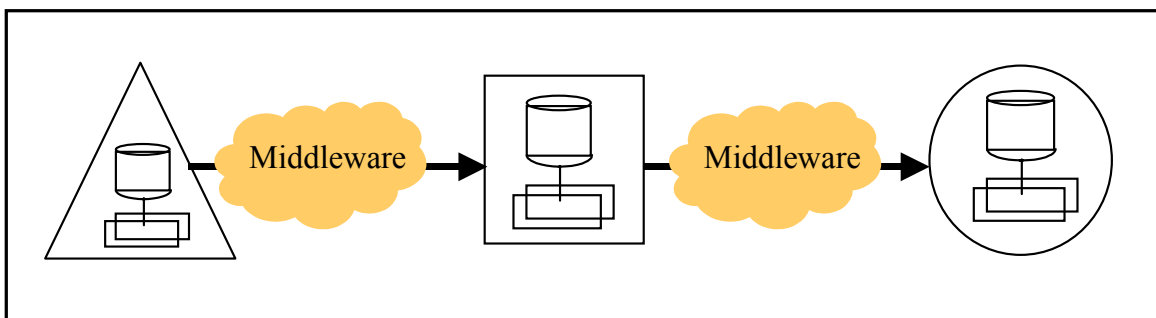
A multi-step process involves a sequence of related steps, each conducted by an application system or person. Process instances — for example, each order or each insurance claim — progress through their life cycles step by step. The applications may be independently designed or purchased at different times by different business units, or they may be application subsystems that are part of the same suite of applications.

When a step is complete, the system or person involved will generate one or more messages, documents, files or database entries to pass data to a subsequent step. Traditional multi-step processes rely on manual data reentry between each step.

### 6.4.2 Straight-Through Processing (STP) Solution

Straight-through processes (STPs) are fully automated from end to end, reducing not only elapsed time and data entry costs, but also data entry errors. In principle, STP can use file transfer, but immediate transfer of individual transactions will provide NIH with greater benefits. The most advanced forms of multi-step processes use BPM or workflow software to track and, in some cases, control the flow of execution. Applications in a multi-step process are logically dependent on the previous steps in the process, because the output of one step is the input to the next step, but each step executes serially.

Figure 20. Logical Design Pattern: Straight-Through Processing (STP)



Source: Gartner

### 6.4.3 Benefits

- Elimination of manual steps lowers labor effort.
- The elapsed time for the business process is reduced.
- Fewer errors are made.

### 6.4.4 Limitations

There are few, if any, limitations, assuming a good fit with the problem.

## 6.5 Pattern: Composite Applications

### 6.5.1 Description

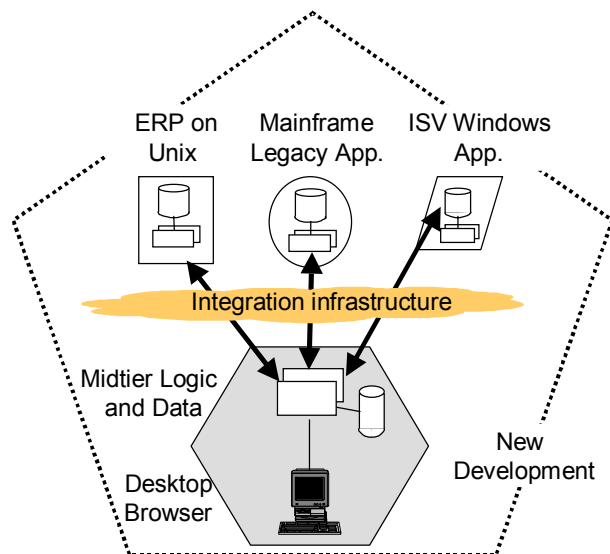
A *composite application* seems to the user like a new application. It has some new functionality itself, plus functionality from multiple existing applications.

### 6.5.2 Composite Application Solution

A composite (or “virtual”) application needs to be near zero in latency (usually under a minute, often within a couple of seconds). It may be implemented with any of a wide range of middleware technologies. Plain, direct gateways are the most popular choice for opportunistic, tactical, request/reply applications, especially when extending only one or two back-end applications with a Web front end. Gateways may operate with a remote procedure call (RPC) model (e.g., COM-CICS or COM-CORBA gateways), a screen scraping model or a database gateway model (e.g., JDBC, ODBC or OLE DB).

For systematic composite applications with multiple participating systems that may change over time, integration brokers, MOM and/or BPM may be used. These introduce an incremental layer of communication semantics and administration, so they are overkill for tactical projects. However, because they enable the asynchronous forms of integration (data consistency and multi-step patterns), in addition to demanding composite applications, they are a good fit for developers who seek to have one comprehensive solution toolset.

Figure 21. Logical Design Pattern: Composite Application

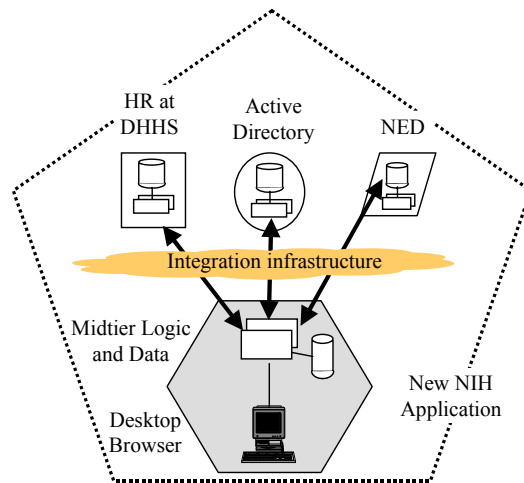


Source: Gartner, 2004

### 6.5.3 Benefits

- NIH can use the composite application pattern to “glue together” some of its partially redundant applications and give users a much more convenient interface.
- Extension systems can use this pattern to construct an interface that uses and updates both enterprise and IC-specific data at the same time. (Note the new logic and data inside the shaded hexagon).

An NIH example is shown in the diagram to the right. *This is an illustration only.*



Source: Gartner Research

### 6.5.4 Limitations

- Will be difficult to apply to monolithic applications, especially those that currently use no form of middleware. (The application architecture survey indicated that these types of applications are rare at the NIH.)
- Coordinated and reliable transaction control (commit/rollback) can be a challenge.

## 6.6 Pattern: Large-Scale Integration

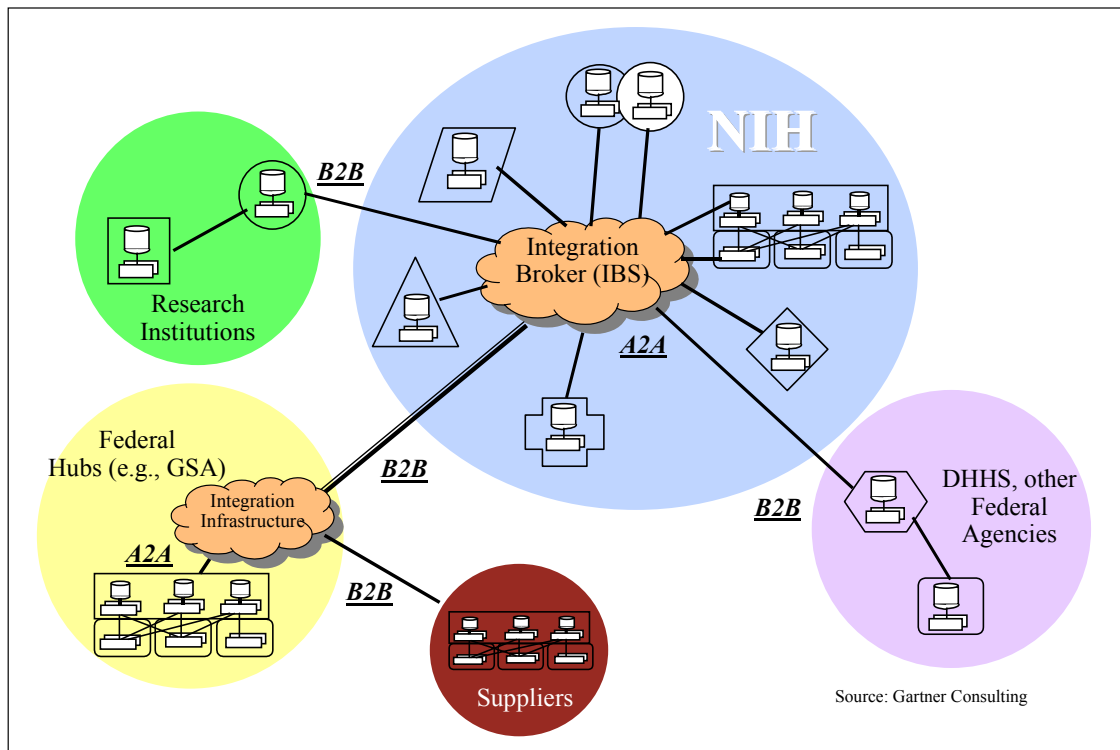
### 6.6.1 Definition

This pattern looks at integration from a high-level perspective. It could be decomposed into hundreds or thousands of the basic integration problems previously described.

### 6.6.2 Hub-and-Spoke Solution

This is the recommended alternative to point-to-point integration. At the center of the diagram there is a “hub” — an IBS. Applications are at the end of the “spokes”; they connect to the hub via adapters and MOM. The central hub may also connect to other hubs owned by other enterprises.

Figure 22. Logical Design Pattern: Large Scale Integration Hub-and-Spoke Solution



Source: Gartner, 2004

### 6.6.3 Benefits

- The hub-and-spoke topology lowers the number of connections that must be built from  $N*(N-1)$  to  $N*2$ .
- Specialized IBS tools are more efficient for building and maintaining interfaces and transformations.



## 6.6.4 Limitations

- The NIH has not yet purchased a general-purpose IBS, and the cost is substantial.

After an IBS product has been purchased, a physical design pattern should be developed.

## 6.7 Pattern: Broker/Operational Data Store/Warehouse

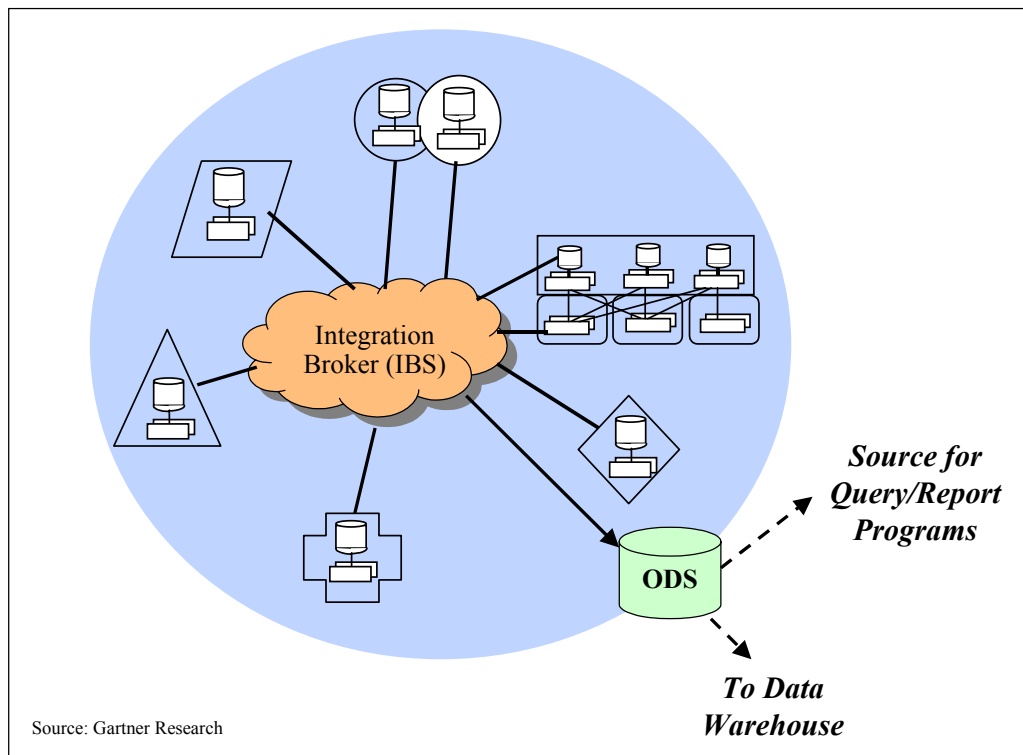
### 6.7.1 Description

This pattern shows how an IBS can be used to maintain an operational data store (ODS). The ODS can then be used as a source for queries and reports. It can also be used to populate a data warehouse or data marts.

### 6.7.2 Operational Data Store (ODS) Solution

Data warehouse construction has typically been hampered by the need to extract information from a large variety of existing applications. But once a broker is in place — and most transactions are flowing through it anyway — it is relatively easy to have the broker populate an ODS.

Figure 23. Logical Design Pattern: Broker/Operational Data Store/Warehouse



Source: Gartner, 2004

### **6.7.3 Benefits**

- Lowers data warehouse implementation and maintenance costs.
- Provides a combined source of data in a standardized format for query and reporting use.
- Lessens the need for direct access to important transactional systems that may be overloaded.

### **6.7.4 Limitations**

- Some attributes of the source data may be lost when it is transformed and sent to the ODS.
- Historical data may be lost unless special steps are taken to preserve the view of data over time.

## 7.0 Application Integration Bricks

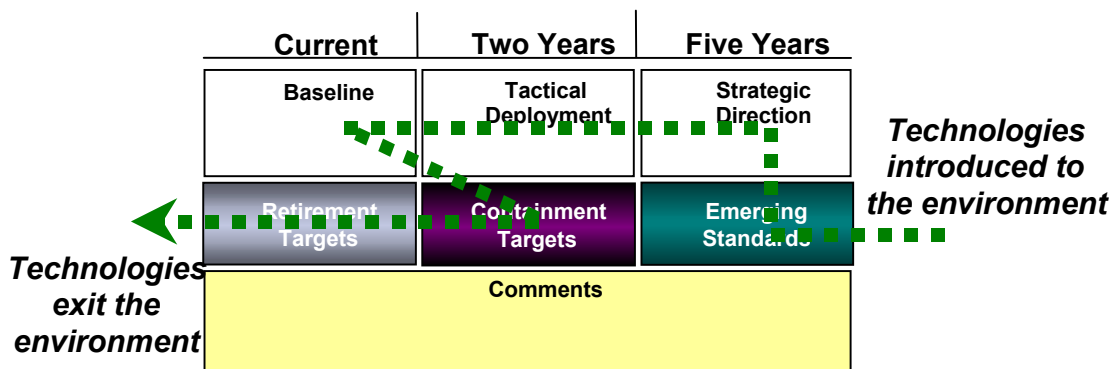
In the Technical Reference Model (TRM), baseline and planned technology choices for elements meet in a chart called a “brick.” Bricks represent the physical building blocks of the enterprise IT systems — they identify specific technologies used to implement solutions. Bricks document both NIH’s current (“as is”) environment and future (“to be” or target) states. The planning horizon is five years.

Each brick captures:

- A description of the technology and its role
- Specific implications, dependencies, and deployment and management strategies
- Technology elements, categorized.

A brick template is shown below.

Figure 24. Technology Planning “Brick”



Source: Gartner

The technology categories for architectural elements are:

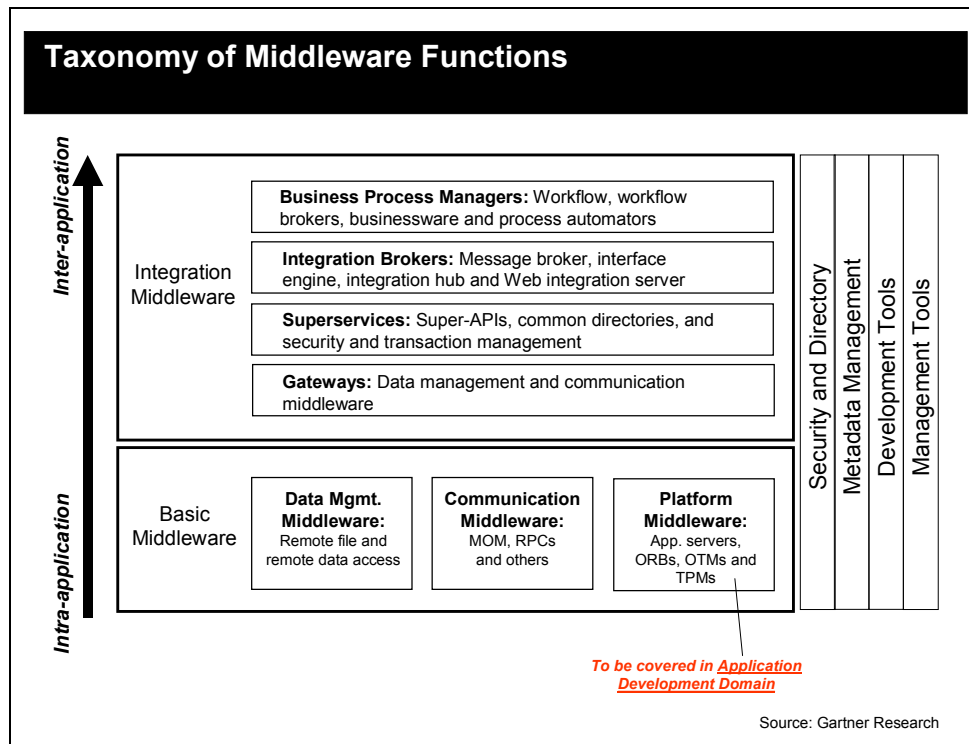
- **Baseline** technologies include current technology and/or process element(s) in use.
- **Tactical** technologies are recommended for use in the near or tactical time frames (next two years). Currently available products needed to meet existing needs are identified here.
- **Strategic** technologies provide strategic advantage and might be used in the future. Usually, marketplace leaders are identified here, as they are likely to provide better benefits and meet the anticipated needs of the business.
- **Retirement** technology and/or process elements targeted for de-investment during the architecture planning horizon (five years).
- **Containment** includes technology and/or process elements targeted for limited (maintenance or current commitment) investment.

- **Emerging** technology and/or process elements are to be evaluated for future use based on technology availability and business need. These technologies may not be new to the marketplace, but are simply not yet in use at NIH. In this case, the products may be a fit for emerging needs at NIH.

## 7.1 Definitions and Taxonomy

*Middleware* is defined as runtime system software that directly enables application-level interactions among programs in a distributed computing environment. *System software* means software that is positioned between an application program and lower-level operating system, data management and networking services. A *computing environment* is physically *distributed* when its programs or databases are spread across two or more computers. Middleware is also useful when logically distributed components run on the same computer. Enterprises will note that a database management system (DBMS) is a set of programs, so distributed data and distributed application code are covered by this definition. Application-level interactions are those that transfer business data or information about its semantics or context (not just technical “housekeeping” data) to or from an application program. The next figure shows the middleware taxonomy.

Figure 25. Middleware Taxonomy



Source: Gartner, 2004

The application integration middleware architecture is comprised of the following bricks, grouped according to the taxonomy shown below.

- Data management middleware
- Communication middleware
- File transfer middleware
- Integration middleware — Integration brokers
- Integration middleware — Adapters
- Integration middleware — Business process managers
- Integration middleware — Gateways.

Web services are not middleware; they are represented in a "protocols and standards" brick (not part of the middleware taxonomy).

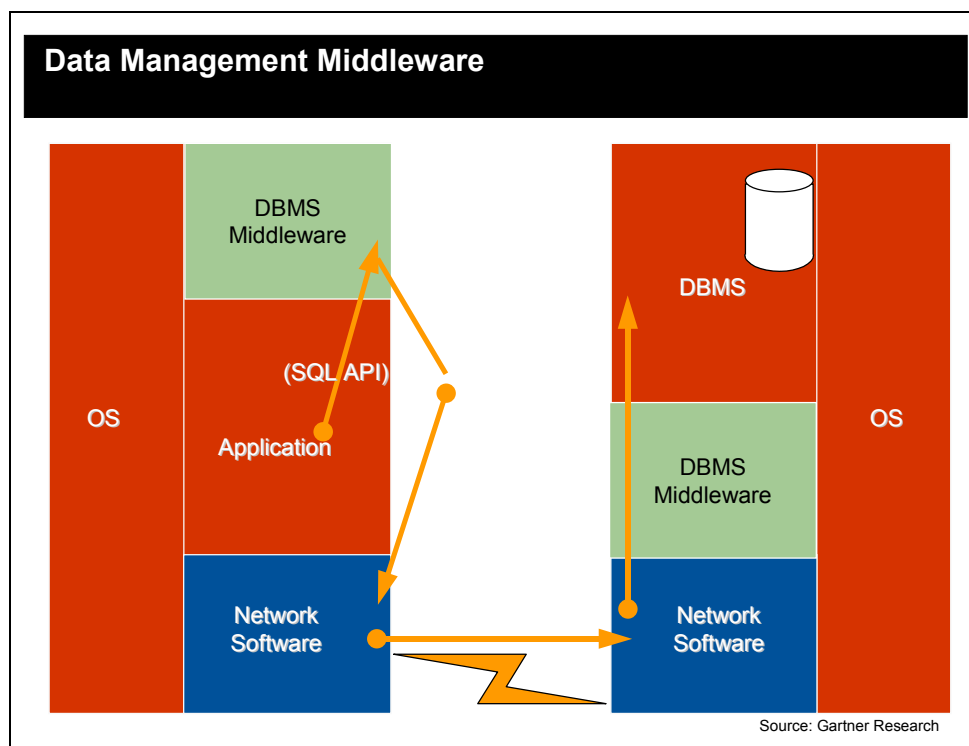
- Web services.

## 7.2 Brick: Data Management Middleware

Data management middleware functionality helps programs, including application programs and database management systems (DBMS) read from and write to remote databases or files. The most widespread forms of middleware today are the remote database access and remote file access middleware bundled into a DBMS or a network operating system, respectively. These support traditional two-tier client/server architectures and can also be used for more sophisticated multi-tier applications.

All modern relational DBMSs include a networking capability so that the DBMS engine can (optionally) be called from a client application located elsewhere.

Figure 26. Data Management Middleware



Source: Gartner, 2004

**Table 7. Data Management Middleware Brick**

<b>Baseline Environment (Today)</b>	<b>Tactical Deployment (0-2 years)</b>	<b>Strategic (2-5 years)</b>
<ul style="list-style-type: none"> <li>■ Neon Shadow Direct 32-bit ODBC driver</li> <li>■ IBM DB2 Connect</li> <li>■ Oracle Net Services</li> <li>■ MS Data Transformation Services (DTS)</li> <li>■ AFS — Andrew File System (CIT)</li> <li>■ ODBC</li> <li>■ JDBC</li> <li>■ OLEDB — Microsoft Object Linking and Embedding Database</li> </ul>	<ul style="list-style-type: none"> <li>■ IBM DB2 Connect</li> <li>■ Oracle Net Services</li> <li>■ MS Data Transformation Services (DTS)</li> <li>■ ODBC</li> <li>■ JDBC</li> <li>■ OLEDB</li> </ul>	<ul style="list-style-type: none"> <li>■ Oracle Net Services</li> <li>■ OLEDB</li> <li>■ Additional, TBD</li> </ul>
<b>Retirement (Technology to eliminate)</b>	<b>Containment (No new deployments)</b>	<b>Emerging (Technology to track)</b>
<ul style="list-style-type: none"> <li>■ AFS</li> </ul>	<ul style="list-style-type: none"> <li>■ Neon Shadow Direct 32-bit ODBC driver</li> </ul>	<ul style="list-style-type: none"> <li>■ XDBMS — XML database management systems</li> </ul>
<b>Comments</b>		
<ul style="list-style-type: none"> <li>■ ODBC/JDBC drivers are supplied by multiple vendors.</li> <li>■ IBM DB2 Connect, Oracle Net Services, MS DTS, ODBC/JDBC, OLEDB were listed as Tactical and Strategic to leverage NIH's investment in products that are a proven fit for NIH's known future needs. Leveraging baseline products in the future will minimize the operations, maintenance, support and training costs of new products.</li> <li>■ Neon Shadow Direct 32-Bit ODBC Driver and AFS have been designated Retirement and Containment. These products are either not as widely or successfully deployed at NIH, or they do not provide as much functionality, value or total cost of ownership as the selected Tactical and Strategic products.</li> </ul>		

XDBMS products support the storage of XML documents in their native format. This is usually achieved via a proprietary database structure in which XML documents or fragments form the foundation of the database. Knowledge of the complete physical structure of the XML document is maintained in the database, enabling the document to be retrieved in its original state. In addition, no predefined knowledge of the document structure is required to store it — the self-describing nature of XML allows creation of the database "schema" on the fly. This enables the database to store XML documents of varying and dynamic formats, and can potentially reduce the administration and support effort. Access to XML data in the database (storage and retrieval) is achieved via XML-standard interfaces (for example, XPath, the Document Object Model [DOM] or other XML-based APIs).

### 7.3 Brick: Communication Middleware

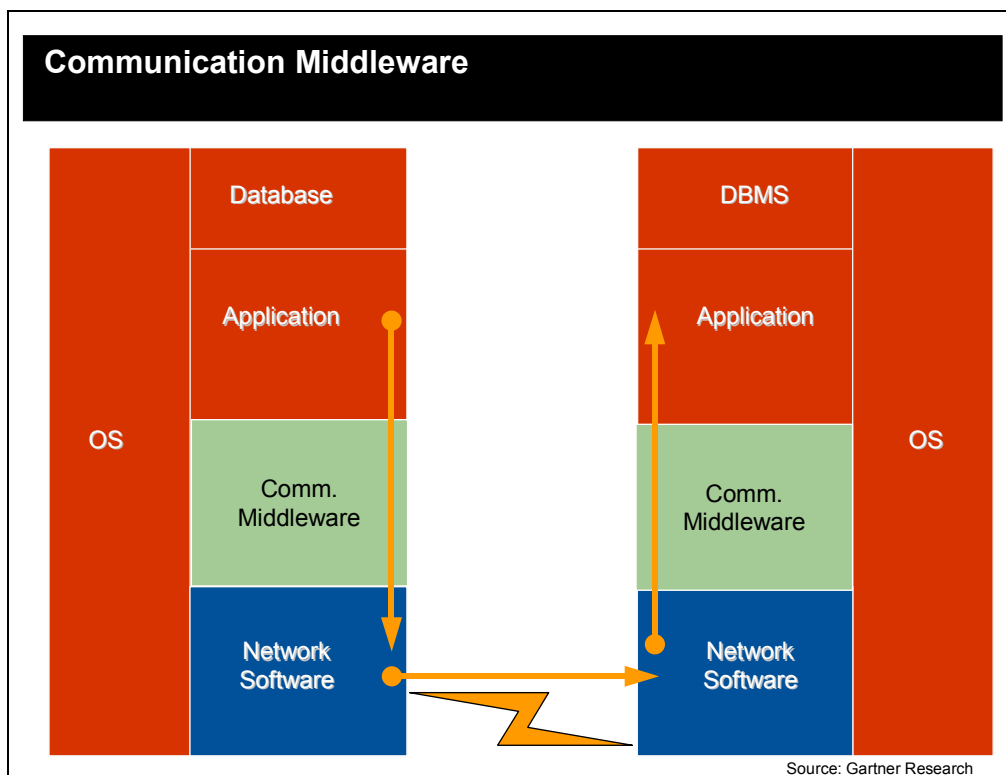
Communication middleware helps programs talk to other programs. It is software that supports a protocol for transmitting messages or data between two points as well as a system-programming interface to invoke the communication service. MOM also provides for the safe and reliable delivery of messages.

Today's communication middleware generally runs on Internet-based protocols, but also may implement higher-level protocols, including industry standards (e.g., XML) and proprietary protocols, and it may run over the Internet or private networks.

Although simple forms of communication middleware do not inherently provide them, a variety of services are provided by more sophisticated products. Such features include reliable delivery, transactional support/integrity, message queuing, offline message handling, once-and-only-once delivery as well as first-in, first-out and other message-ordering variations.

Although communication middleware is an essential requirement for application integration projects, no single solution or industry standard can address requirements for every integration problem or scenario.

Figure 27. Communication Middleware



Source: Gartner, 2004



**Table 8. Communication Middleware Brick**

<b>Baseline Environment (Today)</b>	<b>Tactical Deployment (0-2 years)</b>	<b>Strategic (2-5 years)</b>
<ul style="list-style-type: none"> <li>■ Oracle Advanced Queuing</li> <li>■ Communication middleware for QDX (Unknown) — CC</li> <li>■ COM/DCOM</li> </ul>	<ul style="list-style-type: none"> <li>■ PeopleSoft Enterprise Integration Points</li> <li>■ Oracle Advanced Queuing</li> <li>■ Communication middleware for QDX</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>
<b>Retirement (Technology to eliminate)</b>	<b>Containment (No new deployments)</b>	<b>Emerging (Technology to track)</b>
<ul style="list-style-type: none"> <li>■ TBD</li> </ul>	<ul style="list-style-type: none"> <li>■ COM/DCOM</li> </ul>	<ul style="list-style-type: none"> <li>■ MOM</li> </ul>
<b>Comments</b>		
<ul style="list-style-type: none"> <li>■ Oracle AQ and the QDX communication middleware were listed as Tactical, because they are a proven fit for NIH's known future needs. Leveraging baseline products in the future will minimize the operations, maintenance, support and training costs of new products.</li> <li>■ Microsoft COM/DCOM has been designated for Containment. Microsoft's ".NET" architecture will provide a replacement, product name as yet unknown. These products are either not as widely or successfully deployed at NIH, or they do not provide as much security or reliability as the selected Tactical and Strategic products.</li> <li>■ PeopleSoft Enterprise Integration Points (PeopleSoft Messaging) may be used in the future.</li> <li>■ QDX uses the HL7 protocol, the industry standard for hospitals and clinics.</li> <li>■ MOM is not a new technology, but is still considered an emerging one for NIH as it may provide alternatives in integration with certain platforms, and can offer transactional integrity if needed.</li> </ul>		

## 7.4 Brick: File Transfer Middleware

File Transfer Middleware is a class of communication middleware specifically focusing on the transfer of files from application to application. The transfer may be secure, insecure or managed.

**Table 9. File Transfer Middleware Brick**

Baseline Environment (Today)	Tactical Deployment (0-2 years)	Strategic (2-5 years)
File Transfer <ul style="list-style-type: none"> <li>■ Native OS Tools*</li> <li>■ WS FTP*</li> <li>■ Physical Transfer (CD/Tape)</li> <li>■ DICOM</li> </ul> Secure File Transfer <ul style="list-style-type: none"> <li>■ Native OS Tools*</li> <li>■ E-mail for integration</li> </ul> Managed Secure File Transfer <ul style="list-style-type: none"> <li>■ Sterling Commerce Connect:Direct*</li> </ul>	File Transfer <ul style="list-style-type: none"> <li>■ DICOM</li> </ul> Secure File Transfer <ul style="list-style-type: none"> <li>■ Native OS Tools*</li> </ul> Managed Secure File Transfer <ul style="list-style-type: none"> <li>■ Sterling Commerce Connect:Direct*</li> </ul>	File Transfer <ul style="list-style-type: none"> <li>■ DICOM</li> </ul> Secure File Transfer <ul style="list-style-type: none"> <li>■ Native OS Tools*</li> </ul> Managed Secure File Transfer <ul style="list-style-type: none"> <li>■ Sterling Commerce Connect:Direct*</li> </ul>
Retirement (Technology to eliminate)	Containment (No new deployments)	Emerging (Technology to track)
	File Transfer <ul style="list-style-type: none"> <li>■ Physical Transfer (CD/Tape)</li> <li>■ WS FTP*</li> </ul> Secure File Transfer <ul style="list-style-type: none"> <li>■ E-mail for integration</li> </ul>	
Comments		
<ul style="list-style-type: none"> <li>■ <i>Decrease unsecured file transfer for integration purposes as per NIH policy. Increase use of secure FTP server and managed secure file transfer (file transfer with scheduling, admin, management, etc.) such as Connect:Direct.</i></li> <li>■ <i>DICOM is used for transfer of medical images</i></li> <li>■ Tactical and Strategic products (DICOM, Native OS Tools, and Sterling Commerce Connect:Direct) were selected to leverage NIH's investment in products that are a proven fit for NIH's known future needs. Leveraging baseline products in the future will minimize the operations, maintenance, support and training costs of new products.</li> <li>■ Physical transfer, WS FTP, and e-mail for integration have been designated as Containment. These products are either not as widely or successfully deployed at NIH, or they do not provide as much security or reliability as the selected Tactical and Strategic products.</li> <li>■ "ssh &lt;cmd&gt;" (secure shell plus a transfer command) is classified as a Native OS Tool.</li> <li>■ ftp/sftp &lt;cmd&gt; is classified as a Native OS Tool.</li> </ul>		
*Items with an * support standard FTP protocol		

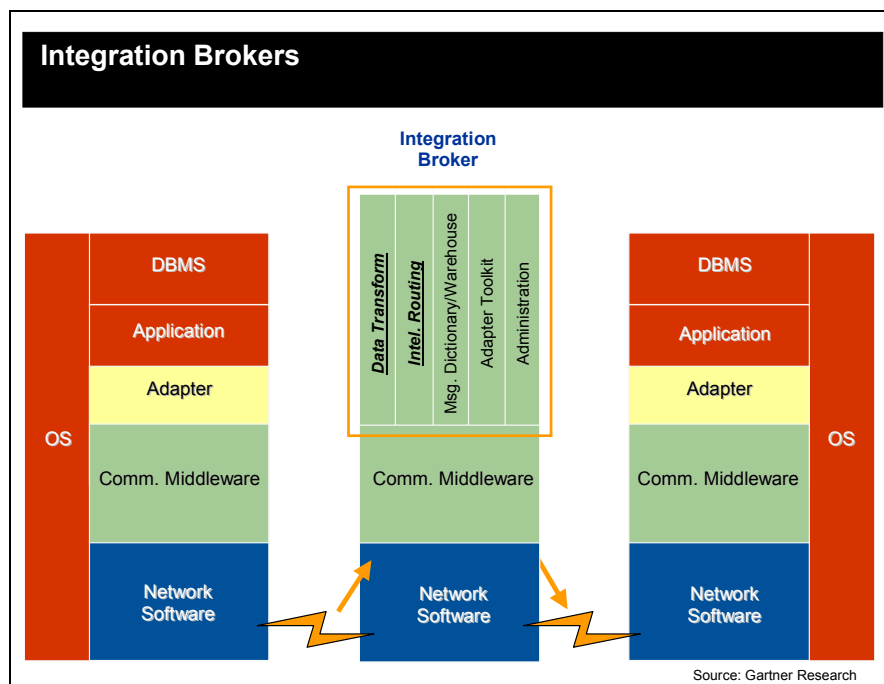
## 7.5 Brick: Integration Broker Suites (IBS)

An integration broker is a third-party intermediary that facilitates interactions among application systems. By definition, the broker itself provides two primary value-added application-layer functions:

1. Transformation — translates message or file contents, including both syntactic "conversion" and some degree of (greater or lesser) semantic "transformation."
2. Routing (flow control) — some form of smart addressing, such as content-based routing and/or publish-and-subscribe. Note that intelligent routing is stateless.

To enable these services, a broker has some form of repository that holds metadata descriptions of the input and output message formats (i.e., a message dictionary), and the transformation and routing rules. It will also have some administration and monitoring facilities to manage the broker configuration, and may also offer application-specific or technical adapters, along with some related development tools, gateways and templates for connecting to packaged applications. An integration broker may optionally also support a message warehouse (a mechanism to store and retrieve copies of messages).

Figure 28. Integration Brokers

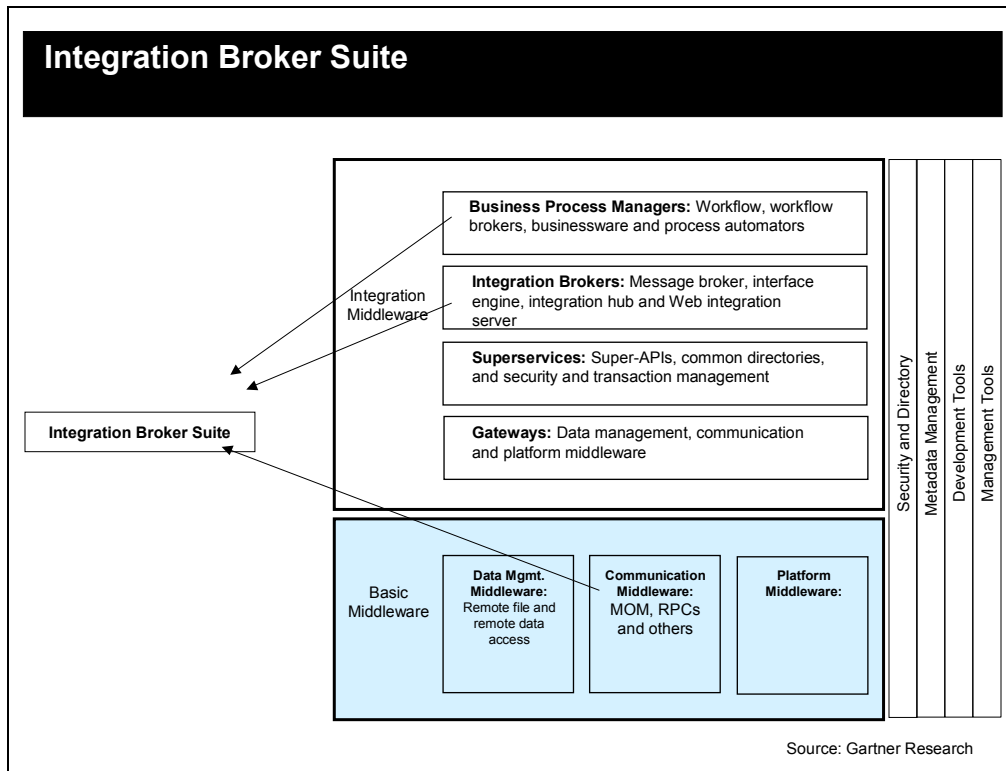


Source: Gartner, 2004

Integration broker suites (IBSs) are broker products with added features such as BPM, adapters, adapter development toolkits, Web services, communication tools, and better metadata and management facilities.

IBSs reduce the time to implement systematic application development projects that have demanding integration requirements. They improve business processes by making a broader and deeper range of integration practical across heterogeneous application systems. More than 75 percent of large enterprises use a broker somewhere, but only 10 percent of integration projects in 2002 used a broker.

**Figure 29. Integration Broker Suite**



Source: Gartner, 2004

**Table 10. Integration Brokers Brick**

<b>Baseline Environment (Today)</b>	<b>Tactical Deployment (0-2 years)</b>	<b>Strategic (2-5 years)</b>
<ul style="list-style-type: none"> <li>■ QDX Integrator</li> <li>■ Caredata Engine</li> </ul>	<ul style="list-style-type: none"> <li>■ QDX Integrator</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>
<b>Retirement (Technology to eliminate)</b>	<b>Containment (No new deployments)</b>	<b>Emerging (Technology to track)</b>
<ul style="list-style-type: none"> <li>■ Caredata Engine</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>
<b>Comments</b>		
<ul style="list-style-type: none"> <li>■ Tactical and Strategic products were selected to leverage NIH's investment in products that are a proven fit for NIH's known future needs. Leveraging baseline products in the future will minimize the operations, maintenance, support and training costs of new products.</li> <li>■ Caredata Engine has been designated Retirement. This product does not provide as much functionality, value or total cost of ownership as the selected Tactical and Strategic products. Caredata Engine is also considered retirement due to the new CRIS architecture.</li> <li>■ QDX Integrator is specialized for HL7.</li> <li>■ The eRA project may be selecting a "B2B Exchange" broker in the near future.</li> </ul>		

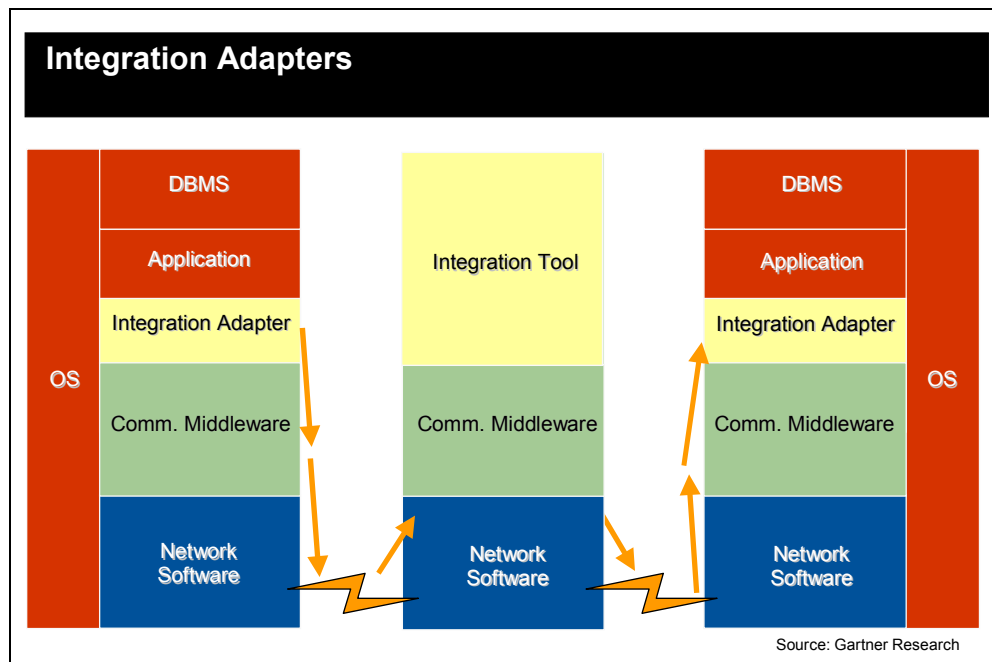
Future product selection should be based on a variety of factors, including:

1. Ability to meet a wide variety of needs at NIH (both A2A and B2B)
2. Other federal agency usage (e.g., DHHS, GSA)
3. Ability to meet eRA project needs
4. Availability of adapters for Oracle Financials, the base software for NBS
5. Availability of adapters for PeopleSoft human resources software at DHHS
6. Additional requirements, to be determined.

## 7.6 Brick: Integration Adapters

Adapters are some combination of design tools and runtime software that act as glue to link applications, which are considered "sources" or "targets" (or both), to other applications or other integration middleware. When interfacing with a source or target application, an adapter generally deals with a group of "touchpoints," that is, one or more entry/exit points, collectively called an "interface." Adapters can be deceptively complex, with "thick" adapters performing a variety of functions that include recognizing events, collecting and transforming data, and exchanging data with platform, integration suite or other middleware. On the other hand, "thin" adapters may only "wrap" a native application interface, exposing another, more standard interface for application access. Adapters can also handle exception conditions, and can often dynamically (or with minor reconfiguration changes) accommodate new revisions of source or target applications.

Figure 30. Integration Adapters



Source: Gartner, 2004

Two common types of adapters are:

- **Technical Adapters** — Technical adapters may connect into DBMSs, communication middleware or other software environments. By definition, technical adapters are not inherently configured to be business process-aware.
- **Application Adapters** — Application adapters interface to packaged application modules or vertical-industry protocols (like HL7 or HIPAA). By definition, application adapters are inherently configured to interact with a source or target interface and read or write specific business documents or messages. Many application adapters include technical adapters within them. For example, an

application adapter that is used to import or export purchase orders from a procurement application can leverage a technical adapter, which accesses the application at the database or low-level API level. While the technical adapter could be licensed and used by itself, the value of the application adapter is that it eliminates the need for complex logic that is often necessary to navigate what are often complex database or low-level interfaces.

Adapters are generally bundled with integration middleware products such as Enterprise Service Buses (ESBs), integration suites, or portal servers; or offered as a stand-alone product such as an adapter suite. Ideally, every adapter, like most application integration tools, should be noninvasive, such that it can interact with the source or target without requiring any customization in the source or target. Such independence helps insulate the adapter from its source or target's upgrades — for example, for new versions of software.

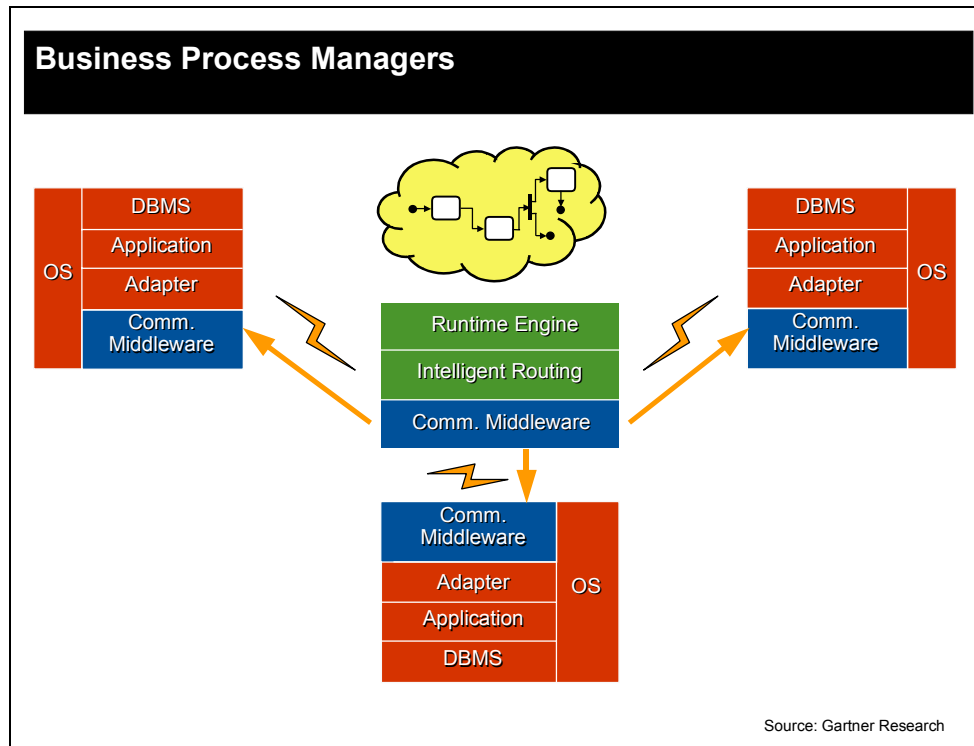
**Table 11. Adapters Brick**

<b>Baseline Environment (Today)</b>	<b>Tactical Deployment (0-2 years)</b>	<b>Strategic (2-5 years)</b>
<b>Retirement (Technology to eliminate)</b>	<b>Containment (No new deployments)</b>	<b>Emerging (Technology to track)</b>
		<ul style="list-style-type: none"> <li>■ Selection of adapters will be determined when an IBS is purchased (may be bundled).</li> </ul>
<b>Comments</b>		

## 7.7 Brick: Business Process Managers (BPM Tools)

Business process manager (BPM tool) is a general term describing a set of services and tools that provide for explicit BPM (for example, process analysis, definition, execution, monitoring and administration), including support for human and application-level interaction. BPM tools have emerged from many sources: workflow, applications, collaborative tools, integration suites, Web integration servers, application servers, development tools, rule engines and commerce offerings.

Figure 31. Business Process Managers



Source: Gartner, 2004

BPM tools track and direct each instance of a business process, such as each individual order or medical insurance claim, through a life cycle that may consume seconds, minutes, hours, days or weeks. Unlike simpler forms of flow automation, a BPM tool "remembers" (maintains in memory or a persistent file or database) context information for the duration of a process that potentially spans many individual activities. BPM tools are called by many names, including "workflow systems," "businessware," "enterprise work management systems" and "business process automation managers." BPM may be a feature in a larger product, or may be the primary role of a particular product.

BPM is a composite market, and can be categorized as either Pure Play (application-independent) or Integrated BPM (part of an IBS). Business-driven BPM decisions usually go in the direction of the pure-play vendors; **however, where architects are involved, integration-centric solutions should not be underplayed.** Architects and integrators should be patient. Eventually the success of BPM will drive



more technical integration purchases, to augment the pure-play systems, which are justified for business reasons now.

Below are some of the advantages/disadvantages of Pure-Play and Integrated BPM.

**Table 12. Pure-Play vs. Integrated BPM**

	<b>Advantages</b>	<b>Disadvantages</b>
<b>Integrated BPM</b>	<ul style="list-style-type: none"> <li>■ Strong routing and transformation</li> <li>■ A rich set of adapters</li> <li>■ High-performance behavior</li> <li>■ Deep system-to-system process control</li> <li>■ Rich programming environments</li> <li>■ Natural composite applications</li> <li>■ Rich technical auditing facilities</li> </ul>	<ul style="list-style-type: none"> <li>■ Reluctance to sell BPM separately</li> <li>■ Lagging human supports (some the exceptions)</li> <li>■ Difficulties experienced by business users</li> <li>■ Expensive implementation</li> <li>■ Absence of rule engines (in some cases)</li> <li>■ Support aimed at technicians and architects</li> <li>■ Sales force aimed at a technical sale</li> </ul>
<b>Pure-Play BPM</b>	<ul style="list-style-type: none"> <li>■ Rich human-to-human support</li> <li>■ Highly visual software</li> <li>■ Ease of development for power users, as well as developers</li> <li>■ Web-service-friendly software</li> <li>■ Easy play with many vendors</li> <li>■ Lower cost for BPM (but brokers are additional)</li> <li>■ Built-in business audit trail</li> <li>■ Vertical and horizontal business process templates</li> </ul>	<ul style="list-style-type: none"> <li>■ Weak system-to-system support</li> <li>■ Need for an integration partner (multiple vendors)</li> <li>■ Less financially strong vendors (some exceptions)</li> <li>■ Reliance on integration partners for technical performance</li> <li>■ Overly simplistic software (for some solutions)</li> <li>■ Misleading expectations (seems simpler than it is)</li> </ul>

Source: Gartner Research, 2003

**Table 13. Business Process Managers Brick**

<b>Baseline Environment (Today)</b>	<b>Tactical Deployment (0-2 years)</b>	<b>Strategic (2-5 years)</b>
<ul style="list-style-type: none"> <li>■ Handysoft Bizflow (Pure Play)</li> </ul>	<ul style="list-style-type: none"> <li>■ Handysoft Bizflow (Pure Play)</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>
<b>Retirement (Technology to eliminate)</b>	<b>Containment (No new deployments)</b>	<b>Emerging (Technology to track)</b>
<ul style="list-style-type: none"> <li>■ TBD</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>
<b>Comments</b>		
<ul style="list-style-type: none"> <li>■ No current IBS tools in place at NIH, therefore no Integrated BPM tools in place.</li> <li>■ Tactical and Strategic products were selected to leverage NIH's investment in products that are a proven fit for NIH's known future needs. Leveraging baseline products in the future will minimize the operations, maintenance, support and training costs of new products.</li> </ul>		

## 7.8 Brick: Integration Middleware — Gateways

There are two types of gateways:

1. Database gateways enable access to heterogeneous DBMSs, usually through a common SQL interface.
2. Communications middleware gateways connect MOM products on the market.

Database gateways enable connectivity to heterogeneous DBMS engines, sometimes including non-relational databases, using a common API (usually SQL) and protocol.

Figure 32. Gateways

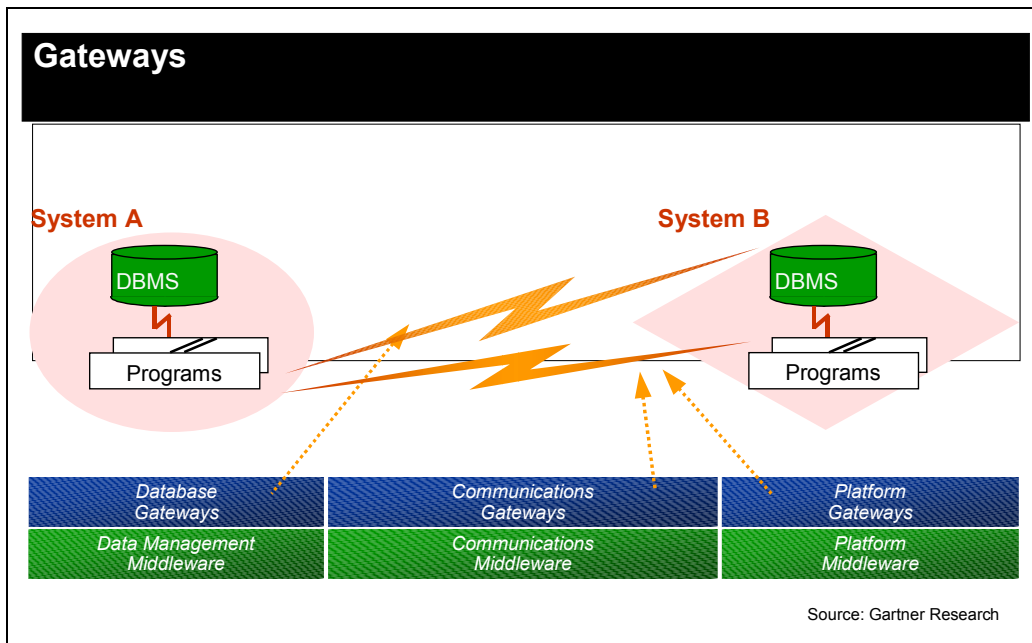


Table 14. Gateways Brick

Baseline Environment (Today)	Tactical Deployment (0-2 years)	Strategic (2-5 years)
Database Gateways <ul style="list-style-type: none"> <li>■ Oracle Transparent Gateway (Oracle to DB2)</li> <li>■ Linked-Server Database Gateway (SQLServer to Oracle or SQLServer to DB2)</li> </ul> Communication Gateways <ul style="list-style-type: none"> <li>■ None</li> </ul>	Database Gateways <ul style="list-style-type: none"> <li>■ Oracle Transparent Gateway (Oracle to DB2)</li> <li>■ Linked-Server Database Gateway (SQLServer to Oracle or SQLServer to DB2)</li> </ul> Communication Gateways <ul style="list-style-type: none"> <li>■ TBD (see "emerging," below)</li> </ul>	<ul style="list-style-type: none"> <li>■ TBD</li> </ul>

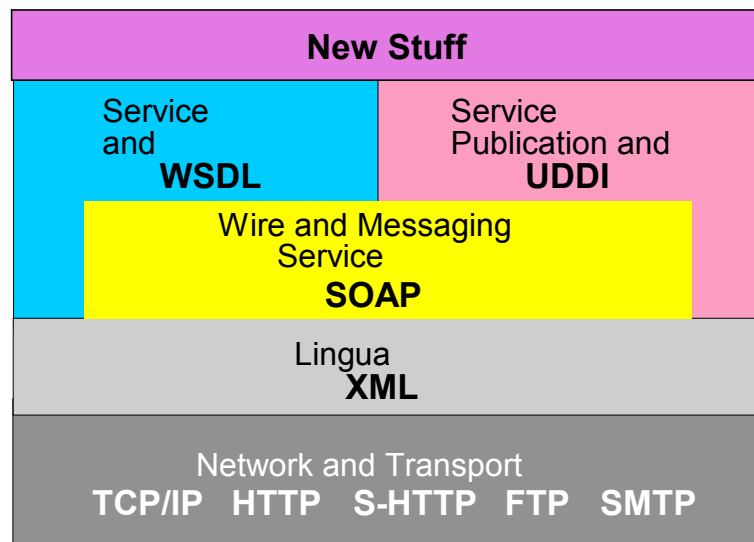
Retirement (Technology to eliminate)	Containment (No new deployments)	Emerging (Technology to track)
■ TBD	■ TBD	■ Communication gateway for PeopleSoft Enterprise Integration
Comments		
■ Tactical and Strategic products were selected to leverage NIH's investment in products that are a proven fit for NIH's known future needs. Leveraging baseline products in the future will minimize the operations, maintenance, support and training costs of new products.		

## 7.9 Brick: Web Services

Web services are not really a technology; they represent software components and a common set of standards supported by multiple, different technologies and vendors. Web services are Web-based services that use any one or more of three related XML-based standards<sup>1</sup> including:

- SOAP — A simple wire protocol for interprogram communication)
- WSDL — Web Services Description Language, an interface-definition syntax
- UDDI — Universal Description, Discovery and Integration, defines how a directory is used to register Web services.

Figure 33. Components of Web Services



Gartner

Copyright © 2003

Web services can operate over Internet protocols. These include TCP/IP, the standard Internet transport; secure sockets; FTP for uploading and downloading files to and from

<sup>1</sup> See Appendix D, Introduction to Service-Oriented Architecture, for more on both SOA and WSA.

the Internet; HTTP and secure HTTP (S-HTTP) for sending information over the Web; and SMTP (and POP3) for e-mail messaging; and even MOM and Java Messaging Service (JMS). The second fundamental technology is XML, which is the language used to create the messages, files, metadata and documents that define and describe Web services. In addition to HTML, Web services make use of one or more of these technologies:

- SOAP lets one application invoke a remote procedure call (RPC) on another application, or pass structured data to a remote location using XML messages and the Web.
- WSDL is a formal XML vocabulary for describing Web services, their interfaces and basic implementation information for use in Web services registries and repositories.
- UDDI is a platform-neutral registry for publishing, querying, finding and invoking Web services via metadata and interfaces.

Taken together, SOAP, WSDL and UDDI form the Web services technology canon that fits atop the XML and Internet infrastructure.

Here are some of the many sources for Web services:

- Applications written in Java J2EE
- Applications written in Microsoft.NET (all Common Language Runtime<sup>1</sup> languages)
- Applications developed with ColdFusion MX
- “Wrapped” service programs from legacy applications
- IBSs
- Commercial off-the-shelf applications
- Commercial service providers (Internet).

---

<sup>1</sup> The Common Language Runtime (CLR) provides a solid foundation for developers to build various types of applications. It provides benefits such as vastly simplified development, and seamless integration of code written in various languages.

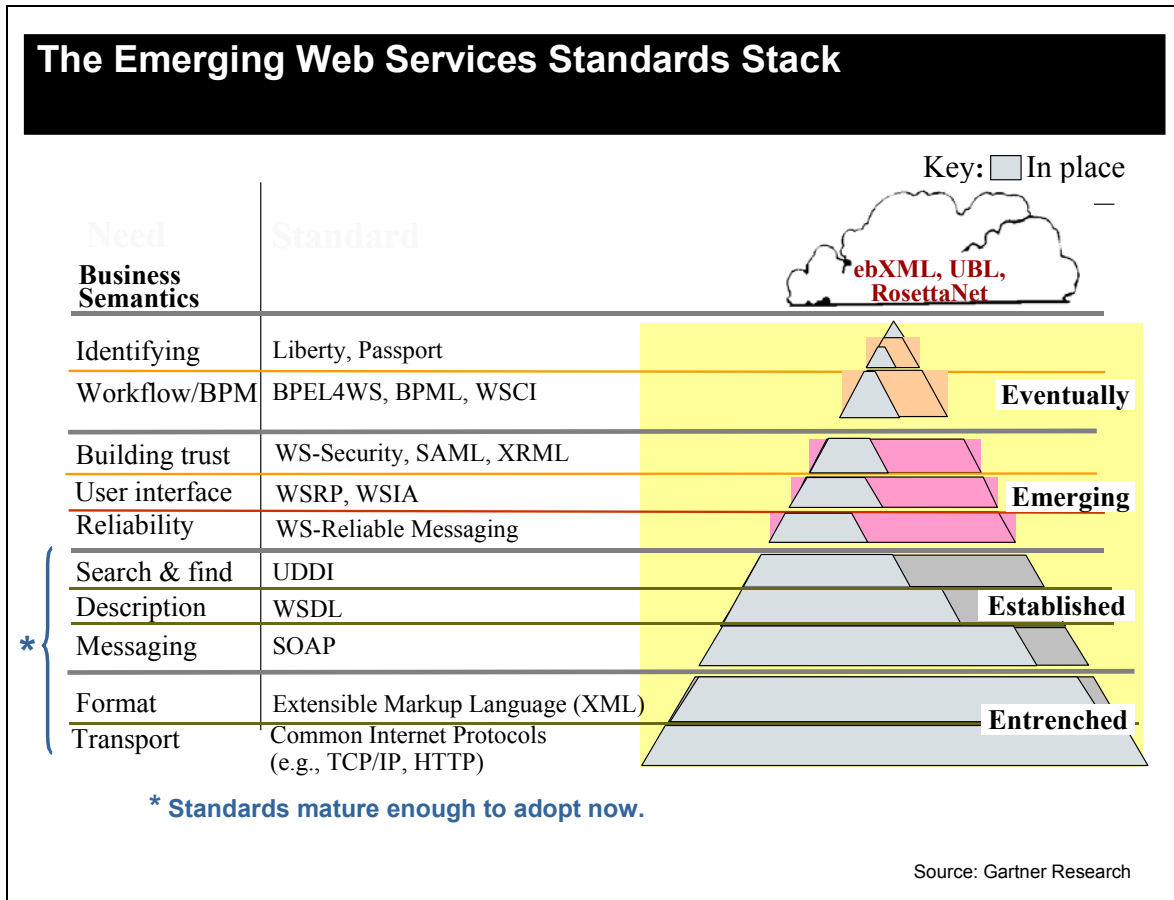
**Table 15. Web Services Brick**

<b>Baseline Environment (Today)</b>	<b>Tactical Deployment (0-2 years)</b>	<b>Strategic (2-5 years)</b>
<ul style="list-style-type: none"> <li>■ XML (in other contexts)</li> <li>■ Some experimental use of Web services</li> </ul>	<ul style="list-style-type: none"> <li>■ XML</li> <li>■ SOAP</li> <li>■ WSDL</li> <li>■ UDDI</li> </ul>	<ul style="list-style-type: none"> <li>■ XML</li> <li>■ SOAP</li> <li>■ WSDL</li> <li>■ UDDI</li> <li>■ Plus additional standards as they mature</li> </ul>
<b>Retirement (Technology to eliminate)</b>	<b>Containment (No new deployments)</b>	<b>Emerging (Technology to track)</b>
<ul style="list-style-type: none"> <li>■</li> </ul>	<ul style="list-style-type: none"> <li>■</li> </ul>	<ul style="list-style-type: none"> <li>■ Web Services standards (see figure in report)</li> </ul>
<b>Comments</b>		
<ul style="list-style-type: none"> <li>■ Tactical and Strategic products were selected to leverage NIH's investment in products that are a proven fit for NIH's known future needs. Leveraging baseline products in the future will minimize the operations, maintenance, support and training costs of new products.</li> </ul>		

The beauty of Web services today is in their simplicity. Eventually, however, complexity will creep in. Vendors (and enterprises) are developing additional layers to the existing Web services stack to address perceived (and real) issues, such as security, transaction management, user interface development, collaborative and peer-to-peer environments, business-to-business (B2B) interactions and more. The emerging stack comes in multiple flavors, depending on the vendor, industry association or standards organization that is authoring the additions.

There will be recurring attempts to build an entire stack of Web services standards that might satisfy every requirement that an enterprise might foresee, and without exception, these attempts will fail due to the vastness of their scope. Electronic business XML (ebXML) might be one such example. More importantly, Web services standards need to fit within a larger framework that can support comprehensive enterprise requirements. One such framework is depicted in the next figure.

**Figure 34. Emerging Web Services Standards**



Source: Gartner, 2004

The Web Services brick should be updated from time to time as the emerging standards mature and become established standards.

## **8.0 Gap Analysis**

### **8.1 Enterprise API Design and Implementation**

- Architecture education and technical training are required to equip developers and project staff to plan for and implement the recommendations of this architecture.

### **8.2 Integration Patterns and Related Middleware**

- Requisite enterprise middleware tools must be purchased and made available to NIH.
- Architecture education and technical training are required to equip developers and project staff to plan for and implement the recommendations of this architecture.

## 9.0 Next Actions

The integration domain team makes the following recommendations:

1. Web services architecture and standards should be mandatory for integration APIs. If the established standards cannot meet business or technical needs (e.g., for workflow or distributed transaction control) an emerging standard may be used or a waiver should be sought.
2. NIH should establish an integration competency center.
3. NIH should purchase specialized integration middleware for use by all enterprise systems and all ICs who have this type of need. This should be for common use, and should be supported by the competency center. So far, integration middleware is rare at NIH, so there is an opportunity to standardize it now — before individual point decisions are made.
4. NIH data standards should be defined for enterprise data classes. These may include encoding standards, business rules (edits), lists of valid values, defined syntax and semantics for NIH-specific data fields, etc.
5. In addition to improved integration methods, the need for extension systems should be reduced. (This would reduce the scope of the integration problem.) If both are done, the extension system problem will be greatly reduced.
6. Training is needed around application integration decisions, and tools selected.
7. NIH should perform a pilot application integration project.



## 10.0 Change History/Document Revisions

Date	Change Author	Change Authority	Change Event	Resulting Version
26 July 2004	Jay Shah, Gary Long	Jack Jones	Original Production	1.0



**11.0 Appendix A — Glossary of  
Terms/Acronym Key**

## Appendix A — Glossary of Terms

Term	Definition
A2A	Application-to-application
ADB	Administrative Data Base (ADB) — An IMS application that provides computer support for a broad range of NIH business functions including the purchase, receipt, and payment of goods and services; the tracking and supplying of nine inventories; sixteen service and supply fund activities; foreign, domestic, and local travel; and property management.
AdEERS	Adverse Event Expedited Reporting System
ADK	Adapter development kit
AFS	Andrew File System
AIR	Acquisition information reporting
AMBIS	Administrative Data Base Information System — an information system that provides timely and accurate information from the Administrative Data Base and Central Accounting System (CAS) to the NIH user community.
AO	Administrative Official
API	Application programming interface
APS	Application platform suite
ASC	Accredited Standards Committee
ASP	Application service provider
B2B	Business -to- Business
BAM	Business activity monitoring
BI	Business intelligence
BOD	Business Object Document (OAG)
BPE L4 WS	Business Process Execution Language for Web Services
BPM	Business process management
BPML	Business Process Modeling Language
BPN	Business process network
BPN	Business partner network (in Figure 10)
BRE	Business rule engine
CAPS	Childhood Asthma Prevention Study
CAS	Central Accounting System
CBD	Component-based design
CC	Warren Grant Magnuson Clinical Center (CC) — Established in 1953
CDR	Clinical data repository
CDUS	Clinical Data Update System
CDW	Clinical data warehouse
CGI	Common Gateway Interface
CI	Content integration

Term	Definition
CIT	Center for Information Technology (CIT formerly DCRT, OIRM, TCB) — Established in 1964
CM	Change Management
COM	Component Object Model
CICS	Customer Information Control System
CORBA	Common Object Request Broker Architecture
CRIS	Clinical Research Information System
CRISP	(Computer Retrieval of Information on Scientific Projects) is a searchable database of federally funded biomedical research projects conducted at universities, hospitals, and other research institutions.
CSR -	Center for Scientific Review (CSR) - Est. 1946
CSV	Comma Separated Values
CTEP-ESYS	Cancer Therapy Evaluation Program Enterprise System
CTSU	Cancer Trials Support Unit
DB	Database
DBMS	Database management system
DCE	Distributed computing environment
DCIS	Department Contracts Information System
DCOM	Distributed Component Object Model
DFM	Desktop File Manager
DHHS	Department of Health and Human Services
DICOM	Digital Imaging and Computing in Medicine
DML	Data manipulation language
DNS	Domain name service
DOM	Document Object Model
DTP	Developmental therapeutics program
DW	Data Warehouse
EAD	Enterprise application development
ebXML	Electronic Business XML
ECB	The Electronic Council Book is a web-based utility that provides on-line Summary Statements, using World Wide Web and Internet capabilities for database search and retrieval.
EDA	Event Drive Architecture
EDI	Electronic data interchange
EHRP	Enterprise Human Resources and Payroll
EII	Enterprise information integration
EJB	Enterprise Java Beans
e-Log	The fundamental focus of e-Log is to help Grants and Program staff with their daily operations. The interface is very easy and requires little to no training to use. E-Log works with IMPAC II data, which is downloaded nightly
EMIS	Ethics Management Information System
ENS	An enterprise nervous system (ENS) implements a new layer

Term	Definition
	of application-level data and logic, extending the enterprise network and making it as smart as application systems.
Enterprise applications	Enterprise Applications are major computer applications that have “enterprise scope” as defined in the Application Architecture report <sup>1</sup> .
eRA	The electronic research administration (eRA) is NIH's infrastructure for conducting interactive electronic transactions for the receipt, review, monitoring, and administration of NIH grant awards to biomedical investigators worldwide.
ERP	Enterprise Resource Planning
ESB	Enterprise service bus
ESP	External service provider
ETL	Extraction, transformation and load
Extension System	<i>Extension System</i> is an NIH term that is used to describe any application add-ons that extend the capabilities of a core application.
F&A Rates	Facilities & Administrative Rates
FAC	Functional Advisory Committee (FAC) membership consists of leaders in the scientific and business communities who represent the management and policy level end user.
FAES	Foundation for Advanced Education in the Sciences
FIC	John E. Fogarty International Center (FIC) — Established in 1968
FMS	Financial Management Services
FPS II	Fellowship Payment System II
FTP	File Transfer Protocol
GOB	Grants Operations Branch
GPO	U.S. Government Printing Office
GSA	General Services Administration
HHS	Department of Health and Human Services
HIPAA	Health Insurance Portability and Accountability Act
HL7	Health Level 7
HRDB	Human Resources Database
HTTP	Hypertext Transfer Protocol
IAE	Integrated acquisition environment
IBS	Integration broker suite
IDMS	Pharmacy Investigational Drug Management System
iEDISON	U.S. Government's Internet center for reporting inventions developed with government funding. Website: <a href="https://s-edison.info.nih.gov/iEdison/">https://s-edison.info.nih.gov/iEdison/</a>
IGT	intra-governmental transactions

<sup>1</sup> Separately published and available on the EA Portal, September 2003.

Term	Definition
ILS	Integrated Library System
IMPAC II	NIH's extensive internal information management system for application and award data.
Innopac	UNIX-based system for public access to catalogues and modules to support cataloging, circulation, serials and acquisitions.
ISB	Institute for Systems Biology
ITAS	Integrated Time and Attendance System
JDBC	Java Database Connectivity
JEFIC	JE Fogarty Database of Foreign Visiting Scientists
JMS	Java Messaging Service
LDAP	Lightweight Directory Access Protocol
LIS	Laboratory Information System
MeSH	Medical Subjects Heading
MIS	Medical Information System
MOM	Message-oriented middleware
MS DTS	Microsoft Data Transformation Service
NARA	National Archives and Records Administration
NBRSS	The NIH Business and Research Support System (NBRSS) is the combination of the NIH Business System (NBS) and the Enterprise Human Resources and Payroll System (EHRP)
NBS	NIH Business System
NCCAM	National Center for Complementary and Alternative Medicine (NCCAM) — Established in 1992
NCI	National Cancer Institute
NCMHD	National Center on Minority Health and Health Disparities (NCMHD) — Established in 1993
NCRR	National Center for Research Resources (NCRR) — Established in 1962
NED	The NIH Enterprise Directory (NED) is a centrally-coordinated, electronic directory that CIT is developing to maintain accurate, current information for all individuals using NIH services or facilities.
NEI	National Eye Institute (NEI) — Established in 1968
NFS	Network File System
NHGRI	National Human Genome Research Institute (NHGRI) — Established in 1989
NHLBI	National Heart, Lung, and Blood Institute (NHLBI) — Established in 1948
NIA	National Institute on Aging (NIA) — Established in 1974
NIAAA	National Institute on Alcohol Abuse and Alcoholism (NIAAA) — Established in 1970
NIAID	National Institute of Allergy and Infectious Diseases (NIAID) — Established in 1948
NIAMS	National Institute of Arthritis and Musculoskeletal and Skin

Term	Definition
	Diseases (NIAMS) — Established in 1986
NIBIB	National Institute of Biomedical Imaging and Bioengineering (NIBIB) — Established in 2000
NICHHD	National Institute of Child Health and Human Development (NICHD) — Established in 1962
NIDA	National Institute on Drug Abuse (NIDA) — Established in 1973
NIDB	The NIH Intramural Data Base (NIDB) is a tool that makes intramural research information available online to the NIH community, to extramural collaborators, and to the public
NIDCD	National Institute on Deafness and Other Communication Disorders (NIDCD) — Established in 1988
NIDCR	National Institute of Dental and Craniofacial Research (NIDCR) — Established in 1948
NIDDK	National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK) — Established in 1948
NIEHS	National Institute of Environmental Health Sciences (NIEHS) — Established in 1969
NIGMS	National Institute of General Medical Sciences (NIGMS) — Established in 1962
NIHITS	NIH Online Training Nomination System
NIMH	National Institute of Mental Health (NIMH) — Established in 1949
NINDS	National Institute of Neurological Disorders and Stroke (NINDS) — Established in 1950
NINR	National Institute of Nursing Research (NINR) — Established in 1986
NLM	National Library of Medicine (NLM) — Established in 1956
NSM	Network and systems management
OAG	Open Application Group
OCI	Oracle Call Interface
OD	Office of the Director
ODBC	Open Database Connectivity
ODS	Operational Data Store
OLE DB	Object Linking and Embedding Database
OLTP	Online transaction processing
OMB	Office of Management and Budget
ONC	Open Network Computing
OOD	Object-oriented design
OPM	Office of Personal Management
ORB	Object request broker
OS	Operating System
OTM	Object transaction monitor
PCA	Packaged composite application
PH	E-mail Directory and Forwarding Service (PH)

Term	Definition
PIP	Packaged integrating process
PL/SQL	Procedural Language/SQL
POM	People Organization Module Database
POPTRACK	Population tracking system
QOS	Quality of service
RIMS	Robocom Inventory Management Daily System
RIS	Radiology Information System
RPC	Remote procedure call
SAML	Security Assertion Markup Language
SCCM	Software Configuration and Change Management
SCP	Secure Copy
SDF	Standard File - These are text files containing data arranged in a standard format
SES	Smart enterprise suites
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
SPI	System programming interface
SQL	Standard Query Language
SSA	Social Security Administration
SSH	Secure Shell
STP	Straight-through transaction processing
TCP/IP	Transmission Control Packet/Internet Protocol
TDN	Transaction delivery network
TFS	This Fine System
TI	Transport independent
TOXNET	Toxicology Data Network
TP	Transaction processing
TPM	Transaction processing monitor
TRM	Technical reference model
UDDI	Universal Description, Discovery, and Integration
UN/EDIFACT	United Nations Electronic Data Interchange for Administration, Commerce and Transport
URI	Uniform resource identifier
VAN	Value-added network
VEDS	Visual Employee Data System
VSOE	Visual Status of Funds
WAS	Web Application Server (SAP)
WSA	Web services architecture
WSAM	Web services application management
WSB	Web services broker
WSC	Web services controllers
WSCl	Web Service Choreography Interface
WSDL	Web Services Description Language
WSIA	Web Services Interactive Applications



Term	Definition
WSM	Web services middleware
WSN	Web services network
WSRP	Web Services for Remote Portlets
XDBMS	XML database management system
XML	Extensible Markup Language
XRML	Extensible rights Markup Language
ZLE	zero-latency enterprise



**12.0 Appendix B — Current State Survey Results**

## Appendix B — Current State Survey Results

### Presentation Layer

- NINDS uses presentation layer integration for the FinEX system.
- OD is using Handysoft's Bizflow workflow tool in one application.
- NIAID integrates applications through the use of dashboard applications and workflow products at the presentation layer, using BizFlow that touches other Windows applications and databases — this may be an example of BPM.

### Process Layer

- OD has listed Crystal Reports and Lotus Notes as Middleware — APIs are being used for integration; ActivePDF converter is also used.
  - Note that OD and NIAID both use .NET to call the Crystal API as a formatting utility.
- OD's NIHITS II application uses Java/Perl components for import/export with other applications/sources.
- NIAID uses direct executable calls for application layer integration for archive/retrieve capability to gather resources that are presented to users.
- Web services are being used by the following ICs for integration:
  - NIAID is developing Web services in .NET to expose data resources — no specifics provided.
  - NHLBI is using Web services in a limited fashion such as allowing applications to share look up lists.
- CIT also using Web services for Parts Logic (application owned by OD). Used for internal authentication.
- NIAID uses COM APIs for Crystal Reports, Office, Acrobat, custom components, WinZip, and other applications. BizFlow is being used for automated workflow and event-driven actions.
- Office of Science Education (OD) using Lotus Notes for Web server integration
- Many ICs are using IMPAC II APIs (stored procedures) for data transfer bi-directionally, for inputting and retrieving data.
  - NHLBI uses IMPAC II APIs to export data from the e-Log application to IMPAC II.
- NCRP has listed numerous applications for retrieving information — Convera Retrievalware (search engine), ASP, Cold Fusion, etc. — these tools provide a means for scientists to login and submit their progress reports.
- NHLBI is using DICOM (industry standard for medical images) for transferring, retrieving and parsing medical images.

- NHLBI is using SQL View and APIs for export of data from IASP to the AMIS and Sirius applications.
- CC uses a QDX Integrator (HL7 messaging) as an interface engine for transferring data between IDMS, LIS, MIS, EKG System, Transcription System, Clinical Data Warehouse and RIS (this is before Phase 1 implementation of CRIS, which replaces MIS for certain functions).
- CC uses the CareData Interface Engine for data transfers between MIS, Wristband System, PYXIS, and Clinical Data Warehouse (this is before Phase 1 implementation of CRIS which replaces MIS for certain functions).

## Data Integration Layer

- FTP is a popular integration mechanism in all ICs:
  - OD currently has three to four ways that Archives accepts information.
  - NHLBI uses FTP to download CAPS data and load into SQL Server.
  - NHLBI downloads indirect cost rates and rate agreements from the DHHS FTP site.
  - NCRR is using FTP.
  - CA uses FTP and database dump/load, and XML for import/export of CTEP data to/from AdEERS and CDUS systems — no specifics on the role each mechanism plays.
  - NLM uses FTP to import data into TOXNET from government/private providers and the Swedish Riskline organization and to export data from TOXNET to commercial DB repositories, labs and universities.
  - NIAMS uses FTP of DB2 files from HRDB and Payroll to VEDS.
- Several ICs are using XML files to transfer data:
  - OD uses XML (file based) to integrate three applications with GemCRIS.
  - NHLBI uses ASP to retrieve IMPAC II data and XML to transfer that data to GOB spreadsheets.
  - CIT uses XML file transfer to export data to PPIRS.
  - CIT uses XML to import data from NCI and PubMed.
  - NIDB gets xml from PubMed.
- Tape/CD imports are also used for data transfer:
  - For NARA (National Archives) transfers, etc. Also, CDs are created and sent (CPS).
- E-mail is used as input/output tool in several cases.
- ODBC/ODBC tools are common for data access within ICs:
  - NINDS uses ODBC tools and database drivers for data access/sharing — no specific tools or systems listed.

- Linked/Server is a DB gateway used by SQL/Server and Oracle (used by many ICs).
- NIAID has identified ODBC as a means for retrieving data from sources such as IMPAC II and the Central Accounting System going to stored procedures or to raw data.
- CIT uses ODBC tools.
- NIAAA uses ODBC as an integration mechanism — no specifics provided.
- NLM uses ODBC and SSH tunnel to import data into TOXNET from Sseus.
- Many ICs have identified common data sharing across multiple applications:
  - NINDS uses data sharing (via views, links, etc.) for several systems and databases — FinEX, People Organization Module Database, iWin, Coding System.
    - POM is updated by a daily script that applies updates from Exchange.
  - NIAID's VED database provides source data for several administrative applications.
  - NHLBI uses data sharing (via views, links, etc.) from a common database server for interfacing internal applications.
- NHLBI is using OLEDB and JDBC for retrieving some run-time IMPAC II data and OLEDB for nightly batches downloaded from Data Warehouse.
- NHLBI is using scripting languages to load and/or update batch data in IMPAC II and Perl Script for loading indirect cost rate files into the database and Web server.
- NHLBI is using a Linked/Server (gateway) on its MS SQL Server to IMPAC II IRDB and OLTP servers and for nightly batch downloads from Data Warehouse.
- NHLBI is using Microsoft Data Transformation Services (DTS, a component of SQL/Server that can schedule transactions as packets, and can also do rollback) for data loads from IMPAC II and the Data Warehouse.
- CC's MIS system sends clinical data extracts to the CDW file repository — no specifics provided.
- CC's MIS system sends ADT data/diet orders to Nutrition (DFM) system — no specifics provided.
- CIT's ADB system imports files from Self Service Stores, PRB, CSR, GPO, CAS, USA Bank, RSB and the NIH Training Center. The Self Service Stores send the file to the mainframe via FTP; the ADB then picks up the file.
- CIT's ADB system exports files to IRS (files are placed on a magnetic device via tape), CAS, Data Warehouse (files are created on the mainframe), RIMS, DES, OPM, Treasury, Self Service Stores, RSB, NCRR, and SSA.
- CIT's ADB/nVision/FMS integrates with IMPAC II (via FTP to the mainframe to be processed by the CAS).

- CIT's ITAS application uses a data load directly from NED to import and exports data to ePayroll with a yet-to-be-determined mechanism.
  - According to project lead, data movement is using FTP.
  - According to technical guide, Kermit (file transfer program) is currently used to move data in a flat file to Treasury.
  - In the future, this data transfer may go to DFAS, which would then feed Treasury.
- CIT uses MS DTS tool to check for the existence of a file, move it and then load data.
- CIT does have some instances of data pushing:
  - CAS -> NBS ->Data Warehouse
  - Push and pull from Treasury
  - Push from CAN maintenance.
- CIT's NED application imports data from JEFIC/DB2, HRDB/DB2, and FPS2/DB2 using a custom Perl connector/DB2 connect; Continuum using a custom Perl connector and shared folders; PH using a custom Perl connector and the Andrew File System (AFS); and Active Directory using custom Perl scripts and LDAP.
- CIT's NED application exports data to Continuum and Conveyant using a custom Perl connector, shared folders, and CSV files; Innopac using a custom Perl connector and FTP; and MVS Customer Registry, NBRSS and ITAS using a custom Perl connector/DB2 connect.
- CIT's NIDB application imports data from NHGRI and exports to intramural ICs via excel spreadsheets (manual export).
- CIT's NIDB application exports data to NIEHS, NINDS and NCI via database views.
- CIT's NIDB imports NED data through direct database queries.
- NLM uses a batch database read using stored procedure to import data to Medlars from SEF, Contractors and NIH Lister Hill Center, MeSH database, Journal publishers, and ILS.
- NLM uses an ad hoc database read to import data to Medlars from SEF.
- NLM uses a put-to-a-public-FTP server for licensed access for export of data from Medlars to the public.
- NLM uses a put-to-internal-FTP server to export data from Medlars to NCBI, Medline Plus, and Voyager.
- NLM imports data from Chem-ID and government/private providers to TOXNET using transfer through secure copy (secure FTP).
- NLM uses a CD-ROM to import data from EPA into TOXNET.
- NLM uses Internet transfer to import data from the Internet into TOXNET.

- **HR Data:**
  - A Peoplesoft feed is available from HHS, but it is not allowed.
  - NIDB would like to use a local HR system.
- Neon DB2 drivers and DB2 direct drivers are in use for extracting data from DB2. NIAID is using batch updates from flat file to import data from ADB into AMBIS.
- At NIAID, a lot of scientists download a lot of data and massage it using standard MS-Office tools.
- NIAID imports data into a secondary data repository using Oracle\_Download for transformed IMPAC II data to be accessed by various applications (Oracle\_Download is the name of a shared database implemented in SQL Server that holds data downloaded from IMPACII that is of interest to NIAID).
- NCCR's Scientific Information System downloads data from IMPAC II.
- NCCR passes data from grantees in files to other systems — no specifics provided.
- NCCR's budget office uses a software package, VSOFF, which uses data from the Data Warehouse (uses Neon Shadow Direct ODBC driver).
- OD is using SDF file to mainframe for integration of eRA to FMS (not FTP — follow up on mechanism).
- OD imports HR data from HRDB to Ethics Management Information System (EMIS) via stored procedures and ODBC drivers.
- Flat ASCII file transfer is being used for data sharing at OD (TechTracs exports to Data Warehouse) and NIAID for systems such as ITAS.
- There is a lot of manual integration where users pull data from reporting tools, download data into Microsoft Office, or use one-off applications to get data from multiple sources.
  - Office 2003, just released by CIT, allows a user to establish a live link to source data through an Excel spreadsheet — updates to the data in Excel will be replicated in the source database.

### **eRA Current State**

eRA currently integrates with CAS (central accounting system), integration is not efficient. eRA submits a transaction set, which is then processed. Sometimes an error log is returned, but there is no real-time data transaction.

eRA as the enterprise system integrates with many extension systems used by individual ICs (or groups of ICs).

Ways ICs integrate with eRA:

- Use data directly by SQL queries, ODBC calls, JDBC.
- Link eRA data to their own data via interface.

- Download eRA data to a local server, supplement with local IC data.

Other than sensitive data, eRA allows read-only access to all data. Updates to eRA data must be performed within an eRA application or via an API developed by eRA. APIs are designed for specific data fields and have business rules and other integrity logic incorporated.

Mechanisms used by eRA today:

- XML, SOAP with attachments for communicating with external partners
- OCI (Oracle Call Interface)
- ODBC
- Custom build processes for bridges with legacy systems.

### **CTEP-ESYS**

CTEP-ESYS (an NCI system) has been built as a single enterprise solution incorporating multiple, branch-specific applications on a single database with appropriate sharing of data and status information resulting in inter-departmental accumulation of information and an integrated workflow (e.g., PATS, Address Module).

Although the CTEP-ESYS does not integrate with eRA, this enterprise solution includes a number of well defined interfaces:

- CDUS — FTP transmission using flat files
- Blinded Orders — FTP transmissions using flat files
- AdEERS data — XML data sets (generated at run time) for authorized users
- Registration data (two-way) with CTSU systems — Oracle advanced queuing
- Real-time access to Medline and Developmental Therapeutics Program (DTP) using standard published APIs from the respective organizations
- Upload of accrual data for Population Tracking System (POPTRACK), an eRA application) using published interfaces and APIs
- Common data elements — static snapshots.

Mechanisms include:

- Published APIs
- XML
- Oracle Advanced Queuing
- File transfer products.

CTEP-ESYS supports the needs of over 10,000 physicians providing care to over 500,000 patients in relation to over 2,000 protocols related to 300 drug agents at 6,000 treatment sites.





IC Short Name	FTP	Secure FTP	Kermit	XML	ODBC/JDBC	DB2 Connect	Data Sharing	Direct DB Queries	OCI	Database Gateway	API's	Custom Interfaces	Cold Fusion	QDX	CareData Engine
CC								✓						✓	✓
CIT	✓		✓	✓	✓			✓		✓		✓			
CSR															
FIC															
NCCAM															
NCI (or CA)	✓			✓				✓							
NCMHD															
NCRR	✓				✓			✓				✓	✓		
NEI															
NHGRI															
NHLBI	✓			✓	✓		✓			✓	✓	✓			
NIA															
NIAAA					✓										
NIAID					✓	✓	✓				✓				
NIAMS	✓														
NIBIB															
NICHD															
NIDA															
NIDCD															
NIDCR															
NIDDK															
NIEHS															
NIGMS															
NIMH															
NINDS				✓	✓		✓			✓	✓		✓		
NINR															
NLM	✓	✓			✓			✓							
OD	✓	✓		✓	✓	✓		✓			✓	✓			
ERA				✓	✓			✓	✓			✓			
CTEP-ESYS	✓			✓				✓			✓				



## 13.0 Appendix C — Introduction to Web Services

## Appendix C: Introduction to Web Services

### What is a Web Service?

Web services are software components that employ one or more of the following technologies: SOAP, WSDL and UDDI to perform distributed computing. Use of any of the basic technologies — SOAP, WSDL or UDDI — constitutes a Web service. Use of all of them is not required.

### What is SOAP?

SOAP is designed to enable enterprises to easily publish data and services via the Web. It lets one application invoke an RPC on another application, or pass an object or other information to a remote location using an XML message and the Internet. SOAP satisfies the growing need for business partners to exchange structured data via the Web independently of each other's underlying application platform and operating environment. As such, it functions as a wire protocol to connect multiple sites which each might use as an information server, object broker or other facilities to integrate and process the information.

SOAP's implementation requirements are simple: a set of mutually acceptable XML message formats; basic XML processing (that is, an XML parser and an engine to translate information to and from XML) at both the requestor and the responder; and a Web connection in the middle. XML namespaces and schemas are optional, but they can ease processing by providing references for XML elements and attributes.

Figure 35. SOAP

## SOAP: A Communication Solution

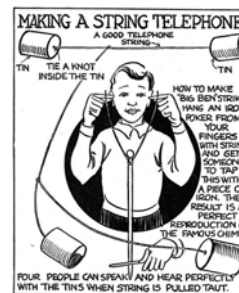
### Simple Object Access Protocol

Enables enterprises to publish data, expose services, and invoke RPCs via XML and the Web.

Supports multiple platforms and operating systems.

SOAP has three key parts:

- **Envelope:** Identifies message content, recipients, processing information and governing schemas
- **Encoding Rules:** Procedures for serializing and deserializing data and methods
- **Request and Response Conventions:** Enables bidirectional and unidirectional communication using RPCs and messages



## What is WSDL?

Having a standard set of service-oriented messaging facilities and a standardized mechanism for discovering services are of no use if there's no way to describe what a service is and how its facilities can be accessed. As communication protocols and message formats are standardized in the Web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. Enter WSDL, which is maintained by the W3C Web Services Activity. WSDL takes SOAP one step further. It originated because of the need to tighten up SOAP semantics and provide a standard set of interfaces to Web services. WSDL addresses these needs by defining an XML grammar for describing network services as collections of communication endpoints that are capable of exchanging messages. WSDL also provides a way to publish data to XML registries and repositories.

Figure 36. WSDL

## WSDL: Describe Your Stuff

### Web Services Description Language

A formal XML vocabulary and grammar that lets organizations describe, discover and use Web services in a UDDI registry or other location.

WSDL's key concepts are:

- Describes Web services as pairs of endpoints, which exchange messages about each other's capabilities
- Messages may contain document- or procedure-oriented information
- Documents include abstract (implementation-independent) and concrete (instance-specific) elements
- Supports various core technologies, including SOAP XML schemas, HTTP GET/POST and MIME e-mail



Gartner

Copyright © 2003

## What is UDDI?

UDDI is a technology for publishing, querying, finding and invoking Web services using a registry that provides data and metadata about the services and pointers to where the services are located. The technology consists of the registry itself — which may be privately operated, publicly available (that is, the Universal Business Registry hosted by IBM, SAP, Microsoft and NTT Communications), or semiprivate — instructions for operating the registries, and APIs for managing the registry information and performing queries. The registry is modeled on the Domain Name Service (DNS), a commonly available and familiar registry technology. In July 2002, UDDI was handed over to OASIS by the UDDI Committee. UDDI 3.0 — now under the auspices of OASIS — is the latest version of the standard; it provides much stronger querying and categorization facilities, a much richer API set, and new information for operating private and semiprivate registries. The trend in UDDI seems to be toward private or semiprivate

registries, which have more control over the type and format of registry data, as well as those who use it.

Figure 37. UDDI

## UDDI: Discovering Web Services

### Universal Description, Discovery and Integration

UDDI gives enterprises a uniform way to describe their services, discover other companies' services and understand the methods required to conduct business with a specific entity.

UDDI key concepts are:

- It is a registry, not a repository
- Provides detailed instructions for operating private, public and semiprivate registries
- Registries offer data, metadata, bindings, pointers and documents for finding and invoking Web services
- Includes client and server APIs for publishing, editing and querying registry entries



**Gartner**

Copyright © 2003

UDDI is:

- A registry, not a repository
- Documents, interfaces and metadata about Web services
- Models that reuse WSDL documents
- A set of APIs for finding and invoking these services.

UDDI is not:

- Software and hardware on a node
- Fail-safe validation of publishers and their Web services
- How publishers and subscribers conduct business
- Web services implementations
- Messaging or wire protocols.



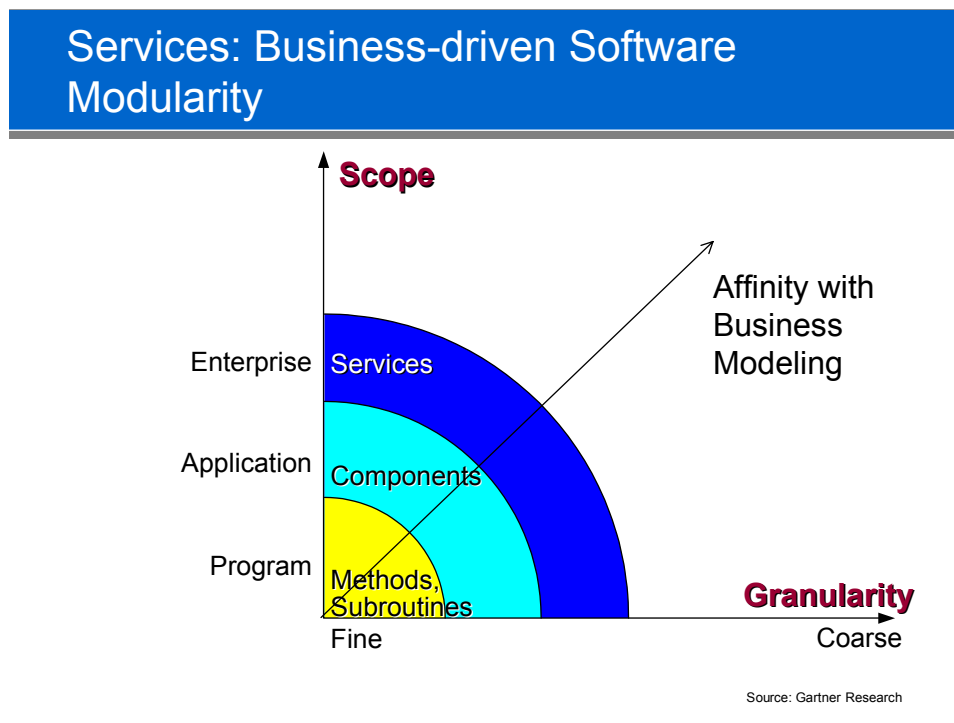
## 14.0 Appendix D — Introduction to Service-Oriented Architecture

## Appendix D — Introduction to Service-Oriented Architecture

### Services and SOA

The modern business application topology is multilayered services and Web services are composed of components, themselves built of objects. In each layer, encapsulation is at different granularity. Object methods (C#, Java) are fine-grained and accessed in the same program context via shared memory. Components (EJB, .NET Remoting) may consist of hundreds of objects and are accessed via distributed computing middleware (RPC, ORB) by other components of the same application. Service interfaces are designed to be accessible by other applications. The usage patterns determine the different granularity and scope of object methods, components and services.

Figure 38. Service Scope vs. Granularity

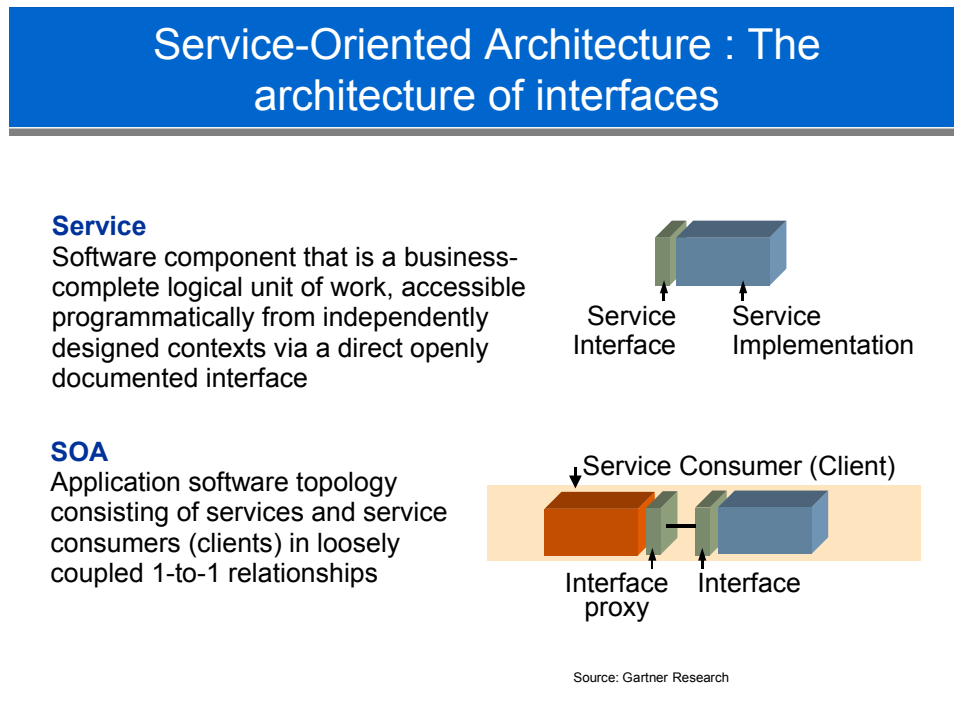


Technically, services are software modules that use a separable platform-independent and well-defined public programmatic interface. Because services may be advertised to other applications or other enterprises as the business service of the application or the enterprise — semantically, services have business identity and completeness. Services can be entities in business analysis, while subroutines and components are technically inclined, intended to improve the technical architecture of software.

A service is a software component that is suitable for cross-application access. A service represents a business function, though it is implemented as a technical component. A service is the point of linkage between business and technical design of business applications.



Figure 39. SOA — The Architecture of Interfaces



A service is never a complete application or a complete transaction. It is always a building block. SOA is the architecture of an application that utilizes services. While services define potentially-reusable business functions, SOA binds services into applications. SOA applications consist of services and service consumers. Services are defined by their interfaces, which wrap their implementations (sometimes complex integrated flows, other times a simple single program). Logical design of SOA is focused on the definition of service interfaces and design of interactions between service interfaces. Technically, the design of SOA also includes design of service implementations.

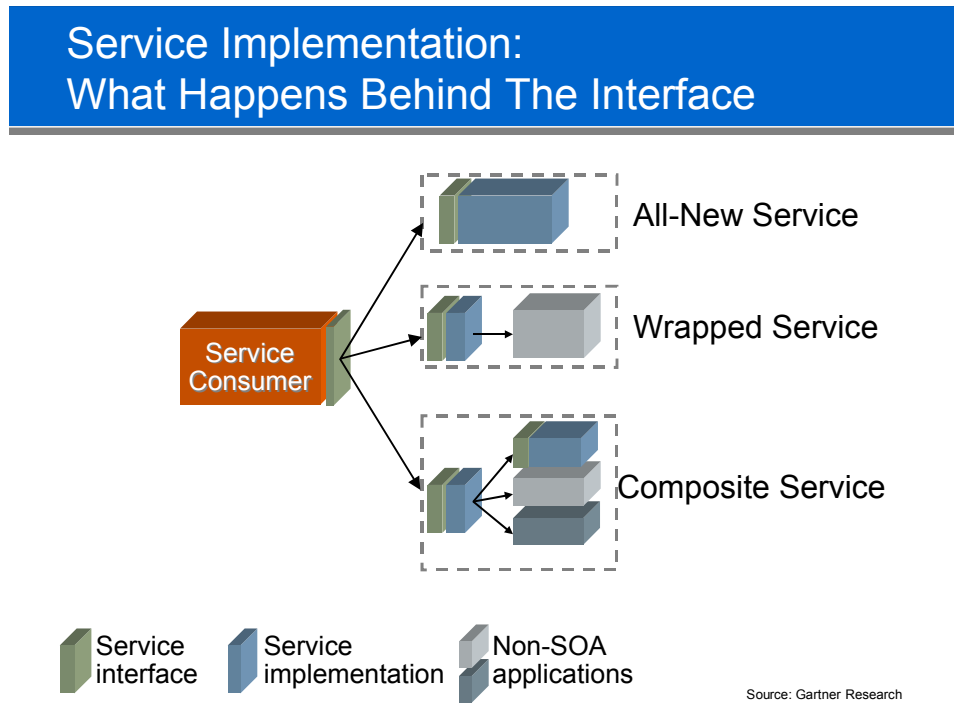
SOA is a loosely coupled (but not decoupled) architecture. Loose coupling of SOA manifests itself in flexible association of services and service consumers. A new service consumer can access a pre-existing service entirely un-intrusively (a poorly designed service may lock a service into a particular service consumer, denying the key benefit of SOA).

## Service Implementation

A logical definition of a service simply indicates the business function that a service performs. In reality, the service implementation may translate to a relatively complex process and depend on many sources of information to fulfill the functional requirement designed for the service. What makes it more complicated is that the technical design of the service implementation cannot make any assumptions about the context in which the service would be invoked. The service may be used stand-alone, as part of a sequence of calls on behalf of a real-time transaction or as a subordinate component in

an asynchronous EDA. Due to this inherent complexity, most service implementations likely will be relatively simple, self-contained and stateless. (All three principles — simplicity, isolation and statelessness — are the best practices of design for all distributed systems.)

**Figure 40. SOA — What Happens Behind the Interface**



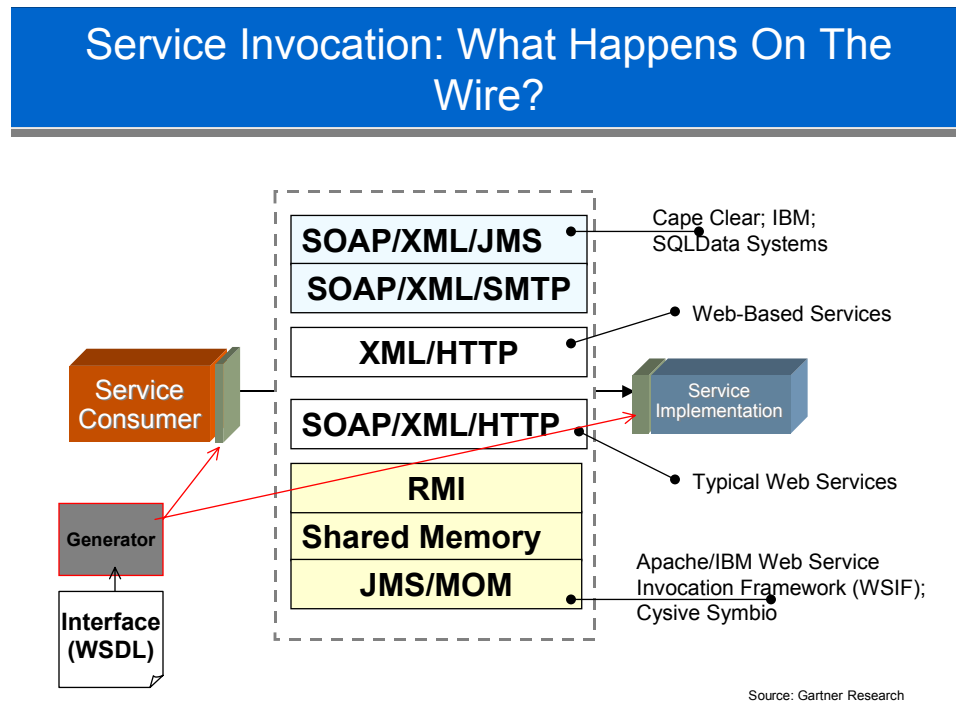
Some service implementations will be developed as new components. This is the simplest case and also the least frequent through 2006, given that the composite application style will likely be the leading driver for adoption of SOA. Some early Web services implementations have been automatically generated (wrapped, pre-existing JavaBeans, CORBA objects, CICS DPL transaction programs, C++ classes). These wrapped services are the easiest to implement but are often the least effective, because the design objective of an object class or a component is different from that of a service — the resulting service may be too fine-grained, may spur massive network traffic and may flood the services repository).

## Service Invocation

Design of SOA starts with the design of interfaces. At run-time, interfaces are represented by a pair of interface programs — the client proxy, used by the caller and the server stub, front-ending the service implementation. The two principals of a service call, the service consumer program and the service implementation program, never talk directly, but rather via the pair of the proxy and the stub. These two programs implement the communication between the service consumer and the service: marshal and encode the message, use a communication method and a transport protocol. Although ideally the proxy and the stub may be created independently, in reality they

are typically generated by the same generator, using the interface definition (WSDL) as the source. What happens on the wire between the client and the service is entirely confined to the pair of the proxy-stub programs and is thus typically determined by a single stub/proxy generator. This offers an opportunity to optimize service communications by deviating from the common lowest denominator standards. As the intra-enterprise use of Web services increases, default use of XML, SOAP and HTTP becomes problematic.

Figure 41. SOA — What Happens On the Wire



## SOA and Web Services

Web services are often seen as architecture, yet at their minimal level, Web services are a set of standards not amounting to an architecture. As the number of proposed standards increases and expands the scope of application behavior issues that are covered under the umbrella of Web services, and as there emerge organizations that are responsible for managing and certifying Web services standards, Web services may yet consolidate into a new application architecture.

The use of Web services standards does not guarantee that the resulting application is service-oriented; a service-oriented application can be implemented without any use of Web services standards. Still, the industry momentum of Web services and the affinity of WSDL to SOA link the two initiatives. Most new SOA application projects intend to use Web services, and most Web services projects intend to use SOA. Unfortunately for many projects there is also an erroneous assumption that the two are one and the same and that use of Web services guarantees the benefits of SOA. In reality, good SOA

application requires a systematic design effort and, for more demanding enterprise projects, the quality of service guarantees. Many new standards are proposed and will be delivered in products through 2006 to fulfill this requirement. Until then Web services will remain the format of SOA, mostly applicable to smaller and simpler applications.

In an SOA scenario, Web services wrap around pre-defined normalized business services running on an application server or TPM. Web services interface facades can be added to complex, albeit fast-executing, business processes, engaging both SOA and non-SOA applications. In this scenario, Web services invocations trigger the execution of multistep business processes spanning multiple systems, within and outside the enterprise, conducted through the cooperation of multiple middleware layers providing the necessary integration infrastructure.

## When To Use SOA

SOA is architecture of services. It is based on the premise that software components operate as components (building blocks) of a larger immediately executing transaction (unit of work). Services are designed to perform reusable partial processes on behalf of a bigger transaction. Some styles of applications operate exactly in this manner. New interactive multi-channel applications (multiple user devices or types of users) benefit greatly from SOA design. All channels are able to access the same consistent and available set of back-end functionality. Building composite applications — applications that draw data from both new and old resources — is also well fit for SOA. Older applications can be wrapped and modernized to expose their functionality through programmatic interfaces — then accessed from new calling applications. Multi-channel and composite applications — are the best candidates for SOA.

SOA is not intended for building autonomous business components, those that operate independently, not on behalf of a larger immediate unit of work. These off-line processes (disconnected from the originator, yet possibly real or near-real time) are required for back-end post-transaction processing, for processing that is triggered by time or by changes of state. Services are not the right architecture for this software, although many of these applications are service consumers and thus are still participants in SOA. The architecture of events is often the best fit for off-line application styles.

**Table 16. When To Use SOA**

<b>Best Practice SOA:</b>	<ul style="list-style-type: none"> <li>■ Composite applications</li> <li>■ New request-reply applications</li> <li>■ Multi-channel applications</li> <li>■ Information retrieval</li> </ul>
<b>Consider Alternatives:</b>	<ul style="list-style-type: none"> <li>■ Process-monitoring applications</li> <li>■ B2B applications</li> <li>■ Post-transaction, batch processing</li> <li>■ Robotic (human-less) applications</li> <li>■ Less predictable, state-dependent processes</li> </ul>

## **Benefits of SOA: Reality vs. Hype**

### ***Benefits***

- Incremental development and deployment of business software
- Reuse of business components in multiple user experiences (channels)
- Low-cost assembly of new business processes
- Clarity of application topology.

### ***Hype***

- Simple software engineering
- Free interoperability
- Free integration
- Technology-independence
- Vendor-independence
- Ultimate architecture for the enterprise
- SOA = SOAP.

### Selecting the Proper SOA Integration Platform

	Project Style	Scalability/ Performance	Consumer Application Number	Service Provider Plat. Number
Web Services	<i>Opportunistic</i>	<i>Medium</i>	<i>Low to High</i>	<i>Low to High</i>
Adapters	<i>Opp./Syst.</i>	<i>Medium/High</i>	<i>Low</i>	<i>Low</i>
RPC/MOM/ORB	<i>Systematic</i>	<i>High</i>	<i>High</i>	<i>Low</i>
Enterprise Service Bus	<i>Opportunistic</i>	<i>Medium</i>	<i>High</i>	<i>High</i>
Programmatic Int. Servers	<i>Opportunistic</i>	<i>High</i>	<i>High</i>	<i>High</i>
Integration Broker Suites	<i>Systematic</i>	<i>Medium</i>	<i>High</i>	<i>High</i>

Service-oriented composite applications and Web services conceptually fit perfectly with one another. However, the perfect pair today works only in limited cases due to Web services' current limitations. To link service provider and consumer platforms, adapter products take advantage of proprietary protocols, screen-scraping techniques and standard protocols. Most adapter products are well-known and proven, even in very large, high-performance, business-critical projects. Therefore, they can also be used for composite applications, even in demanding, high-performance and scalability scenarios. In a many-to-many integration scenario, a communication middleware-based decoupling layer is needed to reduce the number of point-to-point links, provide a consistent set of APIs and simplify operations and management. Integration broker suites (IBSs), Enterprise service buses (ESBs) and programmatic integration servers are powerful integration middleware platforms that will extensively be adopted in the most complex service-oriented composite application scenarios, although IBSs and ESBs are not specifically optimized for high-performance service-oriented interoperability, and programmatic integration servers have a limited track record in large-scale projects.

## **Client Contact Information**

John F. Jones, Jr.  
Chief IT Architect  
Telephone: +1-301-402-6759  
E-mail: [jonesjf@mail.nih.gov](mailto:jonesjf@mail.nih.gov)

## **Gartner (Contractor Support) Contact Information**

Terry McKittrick  
Gartner Consulting  
Telephone: +1-703-226-4779  
Facsimile: +1-703-226-4702  
E-mail: [Terry.McKittrick@gartner.com](mailto:Terry.McKittrick@gartner.com)

# **Gartner®**

