Middleware Architecture Report:

"A Middleware Framework for Delivering Business Solutions"

Version 1.0, May 2001

Prepared for:
The Council on Technology Services
Commonwealth of Virginia

By:
The COTS Enterprise Architecture Workgroup,
Middleware Domain Team

Middleware Domain Team Members

Ted McCormack, CoChair	Commission on Local Government
Troy DeLung, CoChair	Department of Environmental Quality
Matt Blaes	VA Geographic Information Network
Will Burke	Department of Motor Vehicles
Bob Green	Department of Information Technology
Jim Jokl	University of VA
Dick Jones	VA Department of Transportation
Paul Hendricks	Department of Motor Vehicles
Paul Bucher	VA Department of Transportation, Consultant (Domain Staff)
Diane Wresinski	Department of Technology Planning (Domain Staff)
Paul Lubic	Department of Technology Planning (Enterprise Architecture Manager)
Brian Mason	Department of Technology Planning (Enterprise Architecture Consultant)

COTS Enterprise Architecture Workgroup

David Molchany, Co-Chair	Fairfax County, Local Government Representation
Murali Rao, Co-Chair	Department of Transportation, Secretariat of Transportation Representation
Tim Bass	Virginia Retirement System, Independent Agency Representative
Bethann Canada	Department of Education, Secretariat of Education Representative
Troy DeLung,	Department of Environmental Quality, Secretariat of Natural Resources Representative
Linda Foster	Department of Taxation, Secretariat of Finance Representative
Bob Haugh	Department of Corrections, Secretariat of Public Safety & Large Agency Representative
Randy Horton	Department of Rehabilitative Services, Secretariat of Health and Human Services Representative
James Jokl	University of Virginia, Higher Education Representative (UVA)
Ted McCormack	Commission on Local Government, Secretariat of Administration & Small Agency Representative
Bill Mize	Department of Information Technology, Secretariat of Technology Representative
Bob Pontius	Employment Commission, Secretariat of Commerce and Trade Representative

Table of Contents

I. Executive Summary of Middleware Architecture 5
II. Mission 7
III. Introduction and Background 8
IV. Methodology 9
A. Middleware Definition 9
B. Benefits 9
C. Types of Middleware 10
V. Principles 12
VI. Middleware Functions 13
A. Database Middleware 13
A.1 Directory Services 13
A.2 Database Metadata Services 14
A.3 Database Access Services 15
A.4 Database Middleware Guidelines 16
B. Message Middleware 17
B.1 Message Formats 17
B.2 Message Transfers 19
B.3 Message Oriented Middleware 20
B.4 Messaging Middleware Standards 21
B.5 Message Middleware Guidelines 22
C. Transaction Processing Monitor Middleware and Services 24
C.1 Two-Phase Commits 24
C.2 Failure/recovery 24
C.3 Synchronization 25
C.4 Scheduling 25
C.5 Repeat attempts 25
C.6 Message queue management 25
C.7 Business-rule-based transaction workflow services 25
C.8 Load balancing services 25
C.9 Transaction Processing Middleware Guidelines 26
D. Application Integration Middleware Servers and Services 26
D.1 Methods to Integrate Applications 26
D.2 Application Integration Management Services 28
D.3 Application Integration Middleware Examples 29
D.4 Application Integration Middleware Guidelines 31
E. Super-Service Middleware Servers and Services 32
E.1 Value Added Services 32
E.2 Super-Service Middleware Servers and Services Guidelines 33
VII. E-Government Examples 34
VIII. Policies, Standards and Best Practices 35
A. Policies 35
B. Recommended Policies 35
C. Standards 36
D. Recommended Standards 36

E. Best Practices 36
Glossary 38
Appendices 46
Appendix A. History of the Evolution of Separate Middleware Services 46
Appendix B. What Communication Services Are Middleware Domain Service: The OSI and TCP/IP Models 48
Appendix C. Domain Team Working Documents 51
Appendix D. Reference and Links 57
Appendix E: Quick Reference 59

Tables and Figures

Figure 1: Example Middleware Tools and Services 6
Figure 2: Development of the EWTA 8
Table 1: Enterprise-Wide Technical Architecture (EWTA) Domains 8
Table 2: Database Connectivity Technology Ratings 16
Table 3: MIME Examples 22
Table 4: Message-oriented Middleware Technology Ratings 23
Table 5: Transaction Processing Monitor Middleware 26
Table 6: Middleware Application Integration Services 31
Table 7: Example Applications and Potential Middleware Use 34

I. Executive Summary of Middleware Architecture

State agencies today are faced with the challenge of integrating the disparate systems and islands of automation into one enterprise-wide flow of business logic. Often, information needed by knowledge workers is spread throughout various business applications in different departments within agencies or is spread across agencies. Knowledge workers would like to access all the information they need in a transparent and seamless fashion. To accomplish this, programmers must know how to connect information to applications and customers no matter where it resides on the network. A middleware architecture enables agencies to address these connection needs in a consistent and useful manner. Middleware has been described as the software glue that ties applications together across a network. Middleware can allow organizations to share data between systems that do not communicate easily. Middleware is the enabler of application communications in a distributed system and is the tool that improves the overall usability of an environment made up of products from many different vendors on multiple platforms.

In Virginia, the Enterprise Architecture team has divided the architecture into eight domains. To cover every aspect of the information technology architecture in one domain or another the domain teams must break down architecture into components. The middleware domain team has identified a number of components that they wish to cover as part of the middleware domain. In considering these components, they have discussed whether the components belong in the middleware domain or in neighboring domains including the network, application, security, database, and systems management domains.

The middleware domain team has identified five middleware component types: Database Middleware, Message-oriented Middleware, Distributed Transaction Processing Monitors, Application Integration Middleware, and a Super-service Middleware. Super-service middleware is a collection of two or more middleware components. The various types of middleware overlap in some services provided. For example, all play a roll in sending messages (i.e., between applications across the network) and in accessing data.

The Middleware concept is difficult to understand from an enterprise viewpoint without having some understanding of how the introduction of the client-server environment and distributed environments affected the complexity of computer programming. Middleware vendors are trying to address some of these complexities by centralizing certain functions that may have been embedded in the tools of the network architect, the application architect, and the database architect. Appendices A and B provide related history and communication service information in non-technical terms.

Figure 1 provides an overview of many of the services that might be provided by one or more middleware products. The service management tools on the left are examples of functions partitioned from applications and databases for provision centrally in the computing environment (between rather than within the application and databases). The communication services on the right are partitioned from the total “protocol stack” that is required for communication between senders and receivers on a network. So, a big picture look at middleware would emphasize the bringing to the middle certain services

so that they can be created one time, managed centrally, and used many times by the distributed network applications.

Figure 1: Example Middleware Tools and Services

Service Management Tools

1. Environment description tools: example tools might include
   - business rules/workflow definition tools
   - distributed environment definition tools
   - object reuse location repository
   - message type definitions
   - protocol translation information for the environment
   - event registry
   - distributed location information
2. Diagnostic and analysis tools for monitoring transactions:
   - monitoring metrics
   - load balancing services
   - metric reporting/viewing services
3. Scripting tools for configuring each middleware component.
4. Pre-configured message types for metrics and analysis (e.g., directory entries to aid in counting important parts of financial transactions)

Communications Services

- Directory Lookups
- Security Services (e.g., encryption)
- Translation Services (e.g., decryption)
- Database to Database Interconnections
- File Transfers
- Asynchronous Messaging (one-way)
- Synchronous Messaging (two-way)
- Application Interfaces
- Message Publish and Subscribe Services (e.g., like news services)
- Message Store and Forward Services (e.g., like email)

All agency business applications distributed over a two- or three-tiered environment or involving network communications between clients and servers require some of the functionality that may be provided in a bundled middleware product. In agencies without a bundled middleware product, these services are provided through middleware tools acquired along with the operating systems, in database products, as part of shelf-ware, as separate tools and/or through functionality coded into local applications. The acquisition of a super-service middleware product would enable addressing many middleware service needs.

Much of this document is technical in nature and explains in more detail the concepts of middleware. In addition to providing this information, the middleware team addresses the desired Commonwealth technology directions for middleware architecture by applying one of four rating categories to the various technologies that are used to provide middleware services and functions. The rating categories are Obsolescent, Transitional, Strategic, and Emerging. This information will help those evaluating middleware products. The rating terms are defined below.

- **Obsolescent** - The Virginia Enterprise Architecture actively promotes that agencies employ a different technology. Agencies should not plan new deployments of this technology. Agencies should develop a plan to replace this technology. This technology may be waning in use or no longer supported.
- **Transitional** - The Virginia Enterprise Architecture promotes other standard technologies. Agencies may be using this technology as a transitional strategy in movement to a strategic technology. This technology may be waning in use or no longer supported.
- **Strategic** - The Virginia Enterprise Architecture promotes use of this technology by agencies. New deployments of this technology are recommended.

- **Emerging** - The Virginia Enterprise Architecture promotes only evaluative deployments of this technology. This technology may be in development or may require evaluation in government and university settings. Use of this technology may be high risk.

The guidance provided in this document is for agencies and for those who serve agencies in matters related to technology. The word "**agency**" as used in this document means all state, local, and education units of government. In general, the document will provide guidance and information to agencies in the following ways:

1. **Recommendations regarding proposed Information Technology Resource Management (ITRM) Policies, Standards, Guidelines (PSGs) for agencies.**

   The proposed policies, standards and guidelines (i.e., best practices) offered in this document will be reviewed by appropriate stakeholders and then converted into official ITRM PSGs. The policies, standards and best practices begin on page 35. Unless otherwise indicated, policies and standards (existing and recommended) are requirements for state agencies and for local agencies receiving state funding for referenced efforts and related acquisitions.

2. **Discussions of middleware types, services, and technologies.**

   Those interested in a particular middleware function are encouraged to read specific sections as follows: database connectivity on page 13; messaging on page 17; high-volume transaction processing monitors, page 24; application integration servers, page 27; super-service middleware, page 33.

3. **Example e-government applications.**

   Example uses of middleware are described beginning on page 35.

4. **Tables indicate recommended strategic technologies.**

   Strategic technologies may meet an agency's needs. These tables provide a starting point for product component evaluation, but each agency is advised to pilot actual products. The obsolescent, transitional, strategic, and emerging technology ratings are provided in tables. See Tables 2 for database connectivity, 4 (messaging), 5 (transaction processing), and 6 (application integration).

5. **A Glossary of middleware terms** (see page 39).

6. **Web links for more information are provided in the Glossary and in Appendix D.**

7. **A Quick Reference is provided in Appendix E.**

II. Mission

To explain the purpose of middleware, to define the tools and services that may be provided in a middleware product, and to provide additional decision-relevant information to help agencies and other responsible parties make informed decisions regarding their middleware architecture.

III. Introduction and Background

Virginia's Enterprise Architecture (EA) includes business, governance and technical components that describe how Virginia will use technology and proven practices to improve the way it does business. The technical components are referred to as the Enterprise-Wide Technical Architecture (EWTA) and are comprised of eight domains. Together, these domains constitute a comprehensive framework for providing technical guidance and related best practices to Virginia's agencies.

The EWTA is being developed in stages and will be updated routinely. The Council on Technology Services (COTS) and its work groups are responsible for the development and updating efforts. Those involved began their efforts by specifying business strategies and information requirements, which were used to determine expectations for Virginia's future enterprise architecture. The following diagram summarizes the development process and identifies the responsible groups (see Figure 2 below and Appendix C., Domain Team Working Documents).

Figure 2: Development of the EWTA

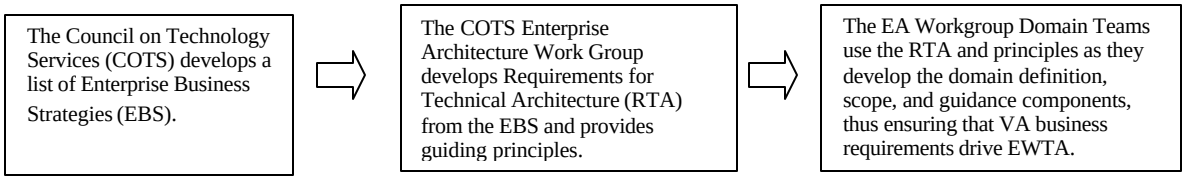

Table 1: Enterprise-Wide Technical Architecture (EWTA) Domains

Base	Functional Glue	Application
Network Architecture Platform Architecture	**Middleware Architecture**	Systems Management Architecture Database Architecture Application Architecture Information Architecture
Security Architecture		

The eight technical architecture domains are listed in Table 1. Each of the eight domains is clearly a critical piece of the overall architecture. The Network and Platform Domains address the infrastructure base. These two areas provide the foundation for any distributed computing architecture. Systems Management, Database, Application, and Information Domains provide vehicles for discussing the business functionality and management of the technical architecture. The Security Domain addresses the many vehicles for enhancing information security across the architecture. The Middleware Domain addresses the interfacing of disparate platforms, systems, databases and applications in a distributed environment. These eight domains provide a useful way of communicating guidelines, policies, standards and best practices of the EWTA to

stakeholders in state and local government agencies and state universities. Only the middleware domain is addressed in this document.

IV. Methodology

The middleware domain team began its work with an orientation to the middleware architecture developed by the State of North Carolina. The team also reviewed the middleware architectures for the states of Ohio and Connecticut, position papers by industry strategists including the META Group and GartnerGroup, and did extensive Web research. Domain team members discussed this information and reviewed existing middleware architectures used by Virginia's agencies to identify what types of Middleware products Virginia will need to enable a citizen centric e-government.

The intended audience for the *Middleware Domain Architecture* is both business and technical leaders in state and local agencies including universities, colleges, and agencies from all branches of government. This document will identify the trends, best practices and emerging standards to help agencies make decisions regarding their middleware architecture. For middleware acquisition decisions, this paper will provide a discussion of when an individual agency might need middleware, how an agency might decide what services should be provided through their middleware tools, and how connectivity rules are evolving for middleware functions.

A. Middleware Definition

Middleware Architecture defines the functions that enable communications in a distributed system and the tools that improve the overall usability of an architecture made up of products from many different vendors on multiple platforms. Middleware is software that allows organizations to share data between disparate systems that do not communicate easily. Middleware has been described as the software "glue" that ties different applications together.

B. Benefits

Middleware products may be used to benefit the Commonwealth in the following three ways.

- Middleware services and tools are key to creating citizen centric service portals[1] that enable information and services to be obtained at one place from multiple State agencies.

[1] According to civic.com in the April 2001 feature on government web portals, "Most states expect the task of having all services available through the portal to take at least five years, if not longer. 'I liken it to building a stained glass window', said Arun Baheti, director of e-government for California. 'We have that overall image – giving citizens one view into government – but we still have to put the smaller pieces of glass the individual projects into the larger mosaic'".

- Middleware services and tools are key to extending the utility of the State's technology infrastructure and skilled workers while developing new services that rely on communications between existing services.

- Middleware services and tools are key to interfacing beyond state agencies to localities, federal agencies and the business sector.

C. Types of Middleware

1. **Database Middleware** enables applications to communicate with one or more local or remote databases. It does not transfer calls or objects. For example, database middleware does not allow for two-way communication between servers and clients. Servers cannot initiate contact with clients, they can only respond when asked.

2. **Message-Oriented Middleware** provides an interface between applications or application parts, allowing for the transmission of data back and forth intermittently. Messaging middleware is similar to an e-mail system that transfers messages between people, except that it sends information between applications. If the target computer is not available, the middleware stores the data in a message queue until the machine becomes available.

3. **Transaction-Process Monitor Middleware** is software that sits between a requesting client program and databases, ensuring that all databases are updated properly. It is a control program that manages the transfer of data between multiple terminals or clients and the application programs that services them.

4. **Application Integration Middleware** provides interfaces to a wide variety of applications. It can provide ways for legacy systems to interface with network clients (e.g., use a thin-client browser to run a legacy system) or ways to execute functions across a network from one application to another.

5. **Super-service Middleware** is the collection, management and integration of multiple (heterogeneous) types of middleware with value added services. These super services are often Web-enabling middleware that allow for the easy integration of back-end applications with new e-commerce and e-government systems. Super-service middleware enables rapid response to changing business requirements.

Note: The above types of middleware are not mutually exclusive. Also, the types do not cover every type of middleware. For example, mobile users of networks require a specialized middleware that stores messages both on the client and on the server so that information is not lost when a connection is broken.[2] For the middleware types listed here, however, examples of overlapping functionality may useful. Database middleware and transaction processing middleware, for example, both enable database connections,

[2] Mobile user middleware will be covered in a separate paper. The use of radio-frequency connections to local area networks is a still evolving area. Many vendors provide proprietary tools that are specialized for the type of network access needed and the tasks that the mobile user needs to perform.

but transaction processing middleware extends and add value to the database communication process. All types of middleware process messages sent between client applications and server applications. All middleware provides ways for messages to get from the application to the network. Also, all middleware provides a way of letting the application know a message was received. The classifications indicate specializations and added services. These differences are presented in considerable detail in this document.

Much of this document is technical in nature and explains in more detail the concepts of middleware. In addition to providing this information, the middleware team addresses the desired Commonwealth technology directions for middleware architecture by applying one of four rating categories to the various technologies that are used to provide middleware services and functions. The rating categories are Obsolescent, Transitional, Strategic, and Emerging. This information will help those evaluating middleware products. The rating terms are defined below.

- **Obsolescent** - The Virginia Enterprise Architecture actively promotes that agencies employ a different technology. Agencies should not plan new deployments of this technology. Agencies should develop a plan to replace this technology. This technology may be waning in use or no longer supported.
- **Transitional** - The Virginia Enterprise Architecture promotes other standard technologies. Agencies may be using this technology as a transitional strategy in movement to a strategic technology. This technology may be waning in use or no longer supported.
- **Strategic** - The Virginia Enterprise Architecture promotes use of this technology by agencies. New deployments of this technology are recommended.
- **Emerging** - The Virginia Enterprise Architecture promotes only evaluative deployments of this technology. This technology may be in development or may require evaluation in government and university settings. Use of this technology may be high risk.

The guidance provided in this document is for agencies and for those who serve agencies in matters related to technology. In general, the document will provide guidance and information to agencies in the following ways:

1. **Recommendations regarding proposed Information Technology Resource Management (ITRM) Policies, Standards, Guidelines (PSGs) for agencies.**

   The proposed policies, standards and guidelines (i.e., best practices) offered in this document will be reviewed by appropriate stakeholders and then converted into official ITRM PSGs. The policies, standards and best practices begin on page 35. Unless otherwise indicated, policies and standards (existing and recommended) are requirements for state agencies and for local agencies receiving state funding for referenced efforts and related acquisitions.

2. **Discussions of middleware types, services, and technologies.**

   Those interested in a particular middleware function are encouraged to read specific sections as follows: database connectivity on page 13; messaging on page 17; high-volume transaction processing monitors, page 24; application integration servers, page 27; super-service middleware, page 33.

3. **Example e-government applications.**

   Example uses of middleware are described beginning on page 35.

4. **Tables indicate recommended strategic technologies.**

   Strategic technologies may meet an agency's needs. These tables provide a starting point for product component evaluation, but each agency is advised to pilot actual products. The obsolescent, transitional, strategic, and emerging technology ratings are provided in tables. See Tables 2 for database connectivity, 4 (messaging), 5 (transaction processing), and 6 (application integration).

5. **A Glossary of middleware terms** (see page 39).

6. **Web links for more information are provided in the Glossary and in Appendix D.**

7. **A Quick Reference is provided in Appendix E.**

The term "agency" means Commonwealth of Virginia executive branch agencies and institutions of higher education. For the purpose of this document, however, any academic "instruction or research" systems/infrastructure that can be isolated from "administrative and business" systems/infrastructure are considered exempt from the stated architecture standards.

Concerning local governments and other public bodies, while they are not required to comply with the standards, the information technology specifications for participation in state programs would be based on the architecture described herein. This architecture was designed with participation of local government and other public body representatives with the intent of encouraging its use in state/local interoperability activities.

V. Principles

The middleware team identified three domain-specific principles. These are presented below.

Principle 1: The Commonwealth should provide seamless access to data and services.

- There is increasing emphasis on the implementation of a single Commonwealth of Virginia portal for citizens to use to obtain data and services from the State.
- There currently are portals for individual State agencies, which provide a separate means to access data from those agencies.

Principle 2: Agencies should strive for inter-operability.

- There is an increasing need for systems to inter-operate within and across agencies.
- Middleware can help in providing the inter-operability needed within the enterprise.

Principle 3: Middleware should provide flexibility, portability, and cost effectiveness in the implementation of enterprise architecture.

- State agencies have limited resources with which to implement enterprise architecture.

- State agencies must be able to react quickly to change.
- State agencies must continue to use legacy systems.

VI. Middleware Functions

In this section, the major protocols in use, usually known by acronyms, will be described and rated (see Tables 2, 4, 5 and 6.). The ratings that are applied are "obsolescent," "transitional," "strategic," or "emerging" (see definitions). Agencies are encouraged to acquire middleware that employs technologies in the “strategic” category.

A. Database Middleware

Database Middleware enables applications to communicate with one or more local or remote databases. It does not transfer calls or objects. For example, database middleware does not allow for two-way communication between servers and clients. Servers cannot initiate contact with clients, they can only respond when asked. The discussion of database middleware is broken into directory, metadata, access services, and related guidance. Guidance information may direct the reader to other domains when those documents become available.

A.1 Directory Services

A directory may be described as a specialized database of lists. Directories serve a wide variety of functions in a computing environment and are used by applications including email, security, and naming services. Directory services are important as tools in the communications process and decisions about directory services are one of the most important foundation decisions an agency can make in planning a distributed architecture and middleware strategy. Deciding on a desired external directory strategy (e.g., external to the database system or network management system) before looking at middleware products will allow an agency to be more critical of how middleware components are integrated, especially in bundled, multi-vendor products. Having a directory strategy is an integral part of promoting interoperability, location transparency, and lower future maintenance costs in a distributed environment. Some directory services can be configured with strong security by attribute so that everyone could see a user email address for example but only the user could update a password or see other personal information. Some example uses of a directory to support government functions are provided below:

- Certificate authority information and public keys for digital signatures
- Single sign-on password information for employees and other authorized individuals
- A statewide citizen-changeable address store that could be accessed by subscribing agencies
- Encrypted agency PIN numbers for citizen access to services

- Object naming for reuse by programmers
- Employee address, office phone or email information for updating by employees

Lightweight Directory Access Protocol or LDAP is based on the X.500 open standard. LDAP specifies the access method and protocol, not the storage structure. LDAP enables extensible access to directories. Using LDAP, directory organization can be configured and extended to add additional categories and attributes. Active Directory Server from Microsoft and Netscape Directory Server are two LDAP compliant directory servers for the NT server networks but LDAP compliant access and storage methods are becoming available on most platforms. Initial implementation of the Microsoft Active Directory Server with Windows 2000 was slowed due in part to changes in the way copies of the directory are replicated and the need for careful planning in organizing the directory structure.

Two additional related directory standards that have been very important to the growth of the Internet are: Domain Naming Service (DNS)--A distributed directory service that may be used on the Internet along with Global Directory Service (GDS) to provide a worldwide hierarchy. This is what enables Internet users to access a Web site by typing a friendly name in the format "www.site-name.com" instead of requiring users to remember complicated series of physical Internet Protocol (IP) addresses with port numbers in the format "127.127.127.127:9999".

(Note: DNS is criticized for its lack of extensibility and its inflexibility in the area of searching. LDAP has both search and extensibility features).

The Open Group's Distributed Computing Environment (DCE) maintains the LDAP standard[3]. For a guide to additional information on LDAP and related standards work, see http://www.opengroup.org/directory/, the Directory Interoperability Forum.

A.2 Database Metadata Services

Metadata is data about data. *A Mars spacecraft crash was attributed to using the wrong interpretation of the type of measurement data in a calculation (metric vs. standard).*

Metadata is all the descriptive information that lets us make sense of the data. It tells us things about the data such as: how to interpret it; the business and technical definitions and descriptions; how it may be used; what constraints there are in its use; where it originated; who creates it; who is responsible for it; what business processes it supports; where it is used; its frequency of use; and any other information deemed valuable by the organization. English language or business names for the data, synonyms (other database table or attribute names that refer to the same data), table relationships and the physical database locations should also be included in a comprehensive metadata repository.

[3] DCE provides a complete Distributed Computing Environment infrastructure. It provides security services to protect and control access to data, name services that make it easy to find distributed resources, and a highly scalable model for organizing widely scattered users, services, and data. DCE runs on all major computing platforms and is designed to support distributed applications in heterogeneous hardware and software environments. DCE is a key technology in three of today's most important areas of computing: security, the World Wide Web, and distributed objects. (From the Open Group DCE Web site.)

Policies should be developed for uniform abbreviations, uniform classifications and access types.

The Object Management Group (OMG) and the Metadata Coalition (MDC) are developing a joint metadata model. In the past, metadata tools followed different formats. A subset of these open metadata repository formats and access tools will enable designers of systems to find and utilize existing data and services. Attaining reuse of data and services has been an elusive goal of the architecture for years. In the future, both application designers and applications will be able to use middleware tools to communicate with metadata repositories and find existing data, services, functions, message format descriptions, etc.

Extensible Markup Language (XML) is a popular method for formatting data message exchange over the Internet. However, so each agency and locality does not develop their own set of formats to describe essentially the same data, sharing and reuse of these definitions will be important. This kind of new information can be placed into an accessible meta-metadata repository to allow developers to share and reuse solutions.

A.3 Database Access Services

1) ODBC

The term Open Data Base Connectivity (ODBC) is often used to refer in a general way to a group of middleware database connectivity drivers and services. The drivers are written to open specifications for accessing data called Structured Query Language (SQL). This includes ODBC, JDBC (for Java), and OLE-DB. Most relational databases support this method of access natively. This method is commonly used for reporting programs to access application tables and for doing lookups in other databases or obtaining data to extract or import into another database. However, since the access is direct and not through the application business rules, ODBC data access should not be used to modify data in different applications by anyone other than the owner of the data. This access method bypasses any security roles and policy maintained through the application interface instead of through database security.

2) Database Gateways / Adapters

Database gateways enable data access and sharing between heterogeneous databases. In order to access non-relational, or legacy databases that do not natively support SQL access, translation database access software needs to be installed on a device with access to the source database. In the past, these gateways enabled the requesting system to utilize the proprietary commands of the host system to access the data, instead of SQL commands. This approach may be advantageous for an application that is being ported to a new platform where the need is to maintain compatibility with the existing application. It is also advantageous when there is a performance penalty for using the open SQL command instead of the proprietary native command. However, for other generic systems, the preferred method is to use the SQL open standard access method. Adapters are essentially pre-built interfaces for connecting one application to another common business application. Adapters in addition to providing access to the data, may also be application program interfaces, object request protocols, etc. Adapters provide a

way to utilize the security and business rules embedded in the application logic. The programmer should be able to extend or modify the adapter if the target application is modified. Even if customization is needed, an adapter can provide a starting point and framework for the programmer.

A.4 Database Middleware Guidelines

Table 2 provides technique and protocol ratings for directories, metadata, and database connectivity. In general, those technologies listed as strategic are based on open standards. Agencies are encouraged to incorporate into their architecture one or more of the technologies and strategies listed as strategic. Commentary regarding the aforementioned technologies and related tools are provided below.

Table 2: Database Connectivity Technology Ratings

Obsolescent	Transitional	Strategic	Emerging
Directories			
	X.500 DAP	LDAP, DNS & GDS Sun JDAP Novell NDS	MS Active Directory (ADSI)
Metadata			
Business rules and meaning hard coded into applications. Hard copy only documentation of metadata.	Configurable metadata separate from application but proprietary to system.	OMG's UML, MOF MDC's XMI (XML, DTD, Schema) OIM's exchange format XIF (XML) Accessible, computer aided metadata documentation (e.g., ERwin modeling tool) and a metadata repository	Active metadata repository
Data Access Methods, Adapters, Drivers, and Contracts			
	OLE (replaced). Screen Scrapers as a mainframe access. Non-ODBC/SQL compliant Gateways Translators for non-standard SQL, XML, etc.	DB Adapters or Drivers: ODBC, JDBC, xDBC, OLE-DB (platform specific) XML point to point contracts (e.g., for Schemas) ODBC/SQL compliant gateways	XML messaging

ODBC drivers may not support all versions or extended features of the host databases. Agencies should use certification labs or groups to test new database ODBC drivers and

new databases especially if any non-standard features are being utilized. Middleware database connectors should support common standards such as ODBC standards and should be able to do intelligent SQL query optimization and rationalization appropriate to each target data source.

Query governance should be part of database management tools so a query submitted by a user cannot bring critical systems to a standstill. The software should permit the setting of time limits or record count limits for users based on their role and need.

Performance monitoring has multiple benefits. It will help identify what sources of data are getting accessed, and how long it is taking users to run their queries on those data sources. Queries that exceed a threshold time limit can be logged for further analysis. This information is important in helping database administrators to reorganize data and create indexes to speed data access. Also, the information may be used to identify priority data for migrating to data warehouses thus decreasing transactions in busy data stores.

B. Message Middleware

Message-Oriented Middleware provides an interface between applications or application parts, allowing for the transmission of data back and forth intermittently. Messaging middleware is similar to an e-mail system that transfers messages between people, except that it sends information between applications. If the target computer is not available, the middleware stores the data in a message queue until the machine becomes available.

B.1 Message Formats

In this section, the term "messages" will be used in the broadest sense to encompass transaction-based messages as well as entire file transfers. To many messaging systems, the format of the content of the message doesn't matter as long as it has the understood envelope/wrapper or an operating system recognizable format. However, the format of the content is very important to the receiving operating system, application, or user. Format translations may be performed by middleware. Also included in this section are messages that are object-oriented. These messages are requests or replies that are issued or received by applications or databases.

When data or even entire databases are transferred between like systems, the entire tables can be copied in their native format. If the systems are dissimilar, data must be converted to a common format understood in both systems. In the past, this format was stated in a standard such as EDI and the encoding was often ASCII, or human readable text that was either fixed-width or delimited with a special character that could be understood by both systems. Sometimes both systems support a common method of formatting or delimiting the export/import file but in other cases, an intermediary program or middleware application is needed to do some transformation.

The ASCII encoding has been used for both file and transaction-based message systems. ASCII is compact, efficient, and compressible. ASCII continues to be used today in newer data access messaging methods such as the Extensible Markup Language (XML). With XML, the standards provide message format and document type definitions (DTD)

much like Electronic Data Interchange (EDI) standards provide file formatting for sharing common documents such as Purchase Orders between different systems. Existing EDI methods are still in wide use for financial transactions when modifications or extensions are not needed. To achieve the same goal of standardization today for a wider variety of applications, the trend is to use the XML tagging standards along with contractual arrangements between the sending and receiving parties. With XML, the method is standard and the content or meaning is flexible.

XML methods can be used to provide structured data formatting for either transaction based messages or entire files. XML is a subset of Standard Generalized Markup Language (SGML) as is Internet Hypertext Markup Language (HTML). It provides start and end tags in a hierarchical structure to define data for example:

```
<XML>
  <EMPLOYEE>
     <NAME>John C. Smith</NAME>
     <DIVISION>Fiscal</DIVISION>
  </EMPLOYEE>
<XML>
```

Many databases now read XML input, have XML tools and provide XML output (e.g., the requested data from an XML or SQL query may be output in the form of XML tagged data). XML messages can transmit DTDs or XML Schemas in the same message with the data or in a linked file. The DTDs and Schemas define the rules for what may be in the file and what it means. One of the benefits of using XML files is that the source system can add a new tag to the message without breaking the message communication.

The human readable ASCII encoding and format tags are helpful to the programmers. Programmers need to tell the application what to look for in each message type and so need to understand what tags to expect in what messages. For this reason it is important for agencies to develop consistent approaches to tag definitions across applications. Any standardization effort needs to take place between communicating Commonwealth government entities or other communicating entities including other states, industry parties, the federal government or even other countries. However, it is also important to keep in mind that one of the benefits of XML is its flexibility. Standardizations should not get in the way of timely and useful solutions. To aid in standardization that would promote XML Schema reuse and tagging standards, the Commonwealth may wish to create a central metadata repository accessible to all Commonwealth agencies.

Efforts in the area of geographic information systems (GIS) provide an example of the reuse that is possible. GIS standards have affected GIS data transmissions in ways similar to the affect of EDI on financial communications. Standards for GIS metadata and messages has been instrumental in the development of GIS mapping servers that can search for data stored on distributed servers and overlay it on a map in the client browser.

One potential area of weakness for XML is its high overhead (e.g., from tagging). For moderate sized messages the automatic compression of HTTP 1.1 (the core protocol of the Internet) or standalone compression tools can improve the transmission efficiency.

B.2 Message Transfers

1) File Transfers

The most common form of message sending and receiving is a request for the transfer of a file. File transfer requests are generally accomplished through the use of operating system "file copy" commands. Middleware compression programs are sometimes used to shrink the size of the message copied.

2) Terminal Emulation

Mainframes have difficulty communicating to the Web natively because their communications protocols were developed and fleshed out before LANs and WANs became ubiquitous. Visual user interfaces other than terminals typically do not exist for mainframes.

Screen scrapers are one middleware method of converting terminal output to browser-viewable output. Thin-client middleware technology is similar but involves running the terminal application on a remote server and transferring the pixels and pixel changes to the end-user's browser. XML conversion of the output is a third approach. Hostbridge, for example, uses a middle-tier application to invoke a CICS transaction using Internet protocols and provides output as an XML document, instead of a mainframe terminal screen. With mainframes, middleware products provide work-arounds because middleware is a distributed system concept and mainframe communications methods do not blend easily with distributed network communications.

3) Translation Services

Some middleware products provide platform-related translation services to ensure that the message is delivered in a language or form that can be understood by the receiving application. Common examples include 7-bit to 8-bit ASCII (American Standard Code for Information Interchange) or ASCII to EBCDIC (binary coded data). Translation service may also include translation from one proprietary implementation of XML to another.

4) File Transfer Protocols

File Transfer Protocols (FTPs) are used to transport whole files over the Internet. This protocol allows files to be transferred between dissimilar systems but it is not a secure protocol. Some middleware tools may enable the scheduling of file transfers for after hours processing, add automated archiving, error recovery, and summary logging. FTP (IETF RFC 959[4]) should not be used if data security is an issue as passwords are transmitted in clear text. A security extension to FTP is provided in IETF RFC 2228[5].

5) HyperText Transfer Protocol

HTTP is short for HyperText Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For

[4] http://ietf.org/rfc/rfc959.txt?number=959 provides the original FTP specifications from the Internet Engineering Task Force (1985).
[5] http://ietf.org/rfc/rfc2228.txt?number=2228 provides the security extension RTF from 1997.

example, when you enter a URL in your browser, this sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

XML is transmitted using HTTP. With XML, the presentation of the data can be separated from the screen format. A programmer may use XML-aware application tools including parsers, extensible style language (XSL) and cascading style sheets (CSS) to create more than one presentation of the data. For example, PDAs and cell phones require presentation styles that are quite different from what would be appropriate for a computer monitor. Yet, because of CSS, the same XML data could be sent to PDAs and computers and a different interface would be shown to each equipment user. Style sheet aware browsers can enable multiple viewing options for the Internet client without requiring the server to resend the data. Browser support for XML style sheets is fairly recent.

HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input. This shortcoming of HTTP is addressed in a number of technologies, including ActiveX, Java, JavaScript and Cookies.

B.3 Message Oriented Middleware

Message Oriented Middleware (MOM) refers to a special set of software applications that are used to manage the message distribution, receipt confirmation, and error handling processes. The messages are distributed network communications between applications. Message tracking on a distributed network is like international package delivery tracking. For example, package shippers today are able to know exactly where their packages are at each step of a physical delivery sequence, which packages are lost, and which are damaged and require resending. To have this level of information detail about application communications, the programmer must rely on middleware tools. To address such complex message sending, tracking and receipt recording requirements, MOM vendors provide several different message services.

1) Store and Forward Middleware and Services

Store and forward services allow the producer of a message (e.g., one application) to identify the recipient location, but if the recipient is not available, the message is held in a central queue or store. The transmission of the message to the recipient may be delayed until multiple messages have accumulated for the recipient, until a time interval has elapsed, or until an event has occurred (e.g., the recipient has become available). This service can enable a busy system to delay processing messages that are not time sensitive to off hours.

Email applications employ a store and forward method of messaging. Email illustrates both the benefits to be gained from this form of messaging and the potential problems that are inherent in messaging systems. Before the advent of Internet mail standards, email systems in the Commonwealth were often incompatible. Middleware was sometimes used to translate between the systems or to provide additional functionality on top of the email. Now state and local agencies are well on their way towards standardization in this area. The email applications used support receiving standard

messages from any system. One of the weaknesses of email messages is that they are one way. A response is procedurally required when appropriate but not mandatory. Complex systems have been built based on email messaging, but they require both sides to trust that the other will respond as appropriate and in a timely manner. Extensions to email systems that are not uniformly implemented include options for notification of message receipt, notification of message being opened, or recall of the message (optional applications that can be invoked by the sending application).

2) Publish/Subscribe Middleware and Services

Publish and Subscribe Middleware services allow the producers of messages to publish the messages to a central location. This central location then uses distribution lists formed by subscription of the recipient.

3) Event Registry Services

For any messaging system, a variety of events may occur between receipt and delivery for one-way messages (asynchronous) or between request and reply for two-way messages (synchronous). To manage and use these events to control messaging, the messaging system needs a way to identify the events and exceptions. Examples of events are "server is not available", "time limit has passed", or "now is 8:00 AM". Example responses are "check for server availability" or "notify sending application". Once a message leaves its application and before it gets to its destination, the use of events and actions falls under the auspices of middleware. Messaging middleware enables the establishment of an event registry and event monitoring services. The event registry can be used to identify thresholds and corresponding actions (e.g., recovery steps). In the case of transaction processing monitor software, the events and responses may be more complex, involving transaction statistics and invocation of special functions. When the transaction process runs cleanly, performance statistics would be gathered but nothing more. But if the process were to fail, the failure event may invoke a paging function to notify the operator on call or if no answer, the backup or supervisor.

4) Intelligent Routing Services

Intelligent Routing Middleware Services ensure the message gets delivered to the appropriate recipients in the correct sequence. The routing service can be configured to handle the exceptions with instructions to forward the message on to another service or to return it if the intended recipient is not available. It could also be configured to transfer a message in sequence from one recipient to another.

B.4 Messaging Middleware Standards

The recommended protocols may apply to mail messaging and/or other application-to-application messaging. Mail programs should support use of MIME, be SMTP/ESMTP enabled, and provide proxy through IMAP4/POP3 servers. Mail programs that interface with Windows clients use Microsoft's MAPI interface. Middleware protocols used by mail applications and/or other applications include: LDAP, DNS, SSL (Secure Sockets Layer), and additional security protocols. Mail uses security protocols for digital signatures and related encryption. Encryption may also be used in the transmission of

sensitive data over LANs including transmissions of passwords, social security numbers, credit card numbers, etc.

MIME stands for Multipurpose Internet Mail Extensions (see examples in Table 3). MIME extensions are important to the recipient mail application because they are used to identify the helper application that enables viewing of the attached file. Several examples of common MIME types are provided below as examples.

Table 3: MIME Examples[6]

MIME Type	Extension	Explanation
text/html	html, htm	Hyper-Text Tag Markup Language
text/plain	txt, text	Plain ASCII text
image/gif	gif	Graphic interchange format
image/jpeg	jpeg, jpg	Joint Photographic Experts Group
image/tiff	tif, tiff	Tagged Image File Format
application/pdf	pdf	Portable Document Format
application/msword	doc	Microsoft Word format
application/zip	zip	compression format

B.5 Message Middleware Guidelines

Table 4 provides information on strategic technologies and approaches related to email, messaging, and secure messaging. In general, those technologies listed as strategic are based on open standards. Agencies are encouraged to incorporate into their architecture one or more of the technologies and strategies listed as strategic. Additional commentary regarding the aforementioned technologies and related tools is provided below.

1) Design Interfaces to be Message based

Interfacing a new application to an existing one is easier when applications are designed for messaging. In an application designed for messaging, the interface mechanism is already defined. The interface process can be automated without modifying the target application.

2) Move towards Asynchronous Messaging

Messages are essentially requests or responses. Interactions between applications and/or databases are of four types from the viewpoint of a programmer:

1. Synchronous messages (requester suspends processing);

[6] The assignment of MIME types is controlled by the Internet Assigned Numbers Authority (IANA). MIME sub-types that begin with 'x-' are not registered with IANA.

2. Deferred synchronous messages (e.g., requester uses polling but continues processing);
3. One-way messages (no reply expected); and
4. Asynchronous messages (requestor continues processing and replier interrupts).

Asynchronous messaging provides greater efficiencies. The systems do not need to have synchronized startups or shutdown procedures. Messages can be stored and processed once the system becomes available. There may be procedures that cannot be completed without a response from the recipient, but the source system can gracefully give a reason and a resolution to the client.

3) Messages may have to be delivered to a variety of platforms.

Although middleware can provide some translation services to assist with application-to-application communications, there are also several protocols and methods that are designed to facilitate communications across platforms. ASCII, EDI, XML, and to a lesser extent MIME types, define open message format standards that can be supported/or translated on most platforms.

Table 4: Message-oriented Middleware Technology Ratings

Obsolescent	Transitional	Strategic	Emerging
Email Messaging (Asynchronous)			
Non-Internet compatible email	X.400 POP3 VIM CMC	IMAP MAPI SMTP/MIME	
File and Data Requests/Replies			
FTP whenever security required		FTP XML file transfer	
Presentation and Translation Services			
	Proprietary style layout separate from application.	XML and CSS (presentation style configurable by administrator for device types) 7 bit ASCII; 8 bit ASCII; EBCDIC (translation)	XSL (presentation style and content configurable by user)
Presentation and Translation Services for Security			
		Encryption/Decryption Services (a wide variety of encryption algorithms are strategic depending on security needs).	

Obsolescent	Transitional	Strategic	Emerging
		E.g., Symmetric Encryption DES Triple DES RC2 RC4	
Presentation: Terminal Emulation			
	SNA/SDLC (OSI level 2)	APPC LU6.2	

4) Use security where applicable including appropriate authentication, authorization, message encryption/decryption, secure transport protocols, and secure storage.
Security services employed by an application are part of a larger set of agency-determined security policies, strategies, and tool. The security architecture document addresses this big picture. Middleware plays a role in deploying security as messages traverse the network and find their destination; however, only a small part of security is provided through middleware. Middleware provides only selected access, communication and authorization tools and protocols for networked applications. Actual security attained by using these tools is dependent on many factors beyond the scope of this paper. As part of a larger plan for building a secure application, the planner may be interested in whether the middleware provides a particular level of encryption determined by key size, for example.

C. Transaction Processing Monitor Middleware and Services

Distributed transaction processing ensures transaction integrity for transactions that involve databases. Transactions often involve multiple steps, all of which must be completed before a database commit can be executed. Transaction processing monitors are critical to n-tier computing, because without them, it would be a very difficult job to write the programs necessary to track transactions across multiple platforms. Some of the services provided by transaction processing monitors include the following: two-phase commits, failure/recovery, synchronization, scheduling, repeat attempts, business-rule-based transaction workflow services, message queuing resource managers, and load balancing. These services are described briefly below. General guidance information follows the descriptions.

C.1 Two-Phase Commits

Commit processing means that a transaction that involves multiple tables or systems is managed so either all of the data is modified or none.

C.2 Failure/recovery

Transaction monitors maintain a sequenced log of all transactions that happened across an integrated set of tables. Transactions logs are used to provide the option of rolling back changes made to the tables to a predefined state by reversing the actions. The logs

can also be used to redo the actions after backup files have been restored from a set point in time, or to redo the actions if an incorrect calculation was being made.

C.3 Synchronization

Transaction monitors can be used to synchronize two different systems. A log is kept of all transactions, which can then be applied to another system so that an identical set of inputs may be provided to both systems.

C.4 Scheduling

The log of transactions can be stored in the message queue until a predetermined event, time, or request occurs to initiate the transfer of those transactions to the other system. Low priority transactions or non-time sensitive transactions can be delayed for after-hours processing.

C.5 Repeat attempts

Transaction monitors can manage multiple repeat attempts to update another system or table, thus, offloading that monitoring responsibility from the requesting system so it can go on to other requests. The transaction-processing monitor may ensure that required processes happen or that the appropriate recovery is initiated.

C.6 Message queue management

Message queues are an important feature of transaction processing monitors. Queues can be established to focus on either availability or reliability. When queues are in memory, they have greater availability and speedier response times. When disk storage is used, this provides greater reliability at the expense of availability. Some middleware message queues provide both disk and memory queues.

C.7 Business-rule-based transaction workflow services

Transacting processing monitors can monitor transaction flows from multiple distributed systems so that business rules can be applied at a central point and so that intelligent routing of certain transactions to other systems or persons may take place. To address customer service and other priorities, business owners may assign priority processing or special handling to selected transactions. Transaction processing monitors allow the business owner to change such rules centrally in the monitor without changing the core application.

C.8 Load balancing services

Load balancing and thread management services are important because transaction processing monitors need to process many transactions on many different systems in a very short time period. The monitor can change traffic patterns, processing parameters, or

increase the pool of processors. This enables the monitor to dynamically adjust to the workload.

C.9 Transaction Processing Middleware Guidelines

Table 5 provides strategic open protocols and example mainframe programs used to define the typical work performed by transaction processing monitors. Transaction processing monitors on a mainframe may not provide distributed transaction processing in the same sense as does a middleware transaction processing monitor in a distributed network environment. However, because today's middleware transaction processing monitors were modeled on mainframe monitors, and because mainframe monitors are still widely used, mainframe monitors are listed here as strategic.

Transaction processing monitors should be used only when transactional integrity is required. Examples of systems that require transactional integrity include integrated financial systems and other core, distributed business systems. Some of the other management capabilities such as scheduling, load balancing, repeat attempts, business rule workflow or logging with recovery can be found in message-oriented middleware and database management middleware.

Table 5: Transaction Processing Monitor Middleware

Obsolescent	Transitional	Strategic	Emerging
		X/Open: XA interface, STDL structured transaction definition language; DTP, distributed transaction processing; CPI-C common program interface for communications	

Historical Note: two TP monitors were widely used in the mainframe world and then later transitioned to the client-server world. These were CICS (customer information control system) and ACMS

D. Application Integration Middleware Servers and Services

Application integration middleware provides interfaces to a wide variety of applications. Application integration middleware might be a service that enables running a legacy system through a thin-client browser or a service that enables the execution of multiple application functions from an integrated user interface. In this section, we will address the methods used to achieve this integration including; application program interfaces (API), remote procedure calls (RPC), and object request brokers (ORB). Summary guidance is provided at the end of this section.

D.1 Methods to Integrate Applications

Methods to integrate applications tend to be aligned with particular application development languages (e.g., C, C++, Java etc.). The methods are often sets of standards developed by particular industry groups. However, even when two vendors deploy implementations using the same set of industry standards (e.g., DCOM—Distributed Common Object Model), their implementations may have differences and may not

interoperate. This result may be due to application tool designers extending the standards or to the standards not being sufficiently specific.

The different industry protocol sets are not designed to interoperate with one another. This is not an issue as long as applications are designed to interoperate only within their intended sphere. When developers try to extend beyond the intended sphere to other applications that implemented a different industry standard, either a middleware gateway must be used to provide protocol translation or the developer must employ both sets of standards in the applications that need to communicate. Remote procedure call differences provide an example. An ONC+ RPC server application cannot interoperate with a DCE RPC client application. Either 1) the server and the client must have both interfaces or a middleware gateway must be employed. In general, it would be advisable for Virginia agencies to use only one RPC method for within-agency distributed functions.

1) Application Programming Interfaces (API)

Many common business applications have a defined interface language to allow programmers to customize or extend the tool. In application-to-application communications, the programmer may use an application provided API or use a middleware provided API to which the programmer adds the required arguments. To the extent that application or database interfaces are open rather than proprietary, future application maintenance will be lessened. APIs often require: customizations to the calling program, use of an appropriate application generation tool and supplied library, or use of a target application software development kit (SDK).

2) Remote Procedure Calls (RPC)

An RPC is a function call issued by the requesting application to run a procedure in a different address space. RPC code is compiled into programs at both ends of a communication. RPCs may require the suspension of processing by the sender until a response is obtained. Implementations today support distributed object components interoperating as a unified whole. The distributed objects may be on different computers across a network, and yet to the application, they all appear as if they were local.

There are competing RPC protocols embraced by major industry providers that do not interoperate. They are Sun's ONC+ RPC and the Distributed Computing Architecture's RPC (DCE RPC). Initially, Remote Procedure Calls were limited to the network protocols for which they were developed. This limited cross platform compatibility. The Internet Inter-ORB Protocol (IIOP) defined a way for any Remote Procedure vendor to map messages to a common TCP network communication protocol.

3) Object Interfaces

Object interfaces simplify addressing a remote call. Object interfaces are part of end-to-end object architecture models. Component Object Model (COM, DCOM), Common Object Request Broker (CORBA), and Remote Method Invocation (RMI) are examples of object models from Microsoft, Object Management Group, and Sun Microsystems, respectively. Object architectures provide the interface definition language (IDL) and blueprints used to define object interfaces (interface stubs for objects). A blueprint

provides guidance for the development of the interface with a particular language binding (dynamic binding). CORBA, for example, provides a variety of blueprints for use by developers.

4) Object Request Brokers (ORB)

Object Request Brokers act as a software bus for objects to intercommunicate. They provide an object location and access service. Microsoft, Object Management Group, and Sun Microsystems all provide this service as part of their object architecture. Applications can request or dynamically invoke objects regardless of location and through metadata services defined as part of the end-to-end architecture (e.g., Naming, Trader, and Interface Repository for CORBA). ORBs also manage object state so that the application does not have to retry if an object is unavailable.

5) Simple Object Access Protocol (SOAP)

Simple object access protocol or SOAP is a remote procedure call protocol that works over the Internet. A SOAP message is an HTTP M-POST request. The body of the request is in XML. The procedure executes on the server, which then replies with message content formatted in XML. SOAP defines what is in a message, which application should deal with it, and whether it is optional or mandatory. SOAP also provides encoding rules for the transport of instances of serialized data. The term ebXML is used to describe the joint use of XML and SOAP to address EDI-like applications for e-business using browsers. This protocol may be an important advance for developers who wish to access objects over the Internet in a standard way without hitting security limitations.

Because SOAP uses HTTP through port 80, it gets around firewall security in ways that DCOM and CORBA cannot. Although SOAP's security avoidance is appreciated by developers who want the connectivity, from a network security perspective, SOAP access is a concern.[7] Arguments exist on both sides regarding whether SOAP may be implemented with adequate security.[8]

D.2 Application Integration Management Services

In addition to utilizing the Object interfaces to applications, many Application Integration Middleware server vendors include other management functions. These may include conversion services, legacy wrappers, and data integration services (access to a set of data such as employee data from several remote sites).

1) Legacy Wrapper Services

Legacy wrappers allow connection to common business systems to allow interfacing with minimal intrusion to the external system. Some package existing legacy code as an object that can be called from another program.

[7] Security concerns are from http://www.vnunet.com/News/1103805 .
[8] For security options with SOAP, see http://www.developmentor.com/soap/soapfaq.htm#16

2) Conversion Services

Conversion services provide a wizard or engineering support for modifying a target system to incorporate new object functionality so that the calling program can integrate with it.

3) Data Mapping and Transformation Services

Data mapping and transformation services can handle the data mapping or transformations on the fly that are necessary when the format of the data is not compatible or complete. Other types of middleware also provide these services.

4) Event Posting Services

Event posting services allow monitoring the status of the interfaced functions and calls to external applications, logging errors, or tracking milestone events. Some ability to correct and reprocess a call may be provided separately or through an integrated Transaction Processing Monitor.

5) Process Trigger Services

Process triggers may be linked to calls to objects to cause variable processing based on the response from the called object. To the extent that the objects and calls have been defined as steps in business functions, business users may be able to rearrange the steps without the assistance of IT staff.

6) Automated Workflow Services

Because Application Integration Servers can call different objects from different applications, they can be used to integrate processes that are handled by different systems.

D.3 Application Integration Middleware Examples

1) Common Address Change Example:

Application Integration Servers that function as a software hub can fuel the imagination of any citizen or manager. It seems this is the method that can be used to write a function such as an address change function one time, package it as an object and now call it from every application that needs to change an address. Or better yet, a developer could have this one object automatically call every application that stores a particular user's address. It would also be possible to call systems that are not connected physically by using the Internet.

Is the above example achievable? Object calls over a network are becoming commonplace and protocols under review by standards committees even enable calls over the Internet. Available technologies can be used to: 1.) connect applications; 2) provide thin-client, browser-based access to a mainframe; or 3) implement client/server applications behind the firewall. However, many older applications were not written for distributed networks and have no exposed API, RPC or COM/CORBA interfaces. Given the hundreds of programs that store a name and address in the Commonwealth, enabling such Commonwealth-wide integration for making address changes would be a monumental task. To integrate to these applications across the Commonwealth, a variety of changes might be required. For example, the developer might have to modify

applications to add appropriate interfaces, and/or replicate application logic in Web servers, and/or modify database middleware used to manage updates to databases. Other applications may not be available around the clock; they may be designed for operation only during office hours or may require batch processing of address changes. Such applications may need message queue services so information could be stored and processed when the system is available. The application may not require the services of a transaction processing monitor. Updating the address in some systems and not in others would be an option preferable to not providing any central updates. Updating at different times may be acceptable for many systems. The queue and error logs on the application integration server would allow addresses changes that could not be made initially to be updated accurately in a locally determined and timely manner.

2) New Enterprise System Example

Some managers have approached the problem of the many different existing programs by replacing the many systems with one new comprehensive system. This approach does not eliminate the need for middleware. Even a comprehensive application may be distributed across the physical architecture or may have client and server functionality, thus requiring services provided by middleware. Applications that access objects need object location and invocation services provided by middleware. Application integration services may be needed to integrate parts of an application. Database middleware may be required to do the initial data transformation and loading for the new application. Often, when implementing a turn-key system, the business may not replace some specialized functions available previously or an external entity may require specialized or ad hoc reports. To meet these needs, some kind of database, messaging or transaction processing middleware will be required. Since Web-enabled applications are a relatively recent occurrence, some of the older major systems do not support thin client browser access, or do not support all functions over the Web. Another drawback to the comprehensive approach is that the extensive scope and timeline can make them costly and time consuming projects to implement, which increases the risk of failure. But, if successful and well designed with new object-oriented interfaces, such replacements could be strategic for building and extending the future architecture.

3) Digital Signature Example

Security is a real concern when extending the application outside of the firewall. The COTS Digital Signatures workgroup implemented several pilot projects. Most applications do not natively support the public key infrastructure (PKI) for encryption or authentication, but most middleware products provide the needed tools. As part of a larger plan, middleware can help in implementing such methods as using digital signatures. The tools enable the current applications to utilize security protocols for authentication or encryption. Also unavailable at present are multi-vendor Certificate Authorities systems that manage trust levels across vendors. Authentication levels may be implemented to the same standard (X.509) but given different names. For this reason, Virginia piloted a deployment of a middleware bridge architecture, for mapping certificate terminology across vendors.

4) Common Portal Example

Middleware can be instrumental in implementing portals. Application integration services may be an important part of the overall architecture that enables the commonwealth to make steady progress towards creating a new citizen-centric, Internet-based government service portal with high-priority applications that citizens want to access online. To simplify portal use by citizens, common functions needed across agencies such as credit card payment processing should be designed once and shared. Methods to verify and authenticate users may also be developed once and shared. Middleware tools are extremely useful in enabling the sharing process. A centrally available middleware tool set can be shared across new development efforts.

The above examples illustrate the need for the last type of middleware, the super-services middleware, which combine all of the above middleware tools and services into a managed suite. Super-services are discussed in section E.

D.4 Application Integration Middleware Guidelines

Protocols and services related to application integration are noted in Table 6. Additional commentary related to applying these protocols and services is provided below.

Those considering XML for RPC and SOAP are cautioned that they are emerging technologies with specifications in draft or first versions but few implemented applications. (XML for message formats is much more mature and strategic.) For more information, see the World Wide Web Consortium (W3C) at http://www.w3.org.

Networks that employ TCP/IP can take advantage of IIOP compliant distributed objects.

New object-oriented business applications should be portable object adapter (POA) compliant. POAs are written using IDL. POA standards replace Basic Object Adapter (BOA) standards, which produced language specific adapters.

Table 6: Middleware Application Integration Services

Obsolescent	Transitional	Strategic	Emerging
Object Request and Request Broker Protocols/Suites			
		MS DCOM +, distributed common object model OMG CORBA, common object request broker J2EE/RMI, Java 2 Enterprise Edition (the distributed version) and Remote Method Invocation GIOP, General Inter-ORB Protocol and IIOP, Internet Inter-ORB Protocol (maps GIOP to TCP) POA, Portable Object Adapter	ebXML (includes SOAP, Object Access Protocol)

Obsolescent	Transitional	Strategic	Emerging
Enterprise Application Integration Services (EAI)			
		Use of Integration Servers/Services	Integration Servers/ Services
Workflow Tools			
		Workflow Tools	
Remote Procedure Calls			
	Suns' ONC+ RPC	DCE RPC DCE secure RPC (integrated with DCE security protocols for authentication, protection level and authorization)	XML synchronous methods SOAP
Object and Application Interfaces			
		IDL (interface definition language) stubs; MIDL (Microsoft); OMG IDL; DCE IDL	XML SOAP . NET

DCOM/CORBA.RMI are all recommended strategic directions. Programming language preferences may be an important influence in object model selection. ORB protocols are still industry based and have interface limits.

E. Super-Service Middleware Servers and Services

Super-service middleware provides the management and integration of multiple (heterogeneous) types of middleware with value added services. These super services are often Web-enabling middleware that allow for the easy integration of back-end applications with new e-commerce and e-government systems. Super-service middleware enables rapid responses to changing business requirements.

Super-service middleware has value added services that may include a common look and feel for APIs, common security and directory lookups, user friendly access to other middleware, audit and logging services, packaging of adaptors, error recovery and exception handling tools, data transformation tools, and mapping repository tools.

E.1 Value Added Services

1) Value Added: Common API / Simplified portability

Super service middleware can provide a common, simplified interface to call objects and invoke application specific API's. This simplifies training for the development team and allows each programmer to call this common routine with a language they understand.

2) Value Added: Common security and directory lookup

A common security and directory lookup can simplify maintenance of the various user identifications and passwords that are needed to access each of the distributed systems and databases. Logons can be mapped from one user identification to another user identification using a password appropriate for the target system(s). A common directory gives the distributed system a common place to find configuration information needed for accessing and running the various middleware services.

3) Value Added: User-friendly access to other middleware

Some middleware is not user friendly. The interfaces provided may have the appearance of terminal screens. Setup ease may vary drastically from tool to tool. Setup and administration screens and procedures may be highly technical and difficult to understand. Super-service middleware packages try to address some of these inconsistencies and problems by providing well-integrated graphic user interfaces to each of the components. The user-friendly front ends make the packaged tools and services easier to configure, use, and administer.

4) Value Added: Integrated Audit and logging services

Super server middleware services can add some of the integrated control and management to the distributed network that existed in the unified mainframe.

5) Value Added: Packaging of "connectors/adaptors" to standard business applications and databases.

Super-service middleware adaptors can provide a quick start to those who are integrating commonly used business applications into their environment. The more widely used an application is, the more likely it is that the super-service middleware vendor has incorporated adapters for that specific application. Yet another possibility is that if not incorporated in the middleware, an application specific adapter may be available in the marketplace such that it can be added to the adapters in the super-service middleware.

6) Value Added: Error recovery/exception handling (business rules on top of reliable service request)

Super-service middleware may offer well-integrated services including event tracking, transaction monitoring, etc. The super-service middleware may coordinate the use of database middleware, messaging middleware, transaction processing monitors, and object calls to handle exceptions and errors in a graceful and appropriate manner and in accordance with flexible business rules.

7) Value Added: Preservation of legacy middleware and systems

By managing multiple middleware systems, a super service can extend and add value to existing middleware products. The super service adds the management. The existing tool does the work.

E.2 Super-Service Middleware Servers and Services Guidelines

1. Super service middleware should be shared or centralized across applications within an agency, and if appropriate, across agencies that interoperate.

2. Maximum reuse should be made of common routines to decrease programmer costs.

3. Developers should buy rather than build interfaces.

4. Those writing middleware specification should investigate what super services have done for other customers in a similar situation before finalizing requirements. For a shared or common system, shared uses should be investigated.

5. The Enterprise Architecture team in cooperation with the Department of Technology planning should identify a project that could be used to evaluate a super-service middleware product. The super-service product would be used to provide necessary middleware services on a project that needs more than one middleware service. The evaluation results could be incorporated into middleware best practices.

VII. E-Government Examples

The following examples hypothetical e-government applications that are used to demonstrate the mix of middleware types that may be applicable. Many compromises need to be made when designing actual systems. The actual middleware needs of Commonwealth applications in these areas may vary considerably from the examples provided, given differences in scope, platforms, and actual requirements.

Table 7: Example Applications and Potential Middleware Use

Application Functions	Interaction Type	Examples	Database	Message	Transaction Processing	Application Integration	Super Service
Administration	C2G	Address Change	X	X		X	X
	B2G	Address Change	X	X		X	X
	G2G	Address Change	X	X		X	X
	E2G	Time Sheets	X		X	X	X
Finance	C2G	Tax, License	X	X	X	X	X
	B2G	Procurement	X		X		X
	G2G	Accounting to Budget	X	X	X	X	X
	E2G	Travel Vouchers	X	X	X	X	X
Legal	C2G	Courts Docketing	X	X	X	X	X
	B2G	Employer Filings	X	X	X	X	X
	G2G	Human Services Coordination	X	X	X	X	X
Information	C2G	General Portal	X				
	B2G	Business Opportunity Portal	X			X	
	G2G	Reports	X			X	
	E2G	Policies and Procedures	X			X	

The government examples are presented based on a framework of the following types of interaction: customer to government (C2G), business to government (B2G), government to government (G2G), and employee to government (E2G). Each of these areas is examined from the perspective of the following major application functions: administration, finance, legal, and information. The need for middleware in each of these functions would vary greatly based on the scope of the function. For example, a travel expense, employee-to-government function may not require use of a separate middleware product if it were being done within one agency from an internal Intranet behind the firewall, and the application did not require Web-enabling. However, if the application were to be used by multiple agencies, or if it were to use an external portal, then more systems could be affected, and more middleware types might be needed to bridge the different platforms, systems, and data structures behind the scenes. The examples in Table 7 demonstrate the considerable potential need for multiple types of middleware in common government applications.

VIII. Policies, Standards and Best Practices

The Middleware domain team has developed several recommendations, which are presented in this section as Policies, Standards, or Best Practices. If applicable, any existing ITRM Policies, Standards, and Guidelines would also be listed. These recommendations are not a summary of information already stated in the prior discussion of middleware types. They specifically address how the Commonwealth agencies should or might approach middleware architecture decisions.

Policies are high-level requirements for agencies and standards are detailed requirements. The policies and standards provided below include existing policies and standards if appropriate (e.g., ITRM Policies or Standards originally created by the Council on Information Management, which is now the Department of Technology Planning). Also included are recommendations of the domain team to continue and/or updated and continued existing policies and standards. Recommended standards are recommendations awaiting COTS approval. Best practices are intended as guidance. Best practices will become part of guidelines for agencies that are not mandatory. Together, the recommendations below constitute the domain team's middleware advice to agencies.

Agencies should consider the following Recommended Standards and Best Practices when designing and implementing middleware services or service enhancements.

A. Policies

No Current ITRM Policies address middleware related issues. An ITRM Policy is a mandatory, high-level instruction to agencies such as "all state agencies must have a security policy."

B. Recommended Policies

No middleware policies are recommended by the domain team.

C. Standards

No Current ITRM Standards address middleware related issues. An ITRM Standard is a set of mandatory, detailed instruction to agencies such as "all agencies must adhere to EIA/TIA building wiring standards".

D. Recommended Standards

Recommended Standard 1. Directory Services. State Agencies must employ LDAP compliant directory services. This will lay the groundwork for uniform decentralized lists that can be aggregated centrally for use by the Commonwealth.

Recommended Standard 2. Email Protocols. State agency email messaging must be SMTP and MIME compatible. Local governments are encouraged to follow this standard as well.

E. Best Practices

Best Practice 1. Planning. Before acquiring a central middleware solution, agencies should map their present middleware sources and uses, and should develop a plan for migration to the central middleware or modifying present uses if needed. Agencies should use middleware strategies that are scalable, extensible, and maintainable.

Best Practice 2. Define Interfaces. Agencies should carefully define their interfaces and business requirements.

Best Practice 3. Testing Middleware Modifications. Middleware tools and services should be thoroughly tested. Consideration should be given to the need to maintain a separate environment for testing modifications.

Best Practice 4. XML Documentation. As XML becomes more widely used in state and local government, the Secretary of Technology should designate a responsible agency for providing Web-based access to common tag definitions, DTDs, XML Schemas and CSSs to promote consistent use of terms and to support reuse of prior work whenever possible. Agencies using XML should participate and monitor state and federal initiatives in their sphere for XML tag standards development.

Best Practice 5. XML Storage. Agencies should only use XML for classifying data content for message interfacing and presentation, not for long-term storage of structured data.

Best Practice 6. XML/SOAP. Agencies considering XML and SOAP should investigate the error detection, and auditing capabilities of their application.

Best Practice 7. Interfaces. Agencies should buy interfaces, not build them.

Best Practice 8. Strategic Investigation. The Commonwealth and its agencies should carefully investigate the success other agencies and states have had in the

deployment of middleware products before considering a separate middleware acquisition.

Best Practice 9. Shared Resource. Agencies may wish to consider whether they should acquire middleware as a shared resource across several agencies.

Best Practice 10. Logical Partitioning. Agencies should use middleware to support logical partitioning and boundaries.

Best Practice 11. Open Interfaces. Agencies should use technologies that support open interfaces, are persistent, and are non-proprietary whenever possible.

Best Practice 12. Efficiencies. Asynchronous messaging provides opportunities for making efficient use of parallel processing capabilities in the network environment.

Best Practice 13. Single Sign-on. Middleware can play an important role in enabling a single sign-on for all applications and services.

Best Practice 14. Security and Directories. Agencies should consider implementing separate directories for internal use and external (i.e., beyond the firewall) use.

Best Practice 15. Email. The Message Transfer Agent (MTA) in email applications should be LDAP enabled.

Best Practice 16. LDAP Schema Coordination. Many universities use the EDUCAUSE/Internet2 eduPerson task force effort as a vehicle for coordinating directory standards for faculty and student access. The EDUCAUSE/Internet2 eduPerson task force has the mission of defining an LDAP object class that includes widely used person attributes in higher education. The URL for the Higher Education LDAP Schema work is: http://www.educause.edu/eduperson. The Department of Technology planning should investigate the utility and applicability of this work to government-wide person attributes.

Best Practice 17. Adapters. New object-oriented business applications should be portable object adapter (POA) compliant.

Best Practice 18. The Enterprise Architecture team in cooperation with the Department of Technology planning should identify a project that could be used to evaluate a super-service middleware product. The super-service product would be used to provide necessary middleware services on a project that needs more than one middleware service. The evaluation results could be incorporated into middleware best practices.

Glossary

ACMS A transaction processing monitor from Compaq that runs on the open VMS operating system.

Active X Microsoft's answer to Java. ActiveX is a stripped down implementation of OLE designed to run over slow Internet links.

ADSI Active Directory Service Interfaces (ADSI) abstract the capabilities of different directory services from different network vendors to present a single set of directory service interfaces for managing network resources

API Application Program Interface or Application Programming Interface.

APPC LU6.2 APPC allows user written programs to perform transactions in a Client-Server IBM network to access a CICS, in MVS "batch" through APPC/MVS, in VM/CMS, in AIX on the RS/6000, and on the AS/400

ASCII American Standard Code for Information Interchange. **"H**uman readable text." The first 128 character codes of any of the ISO 8859 character sets is always identical to the ASCII character set

ASP Active Server Page (Microsoft) A scripting environment for Microsoft Internet Information Server in which you can combine HTML, scripts and reusable ActiveX server components to create dynamic web pages.

Asynchronous/connectionless communication A program-to-program communication model that does not block any of the communicating partners and that allows for time independent interactions.

Authentication Verification a user is who they say they are.

B2G Business to government. Refers to a business process involving electronic interaction of business partners.

BOA Basic Object Adapter protocol. Replaced by POA, portable object adapter.

C2G Customer to government. Refers to a business process involving electronic interaction of citizens with government.

CA Certificate authority. A system for managing certified digital signatures. Manages the implementation of policies to authenticate, authorize and revoke the assignment of keys to users.

CICS IBM mainframe application server that provides industrial-strength, online transaction management for mission-critical applications. on MVS/ESA, OS/390, VSE/ESA and z/OS. Thirty years old but repackaged to turn mainframes into Web servers.

CIM Virginia's Council on Information Management. The forerunner of DTP, the Department of Technology Planning. An agency of state government that supports information technology planning and policy in government.

COM Component Object Model (Microsoft); also DCOM and DCOM+ for distributed systems

Cookies Information stored on a Website visitor's computer regarding a transaction with a Website that may be returned to that Website at each subsequent visit if requested by the Website.

CORBA Common Object Request Broker Architecture. OMG's open, vendor-independent architecture and infrastructure that computer applications use to work together over networks.

COTS Virginia's Council on Technology Services. An advisory groups to the Secretary of Technology that represents state and local government agencies and higher education.

CPI Common Program Interface. IBM's Systems Application Architecture API.

CSS Cascading Style Sheets. An XML protocol used to control formatting of Web pages.

Data marshaling The conversion of data between platform specific representations and the packaging according to the requirements of a particular network protocol in order to perform the data transport between different nodes.

DCE Distributed Computing Environment from Open Computing Group. Includes Remote Procedure Call (RPC), the Cell and Global Directory Services (CDS and GDS), the Security Service, DCE Threads, Distributed Time Service (DTS), and Distributed File Service (DFS).

DCOM+ The Distributed Component Object Model. A set of Microsoft protocols that enable software components to communicate directly over a network.

DNS Domain name system. A general-purpose, distributed, replicated, data query service chiefly used for Internet communications for translating hostnames into IP addresses.

DTD Document Type Definition. An XML protocol for communicating tagging standards that will be used in an XML communication. The definition of a document type in SGML or XML, consisting of a set of mark-up tags and their interpretation.

DTP Department of Technology Planning. An agency of Virginia state government that supports information technology planning and policy in government and supports the work of the Governor's Secretary of Technology.

E2G Employee to government. Refers to a business process involving electronic interaction of citizens with government.

EAI Enterprise Application Integration. The use of middleware to integrate the application programs, databases, and legacy systems involved in an organization's critical business processes.

EbXML ebXML is a set of specifications that together enable a modular electronic business framework. The vision of ebXML is to enable a global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML based messages. ebXML is a joint initiative of the United Nations (UN/CEFACT) and OASIS, developed with global participation for global usage

EBCDIC Extended Binary Coded Decimal Interchange Code. IBM's 8-bit extension of the 4-bit Binary Coded Decimal encoding of digits 0-9 (0000-1001).

EDI Electronic Data Interchange. EDI works by providing a collection of standard message formats and element dictionary that can be used by businesses to exchange electronically. EDI is used for electronic commerce. EDI interchanges use some variation of the ANSI X12 standard (USA) or EDIFACT (UN sponsored global standard).

Emerging Rating category used in this document to rate middleware technologies. "The Virginia Enterprise Architecture promotes only evaluative deployments of this technology. This technology may be in development or may require evaluation in government and university settings. Use of this technology may be high risk."

ERwin A database design and optimization tool from Computer Associates.

ESMTP Extended SMTP. Initially defined in RFC 1869 and extended thereafter

EWTA Enterprise wide technical architecture.

Extensible quality of a system that allows new features and functions to be added to it.

Firewall A dedicated gateway machine with special security precautions on it, used to service outside network, especially Internet, connections and dial-in lines. The idea is to protect a cluster of more loosely administered machines hidden behind it from crackers. The typical firewall is an inexpensive microprocessor-based Unix machine with no critical data, with modems and public network ports on it, but just one carefully watched connection back to the rest of the cluster. The special precautions may include threat monitoring, call-back, and even a complete iron box keyable to particular incoming IDs or activity patterns. Firewalls often run proxy gateways.

FTP File Transfer Protocol. A protocol used to transmit whole files over the Internet. Security with FTP.

G2C Government to Customer. Refers to a business process involving electronic interaction of government with citizens.

GDS Global Directory Services, such as DNS and GDS (X.500), grew out of the computer industry's need to reference objects in distributed networks across an entire enterprise and worldwide.

GIS Geographic Information System.

HTML HyperText Markup Language – A subset of SGML. A W3C standard for formatting Web pages.

HTTP Hypertext Transfer Protocol. The protocol used on the World-Wide Web for the exchange of HTML documents. It conventionally uses port 80.

HTTP MPOST and HTTP POST "A SOAP request can use HTTP's POST verb. In fact, however, the protocol requires that the first request to a server is made using M-POST. M-POST is a new HTTP verb defined using the HTTP Extension Framework (http://www.w3.org/Protocols/HTTP/ietf-http-ext). If a request made using M-POST fails, the client can try again using a standard POST request. (In this case, future requests can also use POST because the server obviously doesn't support M-POST.) M-POST allows sending HTTP headers that can't be sent via the standard POST verb, providing more flexibility for SOAP users. Firewalls can even force the use of M-POST if desired, by simply refusing all HTTP POSTs with a content type of "text/xml-SOAP".

Hypertext Hypertext is text that contains links to other text

IANA The central registry for various "assigned numbers": Internet Protocol parameters, such as port, protocol, and enterprise numbers; and options, codes, and types. The currently assigned values are listed in the "Assigned Numbers" document STD 2. To request a number assignment, e-mail <iana@isi.edu>.

IDL Interface Definition Language defined by OMG is a language for describing the interfaces of software objects. Various Vendors have their own version of IDL (e.g., MIDL by Microsoft).

IETF Internet Engineering Taskforce. A standards group that works on Internet architectural issues.

IIOP Internet Inter-ORB Protocol. A protocol that defines a way for Remote Procedure vendor to map messages to the TCP network communication protocol

IMAP stands for Internet Message Access Protocol. It permits a "client" email program to access remote message stores as if they were local.

Interface repository The interface repository is part of object-oriented middleware. It contains the definitions of all the services that objects can provide. The definitions form the contract by which a client can invoke requests upon a server object.

IP Internet Protocol. A network addressing protocol. Two versions are defined: IPv4 and IPv6.

IP address An identifier for a computer or device on a TCP/IP network. Networks using the TCP/IP protocol to route messages based on the IP address of the destination. The format of an IP address is a 32-

bit numeric address written as four numbers separated by periods. Each number can be zero to 255. For example, 1.160.10.240 could be an IP address. Within an isolated network, you can assign IP addresses at random as long as each one is unique. However, connecting a private network to the Internet requires using registered IP addresses (called Internet addresses) to avoid duplicates.

ISO International Standards Organization.

IT Information Technology

LDAP Lightweight Directory Access Protocol. A protocol for accessing on-line directory services. LDAP was defined by the IETF to encourage adoption of X.500 directories. The Directory Access Protocol (DAP) was seen as too complex for simple Internet clients to use. LDAP defines a relatively simple protocol for updating and searching directories running over TCP/IP.

J2EE Java 2 Enterprise Edition. The distributed version of Sun's Java platform. with Enterprise JavaBeans™ (EJB™), JavaServer Pages™ (JSP™) and Java Servlet API component technologies.

Java portable language from Sun designed to run on any machine with a Java Virtual Machine interpreter.

JDAP Java Directory Access Protocol --an implementation of the Lightweight Directory Access Protocol.

JDOM Java document object model. A way to represent an XML document for easy and efficient reading, manipulation, and writing.

JDBC Java Database Connectivity is a standard SQL database access interface. It comes with an ODBC bridge.

Load balancing Load balancing means that requests from clients are distributed across available servers to achieve better utilization of computing resources. In general, load balancing can be based on network traffic, CPU load, relative power of the server, size of the server's request queue, a simple round robin method, or other mechanisms.

Loosely coupled Architectures based on publish/subscribe communications can provide a lightweight and resilient foundation for applications that do not require tight coordination.

MAPI Messaging Application Programming Interface. A protocol used to write components that connect to different mail servers, provide access to custom address books and provide rich storage facilities.

MDC Meta Data Coalition

Metadata (also Meta data) Data about data that makes the process of finding and using data easier

MIME Multipoint Internet Mail Extensions. An official Internet standard that specifies how messages must be formatted so that they can be exchanged between different email systems.

MOM Message Oriented Middleware

Monolithic application is entirely installed on one machine.

MTA Message Transfer Agent. The internal component of an e-mail delivery system, responsible for mail collection from and distribution to MUAs, and relay of mail between e-mail post offices. Also called e-mail server.

MUA Mail User Agent—Primary entry and exit point for an e-mail system. Also called an e-mail client.

Multi-threaded Sharing a single CPU between multiple tasks (or "threads") in a way designed to minimize the time required to switch threads.

Naming service Naming service refers to the ability of application programs to locate application components offered by other applications in a distributed environment. Typical naming service should support registration of services in the naming service and their subsequent location through the naming service.

NDS Netware Directory Services. A hierarchical, class-based directory structure for accessing network resources.

NIST National Institute of Standards and Technology. Formerly, the National Bureau of Standards. A United States governmental body that helps to develop standards.

N-tier A client-server architecture in which the user interface, functional process logic (the middle tier) and data storage and access are developed and maintained as independent modules, most often on separate platforms. If three-tiered, the middle tier is a single tier. If n-tiered, the middle tier is multi-tiered.

Obsolescent Rating category used in this document to rate middleware technologies. "The Virginia Enterprise Architecture actively promotes that agencies employ a different technology. Agencies should not plan new deployments of this technology. Agencies should develop a plan to replace this technology. This technology may be waning in use or no longer supported. "

ODBC Open Data Base Connectivity. ODBC is based on Call-Level Interface and was defined by the SQL Access Group. Microsoft was one member of the group and was the first company to release a commercial product based on its work (under Microsoft Windows) but ODBC is not a Microsoft standard.

OLE- Object Linking and Embedding. The software capability that enables the creation a compound document that contains one or more objects from one or more applications. Objects can be linked or embedded in the compound document. Changes to linked objects are reflected in the source and vice versa. Embedding objects breaks all links.

OLE-DB Microsoft's interface to data. OLE DB is an open specification designed to build on the success of ODBC by providing an open standard for accessing all kinds of data.

OMG Object Management Group. A consortium aimed at setting standards in object-oriented programming.

ONC+ RPC Open Network Computing (Sun) Remote Procedure Call. A remote procedure call or function call protocol developed by Sun.

Open Group, The The Open Group is a standards development and product approval consortium. "The Open Group's Mission is to offer all organizations concerned with open information infrastructures a forum where we can share knowledge, integrate open initiatives, and certify approved products and processes in a manner in which they continue to trust our impartiality."

Open standards Standards that are available for all vendors to use in product development.

Open system A desirable but unachievable computing architecture state. A system built using open rather than proprietary standards. A product based on widely implemented vendor-neutral standards.

ORB Object request Broker. A software tool that enables the location of and access to objects in a distributed system.

OSI Reference Model Open Systems Interconnect seven layer model. A model of network architecture and a suite of protocols (a protocol stack) to implement it, developed by ISO in 1978 as a framework for international standards in heterogeneous computer network architecture. The OSI architecture is split between seven layers, from lowest to highest: 1 physical layer, 2 data link layer, 3 network layer, 4 transport layer, 5 session layer, 6 presentation layer, 7 application layer. Each layer uses the layer immediately below it and provides a service to the layer above. In some implementations, a layer may itself be composed of sub-layers.

POA Portable object adapter standard. An adapter written using IDL.

PDA Personal Digital Assistants

Persistence service Defines a service when an object state can be preserved in a persistent media such as an object database.

PKI Public Key Infrastructure – a way to distribute security and encryption keys.

POP3 Post Office Protocol version 3. The most common protocol used by MUAs to retrieve mail from a central message store (messaging server). Most commercial Internet Mail post office products include a POP3 server. IMAP is typically a better choice than POP3 for unified messaging.

Portability 1) The ability to pick-up, store and delivery messages everywhere. 2) The ease with which a piece of software (or file format) can be "ported", i.e. made to run on a new platform and/or compile with a new compiler.

Protocol - A set of rules. For example, network protocols are rules that enable connectivity and communication.

Publish & subscribe Providers of information can publish it for consumption by information consumers, without any logical connection between the participating applications. 2) Software or protocols that enable publishing and subscribing.

Quality of service 1) Reliable message delivery (no messages are lost in case of system failure). 2) Guaranteed message delivery (messages are delivered within a defined time limit, even in the case of network or system unavailability). 3) Assured message delivery (messages are delivered at most once).

Repository A repository is a collection of resources that can be accessed to retrieve information. Repositories often consist of several databases tied together by a common search engine.

Reusable component A sub-object derived from an object or a class of objects by taking advantage of inheritance properties. The derived object inherits the instance variables and methods of the super class but may add new instance variables and methods.

RMI Remote method invocation. A J2EE RPC.

RPC Remote Procedure Call—An external form of communication that allows a client to invoke a procedure in a server.

Scalability The ability to expand as higher and higher volumes occur due to high volume operations with a parallel engine.

SDK Software Developer's Kit; Software Development Kit

SDLC Synchronous Data Link Control. An IBM/SNA communications protocol. HDLC, high level data link control was derived using SDLC. SDLC manages synchronous (i.e., uses timing bit), code-transparent, bit-serial communication which can be duplex or half-duplex; switched or non-switched; point-to-point, multipoint, or loop.

Security service Compared to monolithic environments, distributed systems create new challenges for the implementation of security. Middleware must provide authentication, auditing, authorization and encryption services that allow a client to conduct a secure communication with a server.

SGML Standard Generalized markup Language. HTML and XML are subsets of SGML.

SMTP Simple Mail Transfer Protocol (SMTP), documented in RFC 821, is Internet's standard host-to-host mail transport protocol.

SNA IBM's Systems Networking Architecture (SNA) provides a structure for transferring data between a variety of computing platforms.

SNMP Simple Network Management Protocol. The Internet standard protocol, defined in STD 15, RFC 1157, developed to manage nodes on an IP network.

SOAP Simple Access Object Protocol. A minimal set of conventions for invoking code using XML over HTTP

Sockets - virtual connections between processes. They can be of two types, stream (bi-directional) or datagram (fixed length destination-addressed messages). The socket library function socket() creates a communications end-point or socket and returns a file descriptor with which to access that socket. The socket has associated with it a socket address, consisting of a port number and the local host's network address.

SQL Structured Query language. An industry-standard language for creating, updating and, querying relational database management systems.

STDL Structured Transaction Definition Language. a high-level language for developing portable and modular transaction processing applications in a multi-vendor environment.

Store and forward A term used in message processing where a message is saved and then delivered.

Support for standard management platforms Management of large-scale distributed application environments requires appropriate tools. These tools should be based on standards (e.g. SNMP), so that the management of applications can be integrated with popular management platforms like OpenView in order to provide a consolidated picture of the state of network, operating system and application components.

Strategic Rating category used in this document to rate middleware technologies. “The Virginia Enterprise Architecture promotes use of this technology by agencies. New deployments of this technology are recommended.”

Synchronous/connection oriented communication The implementation of a request/reply model for communication, i.e. the client program transfers data and control to the server with each call and it is blocked until a reply is returned.

TCP/IP 1) Transmission Control Protocol over Internet Protocol. 2) The TCP/IP Suite of protocols.

Technical architecture In enterprise architecture, business and technical computing specifications are considered. The technical architecture includes specification for only technical dimensions or components. In Virginia’s enterprise architecture, the technical domains include: middleware, security, platform, network, application, database, systems management, and information architecture.

TP Transaction Processing Monitor

Transaction service Guaranteed “all-or-nothing” execution of update requests against multiple (heterogeneous) resources.

Transitional Rating category used in this document to rate middleware technologies. “The Virginia Enterprise Architecture promotes other standard technologies. Agencies may be using this technology as a transitional strategy in movement to a strategic technology. This technology may be waning in use or no longer supported.”

Triggering Application components are dispatched automatically based on a predefined event condition. The definition of the event concept varies between the different types of middleware, e.g. request for certain elements in a database, arrival of a message in a queue, or method invocation request for an object that is managed by a ORB.

URL Uniform Resource Locator. An address, usually for locating Web pages. (E.g., FTP//: abc.org). The part before the first colon specifies the access scheme or protocol. Commonly implemented schemes include: ftp, http (World-Wide Web), gopher or WAIS. The "file" scheme should only be used to refer to a file on the same host. Other less commonly used schemes include news, telnet or mailto (e-mail). The part after the colon is interpreted according to the access scheme. In general, two slashes after the colon introduce a hostname (host:port is also valid, or for FTP user:passwd@host or user@host). The port number is usually omitted and defaults to the standard port for the scheme, e.g. port 80 for HTTP.

VIM Lotus/IBM CC:Mail.

X.400 International Telegraph and Telephone Consultative Committee (CCITT), now known as the ITU Telecommunication Standardization Sector, completed the first release of the X.400 message handling system standard. The standard provided for the exchange of messages in a store-and-forward manner without regard to the user's location or computer system.

X.500 An ISO OSI Directory Service with an information model, a namespace, a functional model, an authentication framework, and a distributed operation model. X.500 directory protocol is used for communication between a Directory User Agent and a Directory System Agent. To allow heterogeneous networks to share directory information, the ITU proposed a common structure called X.500. However, its complexity and lack of seamless Internet support led to the development of Lightweight Directory Access Protocol (LDAP), which has continued to evolve under the aegis of the IETF. Despite its name, LDAP is too closely linked to X.500 to be "lightweight".

X.509 Standards for PKI or Public Key Infrastructure (e.g., Digital Signatures)

X/A an application program interface (API) specification between a global Transaction Manager and Database.

XSL Extensible Stylesheet Language

XML Extensible Markup Language

XML Schema XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents.

We would like to thank FOLDOC, the Free On-line Dictionary of Computing for being there for references, links, and definitions. http://foldoc.doc.ic.ac.uk/foldoc/

Additionally, we used a variety of other highly recommended resources on the Internet including:

What Is? http://whatis.techtarget.com/

North Carolina ITS Glossary http://www.its.state.nc.us/Information/Glossary/GlossMain.asp

U. of Colorado Computing Standards with Links http://itp-www.colorado.edu/~scig/std_glossary.html

Appendices

Appendix A. History of the Evolution of Separate Middleware Services

The term middleware is relatively new in the definition of computing architectures; separate middleware has mushroomed as a result of the growth of multi-tiered applications, multi-platform, distributed computing environments, and e-business. In the more homogeneous mainframe computing environments of the 60's, 70's and 80's, programmers did not generally have to deal with the complexities of local and wide area networking, logical partitioning of applications, or applications running on multiple physical platforms. Less complex environments made communications simpler. Programmers knew where data resided, used reports or terminal screens as the primary user interface, issued simple function calls for remote processing, and employed simple operating systems commands when necessary. They conducted most of their computing work inside the controlled world of a unified mainframe or miniframe environment.

When local area networks (LANs) began to proliferate in the late 80's and 90's, two-tiered client server applications became commonplace. The computing environments were often characterized by fat client workstations interacting with a multifunctional database servers, which had internal operating architectures that were much like the mainframe—architectures that provided simple integrated solutions for most of the communication needs between the client workstation and the database server over the LAN. The database systems also provided the application languages, metadata stores, internal middleware solutions, and other tools. Similar self-contained, stovepipe applications for massive transaction processing remained on the mainframes. During this period, mainframe applications were generally viewed as legacy systems that should be converted to the new database system or replaced by shelf ware systems. Major enterprise off-the-shelf systems (e.g., PeopleSoft and SAP R/3) sometimes used middleware functions defined within a database environment or created their own built-in network communications functionality. Programmers using the databases or shelf ware had to learn new methods, but generally were still somewhat shielded from the network communications and environmental complexities.

The two-tiered applications and client to database server communications quickly evolved to three or more tiers when applications became Web-enabled and server farms dotted the landscape. Internet connectivity, e-business, corporate mergers and buyouts, and internationally distributed businesses began to define the heterogeneous mix of application types, databases, application languages, and shared services in even more complex, distributed networked environment. Databases, mainframes, and complex applications were all providing redundant pieces of the communications between and among applications and databases. The computing environments had no central controller of all high-level communications. IT managers were faced with finding solutions that would integrate the high-level communications in their increasingly heterogeneous environments. Of extreme importance was simplifying the environment from the viewpoint of programmers who had different middleware solutions with every

application. The growth of Internet and e-business played a very significant role in changing attitudes. One big change was that mainframe and legacy applications became more accepted as a permanent part of the computing landscape. Another, perhaps more important change was that a large proportion of existing systems potentially needed to provide data and services to customers over the Internet. Many new interfaces were needed.

Having to separately program so many new database and application interface in these multi-tiered, distributed environments would have been a very wasteful approach. The marketplace responded to the apparent coordination void by creating bundles of middleware services. Sometimes the products were coupled with application functionality aimed at specific business needs such as e-business. Transaction processing middleware had been around for a long time. It migrated from mainframes to the client server world and was often formed the base layer of the "messaging plus" middleware solutions. Some bundled services were designed to help specific legacy systems to interface with non-mainframe, networked applications. Some products specialized in communications among disparate databases. Others focused on providing a comprehensive solution for object-oriented services. Each provided some tools for building and accessing sharable connectivity services.

Middleware super-services are a more recent phenomena in the marketplace. The super services bundle a variety of middleware services from two or more vendors (e.g., transaction processing from one vendor and object brokering from another vendor) along with additional environment management functionality into one product. Essentially, some middleware providers are offering a suite of services that will envelop the high-level communication needs of an entire distributed, multi-platform environment. Super services try to return central control to interoperability and location transparency tasks by providing all of the brokering, monitoring, metrics, protocols and communication services needed in a particular computing environment. In offering these services, middleware providers assumed that legacy systems would continue and multiple standards would be in use (e.g., in a government setting, the mixed environment might be within an agency or across several agencies with a common system). Other middleware providers focus on the enterprise integration aspects (e.g., enterprise application integration services (EAI), transaction processing, or Web enablement needs (e.g., comprehensive e-business solutions).

One important issue regarding acquiring middleware is that agencies must understand that they may be making some serious architectural decisions in choosing product. Unless agencies understand all of the underlying components and how they work together (or do not), they will be unable to make architecturally sound decisions. The middleware decisions made related to one application may seriously limit future options for providing a comprehensive, coordinated approach to middleware services and functions. Agencies need to know what all the pieces are and what they do. Agencies also need to know how well coordinated the set up is, both from component to component and with the existing architecture. Understanding middleware only from a use perspective (a programmer's viewpoint) or only from a "problems addressed" perspective is not enough. The decision makers must dig into the details to see the big picture.

Appendix B. What Communication Services Are Middleware Domain Service: The OSI and TCP/IP Models

Virginia's EWTA includes a Network Domain and a Middleware Domain. The Open Systems Interconnect (OSI) reference model and the TCP/IP protocol stack provide vehicles for explaining what network communications services are covered in the Virginia's network domain and what communication services are covered in its middleware or systems management domains. A particular middleware product may or may not cover all of the services that are to be discussed here.

The OSI Model is a seven-layer model used to describe what may take place in a particular network communication. (Note: not all communications require all seven layers of functionality.) The network domain team and the middleware domain team have mutually decided that functions in OSI reference layers 1-4 will be addressed primarily by the network domain team and functions in layers 5-7[9], primarily by the middleware domain team. A brief explanation will help.

Figure A provides a picture of client server communications using the OSI reference model. Within a client server communication such as "the client mail application requests 'get new mail' from mail server application," the client performs steps 7 through 1, as needed, to *send the request* and the server performs the same steps in reverse order to *receive the request*. The original point of this division of communication tasks into layers was so that one vendor could provide the functions of one layer and another vendor could know how to interface with the neighboring layer and what tasks to perform at their layer. The model will never be fully implemented, especially for layers 5-7. Never the less, the model does provide a useful vehicle for drawing functional lines in an enterprise architecture.

A couple of points are important for understanding the role of middleware in Virginia's architecture:

1. First, the Virginia's network domain architecture recommends a standard interface between network layers and middleware layers. The interface protocols are TCP/IP. Figure B demonstrates the differences between OSI and TCP/IP.
2. Second, OSI layers 5-7 tend to be implemented as functional stacks rather than as layers. The TCP/IP stack shows only one layer above the transport layer. This is a more accurate picture of how high-level communication functions are actually implemented. Examples of integrated high-level functions are security functions for digital signatures and email functions.
3. Third, there are exceptions to the division between Virginia's network and middleware layer. These exceptions are most likely to surface when vendors are providing a workaround for a proprietary protocol. The middleware solution may use "tunneling" or hiding of one protocol inside another (layers 2 and 3) or may employ a different transport protocol (layer 4).

[9] In actuality, Virginia is recommending that the TCP/IP (5 layer) protocol stack be implemented as network standard with directory services and security services following open protocols. Where middleware is concerned, packetizing belongs to the network domain and higher functionality to the middleware domain and the systems management domain.

4. Fourth, some services performed by middleware products are not communication functions, but are instead environment management and integration functions such as providing a GUI interface for building the workflow components, directory services, etc.

Figure A: OSI Seven Layer Model Showing a Communications Flow

Client Application (E.g., Sender)		**Server Application (E.g., Receiver)**
OSI 7 Application **……?**		OSI 7 Application …… ?
OSI 6 Presentation **……?**		OSI 6 Presentation…… ?
OSI 5 Session**……?**		OSI 5 Session…… ?
standard interface		*Standard interface*
OSI 4 Transport **……?**		OSI 4 Transport…… ?
OSI 3 Network **……?**		OSI 3 Network…… ?
OSI 2 Data Link**……?**		OSI 2 Data Link …… ?
OSI 1 Physical**……?**	Signal over medium ……?	OSI 1 Physical…… ?

The Network Layers in Brief

The four OSI layers that define networking are as follows. Layer 1 is the specifications for the physical layer (e.g., network wiring or other media). Note that the physical layer could be copper or fiber or air (wireless) and still communicate with the data link layer, which is layer 2. Ethernet methods of making sure that only one communication is taking place at a time on the physical medium is an example of this layer. The 3rd layer is called the network layer. One thing it deals with is protocols like IP addressing to locate a client or a server when more than one network is involved. The 4th layer is the transport layer and it uses protocols like TCP to packetize a communication and to ensure the packets are transported properly. The important thing to note here is that network layers are rarely used to provide business application related services. They move communications from one place to another.

Figure B: OSI and TCP/IP views of Network Communications

OSI 7 Application (FTP, HTTP)	**OSI vs. TCP/IP**	Process (FTP, Telnet, RPC, encryption, etc.)
OSI 6 Presentation (ASCII, XML, encryption)		
OSI 5 Session (Sun RPC, DCE RPC, IIOP, RMI)		
OSI 4 Transport (TCP, UDP)		Transport (TCP, UDP)
OSI 3 Network (IP)		Network (IP)
OSI 2 Data Link (MAC address)		Physical (MAC address, signaling, media)
OSI 1 Physical (signaling, media)		

The Middleware Layers in Brief

The middleware layers in the OSI model are called session (layer 5), presentation (layer 6), and application (layer 7). An example of a session service is providing a one-way communication or tracking a two-way transmission. Presentation services include encryption for security, language translations (e.g., from ASCII to EBCDIC), and compression/decompression so that the application programs and operating systems will understand the communication when it is received. The application layer is not the business application program (e.g., email read and write program), but rather, the service that knows when an application wants to communicate with the network (e.g., that the email program is issuing a "file send" command).

What is important to note is that the middleware layers deal with *business functionality* related to transmissions between network clients and servers (e.g., delivering email or accepting an encrypted password to authorize access). The application program has to ask for the middleware layer functionality but the programmer should not have to know how it is provided. Good middleware products should shield the programmer from these complex details.

Appendix C. Domain Team Working Documents

Domain Technology and Business Trends

Middleware Domain Architecture is affected by the following technology and business trends:

•XML is essential to middleware. This trend, in particular, requires a significant amount of cooperation between and among entities to develop standards and agreed-upon definitions. It also requires a great deal of flexibility between and among entities.

•The increasing emphasis on portals for "one-stop shopping" or self-service over the Web by business and the public. This is manifested in the continuous growth of citizen access to services and information over the Web, as well as the increased use of the Web for business-to-business and business-to-government transactions.

•The tension that exists between the requirement that middleware must be both simple and complex

•The assumption by end users that all systems can communicate.

•The heightened expectations about what middleware can accomplish.

•The unsettled environment in which middleware currently exists. For now, it is one of the latest trends (i.e., buzzword) sweeping the computing industry.

•The limited resources that requires entities to employ legacy systems as long as possible.

•The increasing emphasis on asynchronous communication.

The Enterprise Architecture Common Requirements Vision also identifies the following technology trends that apply to the middleware domain:

TT01. Widespread Access to Internet by Citizens. The availability and acceptance of moderately priced computers, coupled with rapid growth in ISP availability in almost all locations, has led to an increasing number of citizens who use the Internet. The emergence of Web-TV and similar devices holds the promise of even greater Web access by the majority of citizens in the next few years. Consequently, governments that Web-enable their services will be positioned to meet the accessibility demands of their citizens. At the same time, governments

must continue to recognize that not all citizens will have access to the Internet. The requirement for other forms of government service delivery will continue.

TT02. Internet and Intranets as Dominant Communications Vehicles. The Internet and Intranets have become the dominant communications vehicle for publishing information and for conducting business in both the public and private sector. This trend will make business solutions that are not compatible with the Internet obsolete. "Internet only" solutions for delivering public services, however, will not completely replace other service delivery mechanisms because many citizens, who are most in need of government services, do not have Internet access.

TT03. Requirement for Secure Transactions Across the Network. As more business-to-business and business-to-customer transactions are processed over the Internet, there is a widespread realization that security is vital. Technologies like encryption, Public Key Infrastructure (PKI) and digital signatures are becoming increasingly viable and reflect the growing demand for absolute authentication, privacy & access control. This growing demand for pervasive security will drive rapid advances in security technologies and require significant investment by the public sector in proven security systems.

TT04. Network Centric Computing. The need to share information and work cooperatively, regardless of time and distance, is causing electronic document handling, electronic commerce, automated workflow and collaborative computing to continue to increase dramatically. Increased use of electronic work processes will escalate the demand for network connectivity and communications bandwidth. This trend elevates the importance of networks and makes them critical factors in the success of business processes.

TT05. Electronic Commerce Expectation of Business Partners. Businesses are discovering that electronic commerce is a viable and productive way to transact business across supply chains. Many businesses now require their partners to transact business electronically. Consequently, state and local governments, which are not positioned to conduct electronic business, will find themselves and their citizens at a significant, and potentially costly, disadvantage. The Commonwealth E-Government initiative was created in response to this trend.

TT06. Emergence of Web Browser as Client of Choice. The Web browser has become a common element on all workstations. Deployment of applications using a Web browser as the client is widely acknowledged to be an efficient choice. This approach mitigates problems inherent in client software distribution and synchronization. Web browser enabled applications will become an increasingly dominant method of delivering information, services, and software.

TT09.Enterprise Servers. Just a few years ago, conventional wisdom was that large systems would soon be extinct, to be replaced by networks of small computers. However, mainframe systems have rapidly evolved into enterprise servers with almost unlimited scalability combined with robust management tools, open protocol support, excellent security and high availability. An example is the OS/390 Enterprise Server, which evolved from the monolithic MVS system.

Improving price/performance ratios have resulted in enterprise servers becoming the lowest total cost of ownership (TCO) option for many large applications and shared systems. While no single technology choice is the right solution for all needs, this trend will drive continued centralization of computing and data storage resources, and make the central provision of applications as services more attractive.

TT11. Convergence of Multi-Media Applications and Networks. Innovations in technology are creating the opportunity to transmit voice, video and data over the same network. For example, video is becoming a common element in many applications. The broadcasting of video, interactive video conferencing, and even conducting client interactions via video is proving to have significant business value. This trend will continue to drive increasing bandwidth requirements and shape both new applications and future networks.

TT13. Enterprise Portals. Large, complex organizations, like state government, are increasingly moving away from a multiple Web site approach for providing services to a "Web portal" strategy, which provides a single gateway to services across the enterprise. Web portals are essentially Web sites that provide various types of services in an integrated format. Often, users of a portal can create personal profiles that allow customized views of the information and services available. The use of portals at the enterprise level can benefit both service providers, and the customer, through the leverage of investment in IT resources, and having a single point of entry for all services provided by the enterprise.

TT14. Mobile Connectivity. The demand for mobile connectivity and information access is increasing in response to increased use of laptop computers, Personal Digital Assistants (PDA), Web-enabled cellular telephones and similar devices. This demand will drive significant advances in wireless technology and a corresponding need for applications that use next-generation mobile devices.

TT15. Increasing Use of Data Warehouse Technologies. The need to accelerate decision-making causes organizations to place lines-of-business operational data into data warehouses that provide enterprise-wide views of information. Powerful graphical user interface (GUI) analysis tools in the hands of decision-makers provide them with immediate access to critical information. This type of processing is being referred to as on-line analytical processing (OLAP) and requires database structures that allow dynamic definition of data relationships. Consequently, large organizations will increasingly deploy data warehousing technologies as separate but complementary to online transaction processing (OLTP) systems.

TT17. Customized Service Delivery. As technology changes more rapidly, the business requirements and expectations of users also change rapidly. Technology solutions that are not designed for adaptability can quickly become obsolete, often before they are fully implemented. This trend makes adaptability to change a vital measure of the value of any proposed technology solution.

TT18. Continued Growth of Online Transaction Processing Systems. There continues to be very strong growth in high volume online transaction processing (OLTP) systems, driven by the equally explosive growth of electronic business activity. These systems often require large-scale processing and data storage, with emphasis on fast transaction processing and positive commitment of transactions in real time. This trend will continue to drive the demand for large systems and high bandwidth networks.

TT19. Standardization of Desktop Workstations. The present standard desktop is the Web-enabled computer with resident applications or client software. Rapid advances in technology are requiring regular refresh of desktop workstations. Costs of support for this desktop environment are increasing. Initiatives such as Seat Management and thin client architectures are providing cost effective solutions that bring current technology to the desktop. Enterprise level implementation of standard desktop configurations, coupled with regular technology refresh strategies, will increasingly be employed to gain technical and economic advantage. (COTS Position: Standard desktop configurations are better implemented at an individual organization, not enterprise, level.)

TT21. N-Tier Computing. The three-tier client server model proved difficult to manage and expensive to implement and maintain. "N-tier" computing models are rapidly replacing this model. N-tier implementations logically separate the database, the application code, the presentation server and the client processing. The number of physical systems involved can be two or more, with multiple parts of the application residing on the same system in many cases. The client of choice is the Web-enabled computer with no application software installation required on the client other than plug-ins or helper apps that are downloaded as needed through a Web browser. The result of this trend is the rapid obsolescence of monolithic applications and older two and three-tier client server architectures.

Enterprise Business Strategies Supported

The Middleware Domain Architecture supports the following Enterprise Business Strategies:

EBS06. Identify and Encourage Improved Service Delivery Mechanisms. Improving government service to citizens is contingent upon improved service delivery mechanisms. The Commonwealth must identify and encourage the use, through appropriate incentives, of the most effective and efficient service delivery mechanisms. As an example, government services provided over the Internet may potentially provide better service at reduced cost when compared to other possible service delivery approaches.

EBS09. **Improve Procurement of Goods and Services.** In order to address the rapid changes in government service delivery demanded by the citizens of the Commonwealth, the resources necessary to implement change must be readily available. State government requires a procurement system that is responsive to rapid change and focused on best value. Movement to an automated procurement process that is fully integrated across state government, while placing increased demands on current infrastructure, will better enable technical innovation that supports improved service delivery to the citizens of the Commonwealth.

EBS10. **Insure IT Interoperability.** The seamless delivery of government services to the citizen implies both the vertical (within an organization) and horizontal (across organizations) integration of information. A cornerstone of integrated information and service delivery is IT interoperability. The Commonwealth must establish interoperability as a prerequisite for the development of technical infrastructure.

EBS12. **Promote Collaboration and Cooperative Systems Development.** Many organizations across state government are engaged in similar, and in some cases identical, core business activities. Promoting cooperative systems development provides the Commonwealth an excellent opportunity to meet common business needs while simultaneously building integrated technical architecture. Cooperative development strategies facilitate the leveraging of resources across government boundaries and the deployment of common service delivery mechanisms

Domain Requirements for Technical Architecture

Middleware Domain Architecture supports the following Requirements for Technical Architecture:

RTA03 The Enterprise Architecture must provide mechanisms to determine and adapt to the service delivery preferences of customers.

RTA04 The Enterprise Architecture must support implementation of multiple service delivery channels for the same service utilizing common underlying information and systems to enable rapid response to changes in business requirements.

RTA05 The Enterprise Architecture must provide the capability to locate and present information seamlessly based on the requestor's needs and context without requiring the requestor to know in advance the source or location of the information.

RTA06 The Enterprise Architecture must support delivery of latest relevant information.

RTA09 The Enterprise Architecture must support mechanisms to detect and resolve data discrepancies, incomplete data and incorrect data.

RTA12 The Enterprise Architecture must provide infrastructure that facilitates collection, analysis and sharing of recruitment data, retention data, and workforce availability information across all levels and branches of government.

RTA13 The Enterprise Architecture must protect the confidentiality and integrity of data being stored or transmitted.

RTA16 The Enterprise Architecture must enable flexible sharing of service delivery channels to provide seamless customer service.

RTA19 The Enterprise Architecture must provide a flexible and scaleable infrastructure to support rapid fluctuations in demand.

RTA22 The Enterprise Architecture must enable collection, analysis and sharing of procurement performance information to support a well managed and auditable procurement process.

RTA23 The Enterprise Architecture must support flexible implementation based on industry standards consistent with mainstream trends.

RTA24 The Enterprise Architecture must support multiple sets of standards to ensure interoperability.

RTA27 The Enterprise Architecture must enable deployment of common applications in both centralized and decentralized implementations as appropriate.

RTA28 The Enterprise Architecture must facilitate implementation of a high capacity and high availability technology infrastructure in all parts of the Commonwealth, in cooperation with business and industry that will attract businesses to the Commonwealth and promote widespread economic growth.

RTA29 The Enterprise Architecture must enable strategic prototyping of new technologies and rapid deployment of technologies and service delivery mechanisms determined to be effective, stable, and appropriate.

Appendix D. Reference and Links

State Sites:

Connecticut Middleware Domain Architecture
http://www.doit.state.ct.us/policy/mdlware.pdf

North Carolina Application Middleware Architecture:
http://irm.state.nc.us/techarch/chaps/pdffiles/chap6.pdf

Ohio Middleware Infrastructure Architecture:
http://www.state.oh.us/das/dcs/opp/pdfs_other/MidwareReq.pdf

Federal Links:

Links: Federal Information Technology Architecture
http://www.itpolicy.gsa.gov/mke/archplus/federal.htm

General Middleware References

GartnerGroup:

Application Integration Middleware Market, R-11-5113, 1-September 2000

Middleware: The Glue for Modern Applications, R-08-2601, 26 July 1999

Middleware Deployment Trends: Survey of Real-World Enterprise Applications, R-07-4976, 2 April 1999

META Group:

X-EAI Standards: Part1-Waiting for Esperanto or Dealing with the Tower of Babel? File: 779 28 September 1999

Open Computing is Dead – Long Live Open Computing. File: 780, 28 September 1999

Host Data-Sharing Framework: Part1-Overview. File 768, 2 August 1999

The Challenge of Adaptive EAI Infrastructures. File 762, 28 July 1999

Microsoft.Net: Too Important to Ignore! File 865, 21 September 2000

From Application Server to Application Integrator: Building the Application Services Layer. File 774, 13 September 1999

Other Sources:

Middleware Vendor Database: Middleware Spectra, an independent resource on business integration and network computing through middleware and message brokering.
http://www.middlewarespectra.com/abstracts/vendordb.htm

Issue No. 12 of "The Enterprise Newsletter" (TEN), published by Clive Finkelstein.
http://members.ozemail.com.au/~ieinfo/ten12.htm

Enterprise Wide Information Technology Architecture (EWITA) links and resources
http://www.ewita.com/

The Once and Future System: The Emergence of Middleware. By Colin Currie
http://www.bcsolutionsmag.com/Archives2/march99/Middleware.html

Advanced middleware services by G. Schussel
http://news.dci.com/geos/csglue.htm

Middleware - The Essential Component for Enterprise Client/Server Applications.
http://www.isg-inc.com/goodies.htm prepared by International Systems Group, Inc.

Special Topics

XML and SOAP

XML Database Products, by Robert Bourret
http://www.rpbourret.com/xml/XMLDatabaseProds.htm

XML Overview
http://www.w3.org/XML/1999/XML-in-10-points

A Busy Developer's Guide to SOAP 1.1. By Dave Winer and Jake Savin (UserLand Software). March 28, 2001.

http://www.xmlrpc.com/aBusyDevelopersGuideToSoap11

SOAP won't clean middleware's messy reality, By James Kobielus, Network World, 05/22/2000
http://www.nwfusion.com/columnists/2000/0522kobielus.html

Object Models

Cataloging CORBA's Problems: A Short List
http://www.chappellassoc.com/artcorba.htm

Comparison of DCOM/COM and CORBA
http://www.quoininc.com/quoininc/COM_CORBA.html

Comparison of CORBA, COM/DCOM and Java/RMI
http://www.execpc.com/~gopalan/misc/compare.html

E-Business

High-Performance Technologies for the Second Wave of E-Business. By Tony Rybczynski
http://www.wsta.org/publications/articles/1000_article04.html

Appendix E: Quick Reference

	Scope	Strategic Issues	Applications/ Services	Protocol/ Standard Examples
Database Middleware	Communication w/ 1or more local/remote database servers. Does not transfer calls or objects.			
• DB directory services	Database of lists	Involved distributed architecture or middleware strategy Interoperability, location transparency, lower maintenance costs	Email, security, naming services	LDAP, X500, DNS, GDS
• DB Metadata services	Data about data, descriptive data to help make sense of data	Descriptions of data, how used, who uses, what constraints there are, where it originated Policies required for it use	Dependent on applications	Joint metadata model (OMG/MDC), XML containers
• DB access services	Connectors from application to Database	Dependent on applications services used	ODBC Database Gateways/ Adapters	ODBC, JDBC, SQL, OLE-DB
Message Middleware	Message-oriented middleware provide an interface between applications			
• Message formats	Requests/replies from applications and databases	Format not important ASCII format evolving XML data content strategic		XML primarily
• Message transfers	File copy, emulation, translation, file transfer, hypertext	Decision on application need (simple to complex)	Various file copy, emulation and transfer methods.	FTP, or general transfer. Proprietary and open methods. HTTP.
• Message-oriented middleware	Manage message distribution, receipt confirmation, error handling of messages	Store-and-forward messaging Publish/subscribe Event registry svcs Intelligent routing		No defined protocols due to encouragment of proprietary and open standardized message formats
• Message middleware guidelines	Synchronous, Deferred, one-way and asynchronous	Guidelines as defined by practice		ASCII,EDI,XML

	Scope	Strategic Issues	Applications/ Services	Protocol/ Standard Examples
Transaction Processing Middle ware	Ensures transaction integrity for distributed transaction environments			
• Transaction middleware types	Two-phase commits Failure/recovery Synchronized systems Scheduling	Selection based on need and infrastructure		No specific controlling protocols
• Transaction middleware guidelines	Requirement of only if environment needs transactional integrity			X/A as standard.
Application Integration Middleware	Providing interfaces to applications from thin client to full integrated user interface			
• Methods to integrate applications	Application Programming Interfaces, Remote Procedure Calls, Object Interfaces, Object Request Brokers, Simple Open Access Protocol	Observe differences that may require interoperability and application redesign		Various API's, RPC implementation standards by developer, COM/COM+/DNA, ORBs by developer, CORBA.
• Application Integration Management Services	Conversion services, legacy wrapper for client-host access, data mapping and triggers	Selection based on need and access to local, legacy or other data requiring specific interconnection and interoperability needs		Services and methods of implementation differ by developer
• Application Integration Middleware Examples	Examples that provide context to using application integration middleware	Address the work to expedite distributed, non-distributed business processes in real-business needs		None.
• Application Integration Middleware Guidelines		Strategic and emerging trends for implementations		XML for RPC, SOAP, TCP/IP, IIOP, Portable Object Adapter, DCOM/ COBRA/RMI
Super-service Middleware	Includes the collection, management and integration of multiple types of middleware			
• Value-added services	All value-added services for super-service middleware	Determination of strategy for using value-added service under super-service		
• Super-service guidelines	Guidelines for procurement and utilization of such services	Centralization, maximum reuse, purchase – do not buy, research super-services that serve your customer's needs		