IT ARCHITECTURE

CONTENTS

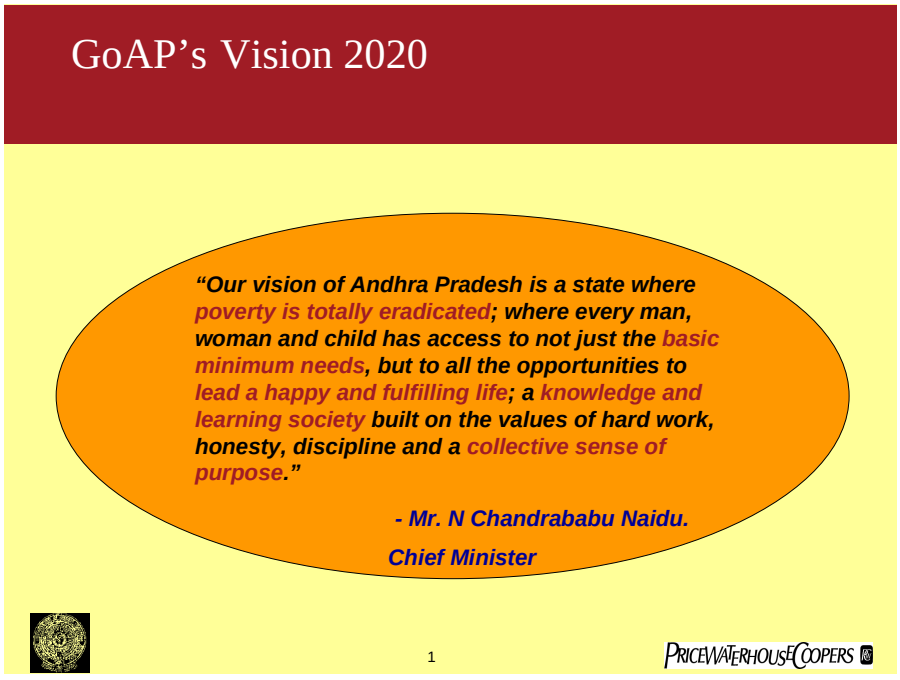
Executive Summary

I. Background

Government of Andhra Pradesh (GoAP) has undertaken multiple initiatives using Information Technology as a strategic tool to provide improved service to citizens. The pilot projects FAST, CARD, MPHS & TWINS have been successfully piloted and widely appreciated. Capitalising on the success of these pilot projects the government departments were keen to go ahead and rollout the solutions to other locations. The Information Technology & Communications (IT & C) department wanted to develop an overall architecture for the state along with standards and policies for key technology components, so that applications developed are interoperable.

II. Vision and Strategy

Government of Andhra Pradesh (GoAP) is a progressive state, which has set itself aggressive targets(Vision 2020) to bring prosperity and well being of its people. In order to achieve this level of development the state has embarked on various efforts to bring about economic growth. Information Technology (IT) is one of the major contributors for achieving this vision.



GoAP envisions to improve its services to citizens and businessmen by providing Simple, Morale, Actionable, Responsive and Transparent (SMART) governance. The government foresees that services to citizens will be made available through various delivery channels (department interfaces, integrated citizen interface counter, Internet, kiosks, etc). Some key benefits that can be achieved through e-Governance are :

- ✓ Improved quality of service to citizens
- ✓ Improved efficiencies within the government
- ✓ Better enforcement of law
- ✓ Education and information dissemination

Key imperatives that will enable the vision to be achieved are:

- ✓ Accessibility of services to all sections of society
- ✓ Affordability of services by citizens and business
- ✓ Information accessed through various delivery channels must be uniform, reliable & secure
- ✓ The services offered through various delivery channels should be economically viable to develop, operate and maintain
- ✓ Processes need to be re-engineered before automation
- ✓ Departments need to collaborate and share information to provide improved services

Information technology can be used effectively to achieve the vision in light of the business imperatives. Key IT imperatives for e-Governance are:

- ✓ Adoption of standards and policies for various technology components
- ✓ Shared database across multiple departmental applications
- ✓ High reliability on hardware and communication
- ✓ Training department employees on various technologies
- ✓ Increased IT investments across multiple levels of governance (Mandal, division, district, etc)

III. Overall Architecture

The overall architecture for e-Governance needs to ensure that the architecture components are extensible and scalable to adapt to the changing environments. In order to minimise the risk, there is an increasing trend where the applications development is moving towards n-tier architecture. Communication cost will continue to fall in the years to come with convergence of technologies. The cost of skilled technical resources will continue to increase in the future.

The architecture options are decentralised architecture, hybrid architecture and centralised architecture. It is recommended that the state government architect its solutions around centralised architecture. The government employees will access the applications over a secure Intranet using browser as their front end. Citizens and businesses can access the services over internet. This target architecture would achieved in phased manner based on availability of infrastructure such as communications, PKI, payment system, etc.

IV. Building blocks of e-Governance

The GoAP Architecture is comprised of a series of interrelated components. The twelve components of the architecture, in concert, provide the basis for the state to take advantage of adaptive systems in support of its business.

The twelve architecture components are:

- **Application**
- **Information**
- **Group ware**
- **Component ware**
- **Data**
- **Applications Middle ware**
- **Integration**
- **Network**
- **Platform**
- **Security and Directory Services**
- **Systems Management**

For each of the architecture components, key guiding principles, best practices and standards are outlined below.

Application Architecture

I. Definition

Application Architecture identifies criteria and techniques associated with the design of applications for the state's distributed computing environment that can be easily modified to respond quickly to the state's changing business needs, as well as to the rapidly evolving information technologies available to support those needs.

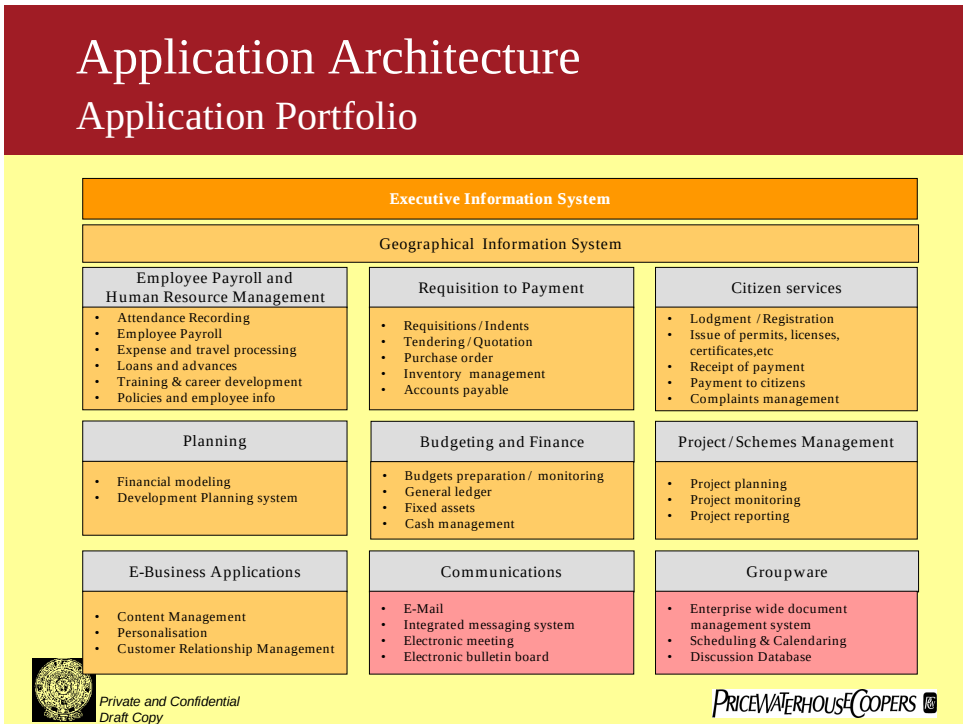
II. Introduction and Background

Recently, application development tools and technology have begun to evolve to help address these problems. A number of options now exist for meeting business needs and delivering information to people when and where they need it. These options include:

- Reuse of Code: Units of code previously duplicated in many applications can be packaged into components or services that can be easily reused by different applications.
- Middleware: Shared software can allow applications to communicate with each other, access data residing on different platforms, and access shared services.
- New User Interface Options: An expanding array of user interface options – including Web browsers, personal digital assistants (PDAs), and interactive voice response units (IVRs) - have been introduced.

Implementing these components in well designed 3-tier or N-tier, client/server application architectures can create solutions capable of satisfying the state's ever changing business needs.

Portfolio of applications for a government department is outlined below:



III. Principles

The principles listed below are key guidelines for the design or purchase of applications and application components supporting distributed, client/server computing across the state.

Principle 1: Design applications to be highly granular and loosely coupled

- The designer should allow for the possibility of re-partitioning an application in the future.
- Being highly granular and loosely coupled provide flexibility in physical implementation (i.e., in the deployment of application components on different platforms).
- Highly granular, reusable application components are key to increased productivity and rapid application deployment. The Componentware Architecture supports a highly granular design.

Principle 2: Plan for extensibility and scalability

- Applications must evolve to support new business requirements and make use of new technologies.
- Extensibility provides functional scalability.

Principle 3: Design application to reuse components

- Applications should be built by assembling and integrating existing components, rather than by creating custom code. Shrinking cycle times do not allow for artisan programming.
- Managing component reuse is supported by the Componentware Architecture.

IV. Technical Topics

Technical Topic 1: Designing and Developing Applications

Introduction

All computer applications - regardless of what they do or with which technology they are implemented - have three general areas of functionality:

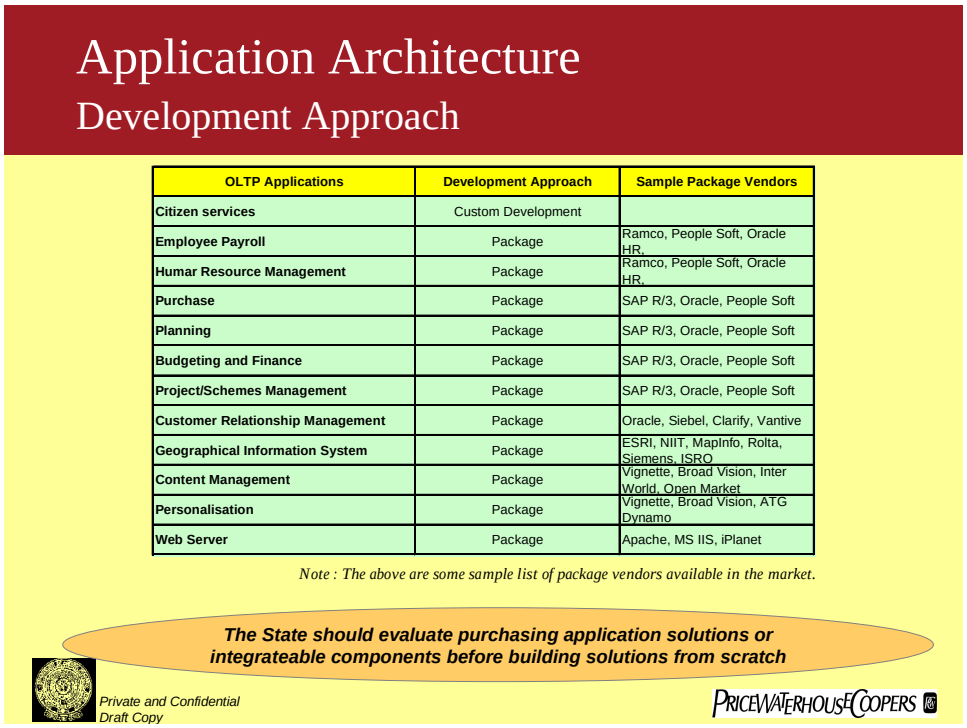
1. Interfaces allow applications to communicate with users as well as with other applications and data resources. In N-tier applications changes in business rules do not usually require changes in interface code. Interfaces may need to be updated for other reasons. Examples include when changes occur in another computer system that interfaces with that application, or when users need a graphical user interface instead of a character-based interface for that application.

Since applications interface with people, the user interface receives the most attention, but other interfaces are equally important. Traditionally, people interfaced with computer applications using character terminals or graphical user interfaces (e.g., Microsoft Windows). Recently new interfaces such as telephones (via IVRs), web browsers, and wireless devices have been introduced.

2. Business rules support the business processes Departments follow. They automate the process, defining what must be done and how it must be done. As the business processes of Departments change, the business rules in the applications that support those Departments must also be changed. Business rules can be isolated into components. (See Componentware Architecture)

3. Data access. Data access code automates the storing, searching, and retrieving of data by computer applications. In N-tier applications, changes in business rules may not require changes to the code that accesses data, but occasionally, they do.

Development approach for the portfolio of applications is as follows:



Recommended Best Practices

The recommended best practices in this section apply to the designing and developing of applications.

Best Practice 1: Design for the N-tier service oriented architecture.

- While many of the problems inherent in the monolithic and two-tier applications can be overcome by implementing applications with a three-tier architecture, large, complex projects that are anticipated to have high usage volumes and/or long life spans will be better served by an N-tier service oriented architecture.
- N-tier applications are easily modified to support changes in business rules.
- N-tier applications are highly scaleable.
- An N-tier architecture offers the best performance of any application architecture.
- Any combination of user interfaces (e.g., character, graphical, web browser, and telephone interfaces) may be implemented in an N-tier application.
- N-tier applications are less expensive to build and maintain because much of the code is pre-built and shared by other applications (see Componentware Architecture).

Best Practice 2: Generalize application interfaces

- Generalize application interfaces.

- The code providing input and output to the user interface should be designed to provide input and output to as wide a range of interfaces as possible. This should include other applications as well as other types of user interfaces.
- Do not assume that application components will always be accessed via a graphical user interface (or any other user interface).
- Avoid assuming a specific page size, page format, layout language or user language whenever possible.

Best Practice 3: Assign responsibility for business rules to business units

- Assign responsibility for defining and maintaining the integrity of business rules to business units.
- IT staff is responsible for coding and administering the software that implements business rules in the network.
- The business units are responsible for the definition and integrity of business rules, and for communicating changes in business rules to IT.
- Every business rule should be assigned to a custodian.

Best Practice 4: Make business rules platform-neutral

- Implement business rules in a non-proprietary, cross-platform language.
- This approach provides platform independence and portability.

Best Practice 5: Implement business rules as discrete components

- Implement business rules as discrete executable components or services (see Componentware Architecture).

Best Practice 6: Access data through business rules

- Design applications so business rules control access to data.
- Data is created and used by business processes. In computer applications, data must be created, used by, and managed by the application component that automates the business process.
- Accessing data in any way other than by business processes bypasses the rules of the module that controls the data. Data is not manages consistently if multiple processes or users access it.
- Centralised data should be used wherever possible to assure data accuracy and simplify data management.

Best Practice 7: Adopt coding standards

- Adopt coding standards, in all languages, on all platforms.

Coding standards make debugging and maintenance easier. They should address (but not be limited to):

- Naming conventions for variables, constants, data types, procedures and functions.
- Code flow and indentation.
- Error and exception detection and handling.
- Source code organization, including the use of libraries and include files.
- Source code documentation and comments.
- Even the earliest code developed in a project should adhere to the standards.

Standards

The standards in this section pertain to the design and development of applications.

Standard 1: Develop 3-tier or N-tier Applications

- All new Department applications should be developed using 3-tier or N-tier architecture in order to maximize flexibility and scalability.
- Large, complex projects that have high usage volumes and/or long life spans will be better served by an N-tier service oriented architecture.
- The logical separation of the tiers for: user interface(s); business rules; and data access code allows for simple, straightforward additions to each of the three tiers without undue impacts on the others.
- The logical separation of the tiers also allows for changing the platforms where the tiers are deployed, resulting in a high degree of scalability. As transaction loads, response times, or throughputs change, a tier can be moved from the platform on which it executes to another, more powerful platform - or be spread over multiple machines - without impacting the other tiers.
- While many of the problems inherent in the existing monolithic and two-tier applications can be overcome by implementing applications with a three-tier architecture, the goal should always be true, N-tier applications.
- The maximum benefits of an N-tier architecture are realized when many N-tier applications are deployed across the state, sharing common software services and offering multiple user interfaces.
- Large, complex projects that are anticipated to have high usage volumes and/or long life spans will be better served by implementing applications with a three-tier architecture with access to an N-tier service oriented architecture.
- With three-tier client/server applications, there is less risk in modifying the code that implements any given business rule.
- Three-tier client/server applications can be made to support multiple user interfaces: character, graphical, web browser, telephones, and others.

Standard 2: Isolate Customizations to Purchased Software

- Isolate customizations into separate modules from the purchased software itself to improve the ability to upgrade and move to new releases as required over time. For purchased line-of-business applications, loosely couple custom developed modules from the standard application software.

Standard 3: Avoid Common Gateway Interface (CGI) for business logic or to publish information to the Web.

- The Common Gateway Interface (CGI) does not scale, is not portable and is not easily integrated with application servers. Avoid use of CGI for information publishing, back-end applications or data access.
- Publishing information to the web with HTML or XML via Java servlets reduces overhead and works in conjunction with EJB-based components.
- The use of ASP or other HTML publishing is acceptable for publishing only (not business logic) but JSP and Servlets are preferred.

Technical Topic 2: Managing Applications

Introduction

Due to the state's dependency on computer applications, applications must be managed as carefully as any other business-support infrastructure. *Application management is a necessity, not an option.* Application management requirements are as important to the enterprise as an application's functional requirements. Therefore, management requirements for an application should be documented during the requirements phase of the project.

Recommended Best Practices

The recommended best practices in the section pertain to managing applications.

Best Practice 1: Design for end-to-end management.

- Manage the application as a whole entity by managing every component of the application and everything each component depends on. Application developers must instrument every component of the application to facilitate its management.
- Application dependencies include infrastructure (e.g., middleware, databases, and networks), other applications, and shared software components. Application teams must specify these dependencies when an application is deployed.
- Application reporting should be standards based and must be compatible with the state's SNM tool.

Best Practice 2: Design for proactive – rather than reactive – application management.

- Proactive application management supports the business better. With proactive management, applications report potential problem conditions at predefined thresholds, before errors occur. This gives system administrators the opportunity to take corrective action to prevent an application from failing.
- While applications can be managed by administrators responding to errors, the ideal management is automatically undertaken by the SNM tools, and is proactive.
- Use thresholds to provide early alert to possible error conditions. For example, rather than sending an alarm when an application fails because its database table is full, send an alarm when the table is 90% full, so corrective action can be taken to prevent a business-impacting outage.
- Reactive application management is better than no application management at all. Reactive management is when administrators respond to errors and outages reported by applications after they have occurred

Best Practice 3: Instrument applications to report the information necessary to manage them.

- Applications should report status, performance statistics, errors, and conditions. Decide at design time what status events the application should report to users (e.g., erroneous input); to application managers (e.g., database table 90% full); and to both (e.g., can't find needed file).
- Operations staff must be provided procedures for dealing with all conditions that are detected and reported. For example, if an application reports it can no longer access its database, operations staff must have instructions for handling the situation.
- At design time, decide the specific reporting requirements of an application module. Different applications may have different management needs, depending on their respective impact on the business the applications support.
- Applications should only report. Interpreting the reports and deciding on the appropriate response should be performed external to the application, by agents and the SNM framework.
- Application reporting should include run-time tracing to assist trouble-shooting operational problems. Tracing should be able to be turned on and off by administrators.
- If no SNM environment exists, applications should still report status to local log files that can be monitored by administrators. Applications should still be able to read and respond to commands from administrators.

Best Practice 4: Instrument applications to facilitate administration

- Instrument applications to receive and process commands from administrators.
- Decide at design time what control operators and SNM tools should have over application components.
- Design applications to read and respond to commands from system administrators. Commands may include, but are not limited to, shut down, shutdown and restart, reconfigure yourself, and turn tracing on or off.
- Make application configurations parameter-driven, so applications can be reconfigured without recompiling and redistributing code.

Standards

The standards in this section pertain to managing applications.

Standard 1: All applications deployed must be designed to be managed by SNMP.

- By standardizing on SNMP as the instrumentation protocol, there is an opportunity for the state to benefit from reusing management instrumentation code.

Information Architecture

I. Definition

Information Architecture provides standards for accessing data for online analytical processing (OLAP), including executive information systems (EIS) and decision support systems (DSS).

II. Introduction and Background

Important data is stored in multiple application systems across the state and is used to perform day-to-day operations. If this distributed data is grouped together in a meaningful format, it can provide valuable information to the state's business organizations. Information is a compilation of data used for reporting, analysis, and decision support. Information compiled from data coming from many sources, including historical and summary information, can facilitate business decisions.

III. Principles

The following principles apply to the Information Architecture.

Principle 1: Information is one of the most valuable assets for making business decisions.

- The successful delivery of government services depends on conclusions derived from accurate, timely, well maintained, and secure information.
- The value of information is proportional to its availability. If information is not accessible or is too hard to use, it is of questionable value.
- Information used frequently or used by many organizations is extremely valuable.

Principle 2: In general, there is no new data, but there is new information. Existing data from multiple sources is being transformed into intelligent and proactive information.

- How the data is used is far more important than the data itself (i.e., even if data is accurate, it can still be used ineffectively).
- The most effective use of data is to turn it into proactive information that responds to business events (e.g., the "push model" of information).

Principle 3: Decision-makers should not be overwhelmed with an excessive volume of unnecessary information.

- Too much information gets in the way of focusing on the most important issues that arise at a specific point in time. Some data warehouse efforts put any data in a data warehouse that may be useful in the future, not just the information to assist in a business need. If there is much more information than is needed, it can overwhelm an end user rather than answering their specific decision support need.
- Information that supports OLAP analysis should be proactively presented to business users. Information in a data warehouse can be presented to business end users so that they know it is available and they can use it.

Principle 4: Online transaction processing (OLTP) databases and online analytical processing (OLAP) information databases should have separate data storage areas.

- Separate data storage isolates OLTP systems, which perform mission critical business processing, from large ad hoc queries and online analytical data processing. If data storage is not separate, ad hoc queries and direct access of data for OLAP systems can adversely impact online transactional processing.
- Data design for each type of application, OLTP or OLAP, can be optimized for performance. OLTP and OLAP may require different database designs. OLAP typically includes complex operations involving time series and trend analysis, which do not perform well with relational database technology alone (e.g., sometimes other methods of data storage are needed to support OLAP, such as multi-dimensional databases or flat files).
- For more information about OLTP data, refer to the Data Architecture chapter.

IV. Technical Topics

Technical Topic 1: Data Warehouse

Introduction

A data warehouse is a collection of data designed to support decision making and analytical processing. Data warehouses contain a wide variety of data, usually from multiple data sources, presenting a comprehensive view of a particular business environment. Due to the nature of the data stored in a data warehouse, the size of the data warehouse is usually very large, so it requires special design and planning.

A data mart is a subset of a data warehouse. Where data warehouses are designed to support many requirements for multiple business needs, data marts are designed to support specific requirements for specific decision support applications (i.e., particular business needs). Although a data mart is a subset of a data warehouse, it is not necessarily smaller than a data warehouse. Specific decision support needs may still require large amounts of data. Data marts are typically considered a solution for distributed users who want exclusive control of the information required for their business need.

Data warehouse efforts should begin with a specific requirement for a specific decision support application, similar to the practices of a data mart design. For scalability, the tools and databases

used should be designed to support a very large data warehouse, instead of using data mart specific products.

Recommended Best Practices

The recommended best practices in this section pertain to the implementation of a data warehouse.

Best Practice 1: Begin data warehouse efforts by addressing a specific requirement for a specific decision support application, keeping growth and scalability in mind.

- This practice is similar to data mart design, but the tools and databases used should be designed to support a large data warehouse.
- Use vendor supplied products designed to support a large data warehouse. Vendor-supplied data mart tools are not typically scaleable to support the migration from a data mart to a data warehouse solution. These tools are designed to quickly implement a specific solution.

Best Practice 2: Identify specific requirements for data availability, freshness (i.e., live, 24 hours old, etc.), and recoverability.

- Some data warehouses need to be updated more frequently than others. When the original data is frequently changing or is more volatile, it may be necessary to update the data warehouse on a near real time basis.
- On the other hand, if the original data is fairly stable and not as volatile, the data warehouse may only need daily, weekly, or even monthly updates. For example, a data warehouse that stores criminal data contains more volatile information and needs to be updated more frequently than a data warehouse that stores state registered corporation name and address information for public access.

Best Practice 3: Perform benchmarks on the database design before constructing the database.

- Expect to make changes and adjustments throughout development.
- Changes during the early cycles up to, and including implementation, are primary mechanisms of performance tuning.

Best Practice 4: Allow only read only access to end users of data warehouses.

- Updates should only occur to the operational (OLTP) source where the data originates.

Best Practice 5: Direct all information queries against decision support databases, not OLTP databases. Conversely, operational transactions should be directed to operational databases only, not OLAP databases.

- Data warehouses, and data marts contain data that has been checked for consistency and integrity, and represents a cross-functional view of data.
- Data in transaction (OLTP) systems typically support a specific business group or function.
- OLTP transactions should not depend on a data warehouse database. They require a stable operational environment that is not affected by ad hoc usage or external data.

Best Practice 6: Store atomic-level data in the data warehouse in addition to summary data.

- Atomic data is transaction-level data. It contains much more detail than summary data.
- Atomic-level data addresses the business need to recast history. Due to the fast pace of business change, many organizations are going through multiple reorganizations. After a reorganization, many decision makers want to recast history (e.g., to get a feel for what test scores *would have been like* if the number of school districts was already reduced to respond to legislation or funding).
- If only summary level historical data is kept in the data warehouse, it is not possible to recast history.

Best Practice 7: Perform periodic validity audits against the data warehouse information model to ensure a high level of confidence in the quality and integrity of the data.

- Accelerated decision making requires high quality data. If operational data has changed or additional data is needed, changes must be made in the information model and in the data warehouse itself.
- The data stored in a data warehouse should conform to the information model.
- The source data populating a data warehouse should be verified for consistency and accuracy.
- The data warehouse should still correspond to business needs.
- Ensuring the integrity and quality of data is the responsibility of both the business users and IS.

Technical Topic 2: Repository

Introduction

A repository contains detailed information about the data that is stored in a data warehouse (i.e., metadata). The repository stores the following information:

- Federated data definitions for the data stored in the data warehouse database.
- Aliases that can be used to reference the data.
- Data structures.
- Systems where the original data is found, including the format of the original data.
- Processes used to extract the data from the original location.
- Sources of record for the data. A source of record is an authoritative source for data, where data in a source of record is trusted to be accurate and up-to-date. The original data and the source of record may be the same.

If changes are made that may affect systems and users using the data, it is important to keep this type of information in the repository.

Recommended Best Practices

The recommended best practices in this section pertain to the selection, design, and maintenance of a repository.

Best Practice 1: Maintain a repository for every data warehouse.

- The repository contains metadata, or information about the data, in the data warehouse.
- The repository represents the shared understanding of the organization's data.
- The repository can be built incrementally, in stages, based on data warehouse design and implementation.

- The repository should support multiple types of data elements, such as graphics.
- Changes in the repository must occur before the changes to the data warehouse environment.

Technical Topic 3: Data Hygiene Tools

Introduction

Data hygiene is the process of the *data scrubbing* or *data cleansing* of data stored in a database. Data can become "dirty" due to many reasons. For example, consider a data entry application that has an open text field called "Description." If no limitations are placed on the entry of data in that field, end users can type anything in that field, including misspelled words or multiple text descriptions for the same data element.

Data hygiene:
- Standardizes and elementizes data according to specifically defined rules.
- Corrects data and eliminates redundancy to increase data query accuracy and improve the value of other forms of data analysis.
- Reduces the cost associated with inaccurate, incomplete, and redundant data.
- Reduces the risk of invalid decisions made against incorrect data.

Recommended Best Practices

The recommended best practices in this section pertain to data hygiene.

Best Practice 1: Use the data warehouse metadata repository to document the rules applying to data scrubbing.

- The information about how the data is to be scrubbed should be saved for historical purposes.

Best Practice 2: Ensure data entry quality is built into new and existing application systems to reduce the risk of inaccurate or misleading data in OLTP systems and to reduce the need for data hygiene.

- Provide well-designed data-entry services that are easy to use (e.g., a GUI front end with selection lists for standard data elements like text descriptions, product numbers, etc.).
- The services should also restrict the values of common elements to conform to data hygiene rules.
- The system should be designed to reject invalid data elements and to assist the end user in correcting the entry.
- All updates to an authoritative source OLTP database should occur using the business rules that own the data, not by direct access to the database.
- Attention to detail should be recognized and rewarded.

Best Practice 3: Move to Commercial off the Shelf data hygiene software.

- Over the past few years, data warehousing software products have become a commodity. Use of these existing technologies is recommended to ensure quality and stability. Often times these types of technologies are included with an ETL suite of tools.

Technical Topic 4: Data Extraction and Transformation Tools

Introduction

Data extraction and transformation are used to extract data from existing operational and external systems, transform the data, and put the transformed data in a data warehouse. Typically, data extraction is accomplished through custom-developed programs. They are usually written by application developers responsible for the existing operational systems that are familiar with the existing data required for a data warehouse. However, there are data extraction and transformation tools available from vendors that can be customized to address particular extraction and transformation needs.

Recommended Best Practices

The recommended best practices in this section pertain to data extraction and transformation.

Best Practice 1: During data warehouse design, determine the logic needed to convert the data, plan and generate the extraction and transformation routines, and quality assure the data populating the data warehouse.

- Planning for data extraction and transformation should start at the same time the data warehouse design starts.
- Data extraction and transformation is an important process for populating the data in a data warehouse and for ensuring that the data in a data warehouse is accurate.
- Data extraction and transformation logic includes data conversion requirements and the flow of data from the source operational database to the data warehouse.

Best Practice 2: Assess the source data that will populate a data warehouse for accuracy and quality.

- Data needs to be accurate to ensure good business decisions.
- Data needs to be relevant to the business need and consistent across multiple sources.
- Data must be complete. It must contain the information necessary to answer the data warehouse business need.
- The data assessment also involves evaluating the business rules associated with that data. The appropriate business rules must be applied to the data to maintain accuracy.

Best Practice 3: If a vendor-supplied extraction and transformation product is selected, it should support the same metadata repository that supports the data warehouse. It should also support the physical data warehouse.

- Select products that are capable of interacting with the metadata repository and the data warehouse.
- Metadata drives the operations of the extraction and transformation tools. If the data warehouse repository is not supported by a vendor-supplied extraction and transformation product, a separate metadata repository must be developed and maintained.

Technical Topic 5: Data Replication Tools

Introduction

Replication of data in a data warehouse environment is sometimes needed to address the business needs of distributed users or applications. In the state, there are users located in remote

offices that require data to be loaded in their local environment in order to meet their performance needs. Also, there are mobile users, who carry laptop PCs, who do not have a constant connection to the network. In both cases, when the distributed users are using a data warehouse application, replication is needed to propagate the data warehouse data to the remote systems.

Recommended Best Practices

The following recommended best practices apply to data replication for decision support, executive information, and OLAP systems.

Best Practice 1: Replicated data should be read-only, except where business practices clearly allow inconsistencies.

- It is easiest to manage data quality and integrity when replicated and distributed data is read only.
- Some business applications require updates to occur against the local database.
- Distributed independent updates require a reconciliation process that may be quite complex.

Technical Topic 6: Business Intelligence Tools

Introduction

Business intelligence tools provide the ability to analyze and access data contained in the data warehouse. Typically, several tools are selected within an organization, based on the function needed.

Recommended Best Practices

The following recommended best practices apply to business intelligence tools.

Best Practice 1: Implement decision support and executive information applications using an N-tier application architecture.

- By developing and application system in *N* tiers, systems can respond quickly to changes in business needs.
- Decision support and executive information systems application programs benefit from the use of N-tier and reusable and shared components.
- For more information about N-tier application programs and reusable components, refer to the Application Architecture and Componentware chapters.

Best Practice 2: There should be no *ad hoc* query access to OLTP databases.

Ad hoc query access to online operational databases can severely impact the performance of mission critical operations. A data warehouse should be implemented for users with ad hoc query needs.

Standards

Standard 1: When accessing relational databases, use the industry standard of ANSI Standard SQL.

When using a database access tool that uses SQL calls, do not use any vendor specific extensions.

Standard 2: Use ODBC from any data access programs rather than vendor-specific database access tools.

ODBC allows flexibility in programming. A database can be easily modified or relocated. If a change is needed, the change is made to the ODBC configuration file, not to each business intelligence program or tool.

Standard 3: Implement a server-based ODBC solution rather than a workstation-based ODBC implementation.

A server-based ODBC solution is easier to administer. ODBC database changes and additions are easier to manage, since updates are made to ODBC servers, not every workstation that uses ODBC.

Standard 4: Use domain name system (DNS) names for databases that are accessible via TCP/IP.

- A DNS server provides the capability for a long or complicated TCP/IP location to be accessed by a generic, short alphabetic name. It is basically a lookup service. It maps the generic alphabetic DNS name to its complicated TCP/IP location. The client application programs can be configured to use the generic names when they need to access a database (e.g., a database can be accessed by a client by using the generic name “Summary.” The DNS server accepts “Summary” and translates the address into \\MIX00001\SRV1\DBASE\DATAWAR.FIL. The client then is able to access the database). If the database location changes, the DNS configuration is changed, and no changes are needed to each client configuration.

Groupware Architecture

I. Definition

Groupware Architecture establishes a foundation for collaboration and communication. Collaboration focuses on local and ad hoc workgroups, while communication focuses on sharing information both within and outside the state.

II. Introduction and Background

Groupware is a combination of technologies enabling an organization to create, share, and leverage an accumulated knowledge base. Groupware technologies include electronic mail (email), calendaring and scheduling, electronic document management, shared file and print services, as well as some newer multimedia technologies.

For an enterprise-wide groupware implementation to succeed, the comprised technologies must comply to a set of common protocols and infrastructure standards, allowing them to communicate with one another. Some groupware technologies, such as email, have made considerable progress in the standardization of protocols across software products. Other groupware technologies are still maturing and have not yet standardized on protocols.

III. Principles

The following principles are provided to guide the design and selection of groupware components that will support distributed client/server computing activities.

Principle 1: Groupware requires a consistent infrastructure to truly support collaboration and communication.

- Content exchange, directory services, and authentication services are key infrastructure components necessary to facilitate communication and collaboration.
- Email is the communications infrastructure component within and outside the state. A significant portion of business communications is occurring across email services.

Principle 2: Groupware technologies overcome time and distance barriers

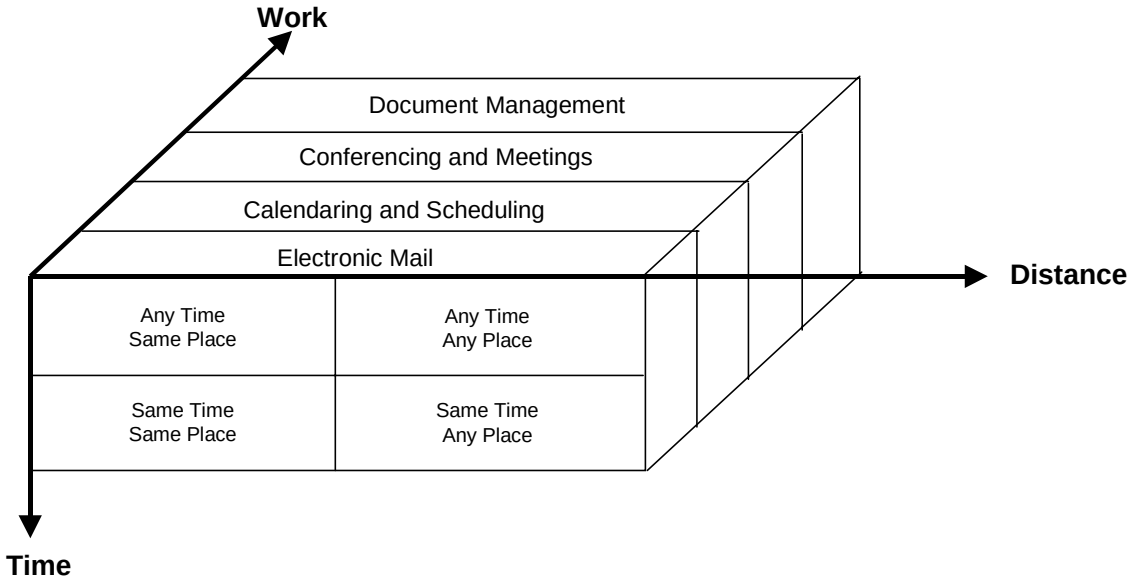


Figure 3-2. Time and Distance Barriers

- Groupware technologies must encompass all scenarios from working the same time at the same place to working any time from any place.
- Geographical boundaries are eliminated by groupware technologies, providing communication and real-time access to information from any location.
- The efficiency and quality of a process are partially measured by the timeliness in which the work is performed. The notion of timeliness must be qualified for each task to determine if groupware is providing value to that process. For example, a month may be considered acceptable for passing a specific group of documents through an organization for approval. In contrast, a month to send email to that same group of people is unacceptable.

IV. Technical Topics

Technical Topic 1: Infrastructure – Content Exchange

Introduction

Content exchange is a critical groupware infrastructure component, enabling the exchange of electronic information and data between individual users and groups. It includes the interchange of editable and non-editable documents between applications and individuals.

Recommended Best Practices

The recommended best practices in this section pertain to content exchange.

Best Practice 1: Avoid proprietary formats in anticipation of document exchange with outside users and applications.

- Proprietary formats inhibit document exchange, and may create future barriers to communication.
- If proprietary formats are used, the capability must be provided to convert documents to standard formats for content exchange. Any vendor using proprietary formats should provide a conversion routine.

Standards

The following standards have been established for content exchange. These standards ensure seamless processing of documents across the state, and are summarized in a table for easy reference.

Standard 1: For non-editable documents, the standard file format is PDF. Typical application software using this file format includes word processing, imaging systems, and World Wide Web publishing.

Typical Document Input Source	Typical Application Software Used	File Format Standard
Non – editable documents	Word Processing Imaging Systems World Wide Web Publishing	PDF
Monochrome documents or drw	Word Processing Archive & Retrieval Work flow Multimedia Digital Publishing Pattern Recognition Geographic Information Systems	TIFF using CCITT/ITU Group IV compression
Colour documents, drawing or photographs	Multimedia Word Processing Digital Publishing Pattern Recognition Geographic Information Systems	GIF JPEG
Facsimile documents	Word processing Archival and Retrieval Workflow	TIFF using CCITT/ITU Group III compression
Multimedia images	Multi-media	MPEG

Content Exchange standards

- PDF is widely deployed for non-editable content exchange.
- The reader is available at no cost.

Standard 2: For monochrome documents or drawing, the standard file format is TIFF using CCITT/ITU Group IV compression.

- Typical application software uses this file format includes word processing, archive and retrieving, workflow, multimedia, medical systems, digital publishing, pattern recognition, and geographic information systems.

Standard 3: For color documents, drawings, or photographs, the standard file formats are GIF and JPEG.

- Typical application software using this file format includes multimedia, word processing, medical systems, digital publishing, and geographic information systems.

Standard 4: For facsimile documents, the standard file format is TIFF using CCITT/ITU Group III compression.

- Typical application software using this file format includes word processing, archival and retrieval, and workflow.

Standard 5: For vector or geometric data, the standard file formats are DGN and DWG

- Typical application software using this file format includes CADD and geographic information systems.

Standard 6: For multimedia images, the standard file format is MPEG-1/2.

- Typical application software using this file format is multimedia.

Technical Topic 2: Communication – Electronic Mail (Email)

Introduction

As one of the fastest growing areas of communication, electronic mail (email) is becoming critical to the state's business operations. It is a powerful medium that allows the exchange of ideas and messages, as well as text documents, videos, images, and sounds. Integrated with other applications, email facilitates timely communication, opens access to documentation, and increases productivity. Email transport capabilities are also used as a delivery mechanism by other groupware services, such as workflow and calendaring and scheduling.

Recommended Best Practices

Since delivering services to citizens requires inter-department cooperation, departments must have compatible communications systems. A statewide email infrastructure facilitates communication on an enterprise-wide basis. For a statewide implementation, the state must implement an email architecture based on the following best practices.

Best Practice 1: Email servers should be administered and managed as a part of the strategic infrastructure.

- A properly structured email system can provide the state with a comprehensive, effective, inexpensive, and widespread method of communication, while permitting a choice of email clients.
- Email is a valuable tool because it provides the structure for easily moving messages and attachments in a timely manner between clients, thereby increasing workflow and productivity.
- Email service should be available at all times from any location. Time, distance, and location should not restrict email service.

Best Practice 2: Email servers should support multiple email clients.

- A properly structured email system can provide the state with a comprehensive, effective, inexpensive, and widespread method of communication, while permitting a choice of email clients.

Best Practice 3: Use a common email directory service throughout the state.

- An enterprise-wide email directory service should be accessible by everyone within the organization. The statewide directory service should be a seamless integration of each department's directory service. If a user in one department requests an email address for a user in another department, the action should be transparent, without the requester knowing where in the organization the address is stored.
- The directory service should be compatible with the directory services of other components in the network. Other applications require use of an email directory service. Use of a single directory service will facilitate reuse of information and directory access routines. It is necessary for heterogeneous components to access the directory service.
- For more information about directory services, refer to the Directory Services sub-topic in this chapter.

Best Practice 4: Select an email client that includes standard APIs for email-enabling other applications.

- Email is a key component of workflow. If a user is working on a document and chooses to send that document to another user, the user should not have to close the document creation application and open email to send it. Instead, the user should be able to mail the document directly from the native application.
- Calendaring and scheduling applications can use email message delivery for meeting proposals.
- Common APIs in use today are the messaging application programming interface (MAPI), vendor independent messaging (VIM), and common messaging calls (CMC).

Best Practice 5: Implement security for email message transport and storage.

- Private and official correspondence will require varying degrees of protection including authentication and encryption. SMTP/MIME was created as a means of "casual" communication over the Internet. It was not created to be a completely secure medium. Protocols are currently being developed bridge the security gap.

Standards

Similar to other groupware products, email protocols and standards are still emerging. For almost every protocol, there are several competing, non-compatible standards. If two email systems conform to different standards to access their mail servers, errors may occur when messages are

sent between the two systems. Approval of new standards is slow, leading to the proliferation of proprietary protocols and protocol extensions. Without consistent standards, a barrier in communication is created between platforms, applications, and components. To overcome these barriers, email gateways have been developed to integrate incompatible email systems.

Standard 1: Use Simple Mail Transport Protocol (SMTP).

- Simple Mail Transport Protocol (SMTP) is the standard transport protocol for sending messages from one MTA to another MTA over the Internet. Using MIME encoding, it enables the transfer of text, video, multimedia, images, and audio attachments. It is the predominate transfer protocol utilized by web browser-based email user agents.

Standard 2: Use Multi-purpose Internet Mail Extensions (MIME).

- Multi-purpose Internet Mail Extensions (MIME), a SMTP message structure, is the standard specification for the attachment of audio, video, image, application programs, and ASCII text messages. The content type is stored in the message header as mail extensions. When the message is delivered, the player or application specific to the content type is opened so that the attachment can be viewed in its native format. If the player or application is not included with the browser, then the user must load it. Common image and video players are included with most browsers.
- The MIME standard will require standardization of certain protocols in the near future. By its definition, MIME is transformable. Although two applications may be MIME-compliant, each application can use a proprietary or custom set of extensions. The data associated with the proprietary extensions may be lost in transfer. Common protocols cut down on the risk of a loss of data occurring.

Standard 3: Use Internet Message Access Protocol version 4 (IMAP4).

- Internet Message Access Protocol version 4 (IMAP4) is the standard protocol for access to the mail server. The user has the option of storing and manipulating messages on the mail server, which is important for job functions that require the user to access email from several different clients. IMAP is also ideal for situations where the user has a low speed link to the mail server. Instead of downloading all messages to the client, IMAP allows the user to select which specific messages to download. If a message has several MIME attachments, the user can specify that only the text portion of the message is to be downloaded for viewing. This practice is considerably more efficient in the event that a high-speed link is not readily available.
- Note: Options sometimes exist to configure email servers and clients without IMAP4 settings. Email servers and clients should be **implemented** using IMAP4.

Standard 4: Use Lightweight Directory Access Protocol (LDAP).

- Lightweight Directory Access Protocol (LDAP) is the standard directory access protocol. LDAP is based on Directory Access Protocol (DAP), an X.500 standard access protocol. X.500 is a set of CCITT/ITU standards for electronic directory services. LDAP has been proven to be more efficient for MUA to directory services transactions. In addition, LDAP can be utilized to access databases other than the email directories, which will add value to other groupware applications, such as scheduling.

Standard 5: Select an email server system that allows multiple standards-based email clients.

- When an email server uses IMAP4 standard, any IMAP4-based client can access that server.

Technical Topic 3: Collaboration – Calendaring and scheduling

Introduction

Calendaring and scheduling (C&S) is the process of scheduling events and accessing calendar information for people, facilities, and equipment. A calendaring and scheduling application manages the calendars and schedules of individuals, groups, facilities, and equipment. Through C&S, events and activities can be easily coordinated through the electronic exchange of scheduling information between individuals and groups.

Recommended Best Practices

The following recommended best practices pertain to calendaring and scheduling (C&S) and should be followed when selecting a C&S application.

Best Practice 1: Select an* open *C&S application, which maintains transparent interoperability with other C&S applications and computing platforms used across the state.

- The C&S system must be capable of supporting multiple server platforms and client platforms. The operating environment within the state is, and will remain, heterogeneous. The C&S system must therefore be capable of transparently transferring schedules and meeting information across each of the operating systems.
- C&S applications are typically purchased independently by each department based on the particular needs that the departments must satisfy. The selected applications must be capable of exchanging schedules, notifications, and materials with the C&S applications utilized by other departments. The user of one application must be capable of viewing a user's calendar created and stored on another application.

Best Practice 2: Select a C&S application that provides a mechanism for attaching supporting documentation, such as meeting materials, to the notification message.

- The capability to include or attach meeting agendas, supporting documentation, and deliverables maximizes the efficiency of C&S as a productivity tool.
- The user should not be required to compose a separate email message to send attachments. The transport mechanism for attachments should be accessible from the scheduling application.

Best Practice 3: Select a C&S application that allows the user to create both public and private notification groups and contact lists.

- Public and private notification groups are lists of people and/or resources that have common calendar and schedule needs, such as project groups or a list of conference rooms. Users can assign selected individuals to their private groups, and administrators can assign selected individuals to public groups. When a group name is entered as a participant, available times can be selected based on the group's information, and a notification message is forwarded to all individuals included in that group. This feature streamlines the use of C&S applications, making them more efficient.

Best Practice 4: Select a C&S application that enables task and resource management.

- In addition to people, the user must have the ability to schedule facilities *and* equipment. For example, the user should be able to reserve a meeting room and an overhead projector through the C&S application.
- The C&S application should be capable of tracking tasks (e.g., "To Do" lists that automatically send reminder messages for upcoming deliverables).

Best Practice 5: Select a C&S application that allows remote and proxy access.

- The user should be capable of disconnecting from the network and still maintain access to personal and shared calendars. Any updates to the calendars or schedule notifications made while the user is offline should be uploaded and synchronized at the time that the user reconnects to network.
- Incoming schedule notifications should be held by the C&S server until the user reconnects to the network at which time the messages are synchronized with the user's local calendar.
- Proxy access enables a C&S user to allow another authorized user or users to administer their personal calendar. This function may be limited to viewing, or allow full maintenance capabilities.

Best Practice 6: Select a C&S application that can be accessed through a web front-end.

- Both Intranets and Internets are accessible by anyone within the organization via a web-browser. Web enabled C&S applications allow users access to C&S information for anyone, anywhere in the state.

Technical Topic 4: Collaboration – Document Management

Introduction

Groupware products in the form of "office automation suites" have come to embody the typical user's view of sharing work by allowing the creation and exchange of many different types of electronic documents (e-docs). These documents include those created with word processors, spreadsheet and presentation software tools. In most local area networks (LANs) there are common areas where e-docs can be stored and accessed by users, if they know where to look for them.

Recommended Best Practices

It is essential that an organization devise a document management strategy as an information "architecture" rather than as just another application or imaging system. This means planning beyond a single standalone application. Recognize that EDM is not necessarily about "imaging." There are real problems with simply managing and sharing the documents created with ordinary word processors and other office automation programs used in the day to day administration of any office.

Best Practice 1: Evaluate potential requirements over a longer term basis and implement a "platform" that can be used to develop document-enabled applications and provide a uniform approach to document storage and access.

- Use the standards guidelines to assure the implementation of "scalable," open systems.

Best Practice 2: Assure the availability of open application program interfaces.

- Adhere to APIs and integration standards being advanced by the Association for Information and Image Management.

Best Practice 3: Select EDM and workflow tools that comply with AIIM open standards, are platform independent, and can be shown to be inter-operable with similar tools and other components of the statewide technical architecture.

- Selecting application components that adhere to industry standards is important for flexibility and adaptability.
- Products must support multiple server, workstation, and application platforms.
- Workflow components should trigger or send some message-based notification when human intervention is needed (using middleware, email, etc.) Partial automation of the office environment will cause confusion as to which activities need human initiative and which activities are being managed by the workflow system. This confusion will cause inefficiencies and leave a wider margin of error in work to be performed.
- Workflow systems should provide a standard interface to other workflow systems for the purpose of passing and processing work items between business units and processes.

Standards

As the state progresses with the development of a departmental and statewide document management infrastructure, and universal access to information, there are many obstacles to overcome related to the inter-operability between departmental systems and their interaction with an evolving enterprise level locator service. In the standards area departments planning for EDM systems and services needs to take into account three general sets of guidelines.

- Existing and evolving standards and guidelines being advocated by public institutions and private industry through the Association for Information and Image Management International (AIIM).
- Other standards and guidelines put forth by related parts of the Statewide Technical Architecture, especially as they relate to the development of document and/or business process-centric applications in an n-tier architecture.

There is no limit to the creativity and innovation occurring in the development of solutions to the problems of document management, workflow, and universal access. The framework embodied by these standards is intended to be moving users and developers toward the goals described previously.

Standard 1: Implement document management systems and components that conform to the Document Management Alliance specifications (DMA 1.0 and ODMA 2.0).

- There are numerous issues related to interoperability among document management applications, services, and repositories. Standards are needed to manage the increased life expectancy and complexity of re-usable electronic documents and content.

- The Document Management Alliance (DMA), is a task force of AIIM. DMA conforming products will support open design for user interfaces, workstations, network operating systems and servers. The DMA provides a framework for vendors to deliver products that provide query services (simple transparent access from every desktop to information anywhere on the network), and library services (including check-in and checkout, version control, security, and accountability). The DMA is working with the Open Document Management API (ODMA) group which specifies the common application program interfaces, and high level call interfaces that enable other client applications (such as MS Office) to work seamlessly with DMA compliant document management systems.

For more information about AIIM standards programs, refer to the Web site: http://www.aiim.org/industry/standards.

Standard 2: Implement workflow systems that conform to the interface specifications of the Workflow Management Coalition (WfMC).

WfMC is another working group of AIIM and is closely aligned with the work of the DMA. As automated workflow systems continue to evolve, the complexities associated with a common approach to process definition, process repositories, object manipulation and transport, and user interfaces are enormous. The Workflow Management Coalition (WfMC) has proposed a framework for the establishment of workflow standards. This framework includes five categories of interoperability and communication standards that will allow multiple workflow products to coexist and inter-operate within a network environment. This framework is contained within a Reference Model for workflow management systems that includes five interface specifications. The model includes the following:

- Process Definition Tools.
- Workflow Enactment Services.
- Workflow Client Applications.
- Invocation of Native Applications.
- Workflow Package Interoperability.
- At this time, there are many companies designing products that comply with one or more of these interface specifications. Departments planning production workflow applications that need to route work outside of the production system for processing or decision making should work carefully with vendors and service provides to determine functional requirements and WfMC standards compliance.
- For more information about the WfMC and the work of the coalition refer to the Web site at: http://www.aiim.org/wfmc/index.html.

Standard 3: Use Adobe Acrobat Portable Document Format (PDF) for Non-editable Electronic Documents.

All documents in final form and prepared for distribution and publishing with no intention for further modification must be stored and delivered in Adobe .PDF format.

Standard 4: Ensure hardware/software and image file compatibility using TWAIN, ISIS, and TIFF standards.

- For typical business document imaging applications, software that controls the operation of the scanner (and some other recognition peripherals) is provided. Not all scanner hardware and scan software are compatible. The industry standards to adhere to are TWAIN and more recently ISIS (Image and Scanner Interface Specification). These are API standards that provide low level integration facilitating the control of the peripherals from many common user

applications. For specialized applications (e.g. hand held devices) other standards requirements need to be investigated.

- The scanned images of typical business documents should be committed to storage in Tagged Image File Format (TIFF) Version 6.0 using CCITT/ITU Group III or IV compression. Organizations planning imaging applications should investigate and demonstrate that any product selected is capable of exporting images in a format that they can be reused. Images that can not be shared are a wasted investment and could result in the loss of critical data.
- Avoid new deployment or migrate away from proprietary image file formats. The current technology direction for image file formats is TWAIN. The emerging technology file format is ISIS.

Standard 5: Select magnetic storage subsystems. Select optical storage subsystems based on smaller standard form factors.

- Typical electronic documents, created with office automation suites, will reside on industry standard magnetic disk that is server or network attached. This will generally be transparent to the users of an EDMS. The images of scanned paper documents might also be stored on standard network attached magnetic disk. Magnetic storage will always provide the most performance in the speed of retrieval, and magnetic disk is increasingly cost competitive with optical disk storage. When selecting any magnetic storage solution, adhere to other parts of the STA that provide the standards for these types of systems.
- Very large document collections (usually image applications) will probably require optical storage subsystems (many are proprietary). Where there is a requirement for the permanent storage of unalterable documents, optical is chosen in the form of Write Once Read Many (WORM) disks. These types of systems generally involve special software that is used to manage the storage and movement of documents from optical to magnetic when documents are requested by users. Optical disks may be mounted in single standalone drive units or they may be loaded into various sizes of “juke boxes.” Software handles the retrieval and loading of specific disks in response to user requests. Typical EDMS systems today will use a 5 ¼ form factor and will be WORM or Compact Disk type formats. Larger disks are available for specialized applications and are generally proprietary.
- Avoid new deployment or migrate away from proprietary or large format optical storage subsystems. The current technology direction is WORM and various types of compact disc in 5 ¼” format. The emerging technology is magneto Optical and DVD.

Standard 6: Use eXtensible Markup Language (XML 1.0) when capturing or authoring document content that requires further automated processing by other information systems and Web based clients using standard XML enabled browsers.

- This standard is promulgated by the World Wide Web Consortium (W3C).
- XML is a subset of the Standard Generalized Markup Language (SGML, an ISO standard).
- XML encodes a description of a document’s storage layout and logical structure with a document type definition (DTD). It provides a mechanism to combine structured data and unstructured information content.
- XML allows information systems and applications to automatically process XML documents when the systems are combined with an XML processor.
- The specification (DTD) describes the required behavior of XML processors in terms of how they read XML documents, and what information they must provide to the processing application. For more information about the W3C and XML refer to the Web site at: http://www.w3.org.

Componentware Architecture

I. Definition

Componentware Architecture enables efficient reuse of existing application assets, faster deployment of new applications, and improved responsiveness to changing business needs. Reusable software components are the building blocks that make a system able to respond quickly to change.

II. Introduction and Background

Components are program modules that provide a complete package of business functionality. Shared components must be designed for portability across platforms.

Components within an application system can be developed in any supported language, with any development tool appropriate for the particular tier where they are deployed. Monolithic applications perform comprehensive business functions and operate independently from other applications. Making changes to a monolithic system is a major undertaking because changes in one area often cause problems in other areas.

III. Principles

These principles provide guidelines for the design or purchase of application components that support distributed, client/server, and adaptive computing across the state.

Principle 1: Componentware Architecture facilitates the reuse of components across the enterprise.

Reusable components increase the productivity of the application development departments within the enterprise.

- Sharing components across the enterprise greatly increases the ability of the system to meet the changing needs of the business.
- The use of proven components enhances the accuracy of information processing.

Principle 2: The focus of Componentware Architecture is to improve business performance.

- A component-based development strategy enables adaptive systems to meet the changing business needs and technical environments.
- A component-based development strategy aligns information technology with the commonly used functions of the business.

Principle 3: Shareable components must be callable by any component-enabled application.

- Reusing existing shared components eliminates duplication of development, testing, and maintenance effort.
- Reusing existing shared components eliminates processing inconsistencies because business rules are maintained in one piece of code.
- Use of components reduces the time and effort required for developing and updating applications.
- Reuse results in quality improvements because many applications are being built based on same components.

Principle 4: New components must be platform independent.

- Components must be developed so they can be deployed on any supported platform.
- If the business needs change or a new platform is required, the component should easily migrate to a new platform.

Principle 5: Purchase rather than build components whenever possible.

- Purchased components must be capable of being implemented in a service-oriented environment; (i.e. can be integrated into an *N*-tier environment with a published Interface Definition Language (IDL) interface).
- Components should be purchased whenever possible, such as class libraries, allowing developers to focus on the development of specialized business rule components.

Principle 6: Design components to be fully self-contained.

- All necessary validation, error detection, and reporting capabilities, logging/debugging/tracing functionality, monitoring and alert functionality, and system management capabilities must be incorporated in the component.
- This facilitates operation, administration, and maintenance functions.

IV. Technical Topics

Technical Topic 1: Component Reuse

Introduction

A successful implementation of an *N*-tier, reusable component service-oriented architecture is not solely dependent on the ability to develop reusable components. Success also depends on the ability to provide the tools and management of the components for reuse. Following a reuse *methodology* and understanding reuse *techniques* are key to developing and managing statewide reusable components.

Recommended Best Practices

The recommended best practices in this section pertain to reusable components.

Best Practice 1: Establish a reuse methodology for the identification and implementation of components.

A methodology that supports reuse contains the following steps:

- Classify the business requirements by service type (e.g., application, interface, support, or core).
- Search the repository for reusable components that support the business or functional requirements.
- Analyze candidate components to ensure they fully meet the requirements.
- Incorporate the selected components into the new or re-engineered application using standard IDL's.
- Harvest new components from new or existing applications that have not been componentized yet. Placing the new component information into the repository.
- Incorporate the reuse methodology into the system development life cycle.
- To successfully implement the Componentware Architecture, the Network and Middleware Architecture must be in place.

Best Practice 2: Establish a component review board to identify common components.

Components used by multiple business units must be commonly understood and consistently referenced by all business users.

Component development can be achieved through the context of projects.

The review board should start with small, achievable, and strategic projects.

In order to create reusable components, cooperation is needed among the business process owners.

A framework needs to be put in place that allows for:

- Centralized management of reusable, shareable components.
- Design reviews of new and existing projects for reusable components.
- Enterprise access to information about reusable components.

Best Practice 3: Establish a repository for maintaining the information about available reusable components.

- The repository provides a place to store documentation about the component API's.
- The repository should be made available to all application developers as a tool for performing their jobs.

Best Practice 4: Every component must have a published API.

- A published API defines the public interface for a component or service. The API is how other applications will communicate with the component. Documentation should include input and output parameters, which parameters are required, which parameters are optional, and the lengths and types of the parameters.

- The API should be entered into the component repository that is available to all application developers.

Best Practice 5: Harvest components from existing applications to initially build the component repository.

- Legacy applications are a good resource for building a component repository.
- There is no need to reinvent a process or piece of functionality if software already exists that performs the desired function.
- If feasible, develop a wrapper that defines the API for the service and allows the legacy application to become a reusable component.

Standards

The standards in this section pertain to component reuse.

Standard 1: No vendor proprietary API calls for infrastructure security services. Use Generic Security Services-API (GSS-API) or Common Data Security Architecture (CDSA) compliant API calls and products.

- Applications requiring security services prior to CDSA products or services being available can use the GSS-API.
- The GSS-API is an Internet Engineering Task Force (IETF) standard (RFC 2748, released in January 2000, obsoletes RFC 1508 and RFC 2078) and supports a range of security services such as authentication, integrity, and confidentiality.
- It allows for plug-ability of different security mechanisms without changing the application layer.
- It is transport independent, which means it can be used with any underlying network protocol.
- Applications using GSS-API can be retrofit to a CDSA foundation without major modifications, therefore providing an interim step to CDSA based services.

Technical Topic 2: Component Services

Introduction

A service completes a task requested by an application. The service itself may call one or more components to complete the task. To the calling application, however, it appears as a single task. Typically, a service is created when it is identified as a common task that would be (repeatedly) coded by several applications.

Recommended Best Practices

The recommended best practices in this section pertain to component services.

Best Practice 1: Component services should be callable by any component-based application or any other component.

- Components must be designed and developed with the understanding that the process that invokes it may or may not be developed in the same language or in the same environment.
- A component should be callable from any supported language on any supported platform.

Standards

These standards in this section pertain to component services.

Standard 1: Custom developed application components must be written in a portable, platform-independent language, such as C, C++, or Java.

- Application components written in a portable language are easier to move from one platform to another because they require fewer modifications to conform to the new host. This portability allows an application to be more adaptive to changes in platform technology.

Standard 2: Statewide infrastructure services must be atleast Common Data Security Architecture version 2.0 (CDSA v2.0) compliant.

- The CDSA version 2.0 architecture is an Open Group specification for providing security services in a layered architecture and managed by a Common Security Services Manager (CSSM). CDSA provides a management framework necessary for integrating security implementations. Version 2.0 of the specification is a cross-platform architecture, which includes a testing suite for inter-operability testing.
- A wide range of vendors has announced support for the specification and products for a broad set of platforms can be expected.
- Security protocols such as SSL, S/MIME, IPSec can all be built from the Common Data Security Architecture base.

Technical Topic 3: Object-oriented Components

Introduction

Object-oriented components encapsulate both the business logic and the data accessed by the business logic. They have the potential to become intelligent, self-managing entities, allowing for more simplified management.

Recommended Best Practices

The recommended best practices in this section pertain to object-oriented components.

Best Practice 1: State application developers should develop Enterprise components using object technology based on Enterprise Java Beans. Departments should enter an appropriate object-oriented analysis and design lifecycle prior to the implementation.

- Object oriented programming starts with the definition of objects. Therefore analysis and design are critical.
- Object-oriented design and development is well understood by many developers and the software industry often provides solutions based on objects rather than APIs.
- EJB together with Servlets hide most of the required infrastructure service requirements programming for web based applications. Where sharing of services is desirable, departments can offload the burden of such programming and can focus on the business logic rather than communication code.

Standards

The standards in this section pertain to object-oriented components.

Standard 1: Purchased applications must be CORBA 2.0 or later and IIOP (Internet Inter-ORB protocol) compliant.

- CORBA and IIOP standards are open standards devised for platform independence.

Standard 2: Build or purchase Enterprise solutions on an Enterprise Java Bean and servlet model. Application servers should be compliant with EJB 1.1 or better.

- Enterprise solutions benefit from reduced requirements to code underlying services and can focus on business logic.
- Vendors provide solutions based on an EJB model. These solutions can be purchased and used without customization.

Standard 3: Avoid OLE/DCOM and the Windows DNA object model for applications with Enterprise or strategic implications.

- OLE/DCOM standards do not scale well and run only on Windows platforms.
- OLE/DCOM applications are not easily portable or integrated into enterprise-wide solutions.

Data Architecture

I. Definition

The mission of Data Architecture is to establish and maintain an adaptable infrastructure designed to facilitate the access, definition, management, security, and integrity of data across the state.

II. Introduction and Background

Data and information are extremely valuable assets of the state. Data architecture establishes an infrastructure for providing access to high quality, consistent data wherever and whenever it is needed. This infrastructure is a prerequisite for fulfilling the requirement for data to be easily accessible and understandable by authorized end users and applications statewide.

III. Principles

The following principles apply to the enterprise Data Architecture. Enterprise principles are relevant to both statewide and department-wide data.

Principle 1: Design an adaptive data infrastructure.

- Design the data infrastructure to easily accommodate changes in the data model and database technology. The data infrastructure is a crucial component of establishing an overall adaptive architecture.
- An adaptive data infrastructure provides extensibility in adding new functionality and facilitates vendor independence.

Principle 2: Design the enterprise Data Architecture so it increases and facilitates the sharing of data across the enterprise.

- Sharing of data greatly reduces data entry and maintenance efforts.
- Data sharing requires an established infrastructure for widespread data access. This includes integration with the Application, Componentware, Integration, Messaging, Network, and Platform Architectures.
- Consistent shared data definitions ensure data accuracy, integrity, and consistency.
- Data sharing reduces the overall resources required to maintain data across the enterprise.

Principle 3: Separate the data sources for online transaction processing (OLTP) data and online analytical processing (OLAP) information.

- Separate data sources isolate OLTP systems, which perform mission critical business processing, from large ad hoc queries and online analytical data processing. If the data

sources are not separate, ad hoc queries and direct access of data for OLAP systems can adversely impact online transactional processing.

- Data design is adapted for optimal performance for each type of application, OLTP or OLAP. For optimal performance, OLTP and OLAP may require different database designs. OLAP typically includes complex operations involving time series, dimensional, and trend analysis, which do not perform well with relational database technology alone (e.g., sometimes other methods of data storage are needed to support OLAP, such as multi-dimensional databases or flat files).

IV. Technical Topics

Technical Topic 1: Centralised Metadata

Introduction

The state should develop applications that cooperate and share data, both within a single department and between departments. In this model, common data elements are defined consistently even when they are stored in multiple databases and data can be shared between applications. This type of data is referred to as centralised data.

When centralised data is defined consistently, data is described the same way in each table where it is defined. Definitions include traits such as name of the field, length, number format, data format, and the values it can have. When the data has the same format, it is much easier to exchange data across system and organizational boundaries.

The way to describe or define data is through metadata. Metadata is “information about data.” Metadata is stored in a repository containing detailed descriptions about each data element. By using the formats described in the metadata repository, whether the data resides in a single location or in multiple databases across the state, the same data management principles apply.

A list of sample Metadata elements are as follows:

Data Architecture
Metadata - Sample

Data Element Attribute Name	Element Definition	Field Length	Data Element Type	Example	Remarks
SSID	Social security identification number	12	Number	123-456-789-123	Unique number for each person
First Name	Person's first name	20	Varchar	Sachin	• International convention (Banks, colleges, etc) • Election commission in India has the same convention • There are some examples given in the end
Middle Name	Persons middle name	15	Varchar	Ramesh	
Surname	Person's surname	15	Varchar	Tendulkar	
Father / Husband's First Name	Father / Husband's first name	20	Varchar	Ramesh	
Father / Husband's Middle Name	Father / Husband's middle name	15	Varchar	Nandan	
Father / Husband's surname	Father / Husband's surname	15	Varchar	Tendulkar	
Date of Birth	Person's date of birth	8	Date	12-09-1965	In DD-MM-CCYY format
Sex	Sex of the person	1	Varchar	M	Male / Female Flag
Caste Category	Caste category of the person	2	Varchar	01	SC/ST/BC/Others etc.

1

PRICEWATERHOUSECOOPERS

Data Architecture
Metadata - Sample

Sample names for data entry – Probable convention

First Name	*Middle Name*	*Surname*
Krishna Reddy VV	--	Kurapati
Manoj	Kumar	Bajpai
Leander	--	Peas
Sachin	Ramesh	Tendulkar
Dipti	Dilip	Bapat
Gandhi LN	--	Challasani
Prasad SRKS	--	Mullapudi
Abdul Aziz	--	Mohd
Srinivasa Rao	--	Palakodeti

- Note: Data entry should be consistent to help
  - No special characters(-,.) should not be used in the names
  - For Data search & query
  - For Reports generation and printing

1

PRICEWATERHOUSECOOPERS

Data Architecture
Metadata - Sample

Data Element Attribute Name	Element Definition	Field Length	Data Element Type	Example	Remarks
Religion	Religion Code	20	Varchar	Hindu	Codified field with name displayed
Address1	Person's address First line	50	Varchar	205, Annapurna Apts., Near Shanti Theatre	First address Field
Address2	Person's address second line	50	Varchar	Street No. 35, Chikkadapalli	Second address field
District	District code	2	Varchar	02	Codified field with name displayed
Division	Division Code	4	Varchar	0103	
Mandal	Mandal Code	4	Varchar	1100	
Village / Town / City	Village / Town / City Code	5	Varchar	25026	
Municipality	Mandal Code	4	Varchar	1155	
Panchayat	Panchayat Code	5	Varchar	22652	
Habitat	Habitat Code	5	Varchar		

1 PRICEWATERHOUSECOOPERS

Data Architecture
Metadata - Sample

Data Element Attribute Name	Element Definition	Field Length	Data Element Type	Example	Remarks
Marital Status	Marital Status	1	Varchar	Y	Y / N Flag
Phone No.	Contact phone no. of the person	10	Number	040-4058364	STD code followed by ph.one number
Fax No.	Fax no. of the person	10	Number	040-4058364	STD code followed by ph.one number
E-Mail ID	Electronic mail id	50	Varchar	Ramachandra.sarma@rediff.com	Electronic mail id
PAN No.	PAN no. of the person	10	Number	ABTPR2507M	Number as allotted by Income Tax dept.
Survey No.	Survey no. of property	30	Varchar	12/24/345/ 496-A1	Survey no. where property is located
Registration No.	Owned vehicle registration no.	10	Varchar	AP37 AX 8420	Number as allotted by RTA
Passport Number	Passport no of the person	8	Varchar	A1155679	Number as issued by Passport authority
Driving License No.	Driving License no. Of the person	20	Varchar	AP37/4235/ BVRM/99	Number as allotted by Licensing authority

1 PRICEWATERHOUSECOOPERS

Recommended Best Practices

The recommended best practices in this section pertain to centralised metadata.

Best Practice 1: Use and actively maintain the Centralised Metadata Repository to store centralised metadata definitions.

- Storing data element definitions in a central repository incrementally builds the enterprise data model.
- The repository must be actively maintained (e.g., changes to metadata occur in the repository *before* the changes occur in operational applications).
- The repository serves as a centralized data administration tool and helps promote data reusability, reliability, and sharing across the enterprise.

Best Practice 2: When designing or modifying a database, review the Centralised Metadata Repository for existing standard and proposed data elements before implementing a new database to ensure data elements are defined according to CMR standards.

- Design reviews are essential to ensure that shared statewide data is defined consistently across all applications. Design reviews also determine whether data that already exists is consistently defined and not redundantly stored.
- Design reviews should document the following:
  - Where is this application getting its data?
  - What other applications are getting data from this application?
  - Is data used by this application defined consistently with statewide definitions? If not, is there a plan to define the data according to enterprise definitions?
- A design review evaluates the data requirements of a project and identifies the following:
  - A data requirement that can be solved by using existing centralised metadata element.
  - Data not already identified as centralised metadata must be proposed as an inter-department or statewide standard to the Metadata Element Review Team to become centralised metadata.
- Access is available for application development projects to reference the CMR in order to actively research data requirements. Review the existing standard and proposed data elements in the CMR before implementing a new database to ensure data elements are defined according to standards.
- Key information about data is stored in the systems that are already implemented in the state. If possible, evaluate existing systems to propose statewide and department data elements.

Best Practice 3: Use the CMR and the Metadata Element Review Team to create centralised definitions of enterprise level data and encourage the sharing of data across departments.

- Data used by multiple business units must be commonly understood and consistently referenced by all business users. Enterprise sharing of data can only be achieved by creating centralised definitions.
- Centralised definitions of data emerge through the context of projects. An enterprise model evolves over time through ongoing projects.
- In order to create centralised definitions of data, cooperation is needed among the business data owners.
- Centralized management of distributed enterprise-level data is provided through the state's Centralised Metadata Repository. This repository is available through the Internet.

Best Practice 4: Identify authoritative sources for centralised metadata.

- Authoritative business sources for centralised metadata must be identified, documented, and actively maintained in the repository. Authoritative business sources are the business units responsible for the accuracy of the data stored.
- A source of record is an authoritative source for data. Data in a source of record is trusted to be accurate and up-to-date. All other data stores should synchronize to the source of record. The data in record sources must be actively managed and the data model should be verified by data administrators. Tools and quality control techniques must be applied to the contents of the data stores themselves, in order to ensure the quality of the data.
- Each application must identify data sources for all data that it does not originally capture. The application capturing the original data is the authoritative source, and is responsible for the quality of the data. All application data models for ongoing projects should be reviewed to ensure that data existing in authoritative systems is reused and not redundantly stored.

Standards

Standard 1: Custom systems must comply with CMR standard data element definitions.

- New databases belonging to custom systems must comply with CMR element definitions. The Metadata Element Review Team will review the data elements to determine if the elements conform to existing standards.
- Any new potential data element standards must be proposed and reviewed for approval as a state standard.
- If the data element definition cannot be customized to conform to existing standards, a waiver must be requested.

Standard 2: Commercial off-the-shelf (COTS) systems that support client-controlled data element definitions must comply with the CMR standard data element definitions. Otherwise, the vendor must provide a conversion routine to conform to metadata exchange standards for data sharing.

- If an off-the-shelf system has data formats that can be modified, the data elements should be adapted to conform to the standard data requirements.
- If the data element definition cannot be customized to conform to existing standards, the vendor must provide conversion routines to conform to the CMR metadata exchange standards.

Standard 3: Use Centralised Metadata Exchange Standards when exchanging data across departments.

- If data needs to be exchanged across department boundaries and the data is physically stored differently, then the data must be exchanged through the exchange standard as specified in the CMR.

Technical Topic 2: Data Modeling

Data modeling is the process of defining a data model for a project or application and is typically performed at the same time as the business model during the design phase of a project.

Recommended Best Practices

The recommended best practices in this section pertain to Data Modeling.

Best Practice 1: Take the Entity-Relation (ER) model to the third normal form, then denormalize where necessary for performance.

- The third normal form is the most commonly recommended form for the ER model.
- In some cases, a denormalized database can perform faster as there can be fewer joins, or reduced access to multiple tables. This process saves both physical and logical input and output requirements.

Best Practice 2: In a dimensional model, use a star schema whenever possible.

- Use a snowflake schema only if it increases user understandability or improves performance. A snowflake schema may add unnecessary complexity to the data model.

Best Practice 3: Restrict free form data entry where possible.

- In the design phase, consider the values that may be input into a field. These values or domains should be normalized so that data is consistent across records or instances. For example, using consistent values for gender or address information.
- Use look-up tables and automate data entry for column or attribute domain values to restrict what is entered in a column.

Best Practice 4: Setup indexes and form relationships carefully.

- Limit the number of indexes on databases that will be experiencing significant insert and update activity. When an insert is performed, not only is the record updated, but all the indexes are updated as well.
- Increase the number of indexes on databases where importance lies in retrieval time. Indexes can increase performance on retrieval time.
- Before creating a database, indexes, or data access programs, verify that all relationships have been documented.

Best Practice 5: Design the data model to allow for growth and/or change.

- Design data models to accommodate any future changes, including growth and changes in business requirements or database technologies.

Best Practice 6: Archive and protect the data model.

- Data models store a wealth of department and statewide information and must be archived and protected.
- Data models should be catalogued with the department's project documentation and used to facilitate future revisions.

Best Practice 7: Each department should standardize on a common data-modeling tool for designing and maintaining all new database instances.

- Department Technical Architectures specify each department's common data modeling tool.

- A data model ensures that data is defined accurately so it is used in the manner intended by both end users and remote applications.

Best Practice 8: Use a data-modeling tool to reverse engineer existing databases.

- Data modeling tools can evaluate an existing database structure and reverse engineer a data model. The reverse engineered data model can be used to capture valuable information about the existing database.

Technical Topic 3: Data Access Middleware

Introduction

Data access middleware is the communication layer between data access rules and the Data itself. Data access middleware is designed to enable communication between a data access tier and a database, as opposed to application communication middleware, which enables communication between the programming tiers of an N-tier application.

Recommended Best Practices

The recommended best practices in this section pertain to data access middleware.

Best Practice 1: Provide centralized administration for data access middleware through central IT staff.

- Typically, workstations and servers are used for multiple applications and require connectivity to multiple databases. If administration for data access middleware is provided by central IT staff, any changes that are required are more easily managed and executed
- Support costs and efforts to support data access middleware are reduced.

Best Practice 2: Avoid use of extensions that create vendor lock-in.

- To differentiate their database from other vendors, many database vendors have implemented special extensions beyond the SQL-compliant commands. Although sometimes these extensions may be useful for a particular function, they are not recommended.

Standards

The standards in this section pertain to data access middleware.

Standard 1: Use OLE DB or JDBC database access middleware when accessing a database.

- Use OLE DB or JDBC to access a database instead of vendor specific database middleware.
- OLE DB and JDBC allow flexibility in programming. A database can be easily modified or relocated. If a change is needed, the change is made to the OLE DB or JDBC configurations, not to each data access program or tool.
- These technologies are widely supported by the industry and make an application more adaptable to changes in database or other technology requirements.

Standard 2: Implement a server-based OLE DB or JDBC solution as opposed to a workstation-based OLE DB, ODBC, or JDBC implementation.

- A server-based solution is easier to administer. Database changes and additions are easier to manage, since updates are made to database middleware servers, not every workstation that requires access.

Standard 3: Use domain name system (DNS) alias names when accessing databases through OLE DB and JDBC.

- If the database location changes or if the server name changes, the DNS configuration is changed, and no changes are needed to each client configuration.

Technical Topic 4: Data Access Implementation

Introduction

Since data is at the core of most applications, data access is a vital component of the Data, Application, and Componentware Architectures. Depending on the application, data can be stored and accessed in numerous databases in multiple locations. When data is centralized or when data is distributed across an organization, data access must be carefully implemented with usability, accessibility, cost, performance, adaptability, and security in mind. This topic provides an overview of the different types of data access and discusses practices and guidelines to use when implementing data access including:

- Overall data access methods
- Specific data access methods
- Data integrity
- Data access design considerations.

Recommended Best Practices

The recommended best practices in this section pertain to Data Access Implementation.

Best Practice 1: Establish a data infrastructure that can accommodate rapid changes in data models based on changes in business requirements or changes in database technologies.

- Business requirements change frequently. The data infrastructure and design must be adaptive and allow for changes to be easily implemented.
- Technology changes are fast emerging. The infrastructure must allow for replacement of the database technology if necessary.

Best Practice 2: Centralize data that needs to be shared and current.

- High-volume transaction data that is shared across locations and that needs to be current for all locations must be centralized so all locations have access to the same data source
- Replicating frequent updates to distributed databases increases systems complexity and network traffic.
- Data must be centralized when one or more of the following criteria occur:
  - Many users need access to latest data (i.e., OLTP systems).

- The number of users is small and there are no distributed sites.
- There is a lack of skills and tools at multiple sites to manage distributed data.
- There is a need to provide a consolidated and integrated database for centralised metadata on an open platform.

Best Practice 3: Design databases to be modular, business driven and aligned with application services, not monolithic.

- Aligning data with the application service facilitates changes in business processes. Only the data associated with a particular business process is potentially affected when a change is needed, not all the data associated with an entire application. It also increases performance for backup and recovery and provides higher reliability, availability, and scalability.
- By modularizing data, this practice provides better performance for backup and recovery, higher reliability, availability, and scalability, and better transaction performance due to parallelism (e.g., a complex request can be broken down and be processed by multiple databases at the same time).
- Very large databases (VLDBs) typically use a terabyte or more of storage. It is recommended that VLDBs be partitioned based on the appropriate business elements to improve application and database performance. VLDB partitioning can enable more efficient backup and recovery capabilities (e.g., it is more efficient to backup five 10 million row tables simultaneously than it is to backup one 500 million row table).
- In order to align data with application services, the following items need to be defined:
  - Business processes.
  - Data required to service the business processes.
  - Business units responsible for providing the business process.
  - Access to the data, and the know-how to use the data, by other business units or applications.

Best Practice 4: Minimize the replication of data within operational applications by replicating only stable data when necessary and based on business requirements.

- It is better to maintain only one version of data whenever possible, particularly for mission critical OLTP systems.
- Replication must not be used unless it is required for performance or decision support.
- A replication infrastructure is simpler to design for stable data. If replicated data is updated frequently, it is much more difficult to design and maintain a replication infrastructure.
- Replication may be appropriate when there are users in different locations needing similar data that does not need to be current 24 hours a day and a central source database is not a possible solution.
- Specific application requirements for data availability, recoverability, and freshness (near real time, 24 hours old, etc.) must be identified.

Best Practice 5: Design the data access infrastructure to support the transparency of the location and access of data by each application.

- This means designing an N-tier architecture where all data access is managed through a middle tier. This design makes databases easy to relocate, restructure, or re-platform the back end services with minimal disruption to the applications that use them. It is essential for adaptive systems.
- A client should not send SQL requests directly to a server. Instead of using SQL code, the client should communicate with the database through data access rules. The application receives a request from a client and sends a message to the data access rule. The data

access rule sends an SQL call to the database. With this method, the client does not send SQL to the server, it sends a request for work.

Best Practice 6: Design for data to be accessed only by the programs and business rules owning the data, never by direct access to the database.

- This practice ensures security, data integrity and accurate interpretation of the data and allows for adaptability to changes in business needs.

Best Practice 7: For data quality management, implement tools, methods, processes and policies to provide high-level data accuracy and consistency across distributed platforms.

- Both business users and Information Technology (IT) staff are responsible for data accuracy and consistency. Policies and procedures must be established to ensure the accuracy of data.
- IT staff is responsible for and must provide security mechanisms to safeguard all data under IT control. The business users must determine functional security requirements, while the physical security must be provided by IT.
- Applied systems management provides safeguards against data loss and corruption and provides the means of recovering data after system failures. This implies that effective backup and recovery systems are imperative and that data can be recovered in a timely basis regardless of the cause of loss.
- To satisfy service-level requirements, if the data is not too volatile, data replication can be used. If the data is extremely volatile, using replicas must be avoided. Additional requirements may include providing redundancy for extra bandwidth for communications volume and for availability in the event of a disaster. For critical functions, plan for survivability under both normal operations and degraded operations.
- It would be ideal to record the flow of data across systems, even if it is built incrementally starting with existing application development projects.

Best Practice 8: Optimize the physical database design to support an optimal total performance solution.

As with all application and data access design, performance is always a factor to consider. However, database performance is only part of the total solution and must be evaluated in conjunction with other components that impact performance, such as network and application. When implementing data access, several practices can help performance, including:

- Limit the number of indexes in a database. When a record update occurs, not only the record is updated, but all the indexes are updated as well.
- Limit ad hoc data access. End user ad hoc access can impact the performance of the database.
- Limit the number of rows returned in a query. In OLTP, most users normally work with only a single row at a time or a few rows displayed in a grid, list or combo box. If a user will only be working with a handful, there is no reason to return all the rows in a table.
- Return only the columns needed. Provide an explicit column list instead of a “SELECT *” query.
- Limit the number of joins. Complex multi-table joins have negative performance ramifications.
- Avoid sorts. Sorts can be slow, especially if sorting large amounts of data. If sorting is required, sort on indexed fields.
- Limit the rows used for pick lists, combo boxes, or lookup tables. If a large list is necessary, find an alternate method to provide the list.

Best Practice 9: Implement a minimal number of data access rules.

- A typical n-tier application has numerous business rules. The data access logic for these business rules should be shared through a minimal number of reusable data access rules. Due to the commonality of database queries, many similar queries can execute using a single, properly planned data access routine.
- Portability to another database platform or vendor is simplified by having a fewer data access rules.

Best Practice 10: Use ANSI-Standard SQL programming language to access a database.

- Data access to relational data stores must be through ANSI-standard SQL programming language access, not proprietary SQL extensions.

Best Practice 11: Implement a minimal amount of data access rules stored in the database as stored procedures and triggers to avoid vendor lock-in.

- Code data access rules into a data access service. When implemented through the Andhra Pradesh Service Broker (APSB), these services are callable by multiple applications. If a database changes, there is minimal impact to the calling applications since the API should not change.
- Stored procedures and triggers are specific to the database vendor and are more difficult to migrate to a new database if required.
- Database triggers must only be used to support referential integrity.
- Other technologies such as object transaction monitor (OTM) can be used to negate the impact of executing dynamic SQL.

Best Practice 12: Use the State Service Broker for intra-department and intra-application data sharing.

- The department owning the data is responsible for writing the shared service to access the data. This ensures data integrity and proper data interpretation.
- The department requesting the data is responsible for writing the request to retrieve shared data according to the shared service specifications.

Best Practice 13: Use the state's interface engine for data sharing of legacy platform data or other data where the application source code cannot be modified or interfaced.

- Some legacy systems do not have an application program interface (API) and there is no access to the source code. When requiring data access to one of these systems, use the state's interface engine.
- The state's interface engine can be used in instances where the source code is unable to be modified or the data layouts are unavailable. It is an unobtrusive interface to accomplish data sharing.
- Use of database gateway or database-specific middleware must be avoided.
- For more information about the interface engine, refer to the Integration Architecture.

Standards

The standards in this section pertain to Data Access Implementation.

Standard 1: Use the State Service Broker (APSB) for inter-department data sharing.

- Service Broker is the standard for inter-department data sharing. Inter-department services deployed using APSB can be easily leveraged by other authorized applications.
- The department owning the data is responsible for writing the shared service to access the data. This ensures data integrity and proper data interpretation.
- The department requesting the data is responsible for writing the request to retrieve shared data according to the shared service specifications.

Standard 2: Use the industry standard of ANSI Standard SQL when accessing relational databases.

- When using a database access tool that uses SQL calls, do not use any vendor specific extensions.

Technical Topic 5: Data Security

Introduction

The state's data is a very valuable resource, and establishing a secure data environment is a key component of the Statewide Technical Architecture, particularly since more and more applications use the Internet to access data. It is critical that the state's data be protected against any unauthorized access. Data security is designed to protect data against the following threats:

- Unauthorized use of the database or application.
- Accidental modifications and deletions.
- Confidentiality and integrity breeches for data in data transport and physical storage.
- Disasters.

Recommended Best Practices

The recommended best practices in this section pertain to Data Security.

Best Practice 1: Use generic, protected user accounts for direct database access to streamline administration, ensure scalability, and protect against non-application data access.

- When a generic, protected user account is used, each individual user account is not defined to the database, so end users are unable to gain ad hoc access to the data. Their only access should be through the application.
- The individual user account is only defined at the application level, and does not have to be maintained in more than one place.
- Implementing generic users makes applications scaleable since each process is not tied to a specific user.
- The generic user account and password used to access data in the back end must not be protected and not accessible to end users.

Best Practice 2: Implement data security to allow for changes in technology and business needs.

- Implement security to be a roadblock to unauthorized access, but not a hindrance to access by authorized users. Implement the minimal number of sign-on or authentication processes if possible.

- An adaptable security infrastructure must be implemented to allow for changes in technology, business needs, and reactions to intrusions.
- Monitor ITS and industry security alerts and recommendations. Security tools and techniques are rapidly changing and enhancements are being made. Monitor the industry and ITS recommendations and implement changes to security configurations as needed.

Best Practice 3: Handle sensitive data carefully.

- Confidential or private data must not be stored on a laptop without password protection or encryption. Laptops are vulnerable to loss of data through hackers, thieves, and accidents. Sensitive data must be secured on a database server with proper policies and procedures in place to protect the data.
- Ensure that passwords are encrypted both inside application executables and across the transport layer.
- Password and data encryption in databases and laptops can be provided by third party products.
- A backup and recovery plan for databases and laptops must be in place.

Best Practice 4: Provide measures for laptops to backup their data, like zip drives, etc.

- Only non-sensitive data should be stored on a laptop. If possible, the authoritative source must be on a server, and data should be replicated to the laptop.
- When data is stored on a laptop, provide easy-to-use backup facilities. Implement policies to ensure and automate backup.

Best Practice 5: Record information about users and their connections as they update and delete data. Auditing can determine who updated a record and their connection data.

The information that can be captured by the application includes:

- The user account the user logged in with.
- The TCP/IP address the connected user's workstation.
- The certificate information (if using certificates) about that user.
- The old values that were stored in the record(s) before the modification.
- The new values that were input to the record(s).

Best Practice 6: Implement transaction logging so recovery of original data is possible and protect the transaction log.

- Transaction logging records activity on the database and can be used to roll back a transaction.
- Protect the transaction log through access control and backup. Only the database should be writing to the transaction log. All other access should be read only.
- The transaction log should be located on a separate physical disk if possible. If not possible, use RAID to protect the integrity of the log file.

Best Practice 7: Implement security scanning and intrusion detection at the database level if possible.

- Scan the database and database server for potential weaknesses before they become a problem. Implement any recommendations of the security management tool. For example, a tool may advise to disable FTP services on a database server.

- Monitor the database for possible intrusions. For example, monitor and alert when multiple invalid login attempts occur. Intrusion detection protects the database server from attacks from both sides of the firewall (e.g., internal network, WAN, or Internet).
- Audit logins, user account creation, and failed login attempts.

Best Practice 8: Ensure data integrity by securing data movement or data transport.

- When high impact, sensitive data is transported through the LAN, WAN, or Internet, ensure that the data is encrypted and protected from alterations. This can be accomplished through Secured Socket Layers (SSL) or Virtual Private Network (VPN).
- Other types of data must be encrypted and protected if there is a risk of the data being altered.

Best Practice 9: Protect database servers from hardware failures and physical OS attacks.

- Database servers must be located in a climate-controlled, restricted-access facility, and preferably a fully staffed data center. Uninterruptible power supplies (UPSs), redundant disks, fans, and power supplies must be used.

Best Practice 10: Protect source code in data access rules, particularly if it contains password information.

- On the back end, an application needs to store account and password information in order to authenticate to a database or other application service. Protect the source code from unauthorized viewing.
- Store passwords in an encrypted format when possible.

Best Practice 11: Do not store credit card numbers in the database for non-recurring charges or infrequent recurring charges. Store authorization numbers and discard credit card numbers after use.

- For infrequent recurring charges (for example an annual fee), or non-recurring charges (for example a one-time fee), storing credit card numbers and expiration dates in a database, even encrypted, can present an unjustifiable risk for the state.
- A credit card number is only necessary to request authorization. Keep the credit card number only until authorization is complete, then discard. The authorization number can be used to track activity and verify authorization.

Best Practice 12: Protect and encrypt credit card numbers when storing for recurring charges. Store personal verification information independently.

- Certain business requirements, such as frequent recurring charges, may require credit card numbers to be stored in a database.
- When it is absolutely necessary to store credit card numbers, encrypt the credit card number in the database. To further protect the credit card, store personal verification information, such as name and address, in a separate database from credit card information. Use different user accounts for each database connection.

Standards

The standards in this section pertain to Data Security.

Standard 1: Change all default database passwords

- System administrator accounts have full access to all databases in a database server. Hackers often attempt a login to a system administrator account using a default password. As soon as a database is set up, change all default passwords.

Application Communication Middleware Architecture

I. Definition

Application Communication Middleware Architecture facilitates and simplifies communication within and between heterogeneous, distributed application systems. The focus of this chapter is limited to application communication middleware, as opposed to data access middleware or network middleware, which are separately discussed in the Data Architecture and the Network Architecture chapters respectively.

II. Introduction and Background

There are two areas that require application communication middleware:

***Intra-applicatio*n.** Handles communication *within* the tiers of an application system.

***Inter-applicatio*n**. Handles communication between the application system and *external* services, such as common shared services and other application systems

III. Principles

The following principles guide decisions on the use of application communication middleware.

Principle 1: Using application communication middleware is required in a heterogeneous, distributed environment.

- The tiers of a distributed application, which often run on different hardware and operating systems, must communicate.
- Application communication middleware enables both inter- and intra-application communications.

Principle 2: Using message-oriented middleware changes the fundamental design for building distributed applications.

- Messaging allows asynchronous processing so applications can continue processing after a message is sent.

Principle 3: Using remote procedure calls (RPCs) offer a good migration strategy.

- RPCs are the easiest transition for mainframe programmers. An RPC is simply a subroutine even though it is running a business rule on the network.

- RPCs are a mature technology. They are already bundled with many operating systems and databases.

Principle 4: Minimize the use of distributed units of work.

- Distributed transaction monitors are becoming less and less a requirement as high speed networks and messaging subsystems are deployed.
- The need for a transaction monitor can be eliminated or reduced by using features of message oriented middleware, combined with application design.

Principle 5: Do not use database middleware for application communication

- Database middleware has limited usefulness. It allows an application component to access data, thereby supporting a two-tier application design.
- Database middleware does not have the capability to provide all levels of inter-component communication. Stretching its use to inappropriate environments will ultimately result in systems that have performance problems.

Principle 6: Using a broker facilitates reuse and shortens development cycles.

- A broker provides access to common services that can be reused and shared, thus reducing development costs. See Componentware Architecture.
- The state can reduce the resources spent on developing and maintaining “islands of applications,” which include redundant code. Application developers can focus on new work rather than rework.
- New applications will be a combination of new business rules and common shared business rules. Since part of the application is “pre-written” and “pre-tested,” delivery of the total application should result more quickly.

Principle 7: Precede selection of application development tools with an application communication middleware strategy.

- In the long term, use of middleware by many applications is of more strategic importance than any one application development tool. Middleware selection should drive the choice of application development tools, not vice versa.
- A range of communication methods is available through middleware. A combination of products may be required.

Principle 8: Select third-party middleware rather than middleware supplied with a development tool.

- De-coupling the middleware from the application development tool provides more flexibility in changing development tools in the future. For example, integrated CASE tools often provide third-party message oriented middleware as well as their own, proprietary message oriented middleware.
- When given the choice of proprietary middleware versus third party middleware, select the third party middleware option. For instance, message oriented middleware provided by the

integrated CASE tool vendor limits flexibility and links to a specific vendor and product strategy more closely.
- If message oriented middleware is linked directly to a specific development package, then there is the risk of limited usefulness with other applications that are not developed with the same tool.

Principle 9: Document application programming interfaces (APIs) and interface definition language (IDL).

- APIs and IDL for components and services must be documented so that developers know where they are and how to use them.

IV. Technical Topics

Technical Topic 1: Application Communication Middleware Types

Recommended Best Practices

The recommended best practices in this section apply to application communication middleware types.

Best Practice 1: When possible, design applications to use asynchronous communication.

- Message oriented middleware supports asynchronous communications.
- Asynchronous messaging requires a distinctly different design. It is implemented with a very basic set of message oriented middleware commands.
- Message oriented middleware provides a reliable form of communication.
- Asynchronous communication offers more flexibility than synchronous communication. The downstream application has more control over its operation.

Best Practice 2: Use Remote Procedure Calls (RPCs) when message oriented middleware is not available.

- RPCs provide an acceptable, albeit limited, method of communication between software components.
- RPCs require synchronous communication and are less efficient in the use of resources; they tie up resources from both the client and the server until the service has been provided.
- Synchronous communication requires error handling in the client application if the request is made while a server is unavailable.

Standards

The standards in this section pertain to application communication middleware types.

Standard 1: There is no Remote Procedure Call (RPC) standard. Use the Andhra Pradesh Service Broker (APSB)[1] for inter-application communication.

- Even with an RPC that is endorsed by a vendor neutral party, such as The Open Group, there is no standard RPC.
- RPCs are available from different vendors, such as The Open Group's DCE RPC, Sun Microsystems'
- ONC/RPC, and Microsoft's RPC.
- Each vendor's version has a different application programming interface and they do not inter-operate with one another.

Standard 2: There is no Message Oriented Middleware (MOM) standard. Use the state of Andhra Pradesh's service broker for inter-application communication.

- At present, all message oriented middleware is proprietary. Products from different vendors have different application programming interfaces, which do not inter-operate with one another.

Standard 3: There is no distributed transaction processing (TP) monitor standard. Use the state of Andhra Pradesh's service broker for inter-application communication.

The applications coordinated by a transaction monitor will run on different platforms with access to different databases and resource managers.

The applications are often developed using different tools and have no knowledge of one another.

Industry standards specify how a TP monitor interfaces to resource managers, other TP monitors, and its clients.

- X/Open XA specification defines specifications for two-phase commits that work with distributed databases.
- X/Open TX standard defines transactions.
- X/Open X/ATMI provides a standard transaction management interface.

Technical Topic 2: Application Communication Middleware Brokers

Introduction
For communication external to the application or access to common services, another technology component called a "broker" is required to establish the relationship between the applications. This broker performs the same function as a real estate broker or stock broker: the broker brings parties together. Brokers are built on top of other communications middleware (e.g., RPC and MOM).

Statewide Recommended Broker Strategy
Implementing an Object Request Broker (ORB) statewide would very likely require a significant investment. An ORB solution would require integrating message oriented middleware, communication protocols, or operating systems in order to provide a complete, statewide solution.

[1] To be formed as a part of the e-Governance initiative

Service brokers are a proven technology. However, there are no industry-standard service broker products available. Successful implementations of service brokers usually require the interface to be developed specifically for the organization. Like other organizations `using a service oriented architecture, the state must invest in the development and maintenance of the service broker's generic interface.

Since object technology promises the greatest gains in productivity and adaptability, the state will ultimately transition software development to a fully object-oriented approach. Requests for service will be conveyed by an Object Request Broker. In the meantime, the state will use a service-oriented architecture, with requests conveyed by a Service Broker.

To assure their long-term contribution to the state, it is important that services developed to support the service-oriented architecture be designed in such a way that they can be accessed via either a Service Broker or an Object Request Broker.

Recommended Best Practices

The recommended best practices in this section pertain to application communication middleware brokers.

Best Practice 1: Manage a statewide broker as a strategic infrastructure component.

- The service broker is a critical part of the distributed computing environment because it allows the technical architecture to meet the three goals of efficiency, sharing of information and Department autonomy.
- Strategic infrastructure benefits all departments and should be centrally managed.

Best Practice 2: Be sure a statewide broker is independent of code development tools.

- The purpose of the service broker is to facilitate communication in a multi-platform, multi-language environment. If the service broker is tied to a single vendor's product, then the goal of facilitating communication in a diverse environment has not been met.
- Implementing a service broker that supports multiple vendors' products helps protect the state from being negatively impacted by market forces.

- A best of breed approach should be taken when selecting the application communication middleware.

Best Practice 4: Use the state' s inter-application middleware, the service broker interface, for inter-application communication between state-developed applications. For interfaces with other applications, such as purchased packages or applications owned by other entities, use the Interface Engine.

- State-developed applications gain performance and flexibility by using the service broker for inter-application communication.
- In-house or out-sourced custom-developed applications requiring inter-application communication should be capable of using a service broker. Applications sharing or requiring services from external application systems should provide the capability to use the standard inter-application communication middleware architecture.

- In instances where the application code cannot be modified, such as purchased applications where the state does not have rights to source code, use the interface engine. For more information about application integration, refer to the Integration Architecture chapter.
- For more information on the Interface Engine, see the Integration Architecture chapter.

Standards

The standards in this section pertain to the application communications middleware brokers.

Standard 1: Use of the service broker is required for inter-application communication.

- The service broker was put in place due to the lack of standards for inter-application communication types such as RPC, MOM, and TP monitors.
- While the lack of standards is not an issue for development of any single application, it poses problems for communication between applications. The broker is proposed as a standard communication paradigm for inter-application communication.

Integration Architecture

I. Definition

Integration Architecture specifies how various automated applications operating on different platforms can effectively work together. Integration techniques should be used when new application systems need to access existing application systems, while maximizing the investment in existing systems and platforms. This chapter includes an introduction of integration, an explanation of application integration, data access integration and XML along with recommendations and standards for each component.

II. Introduction

Integration is key to bridging the gap between heterogeneous operational application systems while still maximizing the investment in existing hardware and client platforms. Integrating new client/server, adaptive, and distributed systems with existing systems while still optimizing performance, minimizing maintenance and utilizing existing platforms is a major technical challenge. When new client/server systems are developed, they need the ability to access business processes and data from legacy and purchased systems developed under different technical architectures or built in features for future systems to access and interact with these systems.

III. Principles

The principles in this section are designed to provide guidelines for Integration Architecture components that will be used throughout the state.

Principle 1: An Integration Architecture addresses the correlating components of data interchange, business processing issues, and end-user presentation.

- The Integration Architecture encompasses the multiple layers of new and existing systems and the middleware in between.

Principle 2: An Integration Architecture meets the needs of linking heterogeneous operational application systems while protecting existing investments.

- The Integration Architecture should take into account the need to use existing workstations, peripherals and existing transports to access existing and new applications.

Principle 3: When making integration decisions, the life span of the solution is a key factor.

- A temporary solution may be engineered very differently than a long-term solution. Cost and effort need to be taken into consideration when providing a solution that is only needed on a temporary basis.
- Short-term solutions are often hard-wired and often have low performance. They are designed to be replaced or easily removed. Cost and effort should also be considered for a short-term solution.
- Long term solutions must be standardized, adaptable, and engineered for high performance.

Principle 4: Integration Architecture relies on middle service tiers such as interface engines, database gateways, messaging, integration services, XML and third party tools.

- It is more cost effective and easier to maintain applications that use middle service tiers than to modify multiple legacy applications.
- New N-tier applications still need access to the legacy information stored throughout the enterprise.

Principle 5: Minimize the impact to existing application systems.

- To the extent possible, the Integration Architecture should enable new applications to use existing resources with minimal disruption.
- Where possible, use non-invasive techniques for integration.
- Integration requires good communications infrastructure. If the basic network infrastructure is not in place, a single integrated network of application communication cannot be achieved. (Refer to the Network Architecture chapter.)

Principle 6: Use statewide technologies whenever possible.

- To the extent possible, use the same technologies in the Integration Architecture that are used in the Statewide Technical Architecture.
- Limit the heterogeneity of the technology used in order to simplify integration and enable migration to future technologies.

Principle 7: Provide maximum flexibility to integrate heterogeneous systems when enhancing existing end-user functionality through the use of a middle service tier.

- Implement the middle tier with standards whenever possible.

IV. Technical Topics

Technical Topic 1: Application Integration

Introduction

An approach to integrating independently developed applications, such as legacy applications, purchased applications, and new client/server systems is through application integration.

An application interface can provide the following services:

- *Data translation and mapping.* Translates the different communications and data interchanges between two applications.
- *Transaction explosion.* If configured properly, an application integration interface can take one client transaction and spawn multiple transactions in remote applications.

Front-ending other applications. An interface can provide a single front end for integrating multiple application systems.

Recommended Best Practices

The recommended best practices in this section pertain to application integration.

Best Practice 1: Use application integration strategy for online transaction program (OLTP) application systems, not decision support systems (DSS).

- Data warehouses or other solutions should be used in decision support applications. (For more information on data warehouses, refer to the Information Architecture chapter)

Best Practice 2: Design an integration solution that does not write directly to an operational database

- Existing application logic or business rules should be used when updating an application database. An external user or application could inadvertently corrupt operational data.

Best Practice 3: Recommended priority of using components of application integration are interface engine first, middle ware systems second, direct program to program interface as third and last alternative.

- This will reduce integration effort substantially.
- The recommendation assumes that all three alternatives are applicable in a given situation.

Best Practice 4: Use direct program-to-program interfaces for high transaction volumes.

- Direct program-to-program interfaces pass only the required information between applications, so performance and throughput is at the optimal level.

Best Practice 5: When designing an application integration solution using an interface engine, give careful consideration to the design and planning of the application interfaces and connectivity.

- At the beginning of the design stage, involve application developers who are knowledgeable in the business rules and interfaces to each system that needs to be accessed.

- Some application systems may have multiple entry or exit points that can be used. If a non-invasive solution is selected, capitalize on using the entry or exit points that best apply to your application needs.

Standards

Standards in this section pertain to the application integration.

Standard 1: Clearly Define Application Interfaces

- To integrate applications for which the state has no source code rights, application interfaces must be clearly defined in order to allow reliable communication between applications.
- To facilitate purchase of best-of-breed software while easing application integration issues, the application interfaces must be clearly defined.

Standard 2: The message structure must be documented.

- A message or transaction is the mechanism for extracting data from an application or sending data to an application.
- Programmers integrating applications need to know record length and type (i.e., whether it is a variable or fixed length record, and if it's variable, the delimiting characters used to separate the fields), and know which fields are optional versus required.
- A description of the data for each field is also necessary.
- Explanations and examples of record formats and field descriptions are helpful and should be included.

Standard 3: The application must be able to transmit and receive messages using a client/server model.

- The client is the process that sends or originates the message. The server is the process that receives the message.
- Clients and servers may communicate using TCP/IP and sockets, or other communication protocols, such as Serial and FTP, as long as they perform the same transmit and receive functionality.
- Packetization characters, which identify the start and end block strings, and message acknowledgment format must also be provided.

Standard 4: Purchase line-of-business application software rather than custom developing it whenever possible.

- Purchasing line-of-business application software can permit the state to respond to business needs in a more timely manner than custom developing software.
- Published API's are insufficient because their use requires custom development of state applications and it may be impossible to interface two purchased applications. Use of an interface engine provides greater flexibility.

Technical Topic 2: Data Access Integration

Introduction

Data access is the accessing and sharing of data between legacy, new, and packaged applications. It can be accomplished through several types of data access including *data extractio*n, *data replicatio*n, and *data sharin*g.

Recommended Best Practices

The best practices in this section pertain to data access integration.

Best Practice 1: Use as few middleware layers as possible when implementing a database gateway

- Additional layers of middleware in between an application and the database gateway could hinder performance of mission critical applications. For example, an application that needs to access a database gateway can implement an ODBC middleware layer that ultimately accesses the gateway middleware. Application performance can be increased if the application was written to make direct calls to the gateway middleware, omitting the ODBC layer.
- If there are fewer middle conversion tiers, there are less operational layers to maintain in the event of maintenance or upgrades. For example, if there is a change to a application database location, or an upgrade or maintenance update to the middleware software, it can effect all end user workstations and servers that access that application.

Best Practice 2: Keep the integration strategy as simple as possible.

- The more complicated the strategy, the more difficult it is to maintain and change.

Best Practice 3: Code data integrity verification rules into the DBMS whenever possible, particularly when external users and programs will be writing data directly to the DBMS.

- Since most DBMS vendors can code triggers and rules into the database, it is recommended to use this technology wherever possible in order to ensure data integrity.
- For more information on databases, refer to the Data Architecture chapter.

Best Practice 4: Separate decision support systems (DSS) from online transaction processing (OLTP) databases whenever feasible.

- If this practice is feasible, it will reduce the impact of ad hoc and large queries from decision support systems onto production operational application databases that are used by online users for day-to-day operations.

Implementation Guidelines

Once the best method for data access has been selected, the following guidelines may apply:

Guideline 1: Implement a hub topology as opposed to distributed data access topology whenever possible.

The distributed data access topology is where each point-to-point connection makes sense by itself, but the infrastructure as a whole is a “tangled” mass of connections.

- The star or hub technology is less complex and easy to maintain.

Guideline 2: Use a database gateway technology to combine queries of SQL data with non-SQL data.

- A gateway allows an application to query legacy data.

Guideline 3: Do not use any vendor specific extensions when using a database gateway that uses SQL calls.

Use the industry standard of ANSI Standard SQL

***Guideline 4: Once the database gateway product is selected for use as an integration tool, use the gateway from the central IT location whenever possible,* particularly in situations where the requirement is to access application data from the central systems/locations.**

- Expertise is centralized so the application developers do not have to duplicate efforts and relearn the gateway technology in each department. The departments also do not have to retain personnel with gateway expertise in addition to hardware and software experts to maintain the system.
- Security is centralized and controlled in a unified manner for all departments. With centralized security administration, the effort to limit access to authenticated users of an application is reduced.
- Dedicated gateway servers can be used that are easily administered, monitored and controlled, which contributes to the state effort for performance monitoring, error recovery and disaster recovery.
- The state has to establish a central license agreement that would simplify the addition of users and allows the departments to share the cost of the gateway more economically.

Technical Topic 3: XML

Introduction

XML (extensible Markup Language) has developed as the de-facto standard for business to business exchange of data on the internet. It is extensible because it allows users to define their own data and document types. It is a markup language like HTML. A mark-up language is a mechanism used to identify structure. The language requires special markers called tags to be added to text documents to give some added meaning to the document.

Recommended Best Practices

Best Practice 1: Choose XML as a preferred mode for all application integration for new systems, wherever possible

- It is a global standard, meaning that tools and solutions to be used for developing applications will either be complying with it already or will comply in their future releases.
- This will significantly reduce the cost and effort for building and maintaining interfaces between applications when compared with similar non-standards based tools.
- Apply the normal cost/benefit analysis criteria while using XML

Best Practice 2: Developing the DTD/schemas can be a top down as well as a bottom up approach

- Some of the DTD/schema can be defined based on metadata, which have been already defined in the Data Architecture chapter. However, while developing and implementing other state applications, a number of DTD/schemas are likely to be defined and can be made available. These can be added to the standard DTD/schemas which the state can use.

Implementation Guidelines

Guideline 1: Check available/accepted schemas before developing them bottom up

- Globally, a number of initiatives are have been underway towards defining DTD/Schemas. These are initiated by various industry groups and address industry specific need for exchange of information. Considerable effort can be saved if some of these schemas can be found to be deployable by the state, either as-is or with some modifications.

Guideline 2 : Develop an organisation and associated processes for developing and maintaining DTD/schemas, statewide

- Developing DTD/schemas and maintaining them, updating them in view of changing needs of the state will be an ongoing process that will require constant monitoring
- Compliance and enhancements can be enforced better if the responsibility is centralised by the creation of a repository and an organisation to maintain it.

Guideline 3: Rely on the network other security infrastructure for building and enforcing necessary secure environment for exchange of data.

- The XML standard addresses data transformation only, hence it relies on other systems for security services.

Guideline 4: Monitor developments on the XML standards development forums.

- A number of initiatives are underway on the standards development front. By closely monitoring them, it will be possible to incorporate expected changes into future plans of the state.

Standards

Standard 1: Clearly define and publish DTD/schemas

- This will facilitate their use and re-use
- Give reference to those DTD/schemas which have been developed based on any other globally defined schema, since this will allow incorporation of future changes
- Wherever the DTD/schemas apply for intra-state application, and are not expected to be shared in the public domain, adequate care needs to be taken in publishing them.

Related Information

www.oasis-open.org – Organisation for the Advancement of Structured Information Standards, that creates XML standards
www.xml.org and www.w3.org/xml – the source of information on XML standards, schemas, tools,etc.
www.w3.org/TR/XSL – information on XSL
www.xbrml.org – source for work on business reporting using xml
www.hr-xml.org – source of information on HR related XML schemas

Network Architecture

I. Definition

Network Architecture defines a common, uniform network infrastructure providing reliable and ubiquitous communication for the State's distributed information-processing environment.

II. Introduction and Background

The Network Architecture specifies how information processing resources are interconnected, and documents the standards for protocols (for network access and communication), topology (design of how devices are connected together), and wiring (physical medium or wireless assignments). The Network Architecture defines a unified, high-speed statewide network based on open systems standards. The biggest benefit to a statewide network solution is the ability to efficiently share information processing resources across the enterprise. Sharing resources is a common theme in all aspects of the Statewide Technical Architecture because economies of scale and efficiencies in operation result from collaborative approaches to technology. When departments share common application services and data, they avoid duplicative efforts and costs. The key to successfully sharing these resources is a network connecting all state departments together in a way that reduces redundancy.
A statewide telecommunications network must be strategically planned, strongly backed, and expertly managed. This network must:

- Utilize standard communication protocols.
- Sustain and support high capacity and high performance communication.
- Be scaleable, reliable, and extensible.
- Provide a variety of advanced telecommunications functions.
- Smoothly integrate with other private and public communication networks.

III. Principles

The following principles are provided to guide the planning, design, and selection of network technology and services:

Principle 1: A single integrated wide area network (WAN) is the backbone of an enterprise architecture and supports a variety of communication requirements including voice, data, image, and video.

- It allows access to a wide spectrum of information, application and system resources regardless of location or business unit. Thus, access to resources can be obtained in a timely and efficient manner by appropriate requesters when and where needed throughout the enterprise.
- It expands the scope of an organization domain by allowing them to reach out to customers and suppliers through access to the Internet and through the provision of dial-in/dial-out services.

- It acts as the delivery mechanism for the distributed computing services required by the fast-paced, dynamic business.

Principle 2: Networks should be available seven days a week and twenty-four hours a day.

- Networks provide an increasingly important and necessary role in the execution of business functions and processes. The availability of the network seven days a week and twenty-four hours a day must be maintained in a consistent and complete manner.
- Networks consist of and rely on many interrelated and often highly complex components distributed across a wide geographic area. Failure of any single component can have severe adverse effects on one or more business applications or services.
- Reliable networks contain no single point of failure. Networks are comprised of many components, and are only as reliable as the weakest link. Reliability must be built-in, not added-on.
- Bandwidth must be sufficient to accommodate new and expanding applications, different types of data (e.g., voice, data, image, and video), and a variety of concurrent users.
- The network must support software distribution and installation to a widely dispersed user community.
- The network must be designed to minimize latency. Data must pass across the network in a timely manner so that business decisions can be based on up-to-date information.

Principle 3: A statewide network must be based on common, open, vendor-neutral protocols.

- An open, vendor-neutral protocol provides the flexibility and consistency that allows departments to respond more quickly to changing business requirements.
- An open, vendor-neutral network allows the state to choose from a variety of sources and select the most economical network solution without impacting applications.
- This approach supports economic and implementation flexibility because technology components can be purchased from many vendors. This insulates the state from unexpected changes in vendor strategies and capabilities.
- Applications should be designed to be transport-independent.

Principle 4: User access should be a function of authentication and authorization, not of location.

- All users must obtain authentication via a user identification method consistent with the standards and usage guidelines set by the enterprise.
- Authorization of users must be performed according to the security rules of the enterprise and the local business unit.
- In order to perform their job functions, users need to access services available from multiple sites within the enterprise, from a variety of public and private networks, and from the Internet.

IV. Technical Topics

Technical Topic 1: Local Area Network (LAN) Architecture

Recommended Best Practices

Recommended best practices assist department staff in the planning, design, implementation and expansion, administration, maintenance, and support of LANs. They are based on experience and proven results. They employ standards and practices designed to support a uniform LAN.

Best Practice 1: Networks must be positioned for future growth in traffic and expansion of services such as voice and video.

- The increasing investment of funds in network infrastructures dictates that the life span of each additional component or enhancement be as long as possible. This can be accomplished if the design supports current needs but includes an anticipated growth potential. For example, installing Category 5 cabling today to run a 10 Mbps network positions a site to upgrade to a 100mbps speed in the future without replacing the cabling.
- As businesses expand, networks expand. A flexible, open network design will allow a business to minimize the costs and disruptions of configuration management while providing timely and responsive network changes when and where required.

Best Practice 2: Configure all servers supporting mission critical applications, including desktop applications, to minimize service interruption.

- Select a computer constructed to perform as a highly available, highly reliable, fault tolerant server with such features as redundant disk arrays, network cards, power supplies, and processors.
- Select a server with sufficient growth capacity to accommodate the anticipated increase in application requirements over time.
- Formalize security, disaster recovery, and backup procedures to ensure the integrity of both the server and the application. Test those practices on a regularly scheduled basis.

Implementation Guidelines

Guideline 1: Configure the topology (physical wiring) in a Star pattern

- Star topology uses a central hub/switch to which each network device is connected.
- Problems with a connection in a star network only affect that one device.
- A star topology provides the capability to easily add and remove devices as necessary.
- A star topology responds well to dynamic infrastructure changes in order to meet the growing demands of data movement. With ever increasing demands of information movement, more data, secure paths, new paths, and faster access, a star topology allows different, changeable, connections.

Guideline 2: Use switched multi-segment design with managed hubs.

- The hub is an ideal point for network management due to its central location and because all network traffic flows through it.
- Network switches provide the ability to break a network up into smaller sub-network segments.

- Switches can be used in conjunction with hubs. They improve LAN performance. With switching, network traffic is balanced across multiple segments thus reducing resource contention and increasing throughput capacity.
- Switching allows networks to assign increased speed or performance capability to particular segments in order to respond to heavy usage or application requirements.

Standards

The following standards have been established to assist departments in the implementation of LANs. The goal is to employ only open systems based on industry approved standards, but a full complement of open standards does not yet exist for all components of LANs. Therefore, a combination of industry standards, de facto industry standards, mutually agreed upon product standards, and open standards are currently required to support the state's heterogeneous operating environment. All standards will be periodically reviewed.

Standard 1:

The standard for LAN cabling is Category 5, 6, or 7 Unshielded Twisted Pair (Cat 5 UTP, Cat 6 UTP, or Cat 7 UTP). Unless specific needs exist, such as high EMI or long distances, UTP should be considered for the horizontal runs in cable layouts.

- CAT 5/6/7 UTP can be certified to carry 10/100/1000 MBPS of data.
- It is an industry standard wiring plan and has the support of the IEEE.
- Wiring, cable, connector, and equipment vendors have standardized on this cabling.

Standard 2:

The standard for standard link layer access protocol is Ethernet, IEEE 802.3 Carrier Sense Multiple Access/Collision Detection Access Method (CSMA/CD).

- Widely accepted format.
- Reliable, the protocol has been used for years and is very stable.
- Scaleable, faster versions are currently emerging to help manage the increase of data flow.
- 1000BaseT Gigabit Ethernet has the bandwidth necessary to support the needs of future voice and video requirements.

Technical Topic 2: Wide Area Network (WAN) Architecture

Introduction

A WAN is used to connect distributed network sites via private or public telecommunication lines. It typically serves as a customized communication "backbone" interconnecting all of an organization's local networks with communications trunks that are appropriate based on anticipated communication rates and volumes between nodes.

Access to the Internet must be obtained from an Internet Service Provider (ISP). With the addition of Internet access, the local network or the enterprise Intranet obtains the ability to connect with other WANs and computer sites throughout the world. While this improves the enterprise's access to information and expanded customer bases, it also increases the enterprises need for security and improved management procedures.

Recommended Best Practices

These recommended best practices assist the state in the planning, design, implementation and expansion, administration, maintenance, and support of an interoperable statewide WAN architecture.

Best Practice 1: Develop one enterprise-wide network infrastructure that is centrally maintained and managed.

- A single uniform network infrastructure allows an enterprise to respond more efficiently when faced with requests by departments for WAN component upgrades and installation.
- A centrally developed and managed infrastructure provides a more cost effective use of infrastructure resources.
- Departments or business units should focus their WAN requirements on functional specifications such as level of service needed, throughput needed, and response time needed. The implementation of an appropriately responsive WAN should be a specialized function performed for the enterprise in its entirety.

Standards

In telecommunications, standards for products and services were created by the originating industry monopoly (i.e., the phone company). Therefore, even though the monopoly has been disbanded, the proven standards that were established have remained. With data communications, however, there have always been many companies offering individual products and services. Therefore, although interim product standards have emerged as one company's product gained market share, there has been a lack of industry level standards. Therefore, until industry standards are established, an enterprise must choose to implement product based standards in order to create a manageable solution to the maintenance and management of its data communications infrastructure.

The following standards have been established for the implementation of department-based components to connect with the statewide WAN. A combination of industry standards, de facto industry standards, and open standards are currently required to support a heterogeneous operating environment.

Standard 1: The standard protocol technology is TCP/IP.

- Open protocol.
- Allows Internet access.
- Allows creation of Intranets and VPNs.

Standard 2: The standard internet access technology is Domain Name System (DNS) and IP address assignments are provided by the State for those departments participating in the Andhra Pradesh State Wide Area Network (APSWAN).

- State must assign IP addresses to allow LANs access to the AP State WAN.
- It allows a structured naming conviction and IP address allocation for the state's WAN and domain names.

Technical Topic 3: Network-Centric Applications

Recommended Best Practices

This section describes rules of thumb for planning, designing, and managing applications and application components in a networked environment.

Best Practice 1: Include network expertise on the requirements and design teams.

- Including network expertise ensures correct planning, documentation, and standard practices are followed.
- Requirements definition should include application performance, as well as capacity planning for network usage (based on the predicted number and size of transactions).
- Define any special networking requirements or constraints and perform the associated network design before development tools are selected. Otherwise, the tools used may not support the network architecture required to support the business.
- The network can be modified (upgraded) while applications are under development. Performance and the cost to move information should be balanced during application design. Multiple perspectives of a cross-functional group can ensure all viable options are considered.

Best Practice 2: Design network-neutral applications.

- Isolate the application code from the network specific code so business rules and data access code can be redeployed on a different platform, if necessary.
- Code to a middleware API, not to the network API.
- For a network to remain scaleable and portable, applications must be developed without regard to the type of network (i.e. WAN or LAN) they are to be deployed on.
- Network-specific design (e.g., wireless or guaranteed high-bandwidth) should only be performed when business requirements dictate.

Best Practice 3: Minimize data movement.

- When possible, schedule heavy network use for off-peak hours. For example, where requirements for data freshness permit, perform database synchronization at night.
- Data warehouses typically are used for decision support applications requiring large amounts of data to be transferred through the network.
- When replicating databases, consider partitioning and distributing subsets, rather than duplicating the entire master database.
- Decoupling the application layers provides the most efficient use of network resources by allowing the data access layer to be placed near the data.

Best Practice 4: Consider the impact of middleware on network utilization.

- Perform all transaction commits locally, between the resource manager and the queue. Asynchronous store and forward messaging can limit the scope of a transaction.
- Decouple transactions as allowed by business rules. Reconcile data at low-cost times.

Using store and forward, work can occur at a site even if the network link is down.

Best Practice 5: When data has to be distributed to multiple points (e.g., software and content distribution), move it once and only once across each data link.

- Use push technology, rather than using client polling. It overloads servers and network links to servers.

- Use multicast, rather than broadcast, to distribute messages to multiple points.

Best Practice 6: When designing distributed applications, make no assumptions about the speed of the network on which the application will be deployed.

Since bandwidth is unpredictable at design time:

- Minimize the amount of data to be moved between components. This will enhance performance regardless of the speed of the network on which the application is deployed.
- Use asynchronous rather than synchronous communications between application components (except in cases where business rules require synchronous communications). This will prevent application components waiting for a response from a server.
- For users and application requests that may be intermittently connected, use store-and-forward messaging to communicate with application components.
- When multiple, independent units of work must be performed, initiate all so they can be performed in parallel, rather than waiting for the completion of one before initiating the next.

Best Practice 7: Perform performance measurement and load testing on distributed applications before deployment.

- Measure application performance often, especially before and after any component is moved to a different platform. This helps quantify the performance impact of the redeployment, and helps isolate any problems associated with a network link or platform.
- Use load testing tools that simulate many users accessing the application. This testing method will provide information that will not surface during single user test scenarios.
- Load testing will identify network bottlenecks (and application bottlenecks) before the application is deployed in the production environment.

Best Practice 8: Deploy heavily used data sources "close" to the applications using them.

- "Close" does not imply physical proximity. It means deployed on platforms that have high-bandwidth connections between them. Do not perform heavy data movement across the WAN during peak hours.
- One of the biggest cost factors in designing a network is the transmission of the data over the communications system.
- For applications requiring very large amounts of data movement, try scheduling the execution of these queries to run during off peak hours to minimize the impact on network performance.

Implementation Guidelines

Guideline 1: Use asynchronous rather than synchronous communications between application components (except in cases where business rules require synchronous communications).

- Asynchronous communications will allow faster application processing, because the application is not waiting on a server response.
- Will allow applications to be used on "slower" WAN links.
- Work can occur at the application site even if the network links is down.

Guideline 2: Where business rules allow, use off-peak hours for scheduled data transfers.

- Allows better utilization of network resources.
- Keeps large data transfers from impacting normal operations and WAN/LAN traffic.
- Will allow commits of a day's worth of work to be processed at one time increasing server response.

Guideline 3: Code applications to middleware APIs where there are no specific business requirements. (e.g., wireless)

- Makes the application network neutral.
- It isolates the application code from the network specific code so business rules and data access code can be redeployed on a different platforms, if necessary.
- Allows networks to remain scaleable and portable.

Platform Architecture

I. Definition

Platform Architecture identifies hardware platforms and associated operating systems supporting the state's business.

II. Introduction and Background

The Platform Architecture describes the platform requirements for building a client/server infrastructure as well as the storage architecture associated in maintaining the data generated. Technical topics to be discussed under platform architecture include client architecture and server architecture.

III. Principles

The principles listed below provide guidelines for the design and selection of platform technology components that will support distributed, client/server computing activities across the state.

Principle 1: Design servers with bias toward granularity in physical servers.

- Using multiple servers from the same vendor with the same operating system release is cost effective because a group of uniform servers is easier to manage and integrate across a wide geographic area and multiple departments.
- A highly granular, loosely-coupled server design supports modular application code sets in an N-tiered application architecture.

Principle 2: Design mission critical systems without a single point of failure.

- Distributed systems can be designed to be extremely robust.
- Small granular servers make it easier to replicate services for increased availability.
- Systems should be designed to permit continued operations, albeit at reduced throughput, when a server fails in normal operations or in the event of a disaster.

Principle 3: Design all servers implementing a particular application, application suite, or tier within an application with binary compatibility.

- With binary compatibility, there would be no need to recompile an application for different platforms. For example, if an application that is going to be deployed on servers located in registration department offices, all servers running that application should be binary compatible -- this must be ensured even if the platforms are from the same manufacturer. The platforms must run the same version of the operating system and must not require any recompilation of the line of business application to deploy from one office to another.
- Total binary compatibility will support automated software distribution across servers and associated strategies which reduce support costs and provide stable computing platforms that can be reliably shared across departments.

Principle 4: Utilize open, vendor-neutral systems standards, wherever possible.

- Open, vendor-neutral systems standards provide flexibility and consistency that will allow departments to respond more quickly in an environment of changing business requirements.
- Vendor-neutral systems support economic and implementation flexibility.
- Vendor-neutral systems also protect the state against unexpected changes in vendor strategies and capabilities.

Principle 5: Design servers to allow multi-tasking and multi-threading.

- Multi-tasking achieves better CPU utilization.
- Multi-threaded processing enables a server to respond to multiple user requests more efficiently.
- These features also facilitate session management. Fewer sessions to manage provide a more scaleable solution. Multi-threading usually provides capability to execute *more sessions* i.e., more users can run the same application simultaneously, or several threads of the same application can run simultaneously. The ability to run *more* sessions or threads would demonstrate a more scaleable solution.

Principle 6: Design servers to be field upgradeable.

- Rapid changes in business processes are enabled in part by implementing a platform technical infrastructure that exceeds the immediate application requirements. This means departments should purchase servers with larger chassis so they are able to be expanded more easily and cost effectively.
- Field upgradeable servers provide maximum flexibility and adaptability for growth and new functionality.

IV. Technical Topics

Technical Topic 1: Server Platform Architecture

Introduction

In a distributed information-processing environment, there are many different types of servers that support unique activities across multiple platforms. Three key components of servers are file and print servers, application servers, and database servers.

Recommended Best Practices

Recommended practices that assist in the selection, maintenance and expansion of an interoperable statewide server platform architecture are listed below.

Best Practice 1: Run mid-range application and database servers on a 32-bit multi-tasking, multi-threaded operating system, at a minimum.

- Migration from 16-bit operating system platforms to 32- or 64-bit operating system platforms will support faster processing, access to more memory, and better memory and process management.
- In an N-tiered, client/server environment, speed, memory capacity, and memory and process management become increasingly important as processing is distributed across platforms.
- The 32-and 64-bit operating systems provide more stable, reliable platforms in an N-tiered, distributed client/server environment.

Best Practice 2: For reliability and ease of support, place each major application on a uniformly configured server. This may require that each major application be implemented on its own server.

- Use the same reference configuration on these servers. Important items to consider when planning for consistency include using the same versions of network software, using the same network hardware cards, etc.
- Tuning performance through configuration changes can make overall maintenance more difficult. In the long run, it may be less expensive to buy more powerful hardware than it is to spend time on individualized tuning and maintenance.
- The Network Operating System should be considered a major application and run on its own platform.

Best Practice 3: Consider normal anticipated future application growth when determining capacity requirements for server platforms.

- A server platform should be purchased that will accommodate the current demand as well as support anticipated normal growth without requiring the purchase of a new server chassis.
- Rather than purchasing a fully configured server, purchase the next larger size platform to allow for expansion. This will permit upgrades to an existing platform to accommodate growth rather than forcing the purchase of another machine.

Best Practice 4: Balance business adaptability and ease of systems management with server platform choices. However, when there is a conflict between business adaptability and ease of systems management, the business requirement for providing adaptability should have the highest priority.

- These two goals will always be in conflict.
- The primary design point of the technical architecture is to provide for change in business operations and its supporting applications. Therefore, even though it is easier to manage a large server rather than multiple smaller servers, the business need to provide flexibility should take precedence over any marginal increases in operational costs.

Implementation Guidelines

The following implementation guidelines pertain to the server platform.

Guideline 1: Make server platform decisions after the business makes some basic determinations regarding growth, scalability, portability, and openness.

- *Growt*h: Must the technology accommodate substantial growth beyond present data and transaction volumes?

- *Scalability*: In accommodating growth, must the technology be able to start small and grow by continuous small increments? Alternatively, is it acceptable for the technology to grow in major, discontinuous steps?
- *Portability*: Will it be necessary to move software across server platforms at some point in the future?
- *Openness*: What are the business implications if a proprietary system is used, thus eliminating the option to choose system components from many vendors?

Guideline 2: Consider several criteria when selecting a server platform.

- *Packaged software* availability.
- *Ability* to meet business needs.
- *Adherence to state standards and direction.*
- *Cost*.
- *Availability of skill sets* for development on the appropriate platform and for management following implementation.
- *Availability of technical support*.
- *Availability of systems management tools for the platform.*
- *Service terms* and conditions.

Standards

Standard 1: Run Distributed application servers on platforms supporting "open" operating systems.

- Open operating systems *are available* from multiple vendors, such as UNIX
- Open operating systems *run on* hardware available from multiple vendors, such as Windows NT.
- Open operating systems are in the public domain, but have significant industry support, such as Linux.

Standard 2: Make sure server platforms are POSIX compliant.

- POSIX is an IEEE standard designed to facilitate application portability and interoperability. This facilitates movement of applications from one platform to another if needed.

Standard 3: Make sure server platforms comply with third party certifications:

UNIX	Microcomputers
Manufacturer is ISO 9002 certified	Manufacturer is ISO 9220 certified
XPG4 Branded UNIX 93	Gartner Group Tier 1 or Tier 2 classified

Third Party Certifications for Server Platform Standards

- Third party certifications foster quality product purchases from manufactures that have demonstrated abilities to deliver and support these products.

Technical Topic 2: Client Platform Architecture

Recommended Best Practices

Recommended best practices in this section assist in the selection, maintenance and expansion of an interoperable statewide client platform architecture are listed below.

Best Practice 1: Use open standards based host-controlled client platforms where standards exist.

To minimize the risk of interoperability problems and maximize the reusability of devices and services, select products based on standards. Where no standards exist, consider the following guidelines:

- Choose a device and host software that is already in use elsewhere in the enterprise.
- Consider other potential uses for the device and host software in the enterprise.

Best Practice 2: Ideally, client platform choices should satisfy both end-user ease-of-use and ease of systems management. When there is a conflict between end-user ease-of-use and ease of systems management, give priority to end-user needs.

Best Practice 3: Choose client platforms that support personal productivity and connectivity. This may require multiple client configurations to support business needs.

- If personal productivity applications are run on the desktop, a 32-bit or 64-bit operating system is required. Migration from 16-bit operating system platforms to 32- or 64-bit operating system platforms supports faster processing, access to more memory, and better memory and process management. The 32- and 64-bit operating systems provide more stable, reliable platforms in an N-tiered, distributed client/server environment thus reducing the number of system crashes caused by lack of client resources. In addition, 64-bit operating systems provide better performance for process intensive applications such multi-media and engineering (CAD) applications.

Best Practice 4: The client platform displays the interface to an application. In the design of applications, minimize dependency on a particular client platform as much as possible.

- Web browsers support multiple platforms.
- 3-tier and N-tier application architectures, using "thin" clients, reduces dependence on a particular client platform because the user interface is isolated from application code

Implementation Guidelines

Following are the implementation guidelines for the client platform architecture.

Guideline 1: When selecting a client platform, consider the financial viability of the vendor, the availability of packaged software, the ability to meet department needs, adherence to state standards and direction, cost, availability of skill sets for development on the platform, support availability, and service terms and conditions.

- Technology and technology companies come and go. Consideration of vendor viability and availability of support, service, and skill sets mitigates risk associated with a dynamic and volatile market place.

Guideline 2: Migrate from character cell interfaces to graphical user interfaces.

- A graphical interface is more natural for users and is more effective.

Guideline 3: Use existing standards for user interfaces, such as Open Look or OSF Motif for UNIX Workstations, Windows GUI for personal computers or web browser for workstations or personal computers.

- This approach increases reuse, makes user training easier, and enhances the maintenance process.

Standards

The standards listed below have been established for the client platform architecture.

Standard 1: Two-dimensional (2-d) bar codes should use PDF417 coding standard.

The PDF417 bar code standard is used by the Department of Motor Vehicles. It is capable of storing data such as product information, maintenance schedule, shipping information or others.

Standard 2: Platforms must comply with third party certifications.

Client platforms must comply with third party certifications as specified in the table below.

UNIX	Microcomputers
Manufacturer is ISO 9002 certified	Manufacturer is ISO 9002 certified
XPG4 Branded UNIX 93	Gartner Group Tier 1 or Tier 2 classified

Third Party Certifications for Client Platforms

This will assure quality of platform hardware and software components.

Standard 3: Avoid proprietary smart cards reader-side APIs.

No standards exist for smart card reader-side APIs for application and platform integration. Use reader-side APIs from established platform vendors, such as PC/SC for the windows environment or use APIs that strictly adhere to the ISO 7816/4 command set.

Technical Topic 3: Storage

Introduction

Platform choices also influence storage selection criteria such as capacity, transfer rate, and cost of ownership. The data transfer rate of a given technology is an important consideration, with users trying to accommodate limited backup windows (the amount of time per day or per week that the site can take the system offline for backups). A small number of companies have implemented a triple redundancy disk solution so their systems never have to be taken offline for backup.

Standards

Standard 1: Use either SCSI or FC-AL technology for the disk drive interface.

Standard 2: Use RAID with fault-tolerance in the storage subsystems.

Standard 3: SAN based fibre channel technology for large scale storage deployments running mission-critical applications to be the defacto standard.

Security and Directory Services Architecture

I. Definition

Security and Directory Services Architecture identifies criteria and techniques associated with protecting and providing access to the state's information resources. It facilitates identification, authentication, authorization, administration, audit, and naming services. The state's technological resources must be available to users across the enterprise regardless of location or platform. Therefore, the state must implement security and directory services in such a manner that its information infrastructure is protected and accessible while, at the same time, its functionality is unimpeded and its business services are readily available.

II. Introduction and Background

The purpose of security is to protect and secure the state's information resources in order to provide an environment in which the state's business can be safely transacted. A directory is a natural place to centralize management of security. It is the vault that contains the most trusted and critical components of an enterprise security strategy. This will require authorization and authentication services and a common enterprise repository of digital certificates that secures and supports E-commerce applications. Security services apply technologies to perform the functions needed to protect assets. Historically, such services have consisted of door locks, vaults, guards, sign-in/sign-out logs, etc. As the state performs more business functions electronically, it must transition to security services designed to protect the electronic environment. For example, the use of face-to-face identification must be superceded by an equivalent electronic method that does not require the physical presence of the person.

III. Principles

Principle 1: Apply a level of security to resources commensurate to its value to the organization and sufficient to contain risk to an acceptable level.

- Security is a business enabler with associated costs. Security costs should be rationalized to the intended benefits.
- Requirements for security vary depending on the information system, connection to other systems, sensitivity of data, and probability of harm.
- Each transaction type will have individual security requirements.
- Security costs potentially increase beyond the value of the assets protected. Don't use more security than is required.

Principle 2: Resetting security assurance levels should not require modification of the architecture.

- Requirements for security vary depending on nature of communication, sensitivity of data, risks to the enterprise.
- Security services should be granular enough to accommodate assurance levels required.

Principle 3: Provide infrastructure security services to enable the enterprise to conduct business electronically.

- An architecture that defines an integrated set of security services permits state departments to focus on the business goals rather than on the implementation of security.
- Integration of security services will enable interoperability and provide flexibility in conducting electronic business across and beyond the enterprise.
- Integration will reduce the costs of protecting the state's resources.
- Integration will increase the reliability of security solutions.
- Centralized Directory services

Principle 4: An accurate system date and time are essential to all security functions and accountability and must be maintained.

- The validity of digital signatures and electronic transactions depends on precise, reliable date and time information.
- Audit accountability relies on placing events sequentially according to date and time.

Principle 5: Base application security on open standards.

- Security services will be provided as infrastructure services. In order to take advantage of security services, application security must be designed for open standards. A clear migration path should be defined for products not yet capable of integrating with the infrastructure security services.
- Products from vendors are often implemented in ways that make it difficult to integrate these products into an overall security architecture.
- Clear identification of integration issues should be part of the design process. If necessary, a migration path should be defined. When selecting software requiring security, selection criteria must include:
- Strict Adherence to open standards, such as X.509v3 Certificates, SSL, S/MIME, LDAP, and SASL.
- Avoiding platform-specific implementations that inhibit integration.

Principle 6: Locate security in the appropriate layer of a communications protocol to ensure maximum usability with minimum future modification.

- Whenever security is required, the location in a communications protocol will have an impact. The impact may be on performance, reliance on an underlying network protocol, and on developers. Choosing the appropriate layer in a communications protocol will maximize usability and minimize future changes.
- Security services can have an impact on performance. The impact is minimized when security services are located at lower layers of a communications protocol.
- Security services can have an impact on developers. For example, services provided at the transport layer have less impact on application programmers than services that run above that layer.
- Security services can increase reliance on a network protocol. An appropriate choice depends on the communication requirements of the business system.

IV. Technical Topics

Technical Topic 1: Identification

Introduction

Identification is used to distinguish one user from all others. Identification techniques provide a means of gaining entry to the state's resources such as workstations, networks and applications. Identification is closely linked to authentication. Authentication is the process of verifying the identity of a user and is covered in the following section.

Recommended Best Practices

The recommended best practices in this section pertain to security identification.

Best Practice 1: Use risk management techniques when considering biometrics identification.

- Biometrics is an emerging technology.
- Biometrics techniques may vary in success in a real environment. Testing under real conditions may be necessary to determine effectiveness.
- Application integration with biometrics is hampered by a lack of standard APIs.
- Biometrics identification complements and can be integrated with other security techniques such as digital signatures, smart cards and encryption.

Standards

The standards in this section apply to security identification.

Standard 1: ISO 7816 Smart Card standards for contact smart cards.

- ISO 7816/1-4 standards define the electrical resistance, positioning of electrical contacts, communication protocol between card and card reader, and command set recognized by smart cards.
- They correspond roughly to the OSI layered model.
- The command set defined by the ISO 7816-4 standard are included in whole or in part by most smart cards on the market.

Standard 2: ISO 14443A and Mifare Smart Card standards for contactless smart cards.

- ISO 14443A standards for contactless smart cards define the characteristics and communication protocols between contactless cards and card reader. These standards are still in development.
- The Mifare architecture is the de facto global interface standard for contactless and is based on ISO 1443A.
- Contactless cards under this standard use RF power and frequency protocols and cover read/write distances up to 10cms of the reader.

Standard 3: Use PKCS #11 or PC/SC for integration of smart cards and host/reader-side applications.

- PKCS #11 from RSA is a widely accepted standard for integrating smart cards to applications supported by many vendors.

- PC/SC is widely accepted for integration of smart cards on Intel platforms.

Standard 4: Speaker Verification API (SVAPI).

- SVAPI is an API used for incorporating speaker-recognition technology into desktop and network applications.
- A consortium of vendors, technology developers, researchers VARs and end-users developed the SVAPI.
- The SVAPI offers interoperability over distributed environments with related APIs.
- They include SAPI, the telecom industry's S100, a standard architecture for developing computer-telephony applications, and JavaSpeech, a standard for speech recognition using Java.

Standard 5: Human Authentication API version 2.0 (HA-API).

- The Human Authentication API (HA-API) is a generic API designed to allow a common set of instructions to integrate biometrics into applications requiring identification.
- It supports the enrollment sampling, processing and verification of biometrics.
- The API supports multiple biometric template types and multiple vendor technologies for each biometric type in one database. This permits an enterprise wide approach to biometric identification while allowing different application-specific biometrics to be used. A single database also facilitates the use of multiple biometrics in a single application.
- The API permits changing the biometric used without requiring application code changes.

Standard 6: Use open standards for smart card masks such as MULTOS.

- Highly secure procedures from manufacturing to card issuer
- Allows multiple applications on the same card, addition and deletion at any point of time during the life of the card
- High application level security
- Manufacturer independent mask supported by several card and chip manufacturers

Technical Topic 2: Authentication

Introduction

Authentication is the act of verifying the identity of a user or process. Authentication answers the question: "Are you who you say you are?" The most common method used to authenticate a user is a password. A password is a secret series of characters and numbers associated with an individual user id by the owner/user.

Recommended Best Practices

The recommended best practices in this section pertain to security authentication.

Best Practice 1: Authenticate users prior to accessing services.

- Allowing only authenticated users to access system resources protects those resources from inappropriate access.
- Authenticating users is the basis for providing accountability.

Best Practice 2: Use Public Key / Private Key technology for authentication when digital signatures are required.

- Public Key / Private key technology is the most widely accepted form of digital signatures.
- Digital signatures are central for most electronic business.

Best Practice 3: Use token-based or strong password based authentication where public key certificates are not feasible.

- Token-based systems are an improvement over passwords.
- Where token-based identification and authentication is not possible, a password policy based on best practices can provide an acceptable level of security.

Best Practice 4: Use an enterprise-wide public key infrastructure.

- Collaboration and co-operation will be required to support security services across the enterprise.
- A unified approach to a Public Key infrastructure enables the state to respond to changing requirements and conditions.
- A fragmented approach to a public key infrastructure will complicate administration and management of security across the enterprise.

Standards

The standards in this section pertain to security authentication.

Standard 1: Public Key Certificates (X.509v3)

- Public Key authentication must be based on Public Key Certificates.
- Public Key Certificates must be based on the X.509v3 standard.
- Despite the widespread acceptance of this standard, care must be taken when dealing with vendors. Projects should require proof of interoperability with existing or proposed enterprise implementations using X.509v3 certificates. Proprietary extensions to certificates could inhibit interoperability and should be avoided.

Technical Topic 3: Authorization & Access Control

Introduction

Authorization answers the question: “Are you allowed to do what you are asking or trying to do?” Access to applications, the data they process and database modifications must be carefully controlled. Authorization is the *permission* to use a computer resource. Access is the *ability* to do something with a computer resource. Access controls are the technical means to enforce permissions. They allow control over what information a user can use, the applications they can run and the modifications they can make. Access controls may be built into the operating system, may be incorporated into application programs or major utilities, or may be implemented in add-on security packages that are installed into an operating system. Access controls may also be present in components that control communications between computers.

Recommended Best Practices

The recommended best practices in this section pertain to authorization and access control.

Best Practice 1: Authorize users based on least privilege.

- Authorize users to the minimum set of resources appropriate to their role.
- Authorizing users on least privilege minimizes the impact of security violations.
- Authorizing users to a minimum set of resources necessary to their function makes it easier to establish accountability.

Best Practice 2: Use appropriate security service levels for each part of the technical infrastructure according to enterprise-wide standards.

- Identifying the necessary security service levels allows appropriate choice of a security mechanism.
- A subdivision of infrastructure along security requirements will minimize security management and response to changes.
- A basic level of communication security will reduce the number of applications that must *be security-aware*.

Best Practice 3: Use open standards-based security solutions.

- Security implementations vary widely. Use of proprietary solutions may make it difficult to adapt to advances in security and standards development.
- Security management across the enterprise requires a consistent and open standards based implementation of security solutions.

Implementation Guidelines

The implementation guidelines in this section pertain to authorization and access control.

Guideline 1: Secure transmission of data where appropriate.

- Data in transit to and from the enterprise must be protected in compliance with legal requirements for confidentiality and privacy.
- Web-enabled applications must protect confidential or critical data from unauthorized access.
- Use secure server-to-server communication to protect confidential or critical data transmission.

Guideline 2: Avoid Virtual Private Network (VPN) solutions for connecting trading partners outside the enterprise that are not IPSec compliant.

- VPN solutions today are proprietary. All outside trading partners are unlikely to use the same or similar technology.
- Most transactions can be done with SSL.
- VPN solutions should be chosen on compliance with IPSec and inter-operability among IPSec compliant VPNs.

Guideline 3: Web-enabled applications that require user authentication should use SSLv3 with client authentication and client public key certificates where appropriate.

- For certain payments over the Web, for example credit card purchases, SSLv3 without client authentication is sufficient protection for client and server confidentiality.
- For purchases or changes to state data, which mandate user authentication, SSLv3 with client authentication should be used.

Guideline 4: Use encryption for stored data or email only when appropriate.

- Encrypted data or email incurs management and performance overhead.
- Encrypted data incurs high overhead to encrypt and decrypt.
- Managing encrypted or archived encrypted data requires effective key recovery and escrow schemes.

Standards

The standards in this section pertain to authorization and access control.

Standard 1: Secure Sockets Layer version 3 (SSLv3)

- SSLv3 is the most commonly supported protocol for communication between Web Server and browser.
- It authenticates the Web Server and optionally authenticates the user browser.
- Current implementations allow for client authentication support using the services provided by Certificate Authorities.

Standard 2: IP Protocol security extension (IPSec)

- IPSec is an extension to the IP communications protocol, designed to provide end-to-end confidentiality for packets traveling over the Internet.
- IPSec works with both the current version of IPv4 and the new IPv6 protocol. IPSec has two modes: sender authentication and integrity but not confidentiality through the use of an Authenticating Header (AH), and sender authentication and integrity with confidentiality through the use of an Encapsulating Payload (ESP).

Standard 3: Cryptography must be based on open standards

- Cryptographic services identified in this document are based on open, industry accepted, standards.
- The following business requirements and associated cryptographic standards have received wide acceptability and can be found in most products. Only full strength cryptography should be used. For example browsers are often supplied with weakened versions such as 40 bit DES, RC2 and RC4. Only browsers with full strength keys should be used for transactions involving the state. Cryptography with variable length keys should use a minimum key length equivalent to 56 bit DES.

Cryptography Algorithm	Standards
Public Key / Private Key	RSA (1024 bit keys), ECC (160 bit keys)
Secret Key	DES, 3-DES, RC2, RC4, IDEA, CAST (minimum DES equivalent or full length keys)
Message Digest	MD5, SHA-1

Figure 11-13. Cryptography Standards

Standard 4: Use S/MIME version 3 for securing email communications.

- S/MIMEv3 provides a consistent way to send and receive secure email including MIME data.
- S/MIME defines a protocol for encryption services and digital signatures.
- Email clients should be evaluated for support of the standard and for interoperability.

Standard 5: Services provided through the Internet (Web-enabled applications, FTP, Mail, News, DNS, etc) must be placed on the DMZ or proxied from the DMZ.

- Application services must be protected from unwanted external access and must be located on a DMZ or proxied from the DMZ.
- All communication from servers on the DMZ to internal applications and services must be controlled.
- Remote or dial-in access to the enterprise must be authenticated at the firewall or through authentication services placed on the DMZ.

Technical Topic 4: Administration

Introduction

All organizations experience change. Keeping security systems synchronized with that change is essential. For example, employee additions, transfers and resignations must be reflected rapidly. Administration of security in a distributed environment is a complex task. This task includes the means to administer user accounts, privileges, authentication and security policy implementation.

Recommended Best Practices

The recommended best practices in this section pertain to security administration.

Best Practice 1: Because security control impacts the entire enterprise, its implementation must be easy to administer, verify, and sustain.

- Administration of user identification, authentication and authorization is required to protect the enterprise.
- In order to sustain security it must be easy to administer.
- The security implementation must be verifiable to ensure continued reliability of the state's IT infrastructure.

Best Practice 2: Identify security policy domains.

- The enterprise is one security policy domain with a specific security policy that must be implemented. Other domains may be executive departments or county and local government.
- Establishing security domains simplifies the analysis of security requirements and focuses attention on security policy requirements.
- Identifying security domains allows policies to be applied at the appropriate locations in the architecture.
- Security policies may vary between domains requiring protective measures or gateways to traverse differences in policies.

Implementation Guidelines

The implementation guidelines in this section pertain to security administration.

Guideline 1: Use role-based administration and multiple security domains.

- Role-based administration and multiple security domains are easier to administer and maintain than user-based privileges and single enterprise security domains.

Technical Topic 5: Directory Services

Introduction

The business of the state is becoming more distributed. The state is developing closer electronic partnerships with businesses outside of state government, some employees are mobile users, some employees are working from their homes, and state services are being brought closer to the citizen electronically. This will require authentication services and a common enterprise repository of digital certificates that secures and supports E-commerce applications. Additionally, the state's technological resources must be available to users across the enterprise regardless of location or platform. To meet these goals, an enterprise directory services infrastructure must be in place. The state's technological resources and users are defined to the enterprise directory and appropriate access controls are applied. A directory is a natural place to provide security and it is the most important function it offers. It is the vault that contains the most trusted and critical components of an enterprise security strategy.

Recommended Best Practices

It is necessary to implement an enterprise directory services strategy in order to improve communications between disparate systems. When planning a project that includes directory services, the strategy must be based on the following best practices to assure its success:

Best Practice 1: Implement a fault tolerant solution to provide 24-hour, 7-day availability to the enterprise directory.

- If the directory becomes inaccessible, the resources to which a user has rights become unavailable. Therefore, a directory must be available at all times to accept authentication requests. This can be accomplished with a planned fail-over strategy to ensure that, if one server fails, another backup server can pick up the requests. This should include a replication strategy with hardware solutions that include disk or system duplexing, disk or system mirroring, disk arrays, and UPSs.

Best Practice 2: Purchased applications and operating systems should be directory-enabled.

- Securing applications and their operating environments is a significant challenge. Security is a natural environment for the use of a directory. Applications can authenticate users to an external source by being directory enabled. The directory is better suited to provide information on the level of security necessary. Applications can be further enhanced when they are enabled to obtain an expanded set of information from the directory as appropriate. Thus making applications more modular and consolidating administration to a central location. For example, an application can gather employee information from the user object in the directory. This facilitates user authentication and authorization by making the resources on that platform available to the enterprise, when the appropriate rights are in place.

Standards

Standard 1: Use the statewide directory services infrastructure.

Using the statewide directory services has several benefits:

- The infrastructure is simplified by providing a common interface to view and manage all available resources.
- Directory services are a critical component to statewide initiatives like E-mail and Electronic Commerce. The current enterprise directory is fault tolerant and highly available from any location that participates. Time, distance, and location do not restrict access to the information contained within the services.
- Coordinated directory services will improve communication between our applications, databases, and network operating systems by providing consistent, reliable information in an efficient and effective manner.

Standard 2: Integrate homogeneous directories into a single tree.

- It is more efficient to link "like" directories into a single tree. Most vendors of directories have implemented either the standards that currently exist or standards that have been proposed. These standards include a mechanism to connect their directories together to build a single tree. This provides the optimum integration of Public Department resources and people without regard to location. A single tree minimizes infrastructure costs while maximizing the potential for departments to choose how they share resources with other departments including local governments. The single tree approach also allows for improved fault tolerance and better performance especially for departments with geographically dispersed operations. Joining a tree, regardless of manufacturer, must be coordinated with Information Technology Services.

- It is necessary to tie these single trees from various manufacturers to the authoritative enterprise directory in order to provide authentication services to the authoritative enterprise directory. However, achieving connectivity from one manufacturer's directory to another is complex and difficult. For example, tying one Netscape directory to Novell's NDS can be done but is difficult to implement and maintain. The state currently has dozens of Netscape directories in place. The process would then need to be performed for each of them. However, tying all Netscape directories together into a single tree is fairly straightforward and facilitated through their product. Then the one Netscape tree can be tied to the one NDS tree. It is a complicated task but it is performed once. Through the Enterprise Directory Services Initiative, this interoperability between dissimilar directories will be implemented. This will be accomplished through the use of meta-directory technologies.

Standard 3: Use the Andhra Pradesh Service Broker (APSB) services for directory functions.

- As in-house applications are developed, we must make use of the services that are already available rather than to constantly build new ones. An enterprise directory services infrastructure provides an addressable security service for authentication and authorization as well as a repository for digital certificates.
- These services are addressable directly from the enterprise directory or through a service via the Andhra Pradesh Service Broker. For more information about the Andhra Pradesh Service Broker, refer to the Componentware Architecture chapter.

Standard 4: Use the Centralised Metadata Repository directory schema attributes and object classes.

- A directory is basically a database that has been tuned to perform massive reads and infrequent writes. Like other databases in our enterprise, directories and their elements must be centralised. For example, where a person object class may have an attribute of “Pager Number”, “Pager Number” should be registered in the Centralised Metadata Repository and populated according to that definition. Therefore, when the directory is queried for that information, the data returned will be as expected. In the past there has been a tendency to populate currently unused directory attributes with data that is not consistent with that attribute. For example, there may be a requirement to enter a pager number in the directory for a user. If there is no attribute for “Pager Number”, there may be a tendency to select an attribute that is unused such as “Title”. Instead, extend the schema to include a new attribute that precisely defines the data that will be placed there and register it with the Centralised Metadata Repository. Do not store inconsistent information in an unused attribute.

Standard 5: Populate directory objects according to the minimum attributes defined in Distributed Computing Standards and Guidelines.

- Any data source is only as good as the data it contains. If that data is missing, incorrect, or incomplete, the data source cannot be depended upon as an authoritative source for that type of information. A directory is no different. Directories have become much more than an authentication point for network users. In order to supply information on our users, network devices, and organizations, directories must be built in as complete and reliable manner as possible.

Standard 6: Use Lightweight Directory Access Protocol version 3 (LDAPv3) for directory access where strong security is not required.

- LDAPv3 is the industry standard lightweight access protocol and does not offer strong authentication or access controls. However, LDAPv3 can provide standards based access to directories for lookups, as a communication mechanism for synchronization tools, public key retrieval, and others. Commercial off-the-shelf (COTS) applications often require their own directories. Access to the application directory from outside or for the application to communicate with an external directory will require a standards based approach. Therefore, when purchasing COTS applications, LDAPv3 compatibility is required. LDAPv3 also provides a standards based access to the directory for lookups, as a communication mechanism for synchronization tools, public key retrieval, and others.

Systems Management Architecture

I. Definition

Systems Management Architecture defines the framework for efficient and effective management of the state's distributed information processing environment in order to support and enhance the productivity of its automated business systems.

II. Introduction

The state's Systems Management Architecture is the framework that identifies the requirements for managing and supporting the enterprise-wide technical architecture with primary emphasis on centrally managing distributed systems at geographically disbursed sites. Resources managed include the systems, databases, applications, networks, and Internet components necessary to conduct the automated business functions of the state.

III. Principles

Principle 1: Business needs should have priority when making systems management decisions.

- System management must facilitate the business process. Business unit needs should play a primary role when identifying requirements and selecting technology and applications. Business units are assuming a larger role in driving technology selection and its application.
- Whenever a business need conflicts with a systems management need, the business need must take priority.
- Business units should have as much autonomy as possible to select applications that meet their needs. As long as the business functionality justifies the cost and the business unit is willing to pay the price, then the selected application is acceptable. Support costs should be considered by the business unit.
- To support business processes, systems management must focus on increasing system stability and availability while reducing costs. It can achieve these goals by setting standards, establishing guidelines and centralizing systems management functions along business functional lines.
- Centralization/standardization should occur within a business function. However, a single standard does not apply to all lines of business. For example, all operators using the same system should have a minimum standard hardware and software configuration to meet their needs. Operators using other systems may require different tool sets to meet the needs of their unique business applications. Configurations can be different, however all configurations can be based on the same architectural components.

Principle 3: Limit the amount of "unique" performance tuning to existing individual network components, particularly servers and desktops.

- Performance tuning for unique/non-standard components is not worth the increased maintenance costs of multiple configurations.
- Performance tuning can inhibit change by encouraging comfort with the status quo.

- It may be cheaper to increase performance by upgrading to an architecturally compliant hardware configuration than to spend time tuning an application.

Principle 4: Increase capital investment when it offsets long-term support costs.

- Identical configurations are easier to support. In the long term it may be much more expensive to support multiple types of configurations than it is to invest in replacing them with consistent configurations.
- Purchasing hardware that exceeds the immediate need often saves money in the long run, as it promotes expandable systems and reduces the need for tuning and support.
- It is more cost effective to use capital dollars to improve operations than to spend support dollars on outmoded technology. The cost of continuing to support an aged configuration is often higher than the cost of new equipment that will improve performance and reduce support costs.
- The practice of using “hand-me-down” equipment perpetuates obsolete technology and can greatly increase the support burden by increasing the number and kind of devices requiring support and its associated costs.

Principle 5: Utilize open, vendor-neutral standards whenever possible.

- Open, vendor-neutral systems standards provide flexibility and consistency that will allow departments to respond more quickly to changing business requirements.
- Vendor-neutral systems support economic and implementation flexibility.
- Vendor-neutral systems also protect the state against unexpected changes in vendor strategies and capabilities.

IV. Technical Topics

Technical Topic 1: Help Desk

Recommended Best Practices

The following practices are recommended to develop a service-oriented help desk.

Best Practice 1: The help desk and user support functions must be re-engineered to provide an integrated support services environment.

- The central help desk provides the focal point to mediate problems.
- Support tools should empower both the help desk analyst and the end user with self-help capabilities.

Best Practice 2: A single consolidated help desk design supports an enterprise model.

- A consolidated help desk does not have to be physically located in one place. However, it should have one constituency, one phone number, one set of procedures, one set of defined services, and one set of integrated network systems management (NSM) platforms and applications.
- The implementation of the virtual data center (VDC), where many remote LANs are managed as a single entity, supports the corresponding development of consolidated help desk services.

Best Practice 3: Each centralized help desk unit must provide a single point of contact (SPOC).

- A SPOC minimizes user inconvenience and confusion. In its broadest sense, SPOC means that the end user makes one attempt at contact and the help desk request is channeled by some automated means to the organization that can best service the request.
- The help desk should mediate all problems.

Best Practice 4: In order to leverage support resources and provide effective client support, multiple tiers or levels of client support are required.

- Tier/Level 1 client support should have end-to-end responsibility for each client request. The help desk analyst should be empowered to resolve as many requests as possible. Tier 1 provides the client contact point (CCP) or call ownership, which is the single point of contact for the end user to request a service. Organizations should retain control of the Tier 1 help desk in order to ensure the quality of the customer relationship.
- Tier/Level 2 client support provides advanced technical expertise to the tier/level 1 client contact points. Their responsibility is to analyze the requests routed to them and resolve the problems. Resources at this level can be composed of staff specialists and/or third party providers/vendors.
- Tier/Level 3 support is composed of highly specialized technical experts. Calls which cannot be solved at tiers/levels 1 and 2 are routed to this level. Resources at this level can be composed of staff specialists and/or third-party providers/vendors.

Best Practice 5: Geographically dispersed help desk units must inter-operate and share information.

- All requests for service should reside in a database that is shared by technology and application-based help desk units serving specific constituencies throughout the state. This process shares information and makes it possible for one help desk to electronically pass a service request to another help desk without forcing the user to make another contact attempt.
- The use of technological advances, such as distributed processing, dynamic control of users desktop, improved telephony, and client support software, make it possible for geographically dispersed help desk groups to function as a cohesive support unit.

Best Practice 6: Resolution databases that contain solutions to recurring problems should be built to improve service quality and contain costs.

- Building and using a knowledge base of prior resolutions to solve problems improves the quality of resolutions.
- Help desk operations should include problem resolution links to external systems.

Implementation Guidelines

Guideline 1: Consolidate help desk services when common functions are being supported across business units

- Support resources can be leveraged more effectively when support for common functions are consolidated.

Guideline 2: Link support functions together electronically

- Service requests or problems often begin with one help desk and require the services of another help desk. Support hand-offs and tracking are made more effective when linked electronically.

Guideline 3: Provide a single point of entry per constituency.

- A single point of entry function, responsible for resolving customer needs and routing cases to the appropriate function, reduces customer issues associated with locating and obtaining support.

Guideline 4: Identify common data elements. Enable electronic interchange of problem and solution information.

- Common data elements facilitate the electronic interchange of problem and solution information because translation of data definitions is not required and information can be more easily re-used.

Technical Topic 2: Operations Management

Introduction

Encompasses the coordination of system and network resources throughout the enterprise. Its goal is to provide reliable availability for mission critical systems. It includes job scheduling to coordinate jobs and processes in the distributed environment, fault/event management, configuration management, backup and recovery and automated software distribution.

Recommended Best Practices

The following practices are recommended for successful operations management.

Best Practices 1: Equipment deployed in virtual data centers must be configured to facilitate remote management and support.

- The VDC should be configured to prevent a single point of failure.
- Identical configurations of rack-mounted servers are placed in secure locations (closets).
- For reliability and ease of support, each major application should be placed on a uniformly configured server. This may require that each major application be implemented on its own server.
- Use the same reference configuration on these servers. Important items to consider when planning for consistency include using the same versions of network software, using the same network hardware cards, etc.
- Systems management tools, consistently applied, allow management of multiple instances of the identical network configurations at remote sites as if they were on the data center floor.
- The VDC should support mission critical applications.

Best Practice 2: Systems management functions for the virtual data centers should be remotely performed.

Some examples of remote systems management services include:

- Backup, archiving and recovery
- System, database and application monitoring
- Software distribution to the server and/or desktop

Best Practices 3: Under the Virtual Data Center concept, responsibilities of customers for systems management are limited.

- Even though the equipment is located close to the customer community, for the most part, local user efforts should be concentrated on performing their business functions rather than on system management tasks such as system configuration, debugging and/or backup.

Best Practices 4: System components should proactively alert in advance of failure including predictive capability.

System generated alarms and alerts should be automatically routed to the appropriate systems management resource. For example:

- Database problems should be routed to the database support group.
- PC hardware problems should be routed to PC support.
- Agents should be able to issue alerts for both hardware and applications.

Best Practices 5: Inventories of hardware and software configurations should be maintained real-time.

- Inventories of configurations are critical to support functions
- Inventory capability requires ‘agents’ on workstations and servers.

Implementation Guidelines

Guideline 1: Centralize remote systems management for mission critical applications

- Systems management components are infrastructure and can be leveraged to provide common functionality to multiple business functions.

Guideline 2: Implement products that use standard protocols and interfaces.

- Use of standard protocols and interfaces promotes interoperability among management products and can reduce required core management infrastructure e.g., management consoles and products.

Guideline 3: Use integrated management suites. Use best-of-breed point products when the integrated management suite cannot substantially meet the business requirements.

- Use of integrated tools reduces the costs associated with implementation and support. Common methods and procedures are used for bringing managed objects into the management framework and for ongoing management of these objects.

Guideline 4: Use a Relational Database Management System (RDBMS) as the underlying store for managed objects, policies, events, and alerts.

- Use of an RDBMS facilitates access to management data for functions beyond those provided by the management framework e.g., query and reporting tools.

Standards

The following standards have been established to support operational systems management for the enterprise.

Standard 1: Use SNMPv1 (simply called SNMP) protocols.

- The Simple Network Management Protocol (SNMP) is a group of internet protocols that is the standard for managing TCP/IP based networks.
- It is built into the devices (e.g., concentrators, routers) in the network and in the network operating systems of the servers and workstations.
- The network management system uses SNMP to collect statistics and other information on network devices.
- SNMP is also used to send commands that control the state of network devices.

Standard 2: Use Remote Monitoring (RMON) products.

- RMON products are predicted to become increasingly used in most enterprise networks.
- RMON products provide packet collection, decoding and analysis to the MAC layer of the Operating Systems Interconnection (OSI) stack using a combination of consoles and hardware and software probes that relied on SNMP MIB data collections.
- In 1992, the Internet Engineering Task Force, IETF, specified the RMON1 standard in RCF 1271. The RMON1 MIB extends SNMP capability by monitoring sub-network operation and reducing the data collection burden on management consoles and network agents.
- The RMON2 standard was approved by the IETF in January 1997 in RCF2021. RMON2 includes a new MIB to extend network monitoring into the application monitoring layer.
- RMON functionality is growing to include functions like applications monitoring, report generation and bandwidth allocation.
- All major network device vendors have added RMON MIB collection capability to their products, although the depth of implementation relative to the full RMON specification varies among vendors and products.

Standard 3: Conform to the Desktop Management Interface (DMI) standard.

- The DMI standard was developed by the Desk Top Management Task Force (DMTF), which sets specifications for the management of the desktop environment.
- The DMI is a set of API's that allow different vendor applications to consistently share the desktop.
- It sets the standard for a management platform that enables a common standardized mechanism for systems management of the desktop while permitting vendor differentiation.
- As vendors build desktops with embedded DMI standards, important desktop management information will become available from the newer desktop units.

Technical Topic 3: Storage Management

Introduction

The use of storage management to control the costs of expensive hardware resources has been in place on host systems for many years. In these situations, a strong approach to storage management has allowed IT departments to maintain a minimal amount of hardware expense and best utilize the storage space available for more immediate uses. IT departments have taken

advantage of other less expense alternatives (ie. Tape, Fiche, etc.) for storing older data that is not accessed as often. In many cases, the process for retention and storage is automated to the point of being able to access Terabytes of data quickly without having the high cost of storage to maintain it.

Recommended Best Practices

The following practices are recommended as guidelines for developing a storage management process within distributed systems.

Recommended Best Practice 1: Structure for audit and policy management

Systems and tools selected for distributed systems should be deployed with tools for auditing and managing storage processes. This would include the following areas:

- Auditing and Reporting on space usage, aging of data/files, and ownership of data
- Trend reporting for space usage to plan ahead for future needs
- Set policy guidelines and auditing for user retention of data

Recommended Best Practice 2: Implement Archival routines for aging of data

Leveraging archival of old data will reduce search and fine of current information, along with maintaining cost effective storage. This is appropriate for both user and system data.

- Implement consistent routines for removal of backup (.BAK, etc.) and log files
- Structure appropriate rights and expectations regarding public stores
- Outlines archival rules for old data, communicating the aging process and where backups of the data would be stored
- Verify the archival process maintains a minimum of 2 backups (separate media) of files before deleting these items. This will protect faulty recovery or failure of the media (ie. Tape break, etc.)

Recommended Best Practice 3: Design an accounting process around data

Future requirements for allocating costs and tracking against users/departments will require the use of an “accounting” tool for storage costs. This can then be used to “charge” for space in the future should business needs require this internal accounting.

- Initially, setup rules and processes for accounting of space usage (automated)
- In the future, design accounting process by user/group/business unit and leverage reporting to track costs of the systems towards their proper owner
- Leverage this information on the short-run for identifying high-cost owners and abusers of space requirements

Technical Topic 4: Performance Monitoring and Tuning

Introduction

Performance tuning in the host environments has been a very important requirement, as IT departments attempt to manage the high-cost devices put into their environments to keep users functioning at top speeds and capture issues early in their development.

Recommended Best Practices

The following practices are recommended as guidelines for the process of performance monitoring and tuning.

Recommended Best Practice 1: Develop and maintain simple designs

An attempt at tuning can often generate worse results when engineers try to make changes that are unproven. The focus of IT should be to minimize changes from the industry defaults/standards unless such change is warranted and proven as appropriate. Changes in defaults are often of little value and always result in future work to keep the standard in place.

Some issues to remember:

- Maintain default values unless tested and proven, or instructed from appropriate resources within the product vendor
- Document changes from defaults and reasoning for changes in a Change Log Manual
- Review changes semi-annually and determine if additional issues should be reviewed or updated

Recommended Best Practice 2: Structure an automated Report Card/Dashboard

To track performance and/or tuning results, interface all areas toward a central repository and owner, with appropriate authority granted to the person.

- Outline a common report cared for monitoring performance results, and provide online via Web-services
- Determine method of tracking a “dashboard” for current values and issues within the system. This can coincide with the report card but should be managed and implemented differently due to the timeliness of data from these mechanisms
- Publish SLAs for both expected results and actuals, and educate the staff on their meanings

Recommended Best Practice 3: Outline and Communicate the Maintenance Schedule/Process

All systems require periodic maintenance, enhancements and changes. Distributed systems required consistent maintenance also, which involves scheduled downtime. It should be communicated to the user community and followed appropriately to ensure the maximum availability for users and mobile staff.

- Outline the expectations and schedule for maintenance and communicate to the users
- Provide a change management process for testing and implementing maintenance that does not impact the users negatively and reduces risk of a poorly implemented change

Recommended Best Practice 4: Leverage the central console/control center

Performance monitoring is a 24x7 operation that requires skill development and significant investment to the structure. The control center must be staffed and equipped to do this analysis periodically. This would include:

- Performance trending, paying close attention to affects of spikes in activities
- Monitoring different activity impacts, based on hardware (diagnostic), wire (performance), internet (activity), and desktop (application) needs. These needs will vary and change based on user requirements
- Outline audit and security measures that identify proper processes for ill-advised activities and breaches of secure solutions

- Quarterly reviews of these trends, and a review of the analysis topics to identify new areas that should be monitored