PARCC TECHNOLOGY ARCHITECTURE

TECHNICAL SYSTEMS ARCHITECTURE

Version 1.1

November 5, 2012

REVISION HISTORY

The following revision chart defines the version history for this document.

Revision	Date	Released To	Description
1.0	09/30/12	PARCC States	Initial Release
1.1	11/5/12	PARCC Public Sharepoint Site	Released as support material for Assessment Administration RFP (IDOE 13-29)

CONTACT

Send questions about this document via http://parcconline.org/contact

Contents

FIGURES

TABLES

EXECUTIVE SUMMARY

This executive summary condenses the key facts and findings of this document into a more concise form. It is intended to provide a reasonably complete shorter form of the PARCC Assessment System technical systems architecture plan that may be read instead of the longer document. However, the executive summary does not necessarily reflect the full view of the technical systems architecture plan provided in the full document.

It follows the organizational structure of the document and includes references to key tables and graphics to help communicate important information.

TECHNOLOGY INFRASTRUCTURE PLAN

This section details the technical systems requirements for the PARCC Assessment System, including: deployment, system management and monitoring, middleware and integration, security integration with existing technical environments, and development and testing processes and environment.

The FURPS Model

The FURPS (**F**unctionality, **U**sability, **R**eliability, **P**erformance, and **S**upportability) model, a widely used tool for identifying and categorizing the requirements of any system, was used to categorize the high-level functional requirements for the assessment system defined in *High-level Application Architecture*. Table 1 – Requirements Categorized According to the FURPS Model on page 13 provides the breakdown of functional requirements by FURPS categories.

Hardware, Software, and Network Requirements

The PARCC Assessment System needs to be able to run on a variety of client-side and server-side hardware and software platforms. Its implementation must be able to meet key technology priorities defined in *Key Technology Priorities Summary*.

In making client-side hardware decisions for a state or district, all hardware decisions should be based on the instructional needs of schools and students. Districts should rely on local expert judgments and discussions with appropriate authorities to determine the hardware that is most appropriate to deliver quality instruction and to support individual students.

In making server-side hardware decisions, Windows and enterprise-class Linux distributions can be used in the assessment system. Both operating systems have licensing costs that include support options, which provide upgrades and security patches. Linux has an open-source license. The speed and size of the central processing unit (CPU) caches are of particular importance for overall server performance. Other factors which should be considered are supportability, compatibility, and virtual machine (VM) support.

Databases, Data Storage, and Data Archiving

There are three basic types of databases available in the market today: relational databases (RDBs), object-oriented databases (OODs), and NoSQL databases. They differ in the way the logical data entities are organized. RDBs are the dominant databases today, and, when properly designed and implemented, they can be used for many types of data workloads. However, both alternatives provide capabilities that might be useful in the assessment system.

Data Storage

There are many considerations when sizing and configuring the storage subsystem for the assessment system. Similarly to server choices, the decision-making in this area will be determined by the overall type of system deployment. If the system is deployed in a cloud infrastructure or in a third-party data center, many decisions will be determined by the cloud/hosting vendor offerings and the storage specifications will be negotiated and written into the contract.

- To provide optimal storage performance, the assessment system storage implementation should ideally use a storage area network (SAN) with low overall disk latency. For optimum performance, the SAN should include either solid-state drives (SSDs) or fast Small Computer System Interface (SCSI) disk drives.
- The Operational Data Store (ODS) and Data Warehouse (DW) components should use relational databases because of the stringent requirements for data accuracy and data integrity. Using a NoSQL database can certainly be explored and prototyped during the development of these components to determine the feasibility of this approach.

Data Archiving

Data archiving is the process of removing selected data, which is not expected to be referenced again, from an operational database and putting it into an archive data store, where the data can be accessed again, if needed. The data design, storage methods, and tuning requirements for an archive data store are different than those for an operational database. The PARCC Information Architecture and Data Governance processes will determine what types of data will be subject to data archiving, when data archiving will occur, how long the data will be retained in the archive, and when (if ever) the data will need to be destroyed.

Server-side Application Platforms

In the world of server-side application development, there are two major application development and deployment platforms: Java 2 Enterprise Edition (J2EE) and .NET. Both application frameworks provide a solid foundation for building enterprise-ready, robust server-side applications. Table 3 – Comparison of J2EE and .NET Technologies provides a summary view of these deployment platform options.

Besides J2EE and .NET, there are other popular application platforms (e.g., Ruby on Rails) that should be explored as possible platforms for development and deployment of loosely-coupled components.

Virtualization

Virtualization is the simulation of the software and hardware upon which other software runs. There are many benefits to using virtualized servers, the main one being increased operational efficiency, because existing hardware can do more work by putting a greater load on each computer. In addition to this benefit, desktop virtualization allows the same computer to run multiple operating systems, which can aid both development and testing efforts. Most virtualization environments allow changes to be reverted easily, unlike a physical server environment, in which changes can be hard to track and revert.

To ensure portability of the assessment system component repository, the assessment system components must be able to deploy and run on both physical and virtualized environments. A useful tool for comparing virtualization solutions is provided at www.virtualizationmatrix.com/matrix.php.

Network Requirements

Internet connectivity and network capacity requirements for the PARCC Assessment System will be fully defined once the development of the test items repository is complete and the designs of the assessment delivery platform are finalized. Output data from the Technology Readiness Tool (www.techreadiness.org) will also be taken into consideration when determining network requirements. A network bandwidth estimation model is presented, which approximates the network capacity needed per test per student in an effort to determine the network requirements.

Bandwidth usage may be a concern with tests containing items using video, audio, or other network-intensive media types. Content pre-loading techniques and HTML5 caching mode should be explored as options to reduce the network requirements for a test containing such items.

The Test Client component may mitigate the high-bandwidth items by using pre-loading or caching techniques.

Component Deployment Options

The internal components of the PARCC Assessment System need to be flexible in their deployment to provide the diversity of hosting options. Component deployment refers to how the component functionality is packaged and exposed to other components. Components are usually deployed within some kind of application server running under a particular operating system on a physical machine.

Depending on their deployment and how they talk to each other, two components can be tightly-coupled or loosely-coupled.

Tightly-coupled Components

Tightly-coupled components are usually deployed within the same application server, talk to each other via local calls, and often share common persistence storage (e.g., a database). There

are advantages to tight-coupling that include better performance and lower implementation costs. However, the disadvantages include reduced flexibility and reliability.

Loosely-coupled Components

Loosely-coupled application components are each deployed in a separate application server running on a separate physical machine. They typically talk to each other via Web service calls. The key advantage of loose-coupling is greater flexibility in technology choices. However, the disadvantages include generally higher bandwidth requirements and reduced performance.

System-level Deployment

The assessment system can utilize traditional or cloud-based system-level provisioning and deployment.

Traditional Hosting

Traditional hosting of the assessment system will require either the implementation of the full spectrum of internal IT infrastructure services or outsourcing those services to third-party hosting providers.

Cloud Hosting

Cloud hosting is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

There are many advantages of provisioning the assessment system in the cloud, as opposed to using traditional hosting options. Capacity planning, disaster recovery, availability, and reliability are system features that a reputable cloud provider will list as part of the cloud service contract. Costs can also be better allocated and managed, because typically the service is paid for only when it is actually used.

Further, there are models for deploying cloud hosting that provide different levels of management and control of the cloud infrastructure:

- Software-as-a-service (SaaS)
- Platform-as-a-service (PaaS)
- Infrastructure-as-a-service (IaaS)

Figure 12 – Service Models in a Cloud Infrastructure Diagram on page 41 summarizes the service models.

System Management and Monitoring

System management and monitoring will be an essential part of the technical administration of the PARCC Assessment System. It will also play an important role in maintaining a secure

environment and enforcing policies (e.g., authorization, privacy, and auditing) and standards compliance.

A system management and monitoring framework typically involves setting up monitoring agents on all monitored system entities (e.g., devices, machines), and a central management server that collects and processes the events generated by the monitoring agents. Management event information in the assessment system will be published via standard protocols (i.e., Java Management Extensions [JMX] or Simple Network Management Protocol [SNMP]) to a central management and monitoring server. This server will also perform health monitoring that involves collecting data representing the overall technical health conditions of the system and its components.

- SNMP is the most common system management and monitoring protocol used today. Most professional-grade hardware devices today come with a built-in SNMP agent.
- JMX is a Java technology for monitoring and managing the performance of the Java Virtual Machine (JVM) at run-time. It is applicable for Java and J2EE-based applications.
- Common Event Expression (CEE) is a framework that enables collaborative efforts in the creation of open, practical, and industry-accepted event interoperability standards for electronic systems.

There is a wide range of commercial and open-source monitoring and management tools that provide the necessary capabilities for the assessment system.

Middleware and Integration Software

The middleware layer in the PARCC Assessment System Architecture will be built using service-oriented architecture (SOA) principles and designs. A layered service approach will be used to package component functionality and expose it as services to be used by other components and services. The exposed services will be stateless, coarse-grained, and loosely-coupled. The three layers are:

- Service Component Layer
- Services Layer
- Business Process Layer

Web Services

Web services represent an increasingly popular technique for developing, deploying, and consuming services in an SOA infrastructure, enabling location transparency by utilizing registries such as Universal Description, Discover, and Integration (UDDI) for run-time discovery. The transport for Web services is HTTP/HTTPS. Clients can locate the desired service dynamically by requesting the service from the registry. The Web services architecture provides benefits of loose-coupling by providing a mechanism to find, bind, and invoke the service dynamically.

Middleware and Integration Software Vendor Capabilities and Offerings

There are numerous software offerings that facilitate the development of middleware:

- Oracle Fusion Middleware (www.oracle.com)
- TIBCO ActiveMatrix (www.tibco.com)
- OpenSAF (opensaf.org)

Security Requirements for Applications and End-user Access

The PARCC Assessment System must enforce stringent security checks and rules involving the operation of its applications, the storage and transfer of its data, and the controlling of end-user access.

End-user Authentication/Authorization and Access Control

All PARCC Assessment System end users will be authenticated to the system using a single sign-on process. *Single sign-on* (SSO) is the ability for users to access multiple software applications from multiple sources and vendors by logging in just once with a single username and password—preferably from any location. Security Assertion Markup Language (SAML) is an Extensible Markup Language (XML) standard that allows secure Web domains to exchange user authentication and authorization data. The SAML protocols for single sign-on will be used.

Once authenticated, the users will be authorized to perform specific functions across assessment system subsystems based on their assigned role. Each role defines what the user can access and the level of this access.

Regulatory Compliance

An important aspect of assessment system security will be regulatory compliance. There are two federal laws that relate to the security implementation of the assessment system, namely Family Educational Rights and Privacy Act (FERPA) and Children's Online Privacy Protection Act (COPPA).

Test Delivery Component Security Concerns

The Test Delivery component has specific security concerns that should be addressed. To prevent fraud and ensure the validity of the test, special requirements must be considered regarding the test environment. In order to provide a secure test environment using a Web browser as the test client, the desktop and operating system environment, where the browser is running, must be locked-down, so that students taking the test can access and control only one window on the screen (i.e., the one with the test).

There is a trade-off between the ability to satisfy all those security concerns in their entirety, and the implementation of the Test Client delivering the test.

Web-based Test Client Implementation

Standard Internet browsers (including popular browsers such as Internet Explorer, Firefox, Chrome, Safari, and Opera) have all been designed with great end-user interface flexibility and

convenience—but they all have very limited features for tightening the security of the end-user experience.

Native Application Test Client Implementation

In this type of implementation, the test client is actually a native application written specifically for the target operating system (e.g., Windows or Linux). Typically, the native application has its own data processing and data persistence capabilities, and can provide much greater control over the security of the desktop environment where the test will be delivered. However, there is a significant cost in terms of development, deployment, and configuration efforts when using a native application test client for delivering PARCC Assessment System tests.

Integration with Existing Technical Environments

The PARCC Assessment System will need to integrate with existing technical environments at the state, district, and school levels. Depending on the deployment model for some or all of the assessment system components, certain components can be deployed at the state, district, or school levels for increased performance (e.g., network bandwidth) or other considerations. External and/or existing components (e.g., state student information system [SIS] or item/test authoring systems) are always going to be deployed as per the specifications by the particular vendor that produces the component—which may or may not be the PARCC deployment level. Regardless of the component deployment level, interoperability among components will not be affected and will be executed according to the overall component interaction architecture.

Development and Testing Processes and Environments

While not required, it would be beneficial to PARCC if each vendor developing PARCC components followed development and testing processes based on established frameworks, tools, and methodologies.

Figure 20 – Recommended Assessment System Development Environment Layout Diagram on page 58 shows a recommended layout for the PARCC Assessment System development environment.

Figure 21 – Recommended Assessment System Testing Environment Layout Diagram on page 59 shows a recommended layout for a testing (validation) environment for the assessment system.

Integration Architecture Plan

The Integration Architecture Plan outlines guidelines and recommended approaches to integration, movement, and security of the PARCC Assessment System data, both inside and outside of the assessment system. It also provides a technology integration template to be used by vendors to ensure that their offerings comply with assessment system architecture.

Data Integration

The assessment system will need to integrate data from a variety of sources within the assessment system itself as well as external data sources (e.g., student information systems,

item and test authoring systems, scoring engines, and state-level data warehouses). As described in *Interoperability Standards Review*, industry-standard high-level data standards such as Accessible Portable Item Protocol (APIP), Question and Test Interoperability (QTI), and Common Education Data Standards (CEDS) will be used for data representation and data transfer between those systems.

Data Integration within PARCC Assessment System Components

Data in the PARCC Assessment System will be stored in two major data hubs: the Operational Data Store (ODS) and the Data Warehouse (DW). Individual components may opt to use their own independent data stores to keep transient data while the component is performing its functions.

Data Integration with Member States' Existing Systems

Existing student information systems at the state level will provide core student data and other data needed for the operation of the PARCC Assessment System. The test registration process executed through the Test Registration component will use the industry-standard Schools Information Framework (SIF) protocol to pull data from the state student information system. This could be implemented as either a real-time or an asynchronous batch process, depending on the availability of the state SIS.

Data Movement

The *Data Movement Model* section in the *Information Architecture* document outlines the different types of data produced and consumed in the assessment system as well as how this data moves through the different components and subsystems, both internal and external, using industry-standard data-exchange protocols such as APIP and QTI.

Data movement between assessment system components during real-time interactions, such as submitting authored item data from the Item Authoring component to the Item/Test Bank component or submitting items from the Test Delivery component to the Operational Data Store component, can be implemented via standard service-oriented technology using Web services (i.e., SOAP/REST).

Moving data from the Operational Data Store component to the Data Warehouse component would be best accomplished using an extract, transform, and load (ETL) tool. ETL tools are used to provide continuous or batch-level movement of data from one data source/data store to another.

Data Security

Data storage and movement in the assessment system need to adhere to applicable regulatory constraints (e.g., FERPA and COPPA). The necessary security mechanisms need to be in place when storing and moving most data entities, especially student data and test results data. Hashing and encryption techniques will be used when sensitive data is stored in all data stores, and secure data transfer protocols (e.g., SSL, HTTPS, and WS-Security) will be used when data is

transferred from one component to another. In addition, any transient data should be subject to periodic purging to minimize the risks of unauthorized access.

Table 5 – Data Security Life Cycle Phases and Activities on page 63 shows the data security life cycle phases.

API Design Guidelines for Vendors

The a*pplication programming interface* (API) is essentially the programming contract between two entities (i.e., systems, components, etc.) communicating with one another using an agreed-upon protocol. This protocol would specify, for example, the name of the operations, the sequence in which they execute, and the format of the data exchanged.

Vendors who will be developing external components interfacing with the PARCC Assessment System, as well as vendors who will be developing some or all of the internal components of the assessment system, need to incorporate a number of general guidelines when designing their components so that they will be compatible with the assessment system architecture.

REFERENCE SECTIONS

The remaining sections provide reference information to assist the reader:

- **Related Documents.** Lists the supporting PARCC Assessment System Architecture documents referenced in this document.
- **External Sources.** Lists the outside sources (including Web sites) used in the preparation of this document.
- **Terms and Acronyms.** Lists the acronyms used in this document along with their definitions.

1. INTRODUCTION

1.1 OVERVIEW

This document describes the technical architecture aspects of the PARCC Assessment System in the form of high-level requirements, recommendations, technical diagrams, design guidelines and specifications, and standard templates. It also addresses the topic of integrating existing technology environments at the state, district, and school levels into the PARCC Assessment System. This document contains these sections:

- **Technology Infrastructure Architecture Plan.** This section focuses on the technology infrastructure for the PARCC Assessment System. It begins by presenting the FURPS model, which is useful in analyzing technical system requirements, followed by an overview of the hardware, software, and network requirements. It then focuses on component deployment options; system management and monitoring approaches and tools; middleware and integration software; security requirements; integration with existing technical environments; and, finally, coverage of the development and testing processes and environments.
- **Integration Architecture Plan.** This section outlines the guidelines and recommended approaches to integration, movement, and security of the PARCC Assessment System data, inside and outside of the assessment system. It also provides a technology integration template to be used by vendors to ensure that their offerings comply with the assessment system architecture.
- **Related Documents.** Lists the supporting PARCC Assessment System Architecture documents referenced in this document.
- **External Sources.** Lists the outside sources (including Web sites) used in the preparation of this document.
- **Terms and Acronyms.** Provides definitions for the terms and acronyms used in this document.

1.2 PURPOSE

The purpose of this document is to address the PARCC Assessment System Architecture Deliverable *7.1.A.5.A/B: Technical Systems Architecture Plan* as defined in the Florida Department of Education *ITN 2012-22.* It provides technical architecture recommendations, requirements, and guidelines for the PARCC Assessment System. It is part of a set of documents that provide a high-level view of the assessment system.

1.3 SCOPE

This document focuses on section *7.1.A.5.A/B: Technical Systems Architecture Plan* in the Florida Department of Education *ITN 2012-22*. However, it also addresses topics in relevant sections from these additional documents:

- *Technology Architecture, Interoperability Standards Development and System Implementation Services – Technical Reply*
- *PARCC Assessment System Architecture – Work Plan for Part A*

The technical architecture recommendations, requirements, and guidelines in this document are based on PARCC Assessment System key technology priorities and architectural requirements as outlined in the following PARCC Assessment System Architecture documents:

- *Key Technology Priorities Summary*
- *High-level Application Architecture*
- *Information Architecture*
- *Interoperability Standards Review*

2. TECHNOLOGY INFRASTRUCTURE ARCHITECTURE PLAN

This section focuses on the technology infrastructure for the PARCC Assessment System. It begins by presenting the FURPS model, which is useful in analyzing technical system requirements, followed by an overview of the hardware, software, and network requirements. The section then focuses on component deployment options; system management and monitoring approaches and tools; middleware and integration software; security requirements; integration with existing technical environments; and, finally, coverage of the development and testing processes and environments.

2.1 THE FURPS MODEL

The FURPS model is a useful tool for identifying and categorizing the requirements of any system. The acronym comes from the names of the five categories used to classify the system requirements: **F**unctionality, **U**sability, **R**eliability, **P**erformance, and **S**upportability. These categories capture both functional and non-functional business requirements.

- **Functionality Requirements.** Define what the system must do, including the features and capabilities of the system, most often defined as use cases or user stories.
- **Usability Requirements.** Define the user interface requirements for the system, its navigation, look-and-feel, accessibility, online help facilities, and other visual and non-visual features.
- **Reliability Requirements.** Define the system's availability (i.e., uptime), the accuracy of the system's calculations, and the ability of the system to recover from failures.
- **Performance Requirements.** Address system behavior with respect to time and resources and define characteristics such as response time, throughput, and scalability.
- **Supportability Requirements.** Define the ability to monitor and maintain the system, and include testability, configurability, upgradeability, and ability to interface with external systems.

Table 1 – Requirements Categorized According to the FURPS Model illustrates the applicability of the requirements of the PARCC Assessment System to the FURPS model. Most of the functionality requirements for the assessment system are covered in detail in *High-level Application Architecture* in the form of business use cases that outline the high-level system flows and functionality. The table also lists the derived requirements from the *Key Technology Priorities Summary* document.

Table 1 – Requirements Categorized According to the FURPS Model

Requirement	F	U	R	P	S
Business use cases and high-level application requirements as captured in *High-level Application Architecture*.	X	X			
BR-01. The assessment system shall be based on an open architecture with well-defined data and interface standards.					X
BR-02. The assessment system shall be open, flexible, and scalable and easily integrate with other systems.					X
BR-03. The assessment system shall minimize bandwidth requirements.				X	
BR-04. The assessment system should be able to incorporate and interoperate with existing state and local systems.					X
BR-05. The assessment system should be accessible using a standard Web browser with an Internet connection.		X			
BR-06. The assessment system should operate with devices that comply with the Technology Guidelines for PARCC Assessments v.1.0.		X	X		
BR-07. The assessment system shall support multiple hosting options, and support components distributed at the school, district, state, or PARCC level.			X	X	
BR-08. The PARCC assessment system shall provide the ability for states to select an external infrastructure provider to host an instance of the PARCC assessment system or deploy components of the PARCC assessment system into their own infrastructure.			X	X	
BR-09. The assessment system shall provide tools or services that deliver full functionality to all stakeholders regardless of their IT infrastructure and capability.					X
BR-10. The assessment system shall provide Recovery Point Objectives for critical systems and data.			X		
BR-11. The assessment system will store, backup, and recover assessment system data in a distributed environment.			X		

Requirement	F	U	R	P	S
BR-12. The assessment system should utilize a finalized version of the APIP 1.0 standard for item interoperability between components that transfer item data.		X			X
BR-13. The assessment system shall comply with the accessibility requirements that the components must satisfy.		X			
BR-15. The assessment system shall incorporate a centralized PARCC-level Item/Test Bank.	X				
BR-16. The Item/Test Bank shall store item, form, statistical data, and metadata for the assessment system.	X				
BR-17. The Item/Test Bank shall provide levels of security to support appropriate user access to items.	X	X			
BR-18. The assessment system shall provide a comprehensive tool set for data and metadata management and data quality.					X
BR-19. The assessment system shall support an integration framework for interoperating with disparate student information systems.					X
BR-20. The assessment system will comply with FERPA and COPPA privacy laws.	X	X			
BR-21. The assessment system will utilize robust, standards-based systems for data storage.	X				X
BR-22. The assessment system will utilize guaranteed message transmission technologies.			X		
BR-23. The assessment system will utilize and enforce published data standards and formats.					X
BR-24. The hosting facility should provide monitoring systems and physical access control through multifactor authentication and physical system segregation.		X	X		X
BR-25. The physical hardware of the assessment system should enforce staff access protocols, user access controls, and encryption strategies for sensitive data at rest.	X				X

Requirement	F	U	R	P	S
BR-26. The network layer of the assessment system shall enforce best-practice security measures, including utilizing firewalls and secure transport of encrypted data.			X		X
BR-27. The assessment system components should utilize industry-proven security standards and protocols.			X		X
BR-28. The assessment system should utilize an identity management system with role-based user authentication and authorization.	X				
BR-29. The assessment system should respond within an average of four seconds during estimated peak usage.				X	
BR-30. The assessment system should support cloud-based deployment with auto-scaling to match demand.			X	X	
BR-31. The assessment system should be available during an assessment administration, not including scheduled downtime for maintenance and upgrades.			X		

2.2 HARDWARE, SOFTWARE, AND NETWORK REQUIREMENTS

The PARCC Assessment System needs to be able to run on a variety of client-side and server-side hardware and software platforms. Its implementation must be able to meet key technology priorities defined in *Key Technology Priorities Summary*, such as system flexibility (Key Priority #7), multiple hosting options (Key Priority #5), high availability and scalability (Key Priority #17), and support for varying levels of technology capabilities at the state, district, and school levels (Key Priority #6). This section outlines the available options for the provisioning of hardware, software, and network for the PARCC Assessment System.

This section addresses the fundamental technical implementation options corresponding to a number of use cases in these functional areas (described in *High-level Application Architecture*):

- 004 – Content Movement
- 006 – Registration
- 007 – Scheduling and Assignment
- 008 – Student Delivery
- 014 – Data Export
- 015 – Report Generation

Requirements for Client-side Hardware and Software

Currently, the minimum specifications for new system purchases that will satisfy PARCC Assessment System client-side requirements are:

- **Hardware.** 1 GHz or faster processor, 1 GB RAM or greater memory, 9.5 inch (10-inch class) or larger screen size, 1024 x 768 or better screen resolution.
- **Operating Systems.** Mac 10.7, Windows 7, Linux (Ubuntu 11.10, Fedora 16), Apple iOS, Android 4.0, Chrome OS.
- **Network.** Must be able to connect to the Internet via either wired or wireless network.
- **Software.** Standard Internet browser. Supported browser versions will be determined later.

All hardware decisions should be based on the instructional needs of schools and students. Some students may need hardware that exceeds these minimum guidelines, and some students may require qualitatively different hardware. Districts should rely on local expert judgments and discussions with appropriate authorities to determine the hardware that is most appropriate to deliver quality instruction and to support individual students.

Test delivery in the PARCC Assessment System needs to accommodate a variety of client devices, each with different display characteristics and different CPU power available for processing graphics, animations, and video. Pacific Metrics and IBM expect that rendering of traditional test item content will be fully supported across all client devices. However, rendering interactive content, associated with the so-called technology-enhanced items (TEIs), might present some challenges in several phases of the TEI life cycle, specifically authoring, distribution, storage, and delivery. There are several competing technologies that can provide Web-based delivery of TEI interactive content: Adobe Flash, Microsoft Silverlight, Oracle Java FX/Applets, and HTML5/JavaScript/CSS. All except HTML5/JavaScript/CSS are proprietary technologies that require a corresponding browser plugin to be installed. HTML5 is increasingly becoming the standard technology for delivering interactive content across all devices and operating systems.

The recommended option for interactive content delivery in the PARCC Assessment System is the HTML5/JavaScript/CSS trio of open standards. Therefore, the item authoring tools should support output/export to HTML5/JavaScript/CSS.

There are several content authoring platforms that can either directly create and edit HTML5 content, or publish their native content to HTML5 format. Among them are:

- **Adobe Captivate 6 –** www.adobe.com/products/captivate.html
- **Adobe Edge –** labs.adobe.com/technologies/edge/ and edge.adobe.com/whatisedge.html
- **IBM Maqetta –** www.eweek.com/c/a/Application-Development/IBM-Launches-Maqetta-HTML5-Tool-as-OpenSource-Answer-to-Flash-Silverlight-669762/

More information about specific multimedia data standards can be found in the *Technology Standards and Protocols Options* document.

Requirements for Server-side Hardware and Software

Any hardware capable of running the selected server-side application platform (see "Server-side Application Platforms" on page 23) should satisfy the server-side hardware needs of the PARCC Assessment System. Also, depending on the selected deployment model (see "System-level Deployment" on page 39), server-side hardware may be a choice made by the cloud infrastructure provider or the third-party host. The hardware specifications details for PARCC Assessment System server-side hardware, hosted on third-party premises (whether traditional or cloud-based), will be defined in the vendor contract and should be provisioned in accordance with the service level agreements (SLAs) that will be part of that contract.

Similarly, operating system choices on the server side will depend on the deployment options and the application platform choice.

- For .NET, Windows is the only option.
- For J2EE, Windows or Unix-like operating systems can be chosen.

Windows and enterprise-class Linux distributions can be used in the PARCC Assessment System. Both operating systems have licensing costs that include support options, which provide upgrades and security patches. Linux has an open-source license, but companies like Red Hat provide enterprise-grade commercially backed Linux distributions.

Some factors to consider in evaluating server-side hardware include: CPU clock-speed, cache size, number of CPUs, number of cores, memory size, and disk input/output (I/O) performance. The speed and size of the CPU caches are of particular importance for overall server performance. Other factors that should be considered are supportability, compatibility, and VM support.

Databases, Data Storage, and Data Archiving

Types of Databases

There are several basic types of databases available in the market today. They differ in the way the logical data entities are organized.

Relational Databases (RDBs)

RDBs organize the data entities in tables consisting of rows and fields. In an enterprise-class RDB, there would typically be hundreds of tables and many thousands of relations describing how the data entities logically relate to each other. Structured Query Language (SQL) is the language used to manipulate (i.e., create, delete, update, and retrieve) the data in a relational database. Most relational databases are engineered to enforce the ACID (atomicity, consistency, isolation, and durability) principle, which guarantees reliable database transactions even in the case of adverse conditions (e.g., crashes or power loss).

RDBs are the dominant databases today, and, when properly designed and implemented, they can be used for many types of data workloads. However, one notable problem of RDBs is the complexity of setting up redundant server configurations (through database clusters or master-

slave setups). Another problem is the necessity for using high-end hardware for processing large quantities of data. Popular commercial RDBs include Oracle Database, IBM DB2, and Microsoft SQL Server. Popular open-source RDBs include MySQL, PostgreSQL, and SQLite.

Object-oriented Databases (OODs)

OODs try to resolve a major discrepancy between:

- How data is represented in the server's memory by object-oriented programming (OOP) languages (e.g., Java, C++, or C#)

  AND

- How the same data is represented on a disk drive by relational databases.

Object-oriented languages organize data in a hierarchical fashion using inheritance to provide code reuse and encapsulation. As described in the previous paragraph, relational databases organize data in tables consisting of columns and rows. This data impedance mismatch, as it is known, is typically resolved using object-relational mapper (ORM) software, which is complex and adds a layer of data propagation because of its own needs (and costs) for development, testing, and maintenance.

Object-oriented databases try to resolve this problem by representing the data entities in the database as objects—not tables—which is exactly what the object-oriented languages do in the server's memory. Object-oriented databases, however, have had mixed results. While resolving one type of problem (e.g., eliminate the need for object-relational mapper and reduce the need for SQL joins), they have problems of their own, including:

- Very tight-coupling with the OOP language code.
- Inability to define ad hoc data integrity constraints.
- Interoperability issues with traditional RDBs.
- Lack of standard tools for database administration.

NoSQL Databases

NoSQL databases began as a movement to avoid expensive commercial RDBs and complex open-source RDBs when implementing data storage mechanisms for Web 2.0 applications, which needed fast access to vast volumes of end-user-generated data. The first NoSQL databases were influenced by Google's BigTable and Amazon's Dynamo. The common features of all NoSQL databases are:

- They do not represent data as related tables—like RDBs do.
- They do not use the SQL language for manipulating data—a notable feature of RDBs described previously.
- They do not necessarily follow the strict rules for enforcing data integrity and constraints that RDBs are known for—a process called *normalization*.

Among the reasons for using NoSQL databases are higher throughput, less complexity, horizontal scalability, and the ability to run on clusters of commodity hardware. Setting up

redundancy and high availability with NoSQL databases is a relatively simpler and cheaper process than the process with relational databases because it does not rely on highly available hardware, but instead relies on clusters of cheaper servers, each of which does not necessarily need to have high-availability features. Most notably, handling huge volumes of data in NoSQL databases seems to be a lot less expensive than using RDBs.

On the downside, NoSQL databases expect relatively simple data models and relationships, often representing relations among data items as simple key-value pairs or unrelated tuples.

Most NoSQL databases can be categorized as one of these types:

- **Key-Value** (e.g., MemcacheDB and Riak). Data is represented as key-value pairs, where the value is a simple data element. This type is primarily used when all access to the database is by primary key.
- **Document** (e.g., CouchDB and MongoDB). Data is represented as key-value pairs, where the value is complex data stored as a single document. The documents often have hierarchical structure described in corresponding formats such as XML or JSON. The documents can be similar to each other, but do not have to have the exact same structure.
- **Column-family** (e.g., Cassandra and Hadoop/HBase). Data is represented as key-value pairs, where the value is complex data stored in turn as a set of additional key-value pairs, also known as "column families." Unlike a relational database, the column families do not need to have the exact same structure.
- **Graph** (e.g., Neo4J and HypergraphDB). Data is represented as nodes with properties and relations between each other. Relations have directions, and nodes can be organized by relationships. Data stored in nodes and relationships can be stored once, and then interpreted in different ways based on the relationships.

Table 2 – Comparison of Relational Databases and NoSQL Databases summarizes the pros and cons of relational and NoSQL databases.

Table 2 – Comparison of Relational Databases and NoSQL Databases

	Relational databases	NoSQL databases
Pros	• The relational model is simple in principle and can handle business domains with high complexity. • Provides solid support for ACID, transactional handling and reporting. • Simple, versatile, and standardized query language (i.e., SQL). • Standardized APIs for relational database access in most programming languages.	• Mostly open-source. • Simpler data models. • Horizontal scalability (i.e., data can be processed easily in parallel). • Inserting new data is very fast, as well as simple operations/queries. • Data model changes do not trigger comprehensive changes in code. • Can store complex documents in a single item of storage.
Cons	• Redundancy and scalability setups can be very complex. • Large redundant and scalable databases require powerful, expensive hardware. • Extensive data model changes can have significant impact on existing code and require extensive refactoring.	• The simpler data models restrict applicability to only specific domains. • Indexing support not as powerful as in relational systems. • No ACID properties. Cannot achieve consistency, availability and partitioning tolerance at the same time. • Not very good for reporting purposes. • No standard APIs or query languages • Still not very mature technology.

Usage Areas

Based on the pros and cons outlined above, relational databases are typically best to use in storing transactional data, and for reporting and business intelligence purposes. NoSQL databases should be considered for logging, caching, session storage and other areas which do not require complex data models but do require fast storage/retrieval.

Recommendations for PARCC Assessment System Choice of Database Types

Pacific Metrics and IBM recommend that the Operational Data Store and Data Warehouse components of the assessment system utilize relational databases in their implementation. These two components play a crucial role in the majority of use cases involving the movement and storage of critical PARCC Assessment System data, such as student registrations, rooms, items, tests, raw scores, test responses, and test results that are described in *High-level Application Architecture* in these functional areas:

- 006 – Registration
- 007 – Scheduling and Assignment
- 008 – Student Delivery
- 009 – Test Administration
- 015 – Report Generation

As described in *Information Architecture*, the fundamental data is complex, with many common attributes dispersed across different functional areas, and, as such, can be best described using a relational data model. In addition, the process of moving and storing this data carries with it stringent requirements for data accuracy, data integrity, and reliable database transactions—all of which are best handled by the ACID characteristic of relational databases.

Database choice decisions for internal data stores in other assessment system components can be made in a similar manner, driven by the static and dynamic characteristics of the data that will be processed and stored internally in the component. For example, a key-value or document-style NoSQL database might be used for the Monitoring and Alerting component because of the simplicity of the underlying data model.

Data Storage Overview

There are many considerations when sizing and configuring the storage subsystem for the assessment system. Similarly to server choices, the decision-making in this area will be determined by the overall type of system deployment. If deployed in a cloud infrastructure or in a third-party data center, many decisions will be determined by the cloud/hosting vendor offerings, and the storage specifications will be negotiated and written into the contract.

Important factors to consider are the type of disk drive used (e.g., Serial AT Attachment [SATA], Small Computer System Interface [SCSI], or serial attached SCSI [SAS]), the use of solid-state drives (SSDs), storage array types (i.e., storage area network [SAN] vs. direct attached storage [DAS]), and the RAID (redundant array of independent disks) configuration of the disks.

An important characteristic of any single traditional magnetic disk is the overall *disk latency*, which is a combination of these parameters:

- **Seek time.** The time, in milliseconds, for the head to physically move across the disk to find the data. This will limit the number of I/O operations per second (IOPS).

- **Rotational latency.** The time, in milliseconds, needed to read the data off the disk. This will limit the *I/O throughput*, the amount of data a single disk can read per second (MB per second).

Typical seek times are in the 5 to 10 millisecond range, while typical rotational latency range is 3 to 4 milliseconds (which corresponds to rotational speeds of 15,000 revolutions per minute [rpm], the current upper limit of most disk drives).

To overcome the limitations of single disk drives, the storage system will have many disks working together, in some level of RAID in SAN-based or DAS-based storage arrays, to increase both IOPS and I/O throughput. DAS are typically used via SAS or SCSI interfaces. A SAN is more expensive because it is a dedicated network that has multiple hard drives (from dozens to hundreds) with multiple storage processors, caches, and other redundant components. SANs provide features not available in DAS (e.g., SAN snapshots).

There are two types of SANs: fibre channel (FC) and Internet Small Computer System Interface (iSCSI). They are different in the underlying wiring mechanism:

- **FC SANs.** Typically use fiber-optics. FC SANs have better overall performance. Some of them are configured with tiered-storage, wherein a group of drives in the SAN can be very fast SSDs, while another group can be relatively slower SATA drives. These groups can be used for different types of workloads (see "Operational Data Store and Data Warehouse Workloads" on page 22).
- **iSCSI SANs.** Use a Transmission Control Protocol/Internet Protocol (TCP/IP) network with standard Ethernet components. The iSCSI SANs are less expensive, because they use standard TCP/IP network infrastructure.

Newer solid-state drives (SSDs) have the potential to replace both individual disks and even SANs when it comes to the ratio of performance to cost, because their seek and rotational latencies are much lower compared to traditional magnetic disk drives—i.e., there are no electro-mechanical moving parts. However, SSDs have less predictable failure rates, which can result in higher supportability cost.

Operational Data Store and Data Warehouse Workloads

There are two primary workload types that a database server commonly deals with:

- **Online Transaction Processing (OLTP).** The OLTP workload consists of many short transactions wherein the data is much more volatile than in a Data Warehouse/Reporting (DW/R) workload. Usually there is much more write activity in an OLTP workload than in a DW/R workload, and most OLTP systems generate more I/O operations per second (IOPS) than an equivalent-sized DW/R system.
- **Data Warehouse/Reporting (DW/R).** A DW/R system usually has longer-running queries than a similar-sized OLTP system—with much higher read activity than write activity—and the data is usually more static. In such a system, it is much more important to be able to process a large amount of data quickly, than it is to support a high number of I/O operations per second.

Recommendations for PARCC Assessment System Data Storage

To provide optimal storage performance, the PARCC Assessment System storage implementation should ideally use a SAN with low overall disk latency. For optimum performance, the SAN should include either SSDs or fast SCSI disk drives.

The Operational Data Store (ODS) and Data Warehouse (DW) components will use two separate storage mechanisms (e.g., two SANs), because these two components belong to two different component groupings (Grouping #1 and Grouping #4) as defined in *Component-based Dependency Matrix in High-level Project Portfolio Schedule*.

In evaluating SAN products, it is important to consider the performance characteristics of all components in the SAN (e.g., processor, disks, switches, and cables) to eliminate the risk that a lower-bandwidth component will set the upper limit on the overall SAN performance.

Data Archiving

Data archiving is the process of removing selected data, which is not expected to be referenced again, from an operational database and putting it in an archive data store, where the data can be accessed again, if needed. The data design, storage methods, and tuning requirements for an archive data store are different than those for an operational database. While the operational database is designed and tuned for high levels of create, update, delete, and query activities executed against high volumes of data, the archive data store needs to accommodate much higher volumes of data with infrequent query activities and virtually no update activities.

The process of accessing the data in the archive data store is also different than accessing data in an operational store. Queries against the archive store are typically simple, but produce large amounts of data. It is also important to design the archive store access mechanism in such a way that the data would not have to be restored to the original system in order to obtain the desired output.

It is important to note that the data subject to archival is not only the data hosted in system databases; the data can also include plain files in various formats, including documents, multimedia, email, operating system files, and other types.

The PARCC Information Architecture and Data Governance processes will determine what types of data will be subject to data archiving, when data archiving will occur, how long the data will be retained in the archive, and when (if ever) the data will need to be destroyed.

The media used for data archival would need to provide access to the archived data, when needed—though not necessarily at the speed and convenience of operational data access. Cost will be the driving factor in determining the media for data archival. SAN and DAS are more expensive than network-attached storage (NAS), and NAS is more expensive than tapes. However, tapes offer the slowest access speed.

SERVER-SIDE APPLICATION PLATFORMS

In the world of server-side application development, there are two major application development and deployment platforms: J2EE and .NET. Both application frameworks provide a

solid foundation for building enterprise-ready, robust, server-side applications. These frameworks promote the usage of established development best practices and design patterns (e.g., Model-View-Controller [MVC] for separating presentation, controller, and back-end logic).

Table 3 – Comparison of J2EE and .NET Technologies contains a comparison of the basic features of these technologies.

Table 3 – Comparison of J2EE and .NET Technologies

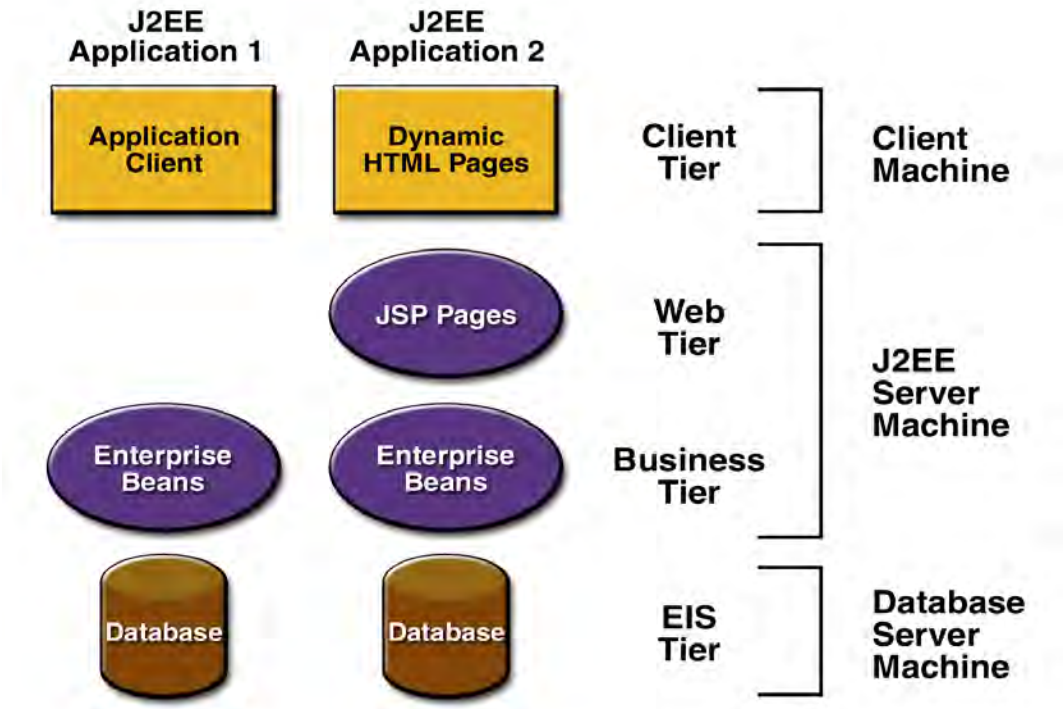
Capability	J2EE	.NET
Open Source	Yes	No
Language	Java	C#, Visual Basic .NET (among others)
Integrated Development Environment (IDE)	Both open-source and commercial including Eclipse, NetBeans, and IntelliJ IDEA.	Visual Studio .NET
Run-time environment	Runs pretty much on any major operating system.	Windows only* *There is an open-source project called Mono which aims at providing cross-platform capabilities to .Net. However, it is only partially compliant.

Both frameworks enable development and deployment of distributed applications, supporting Web-tier and business-tier components as well as integration with the enterprise information system (EIS) tier.

Either technology can be used for development of external or internal PARCC Assessment System components (see "Component Deployment Options" on page 34), as long as the implementation of these components satisfies PARCC Assessment System interoperability requirements. For internal assessment system components deployed as tightly-coupled components (where a single application server execution environment is used), the J2EE technology is preferred and recommended because it would provide maximum flexibility in the deployment options.

Figure 1 – Typical Layout of a J2EE Application System Diagram shows the typical layout in a J2EE application system.

Figure 1 – Typical Layout of a J2EE Application System Diagram



In the world of J2EE:

- Client components include application clients and applets.
- Web-tier server-side components include Java Servlets and Java Server Pages.
- Business components include Enterprise JavaBeans (EJBs).

Typical interactions between J2EE components also involve the use of JavaBeans components, which represent encapsulations of basic data structures with simple data validation rules.

A major requirement for the PARCC Assessment System is for its major functions to be accessible through a regular Internet browser, so the applicability of application client technology is limited. However, see "Network Requirements" on page 26 for a discussion of the trade-offs when using an application client and a browser.

Other Server-side Application Platforms

Besides J2EE and .NET, there are other popular application platforms (e.g., Ruby on Rails) that should be explored as possible platforms for development and deployment of loosely-coupled components. There are two basic requirements regarding the use of such platforms:

- The components developed and deployed on these platforms must support interoperability via Representational State Transfer (REST) and Simple Object Access Protocol (SOAP) Web services.

- The components must be built following the model-view-controller (MVC) pattern. The MVC pattern separates an application's code into three layers:
  - View layer, which is the user interface of the application.
  - Model layer, which is the back-end, server-side storage.
  - Controller layer, which represents the business logic and serves as the bridge between the other two layers.

VIRTUALIZATION

Virtualization is the simulation of the software and hardware upon which other software runs. This simulated environment is called a virtual machine (VM). There are many benefits to using virtualized servers, the main one being increased operational efficiency, because existing hardware can do more work by putting a greater load on each computer. In addition to this benefit, desktop virtualization allows the same computer to run multiple operating systems, which can aid both development and testing efforts. Most virtualization environments allow changes to be reverted easily, unlike a physical server environment, in which changes can be hard to track and revert.

In a typical virtualized environment, one operating system (called the *guest OS*) and the applications it contains are run on top of virtual hardware. The guest operating systems on a host are managed by the *hypervisor*, which controls the flow of instructions between the guest operating systems and the physical hardware (i.e., CPU, disk storage, memory, and network interface cards). Some hypervisors run on top of another operating system, known as the *host operating system*.

PARCC Assessment System components must be able to deploy and run on both physical and virtualized environments. This is to ensure portability of the assessment system component repository. Using virtualization techniques in the assessment system will improve productivity and efficiency during the development and validation phases and enable greater flexibility in the deployment options.

Major commercial products in the virtualization space include VMware VSphere, Citrix Xen Server, and Microsoft Hyper-V. There are also open-source virtualization products, most notably Red Hat Enterprise Virtualization. They are all viable options for use in the PARCC Assessment System and should be explored and compared using comparison Web sites like www.virtualizationmatrix.com/matrix.php.

NETWORK REQUIREMENTS

Internet connectivity and specific network capacity requirements for the PARCC Assessment System will be fully defined once the development of the test items repository is complete and the designs of the assessment delivery platform are finalized. Output data from the Technology Readiness Tool (www.techreadiness.org) will also be taken into consideration when determining networking requirements.

Network bandwidth requirements and their functional impact on test administration in the assessment system are addressed in these specific use cases in *High-level Application Architecture*:

- Edge Case – 005: Low-bandwidth District Implementation Edge Case
- Edge Case – 006: High-bandwidth District Implementation Edge Case

Network Capacity Requirements Model

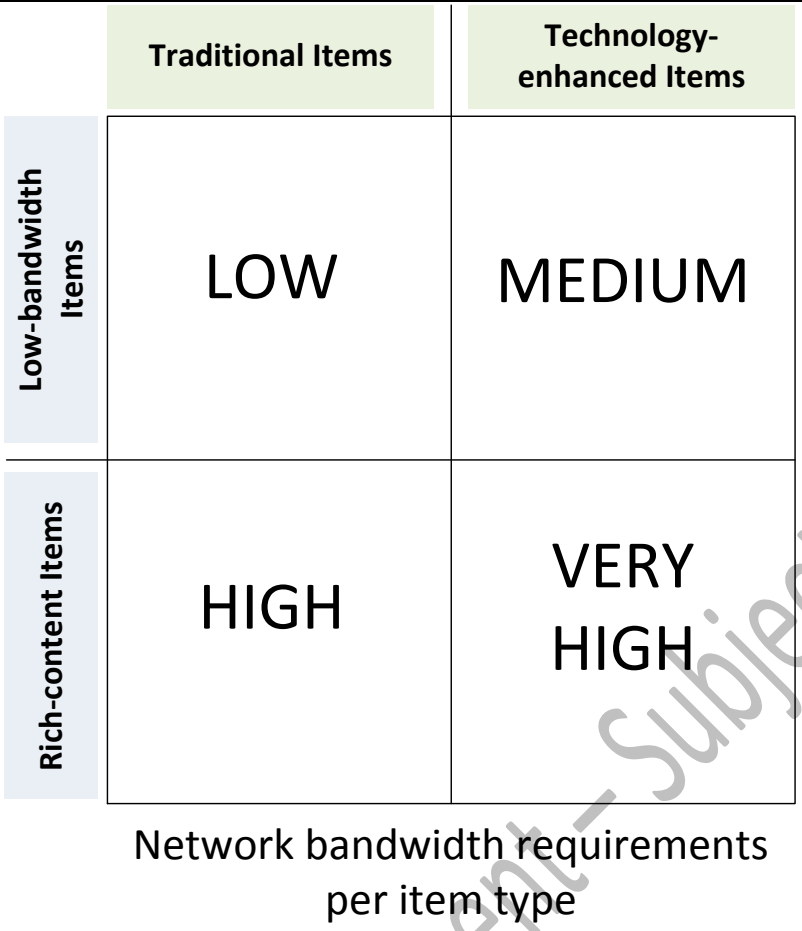
The remainder of this section provides a network bandwidth estimation model to approximate the network capacity needed to administer a PARCC Assessment System test in a variety of network environments.

Model Inputs

The model takes several input variables:

- The total number of test items in the test.
- Overall test duration in minutes.
- The number of test items for each of four categories based on a breakdown in two dimensions as shown in Figure 2 – Relative Network Bandwidth Requirements Diagram. These two dimensions define the item as traditional or technology-enhanced, and its content as rich content or low-bandwidth content. For each category, the expected level of required network bandwidth is shown as well (i.e., low, medium, high, and very high).
- The average size of each item type in bytes.

Figure 2 – Relative Network Bandwidth Requirements Diagram



Example Network Model 1

The output from the model is the required bandwidth in megabits per second (Mbits/sec) for a network environment with a different number of simultaneous test takers (i.e., 1, 300; 1,000; 3,000; and 10,000). The number of simultaneous test takers can represent the school, district, or state level of the test delivery environment. For example, a school level could be represented by 300 to 3,000 simultaneous test takers.

Step 1: Total Size of a Test

The first step in the model is to arrive at an estimated total size for the test. To determine this, the model uses the number of test items and the item size for each of the four item categories described in Figure 2 – Relative Network Bandwidth Requirements Diagram on page 28.

Figure 3 – Example Network Model 1: Estimated Total Size of Test Diagram provides an example for calculating the total size of a test using representative data.

Figure 3 – Example Network Model 1: Estimated Total Size of Test Diagram

	# Test Items	Average Test Item Size	Total Size		
		bytes	*bytes*	*KB*	*MB*
Traditional items with low-bandwidth content	32	2,000	64,000	63	0.06
Traditional items with rich content	8	700,000	5,600,000	5,469	5.34
Technology-enhanced items with low-bandwidth content	16	500,000	8,000,000	7,813	7.63
Technology-enhanced items with rich content	8	1,200,000	9,600,000	9,375	9.16
Total items	**64**	363,500	23,264,000	22,719	**22.19**

Explanation of Figure 3

The total number of items is 64, broken down in a 32-8-16-8 distribution across the four content categories (i.e., with a relatively high percentage of technology-enhanced items with rich content) with the corresponding average item sizes at 2,000; 700,000; 500,000; and 1,200,000 bytes. The resulting total test size is approximately 22 MB.

Step 2: Average Bandwidth Requirement

The second step in the model is to use the total test size and the test duration to determine the average required bandwidth for one test taker, then multiply the result accordingly to arrive at the required bandwidth for multiple, simultaneous test takers. Figure 4 – Example Network Model 1: Estimated Total Bandwidth Diagram shows the results for 1; 300; 1,000; 3,000; and 10,000 simultaneous test takers.

Figure 4 – Example Network Model 1: Estimated Total Bandwidth Diagram

# Simultaneous Test Takers (school/district/state)	1	300	1,000	3,000	10,000
Test duration (min)	150	150	150	150	150
Total Test data (MB)	22	6,656	22,186	66,559	221,863
Required bandwidth (Mbits/sec) (average)[1]	**0.020**	**5.916**	**19.721**	**59.163**	**197.211**

input field

calculated field

[1] *Protocol overhead is not reflected in these estimates.*

Explanation of Figure 4

The data columns in this figure display the number of simultaneous students taking a test. Required bandwidth represents the amount of bandwidth in megabits per second required for the specified number of simultaneous students to take a test for the specified duration with the specified amount of total test data.

Example Network Model 2

This example model utilizes a different composition of content types when compared with Model 1. Figure 5 – Example Network Model 2: Estimated Total Size and Bandwidth Diagram shows another output from the network bandwidth estimation model, where there is a 40-12-8-4 composition of the test items across the four content categories.

Figure 5 – Example Network Model 2: Estimated Total Size and Bandwidth Diagram

REQUIRED NETWORK BANDWIDTH ESTIMATES

	# Test Items	Average Test Item Size	Total Size		
		bytes	*bytes*	*KB*	*MB*
Traditional items with low-bandwidth content	40	2,000	80,000	78	0.08
Traditional items with rich content	12	700,000	8,400,000	8,203	8.01
Technology-enhanced items with low-bandwidth content	8	500,000	4,000,000	3,906	3.81
Technology-enhanced items with rich content	4	1,200,000	4,800,000	4,688	4.58
Total items	**64**	270,000	17,280,000	16,875	**16.48**

# Simultaneous Test Takers (school/district/state)	1	300	1,000	3,000	10,000
Test duration (min)	150	150	150	150	150
Total Test data (MB)	16	4,944	16,479	49,438	164,795
Required bandwidth (Mbits/sec) (average)[1]	**0.015**	**4.395**	**14.648**	**43.945**	**146.484**

input field

calculated field

[1] *Protocol overhead is not reflected in these estimates.*

Explanation of Figure 5

This diagram represents a test with more traditional item content and less content with technology-enhanced items (i.e., a relatively low percentage of technology-enhanced items with rich content). As expected, the required bandwidth is reduced from 59 Mbits/sec to 43 Mbits/sec when compared with the previous scenario depicted in Figure 3 on page 29 and Figure 4 on page 29.

Model Validity

As with any model, the accuracy of the output is directly dependent on the accuracy of the inputs and the assumptions made in the model. At this point, there are still many details related to composition and timing of PARCC Assessment System tests that are yet to be determined. As more information becomes available, the network bandwidth estimation model should be re-used to arrive at more accurate estimates.

Conclusions

Using the two sample results presented previously, we can conclude that in these specific simulation scenarios (i.e., a 150-minute test with 64 items with the specified item breakdowns and sample item sizes), a typical school environment will require between 4 and 45 Mbits/sec of bandwidth to accommodate the delivery of a test for 300 to 3,000 simultaneous test takers.

Implications of Network Capacity on the Assessment System

Figure 6 – Example Connection Speeds for Different Connection Types Diagram shows the typical connection speeds of different connection types and networks.

100-megabit Bandwidth Network

Using the information in this table, we can determine that a 100-megabit Ethernet (i.e., 100Base-T) LAN, which has typical capacity of 80 Mbits/sec after overhead is deducted, will probably provide the needed bandwidth to accommodate delivery of this test for 300 to 3,000 simultaneous test takers. However, the same LAN environment would *not* be able to deliver the test to 10,000 simultaneous test takers, which requires 197 Mbits/sec in the scenario presented in "Example Network Model 1" on page 28 and 146 Mbits/sec in the scenario presented in "Example Network Model 2" on page 30, each one requiring more than the 80 Mbits/sec provided by the 100Base-T LAN.

10-megabit Bandwidth Network

Similarly, a 10-megabit Ethernet (i.e., 10Base-T) LAN, which provides about 8 Mbits/sec, will be able to accommodate delivery to 300 simultaneous test takers—but *not* to 1,000 simultaneous test takers. This larger number of test takers would require 19 Mbits/sec in the scenario presented in "Example Network Model 1" on page 28 and 14 Mbits/sec in the scenario presented in "Example Network Model 2" on page 30.

Figure 6 – Example Connection Speeds for Different Connection Types Diagram

Connection type	Connection speed	Unit	Network type
Dedicated PPP/SLIP via modem	28.8	Kbps	WAN
Integrated Services Digital Network (ISDN)	128	Kbps	WAN
Typical DSL	640	Kbps	WAN
ADSL Lite	1.5	Mbps	WAN
DS1/T1	1.536	Mbps	WAN
10-megabit Ethernet	8	Mbps	LAN
Wireless 802.11b	11	Mbps	WLAN
ADSL2	12	Mbps	WAN
DS3/T3	44	Mbps	WAN
OC1	51	Mbps	WAN
Wireless 802.11g	54	Mbps	WLAN
100-megabit Ethernet	80	Mbps	LAN
OC3	155	Mbps	WAN
OC12	622	Mbps	WAN
Wireless 802.1n	600	Mbps	WLAN
1-gigabit/sec Ethernet	800	Mbps	LAN

Key to Figure 6:

Abbreviation	Definition
Kbps	kilobits per second
Mbps	megabits per second
WAN	Wide area network
WLAN	Wireless local area network
LAN	Local area network

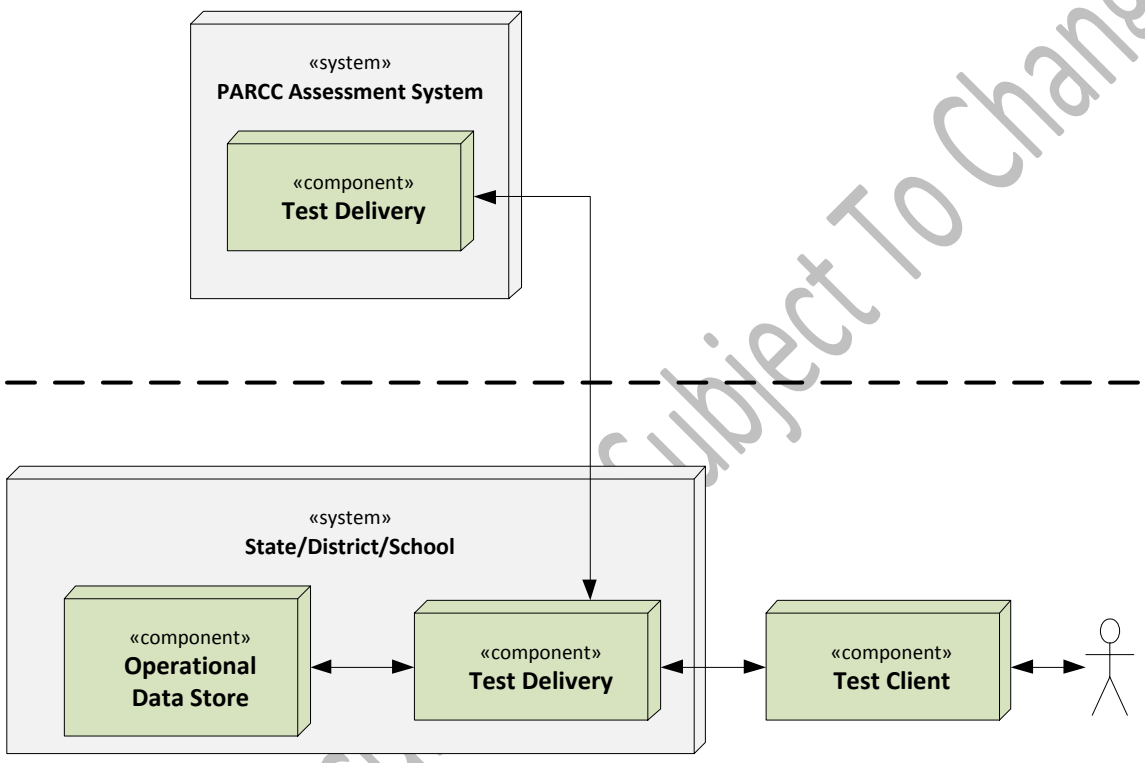
Low Bandwidth Capacity Mitigation Strategies

As discussed previously, bandwidth capacity may be a concern with tests containing items using video, audio, or other more network-intensive media types. Content pre-loading techniques and HTML5 caching mode should be explored as options to reduce the network requirements to enable the assessment system to support tests containing such items. However, because of the diversity of the platforms that are required to be supported (e.g., multiple combinations of device, operating system, and Web browser) and the corresponding diversity of implementations of HTML5, certain items with large memory footprints may not be cacheable across all platforms. For example, the HTML5 Offline Application Cache size limit parameter may have significantly different values across platform implementations, though the HTML5

standard does not set an actual limit in the specification. Sometimes such limitations are not even fully described by the vendors.

The Test Client component can mitigate high-bandwidth items by using pre-loading or caching techniques. The Test Delivery component will be developed with the ability for test-caching, which is the deployment of tests within a local environment. Figure 7 – Example Test-caching Diagram provides an overview of the feature.

Figure 7 – Example Test-caching Diagram



Process Flow Description

1. The student initiates a test by clicking on a link in the Test Client component.
2. The Test Client component contacts a local Test Delivery component.

   Note: The term "local" can mean in the same building, school, district, or state.
3. The local Test Delivery component processes all interactions during the test.
4. The local Operational Data Store component records the student responses, ensuring that the data is persisted.
5. Once the responses are persisted, the local Test Delivery component forwards the responses to the PARCC Test Delivery component.
6. The PARCC Test Delivery component processes the request as if the student were connecting directly.

By using an Operational Data Store component for persistence, the local Test Delivery component can defer the sending of responses if network connectivity is diminished or non-existent.

2.3 COMPONENT DEPLOYMENT OPTIONS

The internal components of the PARCC Assessment System need to be flexible in their deployment to provide the diversity of hosting options as defined in Key Technology Priority #5. While the choices in component deployment will influence the implementation of most use cases, the biggest impact is expected to be on these use case functional areas (described in *High-level Application Architecture*):

- 004 – Content Movement
- 016 – Resource Center
- 006 – Registration
- 007 – Scheduling and Assignment
- 008 – Student Delivery
- 009 – Test Management
- 014 – Data Export

Component deployment refers to how the component functionality is packaged and exposed to other components. Components are usually deployed within some kind of application server running under a particular operating system on a physical machine.

Depending on their deployment and how they talk to each other, two components can be tightly-coupled or loosely-coupled.

TIGHTLY-COUPLED COMPONENTS

Tightly-coupled components are usually deployed within the same application server, talk to each other via local calls, and often share common persistence storage (e.g., a database).

The advantages of tightly-coupled components include:

- Better performance.
- Lower implementation cost.
- Easier security implementation.

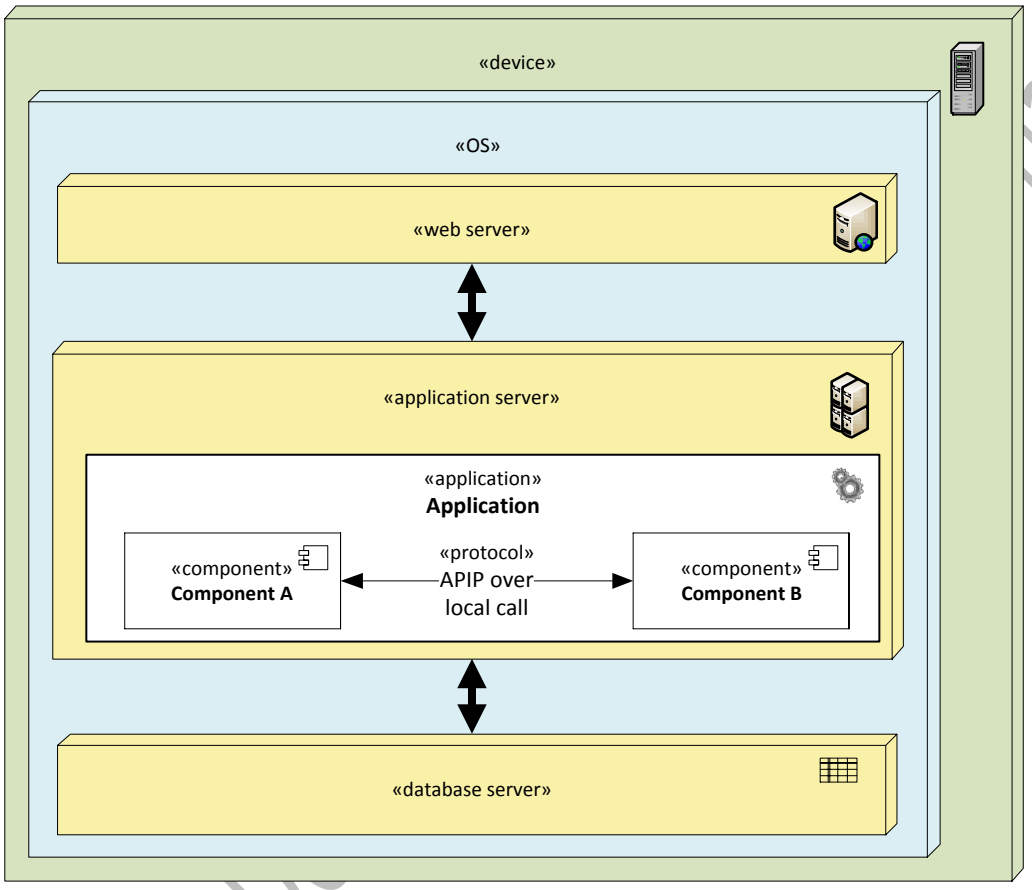
The disadvantages of tightly-coupled components include:

- Less flexibility, because components must be implemented in the same technology utilized by a common application server.

- Reduced reliability, because tightly-coupled components typically share the same in-memory server space, in the case of a server failure, all tightly-coupled components in this memory space will fail.

Figure 8 – Tightly-coupled, Locally Deployed Application Components Diagram illustrates the notion of tightly-coupled components.

Figure 8 – Tightly-coupled, Locally Deployed Application Components Diagram



Deployment: Both components are deployed inside the same application server.
Call type: Components directly talk to each other via local calls.
Persistence: Components share a common persistence layer.

LOOSELY-COUPLED COMPONENTS

Loosely-coupled application components are each deployed in a separate application server running on a separate physical machine. They typically talk to each other via Web service calls.

The advantage of loosely-coupled components is flexibility in technology choices.

The disadvantages of loosely-coupled components include:

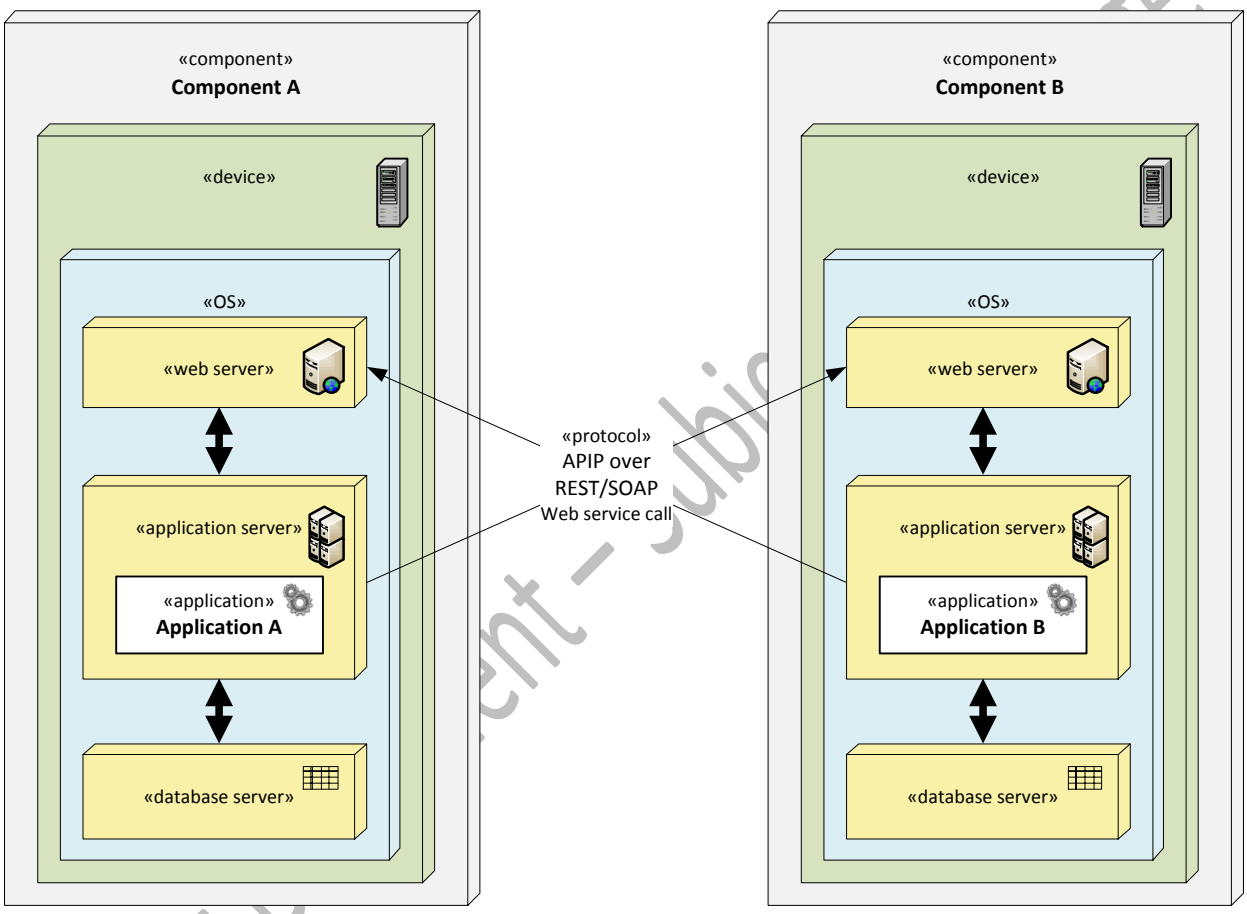
- Higher network bandwidth requirements, because loosely-coupled components, typically deployed in separate application server environments, need an additional network-

intensive server component (a service bus) to orchestrate their communication through service calls.

- Reduced performance.

Figure 9 – Loosely-coupled, Remotely Deployed Application Components Diagram illustrates the notion of loosely-coupled application components.

Figure 9 – Loosely-coupled, Remotely Deployed Application Components Diagram

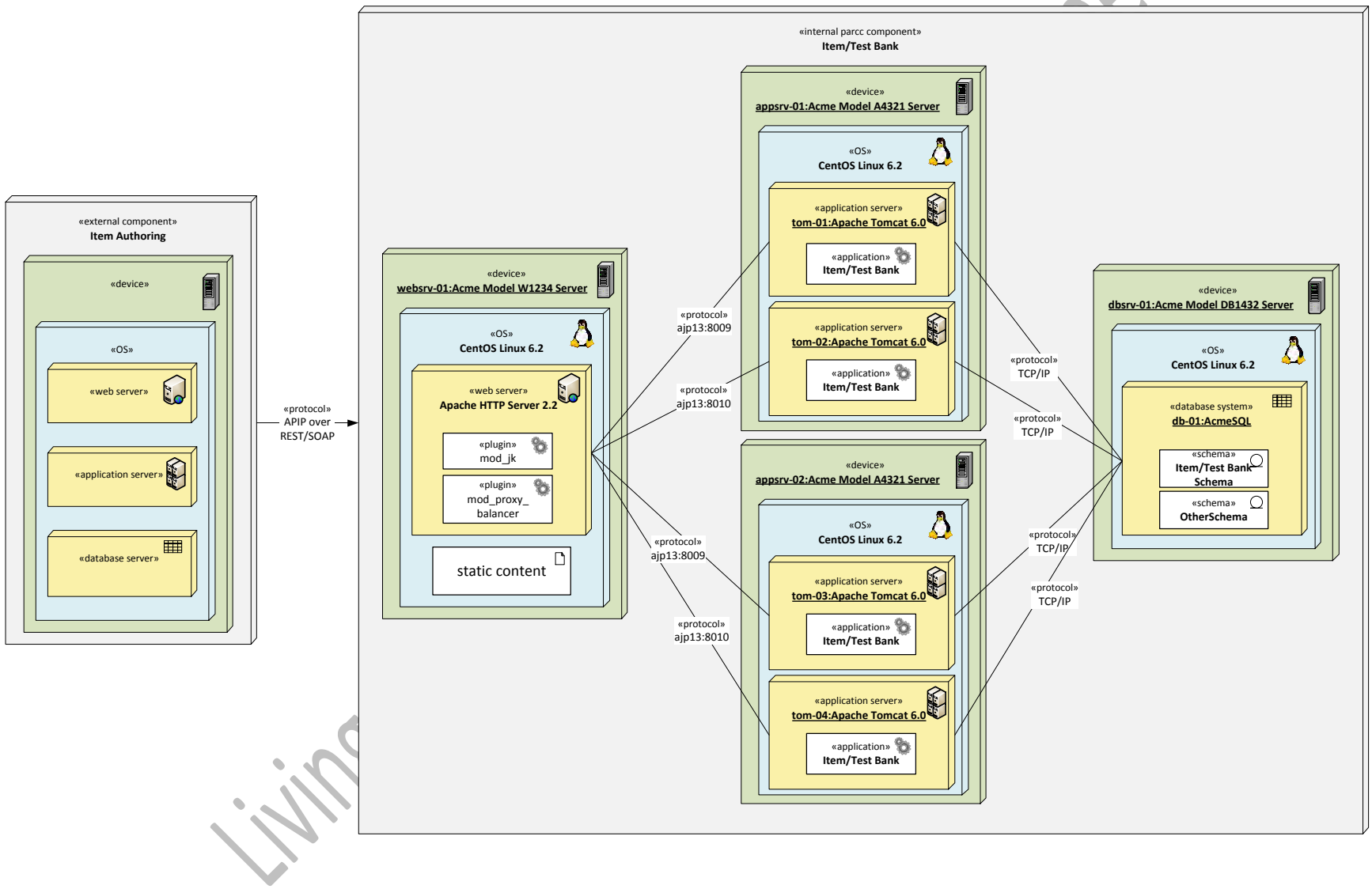


Deployment: Each component is deployed inside its own Device-OS-WS-AS-DB stack.
Call type: Components talk to each other via remote REST/SOAP web service calls.
Persistence: Each component has its own persistence layer.

Other Component Deployment Options

Other component deployment options exist that can help balance the advantages and disadvantages of the two component deployments described previously. The particular deployment type for each PARCC Assessment System component will be determined during system design and implementation. Figure 10 – Distributed Component Deployment Diagram shows a more realistic sample deployment and interactions between two example components (Item Authoring and Item/Test Bank), including possible operating systems, application servers, and network protocols. This sample deployment illustrates a possible internal architecture for an Item/Test Bank instance where a clustered application server setup is used to provide for better availability and scalability.

Figure 10 – Distributed Component Deployment Diagram



This sample design is still limited by the single database instance which can turn out to be a bottleneck. In reality, a database cluster (or master-slave setup for decoupled reads and writes) will be used.

Note: The technologies listed on this diagram are for illustration purposes only. Many other technologies can be used to achieve the same purpose of ensuring higher component availability and performance.

Component-based Dependency Matrix in High-level Project Portfolio Schedule includes a "Component Development Groupings" column that describes the grouping of components based on related functionality.

- Components that fall into the same functional grouping are good candidates for tightly-coupled deployments.
- External components (marked as "Not PARCC Developed" in the matrix) will, by definition, be interacting with internal PARCC components in a loosely-coupled fashion. See Figure 10 – Distributed Component Deployment Diagram on page 38.

System-level Deployment

The PARCC Assessment System can utilize traditional or cloud-based system-level provisioning and deployment.

Traditional Hosting

Traditional hosting of the PARCC Assessment System will require either the implementation of the full spectrum of internal IT infrastructure services or outsourcing those services to third-party hosting providers. Either way, PARCC will typically own or rent the server-side hardware and software and may need to employ technical staff to manage all or some of the IT infrastructure. The advantage of this approach is full control over the deployment environment and IT infrastructure. The disadvantages include: less reliability, inability to change capacity quickly as the load on the system changes, and slower deployments.

Cloud-based Deployment

According to the National Institute of Standards and Technology (NIST), cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

The five essential characteristics of a cloud infrastructure are:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

There are many advantages of provisioning the PARCC Assessment System in the cloud, as opposed to using traditional hosting options. Capacity planning, disaster recovery, availability, and reliability are system features that a reputable cloud provider will list as part of the cloud service contract. Costs can also be better allocated and managed, because typically service is paid for only when it is actually used.

Depending on who uses the cloud infrastructure, who manages the infrastructure, and where it is located, system deployments can be private, public, or community. Figure 11 – Deployment Models in a Cloud Infrastructure Diagram summarizes the features of each deployment model.

Figure 11 – Deployment Models in a Cloud Infrastructure Diagram

Deployment models in a cloud infrastructure:

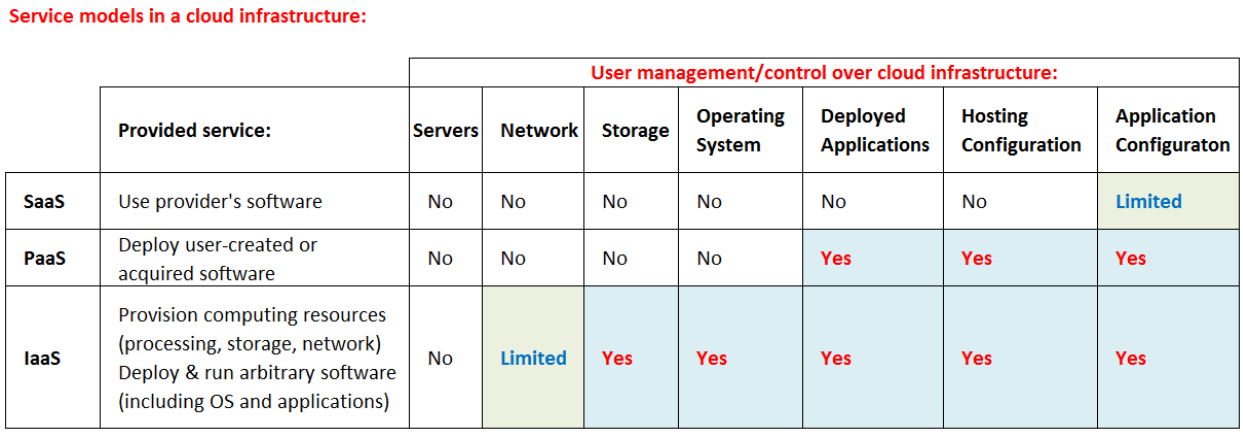
	Cloud infrastucture is:			
	Used by:	**Exclusive use?**	**Owned/Managed by:**	**Located at premises of:**
Private	A single organization and its units	Yes	User and/or third-party	User and/or cloud provider
Public	The general public	No	Business/academic/gov organizations	Cloud provider
Community	Several organizations with shared concerns/mission	Yes	One or more of the organizations and/or third-party	User and/or cloud provider

These are the most common models for deploying cloud infrastructure:

- Software-as-a-service (SaaS)
- Platform-as-a-service (PaaS)
- Infrastructure-as-a-service (IaaS)

These models are based on the types of resources (e.g., servers, network, storage, operating system, and application development platforms) that the customer of a cloud infrastructure can manage and control. Figure 12 – Service Models in a Cloud Infrastructure Diagram summarizes the service models.

Figure 12 – Service Models in a Cloud Infrastructure Diagram



Based on these definitions of deployment models and service models:

- The best option for deploying some (or all) of the PARCC Assessment System components is a community deployment model or some combination of Community and Public (also known as Hybrid).
- The best service model is either PaaS or IaaS, depending on the level of control PARCC would like to have over the network provisioning and the application development technologies that the cloud provider is making available to PARCC.

The SaaS service model is not suitable because PARCC needs greater control and management over the deployed applications. The PaaS model with some cloud providers may be restrictive, if the provider-supplied application development technologies do not match the technologies chosen for internal PARCC development. One other concern of the cloud-based deployment model is the security of the student data. Regulatory and standards compliance in this area must be validated before any cloud-based component deployments are initiated. For a discussion of security in cloud deployments in more detail, see "Data Security" on page 62.

2.4 System Management and Monitoring

System management and monitoring will be an essential part of the technical administration of the PARCC Assessment System. It will also play an important role in maintaining a secure environment and enforcing policies (e.g., authorization, privacy, and auditing) and standards compliance.

A system management and monitoring framework typically involves setting up monitoring agents on all monitored system entities (e.g., devices, machines), and a central management server that collects and processes the events generated by the monitoring agents. Management event information in the PARCC Assessment System will be published via standard protocols (i.e., JMX or SNMP) to a central management and monitoring server. This server will also perform health monitoring that involves collecting data representing the overall

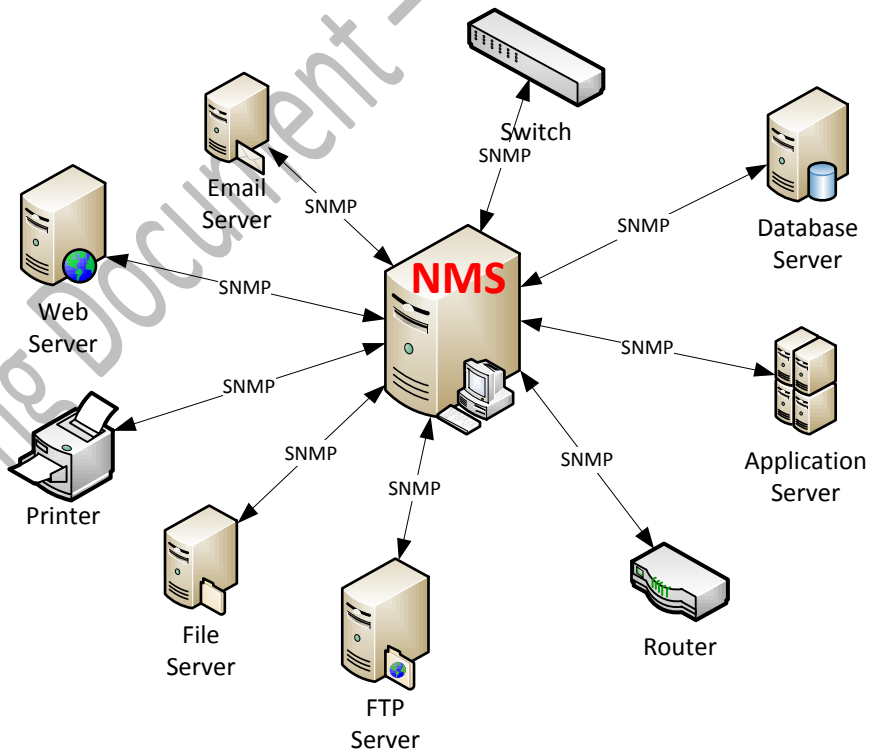
technical health conditions of the system and its components. This data will provide fine-grained, detailed depictions of the health of services, components, and the PARCC technical infrastructure as a whole. Any alerts from the health monitoring will be propagated to support personnel for analysis and action.

Simple Network Management Protocol—SNMP

SNMP is the system management and monitoring protocol most often used today. Most professional-grade hardware devices today come with a built-in SNMP agent ready to be integrated as a network element (NE) in a network management system (NMS). NMS and all NEs in it communicate among each other, exchanging GET, SET, or TRAP messages. An NE can send a TRAP message to the NMS to announce, for example, a particular kind of failure in the device. The NMS can use GET messages to retrieve data from NEs (e.g., the NMS can periodically query an NE for the value of a certain parameter (e.g., consumed bandwidth), and then the NMS can build charts and graphs from that data, warn system personnel of overload conditions, etc. The TRAP and GET messages provide the *monitoring* part of the NMS services. The NMS can also issue a SET message to an NE to change certain parameters on the device (e.g., change routes on a router). The SET message provides the *management* part of the NMS services.

Figure 13 – Example Network Management System with SNMP Diagram illustrates the concepts of NMS, NEs, and SNMP.

Figure 13 – Example Network Management System with SNMP Diagram



The SNMP MIB (Management Information Base) is a collection of variables that are shared between the NMS and the NEs. Hardware and software vendors can extend the MIB by adding new variables to it.

The SNMP protocol is available in several versions. Most hardware and software vendors support SNMPv1 and SNMPv2c. A few support SNMPv3, which supports user-based security.

JAVA MANAGEMENT EXTENSIONS – JMX

JMX is a Java technology for monitoring and managing the performance of the Java Virtual Machine (JVM) at run-time. It is applicable for Java and J2EE-based applications. The JMX has out-of-the-box management tools to monitor and control standard JVM parameters like heap size, CPU utilization, etc.

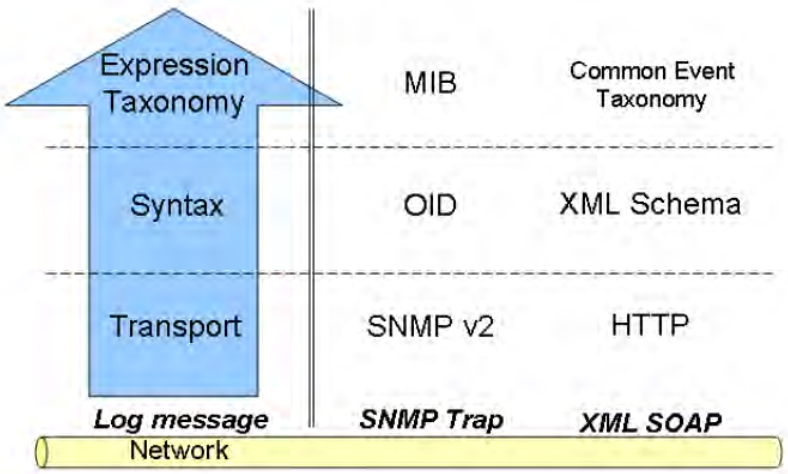
In addition, JMX technology can also be used for managing custom MBeans at run-time without the need to restart the JVM. A custom MBean is a Java component that can expose certain parameters of a J2EE application so that these parameters can be dynamically managed at run-time. In this way, JMX provides custom management and monitoring capabilities for Java-based applications which augment the capabilities of standard SNMP-based management and monitoring solutions.

COMMON EVENT EXPRESSION – CEE

CEE is a framework that enables collaboration in the creation of open, practical, and industry-accepted event interoperability standards for electronic systems. It is developed by MITRE (www.mitre.org) and can be used as a standard for generating, communicating, and consuming log messages across all hardware and software components in the PARCC Assessment System. CEE simplifies the task of establishing and maintaining compliance with various regulatory standards that incorporate audit or security guidelines.

The CEE framework has four sub-elements: log transport, log syntax, expression taxonomy, and logging. These can be thought of as four layers that take different shapes depending on the target area where CEE logging is used. Figure 14 – CEE Application in SNMP and XML SOAP Logging Diagram brings together two monitoring concepts: low-level SNMP-based monitoring (discussed previously) and high-level XML SOAP logging.

Figure 14 – CEE Application in SNMP and XML SOAP Logging Diagram



COMMERCIAL MONITORING AND MANAGEMENT TOOLS

This section provides short descriptions of available commercial monitoring and management tools that illustrate the range of capabilities of the tools.

IBM Tivoli (www-01.ibm.com/software/tivoli/)

The IBM Tivoli Monitoring (ITM) software is a comprehensive monitoring and management solution that can optimize IT infrastructure performance and availability. It can be used to manage operating systems, databases, and servers in distributed and host environments.

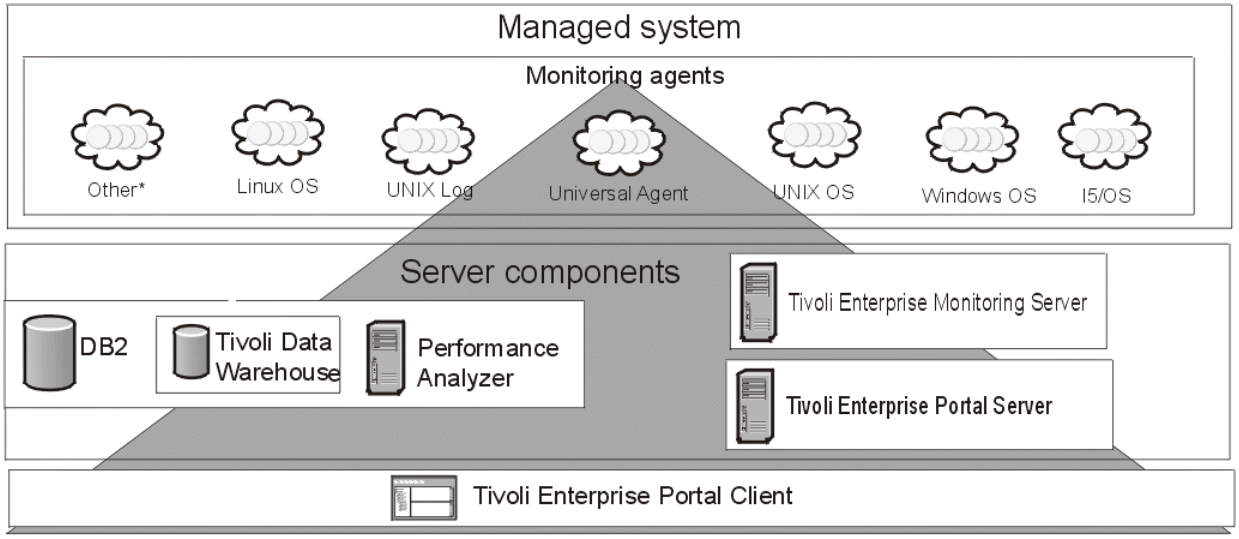
Core Features

The following are the basic features of IBM Tivoli Monitoring software:

- Provides a common, flexible, and easy-to-use browser interface and customizable workspaces to facilitate system monitoring.
- Detects and recovers potential problems in essential system resources automatically.
- Offers lightweight and scalable architecture, with support for IBM AIX, Sun Microsystems Solaris, Microsoft Windows, Linux, and IBM System z monitoring software.
- Includes easy-to-use warehouse and advanced reporting capability.
- Helps to ensure that IT resources and staff are operating efficiently and effectively when combined with composite application, event, network, and service-level management solutions from IBM Tivoli.
- Lowers total cost of ownership with new features that automate the maintenance and support of ITM-based agents.
- Operating systems supported: IBM AIX, Hewlett-Packard HP-UX, Apple iOS family, Sun Microsystems Solaris, Microsoft Windows family.

Figure 15 – Base Architecture of IBM Tivoli Monitoring Software Diagram illustrates the base architecture of IBM Tivoli monitoring software.

Figure 15 – Base Architecture of IBM Tivoli Monitoring Software Diagram



Zyrion Traverse (www.zyrion.com)

Zyrion Traverse is a scalable network and systems monitoring software product that presents correlated views of networks, servers, and applications. In addition to systems and network management, Zyrion Traverse also provides application monitoring of databases, Web applications, Java applications, and mail servers (e.g., Exchange and Blackberry Enterprise Server).

Core Features

- Monitoring features include: Bandwidth monitor, Linux Server monitoring, Windows monitoring, Cisco router monitoring, Oracle, MySQL, SQL Server monitor, Exchange, and Active Directory.
- Reporting features include: SLA, real-time event logs, traps, syslogs, capacity planning, performance, and trend analysis.
- Free trial is available.

NimBUS (www.nimsoft.com)

NimBUS is a service-level monitoring solution (completely developed in-house) that provides scalable, resilient, and reliable monitoring capabilities for organizations that wish to proactively manage critical IT resources against service-level agreements. These resources include, but are not limited to, servers, hosts, applications, databases, network services, and network devices.

Core Features

- Real-time performance monitoring and reporting of potential problems.
- SLA definition, monitoring, and reporting.
- Customizable business service and operations dashboards.
- End-to-end response time measurement with end-user service levels.

For data collection and automation, NimBUS offers a comprehensive suite of infrastructure monitoring robots and probes. NimBUS probes will enable full coverage of heterogeneous IT infrastructures. Monitoring probes include support for networks, databases, servers, middleware, email, applications, Web-based services, directory services, and much more. NimBUS's open APIs, flexible architecture, and out-of-the-box third-party integrations and gateways, ensure that adaptation to other management tools and service-level monitoring processes is easily achieved. With NimBUS, all service-level monitoring functions are inherent; they are written collectively as a single architecture and single code base. The result is easy installation, deployment, configuration, administration, and usability. With NimBUS there is no requirement for strenuous installation integrations and ongoing administration efforts.

OPEN-SOURCE MONITORING AND MANAGEMENT TOOLS

This section provides short descriptions of available open-source monitoring and management tools that illustrate the range of capabilities of the tools.

Nagios (www.nagios.org)

Wikipedia describes Nagios as "a very popular open-source system monitor, network monitoring, and infrastructure monitoring software application. Nagios offers complete monitoring and alerting for servers, switches, applications, and services."

Core Features

- Monitoring network services like SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH.
- Monitoring host resources (i.e., processor load, disk usage, system logs).
- Custom probes via plugins.
- Remote monitoring through SSH or SSL encrypted tunnels.
- Parallelized service checks.
- Alerting.
- Redundant monitoring hosts.
- Optional Web interface.

Wikipedia contributors, "Nagios," *Wikipedia, The Free Encyclopedia,* en.wikipedia.org/w/index.php?title=Nagios&oldid=498419117 (accessed June 22, 2012).

Zenoss (www.zenoss.com)

Wikipedia describes Zenoss as "an open-source application, server and network management platform based on the Zope application server. Zenoss provides a Web interface that allows system administrators to monitor availability, inventory/configuration, performance, and events."

Core Features

- Monitoring availability of network devices using SNMP, SSH, WMI.
- Monitoring network services like HTTP, POP3, NNTP, SNMP, and FTP.
- Monitoring host resources (i.e., CPU, memory, disk usage).
- Time-series performance monitoring of devices.
- Event management tools.
- Automatic network resource discovery.
- Alerting system.
- Nagios plugin format support.

Wikipedia contributors, "Zenoss," *Wikipedia, The Free Encyclopedia,* en.wikipedia.org/w/index.php?title=Zenoss&oldid=491333036 (accessed June 22, 2012).

Zabbix (www.zabbix.com)

Wikipedia describes Zabbix as "a network management system designed to monitor and track the status of various network services, servers, and other network hardware. Zabbix offers several monitoring options. Simple checks can verify the availability and responsiveness of standards services such as SMTP or HTTP without installing any software on the monitored host."

Core Features

- Monitoring host statistics like CPU load, network utilization, disk space, etc.
- Monitoring SNMP, TCP, ICMP over IPMI, JMX, SSH, telnet.
- Supports a variety of real-time notification mechanisms including XMPP.

Wikipedia contributors, "Zabbix," *Wikipedia, The Free Encyclopedia,* en.wikipedia.org/w/index.php?title=Zabbix&oldid=498532690 (accessed June 22, 2012).

MIDDLEWARE AND INTEGRATION SOFTWARE

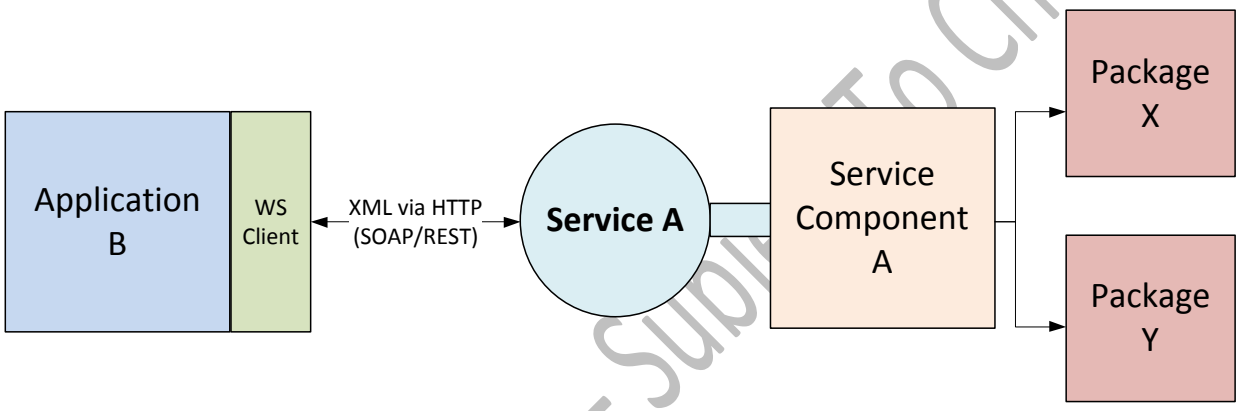
The middleware layer in the PARCC Assessment System Architecture will be built using service-oriented architecture (SOA) principles and designs. A layered service approach will be used to package component functionality and expose it as services to be used by other components and services. The exposed services will be stateless, coarse-grained, and loosely-coupled.

The recommended layered-service mechanism (also known as an SOA stack) is based on the Open Group Standard's "SOA Reference Architecture Technical Standard" (www.opengroup.org/soa/source-book/soa_refarch/index.htm) and will consist of three layers:

- Service Component Layer
- Services Layer
- Business Process Layer

The basic layer is the service component layer. Figure 16 – Service Component as a Façade Diagram shows a service component internally implemented.

Figure 16 – Service Component as a Façade Diagram



This figure shows a service component internally implemented by using the functionality of packages X and Y, and at the same time, exposed as a service, A, consumed by application B, which serves as a Web-service client.

- The consumer (application B) can be another PARCC Assessment System component (internal or external).
- Application B is coupled only to the description of service A (i.e., it is not dependent or impacted by any of service A's implementation details).
- Service component A acts as a service implementation facade hiding the actual physical implementation performed by Package X and Package Y.
- Packages X and Y, in turn, can be either internal or external components exposing their services in a similar way, or they can be low-level standalone packages (e.g., Java packages).

Note: Those packages might be replaced with different implementations in the future, but this implementation detail would be transparent to application B, because it would still see the advertised description of service A that would not be changed by the internal refactoring.

The Service Component layer will be the core implementation layer of the PARCC Assessment System middleware, providing the backbone of the SOA stack.

The next layer up the SOA stack is the Services Layer. Its purpose is to take enterprise components and externalize a subset of their interfaces, which are then made available to

outside consumers. The services layer contains all the services exposed by the underlying service component layer. The specification of each service includes a description of the functionality of the service, which contains (but is not limited to) a formal WSDL (Web Service Definition Language) file. In addition to this, the service specification could often include an informal policy document, SOA governance descriptions, and other documents that show service dependencies or classifications.

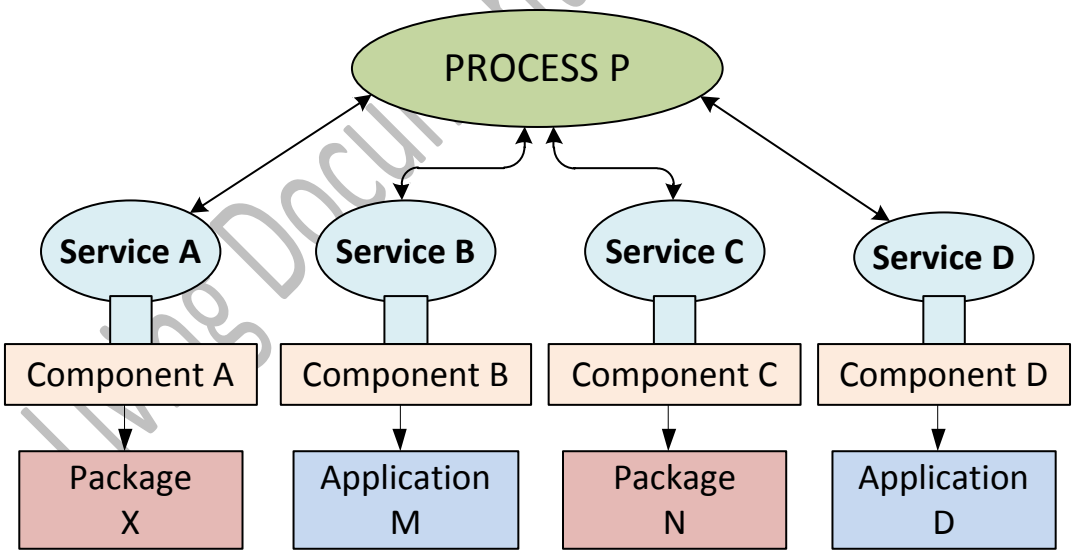
The services in the Services Layer will be accessed using different transports and will provide the fabric of the PARCC Assessment System middleware. The response of each service call can be further transformed into a suitable format (e.g., HTML or XML) and transported to other layers in the assessment system using an Enterprise Service Bus (ESB) or other similar technologies.

The services in the Services Layer can be atomic or composite.

- **Atomic Services.** Use only service components from the Service Component Layer to implement their advertised functionality.
- **Composite Services.** Can use service components from the Service Component Layer, as well as other services from the Services Layer.

The last layer in the PARCC Assessment System middleware stack is the Business Process Layer. This layer provides orchestration by assembling one or more services from the Services Layer to implement a particular business process. Figure 17 – Business Process Orchestration in the Business Process Layer Diagram illustrates this concept.

Figure 17 – Business Process Orchestration in the Business Process Layer Diagram



The business process layer is a set of sequencing processes that organizes the flow of multiple service calls using the services available in the Services Layer as building blocks. Business logic, expressed in Business Process Execution Language (BPEL), will be used to organize those service flows, as parallel or sequential tasks, based on business rules, business policies, and business requirements. Business Process Model and Notation (BPMN) can also be used to depict the

high-level PARCC business processes. BPMN is a graphical notation used to visualize the end-to-end flow of a business process. The primary goal of BPMN is to provide a business process modeling notation understandable by all business users. Since BPEL is currently considered the most important standard for business process execution languages, a translation to BPEL is specified in the BPMN standard.

WEB SERVICES (SOAP/REST)

Web services represent an increasingly popular technique for developing, deploying, and consuming services in an SOA infrastructure, enabling location transparency by utilizing registries such as UDDI for runtime discovery. The typical protocol for Web services is HTTP/HTTPS. Clients can locate the desired service dynamically by requesting the service from the registry. The Web services architecture provides benefits of loose-coupling by providing a mechanism to find, bind, and invoke the service dynamically.

Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST) are two types of Web services. They have different architectural properties and usage scenarios which are described below.

REST is not a protocol or a standard but rather an architectural style, as first defined in 2000 by Roy Fielding in his dissertation *Architectural Styles and the Design of Network-based Software Architectures.* Web services created in the REST architectural style:

- Use the HTTP protocol as the underlying protocol.
- Utilize built-in HTTP features like caching and stateless conversations.
- Expose server-side resources through URI (Universal Resource Identifiers).
- Make use of the standard HTTP operations (e.g. GET, PUT, DELETE, POST, HEAD) for manipulating the server-side resources through their representations.

SOAP is a detailed specification for implementing Web services that:

- Uses many protocols, not just HTTP.
- Relies on XML for its message format.
- Defines method calls and input/output message structures through Web Service Definition Language (WSDL).
- Utilizes service discovery through Universal Description Discovery and Integration (UDDI).
- SOAP is extensible, allowing for accommodation of additional features (e.g., support for Web service security is possible through WS-Security).

Table 4 -- Comparison of SOAP and REST summarizes the pros and cons of REST and SOAP.

Table 4 -- Comparison of SOAP and REST

	REST	SOAP
Pros	• Simpler to understand, learning curve is not steep. • Easier for development and testing. • Uses existing Web infrastructure (e.g., caching, which aids scalability and performance).	• Can use any transport (e.g., SMTP or JMS), not just HTTP/HTTPS. • Industry standard with well-defined protocol. • Supports security and transactions, allowing for more flexibility during API design.
Cons	• Uses only HTTP/HTTPS transport. • Lacks definitive standards. • Considered architectural approach, not a protocol. • Support for security (beyond SSL) and transactions needs to be custom-built.	• High startup costs due to complexity of standards, learning of development tools, and vendor differences. • The additional flexibility comes at the cost of additional complexity.

Based on the summary of pros and cons listed above, ASG and Pacific Metrics can make these recommendations:

- SOAP is recommended for communication between components where security or transactional/reliable messaging is needed and the cost of providing an equivalent custom REST-based solution is deemed too high.
- REST is recommended for create-read-update-delete operations between internal PARCC components or between components in a single component grouping.
- REST is recommended for components hosted on the same local area network (LAN) or internal network.

Middleware and Integration Software Vendor Capabilities and Offerings

This section provides a summary of middleware and integration software products showing their capabilities.

IBM, Red Hat, Oracle, and Microsoft are major vendors providing middleware software. Vendors such as Axway, SAP, TIBCO, Informatica, Pervasive, and webMethods were specifically founded to provide Web-oriented middleware tools. Groups such as the Apache Software Foundation, OASIS, OMG, OpenSAF, and the ObjectWeb Consortium (now OW2) encourage the development of open-source middleware and also push for standards-based development of commercial software.

Oracle Fusion Middleware (www.oracle.com)

Oracle Fusion Middleware (OFM, also known as Fusion Middleware) consists of several software products from Oracle Corporation. OFM spans multiple services, including Java EE and developer tools, integration services, business intelligence, collaboration, and content management. OFM depends on open standards such as BPEL, SOAP, XML, and JMS.

TIBCO ActiveMatrix (www.tibco.com)

TIBCO ActiveMatrix is a technology-neutral platform designed to simplify the development, deployment, and management of composite business process management (BPM) and service-oriented architecture (SOA) applications. The ActiveMatrix family includes products for service creation and integration, distributed service and data grids, packaged applications, BPM and governance.

OpenSAF (opensaf.org)

OpenSAF is an open-source project for developing middleware that is based on industry-standard, open interfaces for applications requiring uninterrupted 24/7 availability. OpenSAF is actively supported by leading companies in the communications and enterprise computing industries.

2.5 SECURITY REQUIREMENTS FOR APPLICATIONS AND END-USER ACCESS

The PARCC Assessment System must enforce stringent security checks and rules involving the operation of its applications, the storage and transfer of its data, and the controlling of end-user access. Most of the decisions in this area will directly impact the implementation of use cases in these functional areas (described in *High-level Application Architecture*):

- 006 – Registration
- 007 – Scheduling and Assignment
- 008 – Student Delivery

The assessment system security implementation must satisfy these basic principles of information security:

- **Confidentiality.** Ensures that the system data and functions are protected from unauthorized access.
- **Integrity.** Guarantees that system data has not been modified or interfered with by a third party (whether malicious or not).
- **Authentication.** Ensures that the identity of a user or a remote system accessing the system is valid and correct and has not been impersonated or compromised in any way.
- **Authorization.** Ensures that that a valid, authenticated user or remote system has the appropriate rights to access system data or execute system functions.

- **Non-Repudiation.** Guarantees that all actions, once performed, cannot be denied by the user or the system itself.

Comprehensive security controls as defined by *ISO/IEC 27001* and *NIST Special Publication SP 800-53 revision 3* must be put in place to ensure that the assessment system is properly secured. The security controls are techniques to avoid or minimize security threats and risks. Examples of such security threats are network sniffing, man-in-the-middle attacks, session hi-jacking, password cracking, cross-site scripting attacks, and SQL injection.

End-user Authentication/Authorization and Access Control

All PARCC Assessment System end users will be authenticated to the system using a single sign-on process. *Single Sign-on* (SSO) is the ability for users to access multiple software applications from multiple sources and vendors by logging in just once with a single username and password—preferably from any location. Security Assertion Markup Language (SAML) is an XML standard that allows secure Web domains to exchange user authentication and authorization data. The SAML protocols for single sign-on will be used. Open-source frameworks for single sign-on (e.g., OAuth, OpenID, and Shibboleth) should be explored as implementation options for the Identity Management component of the assessment system.

Once authenticated, the users will be authorized to perform specific functions across PARCC subsystems based on their assigned role. Each role defines what the user can access and the level of this access. Such end-user access control policy is known as RBAC (role-based access control).

Security attributes can also be used to describe the basic properties of all assessment system internal system entities with regard to the security and safe-guarding of information. Examples of such entities are PARCC student records, test results records, and test report files. The security attributes will then be used to enable access control and flow control policies in the system. End users will be given a set of roles, each defining what resources with what security attributes the end user can access.

The security services of the assessment system must provide for comprehensive account management, access enforcement, and system use notifications (e.g., previous logon/access notifications).

One option to provide a more secure user authentication is the two-factor authentication scheme. It requires that the user present to the system at least two of the three well-known authentication factors:

- Something the user knows.
- Something the user has.
- Something the user is.

While two-factor authentication schemes are more secure, they also have an impact on usability, so this trade-off needs to be further examined before a decision is made.

REGULATORY COMPLIANCE

An important aspect of PARCC Assessment System security will be regulatory compliance. There are two federal acts that relate to the security implementation of the assessment system, namely FERPA and COPPA.

- **FERPA – Family Educational Rights and Privacy Act.** FERPA is a U.S. federal law that protects the privacy of student education records. The law applies to all schools that receive funds from the U.S. Department of Education and will, therefore, apply in full force to the PARCC Assessment System. The law protects student privacy by prohibiting the disclosure of personally identifiable information from education records without prior written consent. FERPA was written specifically for students and guarantees them the right to inspect and review their education records, the right to seek to amend education records, and the right to have some control over the disclosure of information from those education records. FERPA was amended in 2008 to clarify many rules surrounding data sharing.
- **COPPA – Children's Online Privacy Protection Act.** COPPA places parents in control over what information is collected from their young children online. COPPA applies to operators of commercial Web sites and online services directed to children less than 13 years of age that collect, use, or disclose personal information from children.

The PARCC Assessment System security implementation must consider these laws and address relevant sections accordingly.

TEST DELIVERY COMPONENT SECURITY CONCERNS

The Test Delivery component has specific security concerns that should be addressed. To prevent fraud and ensure the validity of the test, special requirements must be considered regarding the test environment. In order to provide a secure test environment using a Web browser as the test client, the desktop and operating system environment, where the browser is running, must be locked-down, so that students taking the test can access and control only one window on the screen (i.e., the one with the test). For example, students should be prevented from switching to another task/application, minimizing the test window, typing a URL in the browser, opening another browser tab in the browser window, going back to the previous page, exiting the test prematurely, etc. Also, the machines on which a test is taken should be prevented from connecting to any other servers on the network except the assessment system servers. This means special network firewall rules must be in place to ensure connectivity only to the designated servers.

There is a trade-off between the ability to satisfy all those security concerns in their entirety, and the implementation of the Test Client delivering the test.

Web-based Test Client Implementation

Standard Internet browsers (including popular browsers such as Internet Explorer, Firefox, Chrome, Safari, and Opera) can all be used to implement a Web-based test client. In such an environment, most of the actual work, including data processing and data persistence, is done on the server computer (i.e., on the other end of the connection and not in the Web-based test

client environment). The test client is limited to providing only the user interface part of the testing functionality (i.e., input data capture and data output).

Note: All browsers have been designed with great end-user UI flexibility and convenience—but they all have very limited features for tightening the security of the end-user experience.

Native Application Test Client Implementation

On the other hand, the test client can be implemented as a native application written specifically for the target operating system (e.g., Windows or Linux). The native application typically has its own data processing and data persistence capabilities and can provide much greater control over the security of the desktop environment, where the test will be delivered. However, there is a significant cost in terms of development, deployment, and configuration efforts when using a native application for delivering the PARCC Assessment System tests.

There is also an intermediate option wherein the regular browser is used for test delivery, but additional software must also be installed on each client machine—along with a corresponding server component in the PARCC infrastructure—to guarantee the requirements of a secure testing environment. This option does incur some additional development, deployment, and configuration costs, but not as much as in the pure "native application" option.

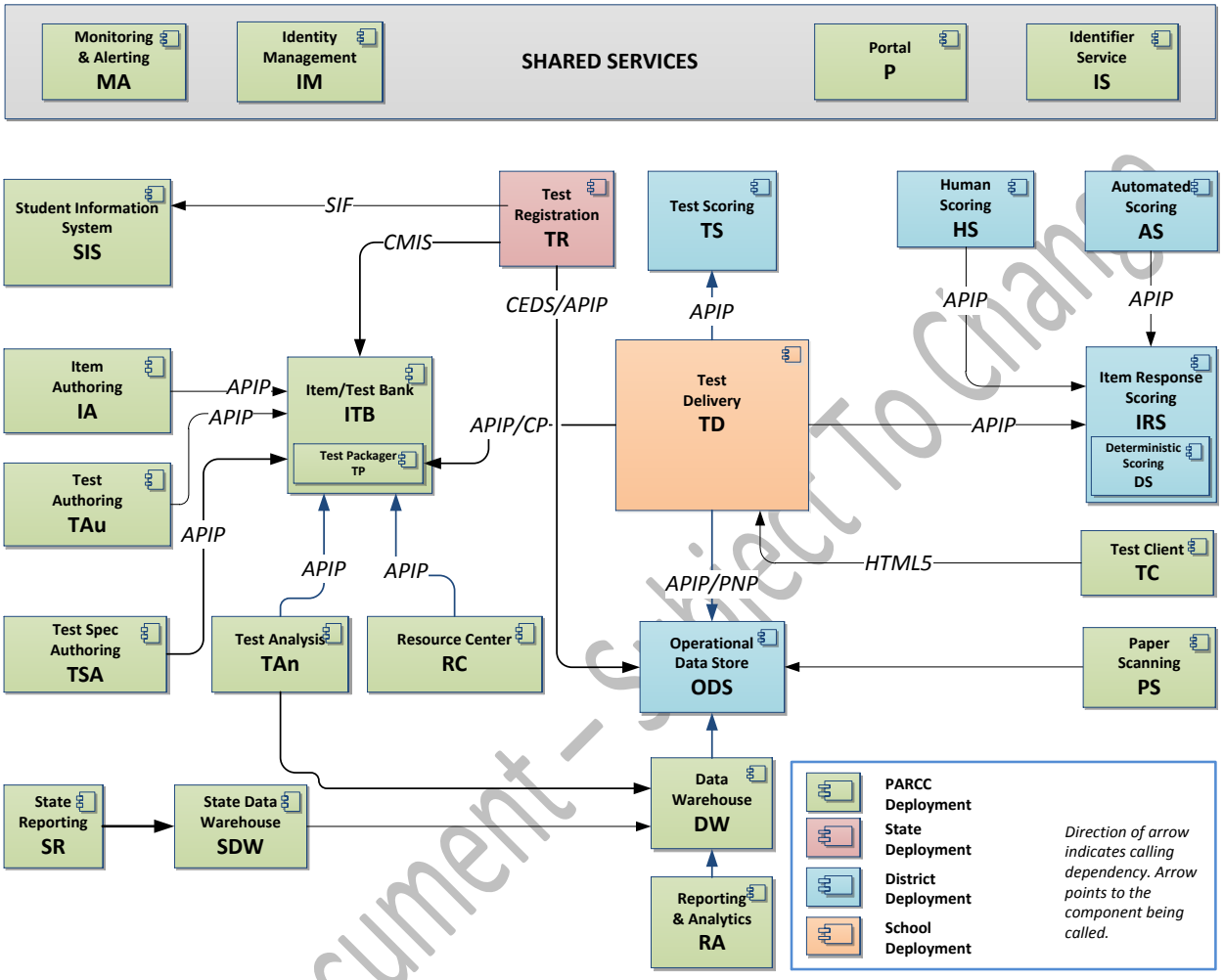
In this option, the Internet browser continues to be the primary vehicle for test delivery in the assessment system. This will be the recommended option for secure test delivery. The additional client software that will ensure a secure desktop testing environment can be commercial off-the-shelf (COTS), procured, or developed as part of the assessment system development effort.

2.6 INTEGRATION WITH EXISTING TECHNICAL ENVIRONMENTS

The PARCC Assessment System will need to integrate with existing technical environments at the state, district, and school levels. Depending on the deployment model for some or all of the assessment system components, certain components can be deployed at the state, district, or school levels for increased performance (e.g., network bandwidth) or other considerations. External and/or existing components (e.g., state SIS or item/test authoring systems) are always going to be deployed as per the specifications by the particular vendor that produces the component—which may or may not be at the PARCC deployment level. Regardless of the component deployment level, interoperability among components will not be affected and will be executed according to the overall component interaction architecture.

Figure 18 – Assessment System Component Deployment at Various Levels Diagram illustrates a typical PARCC Assessment System deployment, wherein some components are deployed at PARCC level, some are deployed at the state level, and some are deployed at the district and school levels.

Figure 18 – Assessment System Component Deployment at Various Levels Diagram

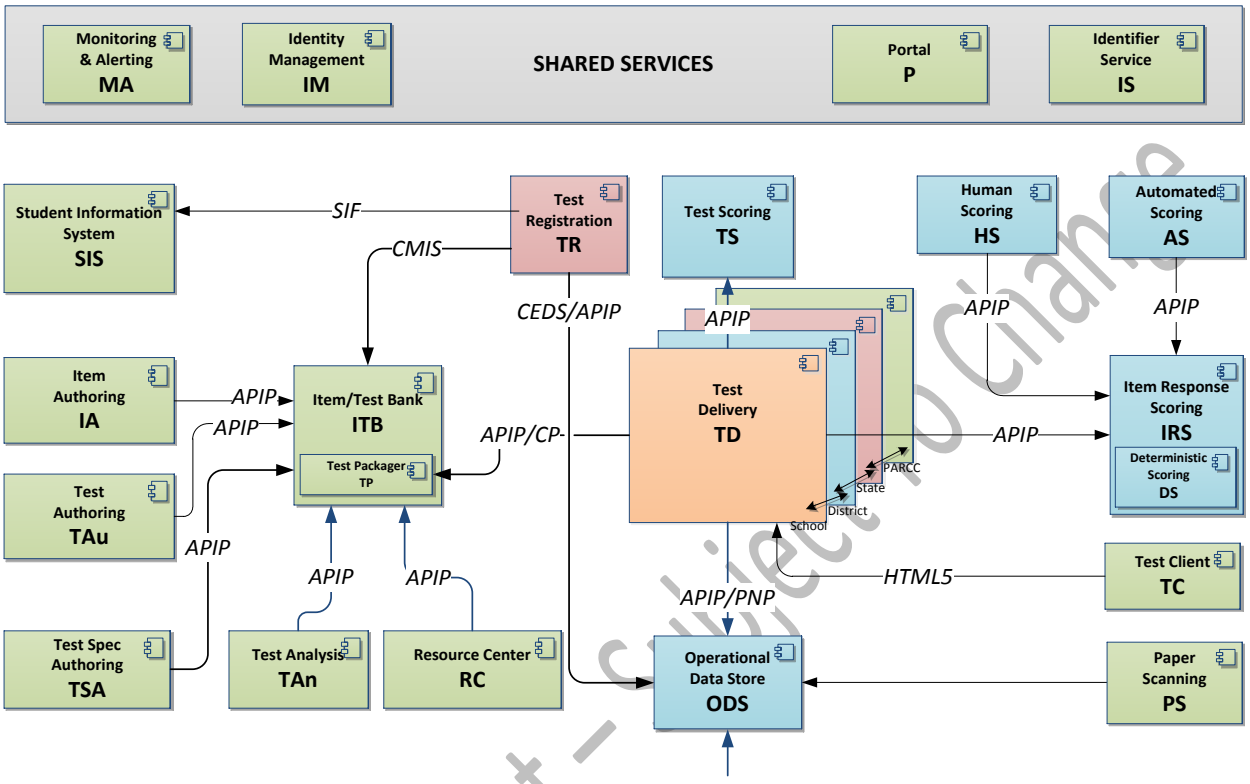


Deploying components at a non-PARCC level will require that these components be developed to expose their functionality via both local and service-oriented interfaces to ensure that the component will function and interoperate correctly in both PARCC-level and non-PARCC-level deployments.

The details of a particular component deployment at a non-PARCC level will be determined by PARCC, the component developer, and the specifics of the technical infrastructure at the non-PARCC level (e.g., state, district, or school). For example, in some cases, this non-PARCC deployment can be implemented with virtualization techniques utilizing existing state/district/school hardware. In other cases, the component may be deployed on new physical hardware.

Figure 19 – Interactions between Components Deployed at Different Levels Diagram shows versions of the same Test Delivery component (deployed at different levels) that will interact with each other to deliver data in and out of the PARCC-level component.

Figure 19 – Interactions between Components Deployed at Different Levels Diagram



2.7 DEVELOPMENT AND TESTING PROCESSES AND ENVIRONMENTS

While not required, it would be beneficial to PARCC if each vendor engaged in developing PARCC components followed development and testing processes based on established frameworks, tools, and methodologies. A typical development environment will utilize:

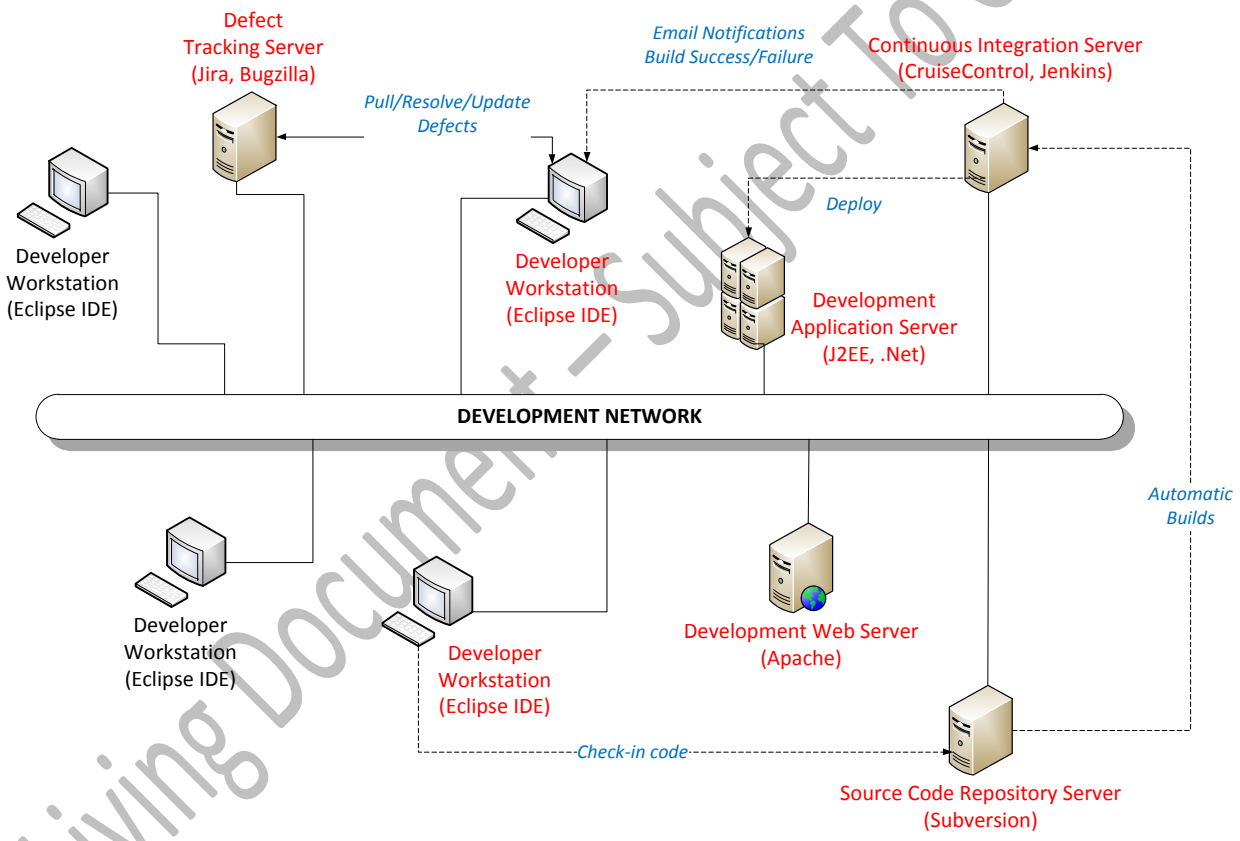
- An integrated development environment (IDE) for code development.
- A version control system for code versioning.
- A build tool for code building.
- A continuous integration tool for automatic builds based on changes in the version control system.
- A unit-testing framework for testing unit-level functionality.
- A test automation tool for automated system testing, possibly integrated with the continuous integration tool.
- A performance testing tool for testing the performance of the system.

There is a multitude of tools, both commercial and open-source, that can be used for PARCC Assessment System development. Each vendor will decide what is the best development toolset and environment to utilize to develop and test the assessment system components.

Development, testing (both functional and performance), and release to production are all activities which happen in their corresponding environments. *Environment*, in this context, means a set of all necessary server-side hardware and software (physical or virtualized) that ensures the operation of the assessment system as it moves through development, testing, integration, and production.

Figure 20 – Recommended Assessment System Development Environment Layout Diagram shows a recommended layout for the PARCC Assessment System development environment.

Figure 20 – Recommended Assessment System Development Environment Layout Diagram



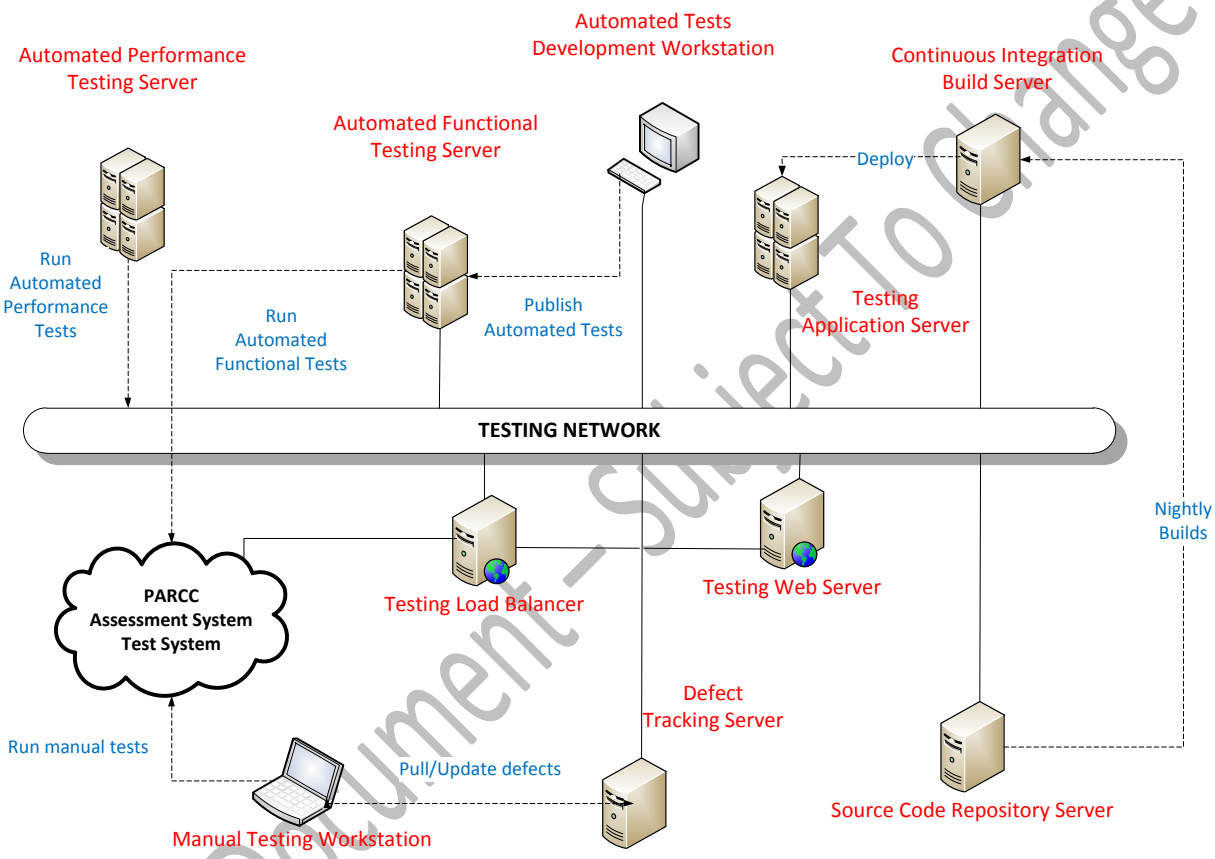
The development network hosts developer workstations with IDE installations, where developers create and change the assessment system programming code as well as execute unit testing.

- Developers check in their code changes into a Source Code Repository Server.
- Upon check-in, code changes are pushed to a Continuous Integration Build Server, where an automatic build occurs and the code is then deployed to the Development Application Server.

- Developers also pull defects from the Defect Tracking Server, work on them by changing the code, and then update the defects' status and description back to the Defect Tracking Server.

Figure 21 – Recommended Assessment System Testing Environment Layout Diagram shows a recommended layout for a testing (validation) environment for the assessment system.

Figure 21 – Recommended Assessment System Testing Environment Layout Diagram



The test environment includes a testing load balancer, testing Web server, and testing application server, wherein the application is deployed after nightly automatic builds that are executed by the Continuous Integration Build Server pulling the source code from the Source Code Repository Server. Both manual and automated testing will be executed to validate the functionality of the PARCC Assessment System.

- Automated tests are developed on Automated Tests Development workstations and published to the Automated Testing Server, where they are executed against the assessment system Test System.
- As defects are encountered during both manual and automated testing, they are entered into the Defect Tracking Server for subsequent resolution by the development team.

The layout also includes an Automated Performance Testing Server for executing automated performance tests.

The integration environment will be of particular importance because of the distributed nature of the development effort for the assessment system. Multiple vendors will deliver components (or groupings of components) at different points in time and the integration environment will be used to validate interoperability among components.

3. INTEGRATION ARCHITECTURE PLAN

This section outlines the guidelines and recommended approaches to integration, movement, and security of the PARCC Assessment System data, both inside and outside of the assessment system. It also provides a technology integration template to be used by vendors to ensure that their offerings comply with assessment system architecture.

3.1 DATA INTEGRATION

The PARCC Assessment System will need to integrate data from a variety of sources within the assessment system itself as well as external data sources (e.g., student information systems, item and test authoring systems, scoring engines, and state-level data warehouses). As described in *Interoperability Standards Review*, industry-standard high-level data standards such as APIP, QTI, and CEDS will be used for data representation and data transfer between those systems.

DATA INTEGRATION WITHIN PARCC ASSESSMENT SYSTEM COMPONENTS

Data in the PARCC Assessment System will be stored in three major data hubs: the Operational Data Store (ODS), the Data Warehouse (DW), and the Item/Test Bank. Individual components may opt to use their own independent data stores to keep transient data while the component is performing its functions.

Internal components developed as part of a particular grouping (see *Component Dependency Matrix* in *High-level Project Portfolio Schedule*) will most likely use a common data store for transient data and will have direct, low-level access to the ODS, so data integration between these components will be realized directly through the common data storage. This will reduce component response times, thus improving overall system performance.

External components and internal components from different component groupings will use higher-level protocols to exchange data, either in real-time or through batch updates. They can do this using Web services or ETL tools (see "Data Movement" on page 62).

DATA INTEGRATION WITH MEMBER STATES' EXISTING SYSTEMS

Existing student information systems (SIS) at the state level will provide core student data and other data needed for the operation of the PARCC Assessment System. The test registration process executed through the Test Registration component will use the industry-standard SIF protocol to pull data from the state SIS. This could be implemented as either a real-time or an asynchronous batch process, depending on the availability of the state SIS.

3.2 DATA MOVEMENT

The "Data Movement Model" section in the *Information Architecture* document outlines the different types of data produced and consumed in the PARCC Assessment System as well as how this data moves through the different components and subsystems, both internal and external, using industry-standard data-exchange protocols such as APIP and QTI. This section focuses on the tools and technologies that can be used to facilitate the technical movement of data. It directly addresses the technical aspects related to use cases in functional areas (described in *High-level Application Architecture*):

- 004 – Content Movement
- 014 – Data Export

Data movement between assessment system components during real-time interactions, such as submitting authored item data from the Item Authoring component to the Item/Test Bank component or submitting items from the Test Delivery component to the Operational Data Store component, can be implemented via standard service-oriented technology using Web services (i.e., SOAP/REST). Payloads will conform to the APIP/QTI data standards. Security and transactions would be handled through the respective WS-Security protocols.

Moving data from the Operational Data Store component to the Data Warehouse component would best be accomplished using an extract, transform, and load (ETL) tool. ETL tools are used to provide continuous or batch-level movement of data from one data source/data store to another. During the ETL process, data is extracted from one or many sources, transformed and normalized through business rules, and then loaded into the target data store. Typically the target data store is a data warehouse but it could be any system that can consume the data.

3.3 DATA SECURITY

Data storage and movement in the PARCC Assessment System need to adhere to applicable regulatory constraints (e.g., FERPA and COPPA). The necessary security mechanisms need to be in place when storing and moving most data entities, especially student data and test results data. Hashing and encryption techniques will be used when sensitive data is stored in all data stores, and secure data transfer protocols (e.g., SSL, HTTPS, and WS-Security) will be used when data is transferred from one component to another. In addition, any transient data should be subject to periodic purging to minimize the risks of unauthorized access.

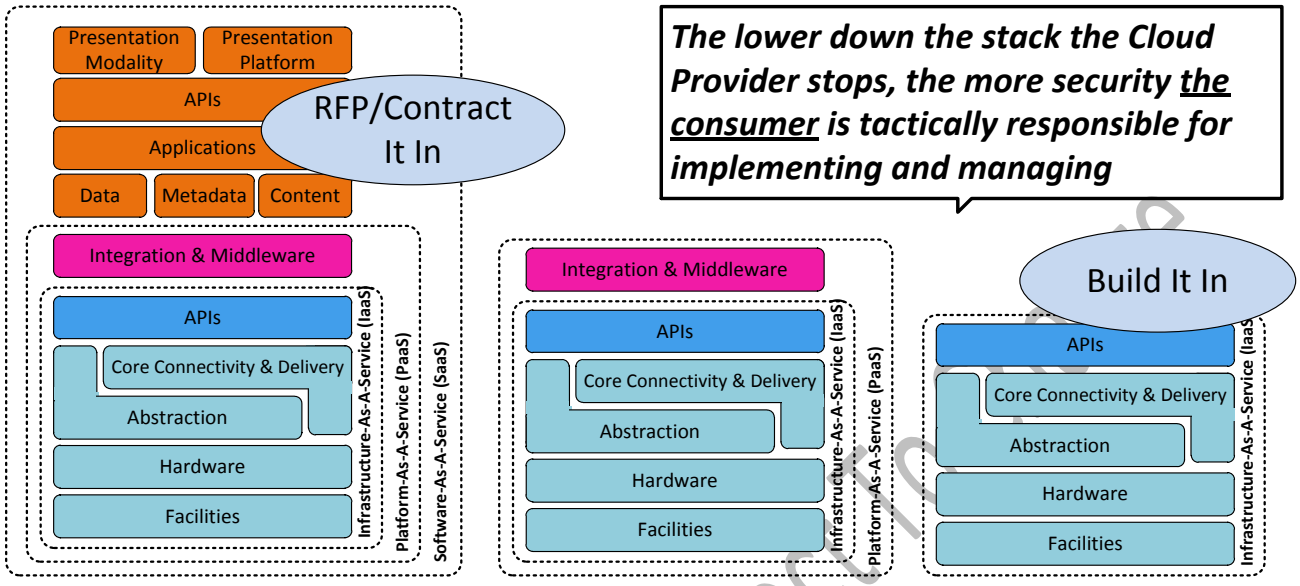
Table 5 – Data Security Life Cycle Phases and Activities shows the data security life cycle phases based on a definition by Securosis (https://securosis.com/blog/data-security-lifecycle-2.0) and related activities that occur during each phase. The security implementation of the assessment system must ensure that at a minimum all of those activities occur as described.

Table 5 – Data Security Life Cycle Phases and Activities

Activities/Phases	Create	Store	Use	Share	Archive	Destroy
Classify	■					
Assign Rights	■					
Access Controls		■				
Encryption		■		■	■	
Rights Management		■	■			
Content Discovery		■				■
Activity Monitoring			■			
Logical Controls			■	■		
Application Security			■	■		
Asset Management					■	
Crypto-Shredding						■
Secure Deletion						■

Depending on the deployment model chosen (internal IT infrastructure, third-party hosting, or cloud deployment), the burden of ensuring data security may lie on PARCC, the third-party hosting provider, or the cloud infrastructure provider. In the internal IT infrastructure deployment model, PARCC will bear all of the security responsibility. In cloud deployments, PARCC's responsibility for data security will be higher if the IaaS service model is chosen. This is a trade-off between the flexibility of the IaaS service model and the higher burden of data security (and other infrastructure concerns) that will be imposed on PARCC. This trade-off is summarized best by Figure 22 – Security in Cloud Deployments Diagram, taken from the *Cloud Security Alliance's Security Guidance for Critical Areas of Focus in Cloud Computing, V2.1, 2009.*

Figure 22 – Security in Cloud Deployments Diagram



The PARCC Assessment System security design must address the following concerns if cloud deployment is utilized:

- PARCC must know the geographic location of the data. This must be stipulated in the SLA and the relevant contracts.
- Understand all circumstances wherein the cloud provider could be obligated to disclose assessment system data.
- PARCC should maintain a "Default Deny All" policy for both PARCC staff and cloud service provider staff. In such a policy, by default, no access to any PARCC data is allowed to anyone. Access to specific data is granted to specific PARCC and cloud service provider staff only after a corresponding request has been formally approved by PARCC's data administration body. This should be stipulated in the contract.
- All assessment system data (i.e., data at rest and data in transit) should be encrypted.
- Require that cloud service provider does not commingle backed-up data with other customers.
- Understand cloud provider policies and processes for data retention and destruction.
- Applications should use end-to-end transport level encryption (SSL, TLS, IPSEC) to secure data in transit between applications deployed in the cloud.

3.4 API DESIGN GUIDELINES FOR VENDORS

The *application programming interface* (API) is essentially the programming contract between two entities (i.e., systems, components, etc.) communicating with one another using an agreed-upon protocol. This protocol would specify, for example, the name of the operations, the sequence in which they execute, and the format of the data exchanged.

In an API contract, the *producer* (i.e., the entity creating the API) announces a public interface containing the advertised functionality, and the *consumer* entity uses the API to access the functionality advertised by the producer. Either of the two entities can be a producer or consumer, depending on which entity initiates the conversation.

In this context, the PARCC Assessment System can be either the producer of an API—wherein one or more of the assessment system components advertise certain functionality made available to third-party systems to consume—or the consumer of an API exposed by a third-party component made by an external vendor.

- An example of a third-party component playing the role of a *consumer* would be an external Item Authoring component which will use the API of the internal Item/Test Bank component to access certain functionality advertised by the Item/Test Bank component, such as the ability to store a new test item to the Item/Test Bank or the ability to update an existing test item in the Item/Test Bank.
- An example of a third-party component playing the role of a *producer* would be an external student information system exposing its API to retrieve student information which will be consumed by the Test Registration component to manage student registration.

Vendors who will be developing external components interfacing with the assessment system, as well as vendors who will be developing some or all of the internal components of the assessment system, need to incorporate the following general guidelines when designing their components so that they will be compatible with PARCC Technology Architecture:

- PARCC will use Web services (both REST and SOAP) as underlying mechanisms for communication between loosely-coupled internal and external components. Any third-party product should be capable of utilizing those protocols as well as creating/consuming the related artifacts for those protocols (e.g., WSDLs) when creating components that need to communicate with assessment system components.
- PARCC will utilize comma-separated values (CSV), Extensible Markup Language (XML), and Java Script Object Notation (JSON) as underlying data structure formats for both input and output data exchanged between communicating components. Third-party products should be capable of parsing and consuming data using these formats.

- PARCC will use the OAuth security framework v1.0 (and v2.0 whenever it is released, tools.ietf.org/html/rfc5849) for authentication/authorization tasks—in addition to other security frameworks. Vendors of third-party components should be prepared, at a minimum, to incorporate OAuth into their components as well. Vendors should also be prepared to incorporate the other security frameworks that PARCC may be using, as they become implemented.

In addition to this, vendors are encouraged to abide by best practices and guidelines for API design when they expose their functionality for the assessment system to consume. Such best practices and guidelines include:

- **Do not expose more than you want**. A minimal API is one that has as few classes as possible and as few public members per class as possible. This makes the API easier to understand and maintain. An API should be as small as possible.
- **APIs should be complete and provide all expected functionality**.
- **APIs should be easy to memorize**. Choose consistent and precise naming conventions. Use recognizable patterns and concepts and avoid abbreviations. Be consistent with the meaning of words used. The same word should mean exactly the same thing across your layers.
- **Choose appropriate names for classes, functions, and parameters**. It should be clear from the name whether a function has side effects or not. Parameter names are an important source of information to the programmer, and it is worth spending some time naming parameters appropriately.
- **Write documentation**. An API is supposed to be read and understood by others. Try to be succinct and precise and cover every single function and class.
- **Avoid long parameter lists**. If you have long parameter lists, your API will not be usable without constant reference to its documentation, because most programmers cannot remember long parameter lists. You can avoid long parameter lists by using helper classes or structures to hold aggregates of parameters.
- **Prefer a factory method to a constructor**. It is more flexible to expose a factory method than to expose a constructor.
- **Always version the API**. This allows for traceability and flexibility.
- **Support multiple data formats**.
- **Provide robust failure handling**.
- **Ensure that data-modifying operations are idempotent**. *Idempotence* is a property of an operation wherein multiple applications of the operation do not change the end result beyond the initial application. Because the network is not always available/reliable, certain API calls can be expected to get retried. In these cases, the API should be capable of detecting and handling duplicate calls reliably without causing data inconsistencies.

4. RELATED DOCUMENTS

Table 6 – Reference Materials lists the supporting PARCC Assessment System Architecture documents referenced in this document.

Table 6 – Reference Materials

Title	Description
Invitation to Negotiate – Technology Architecture, Interoperability Standards Development and System Implementation Services, *ITN 2012-22*	Florida Department of Education solicitation for the PARCC Architecture Program.
Technology Architecture, Interoperability Standards Development and System Implementation Services – Technical Reply, *Final (Dated 04-10-12)*	Pacific Metrics/IBM's technical response to ITN 2012-22.
PARCC Assessment System Architecture Key Technology Priorities Summary	The deliverable that summarizes recommendations, approaches, and considerations for handling PARCC's key technology priorities for sustainability and low-cost impact.
PARCC Assessment System Architecture High-level Application Architecture	The deliverable that defines the high-level application architecture by describing all applications that compromise the assessment system architecture and high-level functional concerns for the architecture.
PARCC Assessment System Architecture Information Architecture	The deliverable that defines the data aspects of the overall assessment system architecture by describing where data resides, how data is delivered and maintained, how data will be managed, and how data will be accessed for reporting.
PARCC Assessment System Architecture High-level Project Portfolio Schedule	An MS Project schedule for the development of the PARCC Assessment System.
PARCC Assessment System Architecture Component-based Dependency Matrix	The deliverable that defines the procurement-related dependencies among the components of the assessment system.

Title	Description
PARCC Assessment System Architecture Interoperability Standards Review	The deliverable that defines the recommended interoperability standards for the components of the assessment system.
PARCC Assessment System Technology Standards and Protocols Options	The deliverable that defines the options and recommendations for technology standards and protocols to be used in the PARCC Assessment System.

5. EXTERNAL SOURCES

This section provides references to some of the external sources used in the writing of this document.

- **API design:**
  - wiki.netbeans.org/API_Design
  - www.symlab.org/wiki/index.php/Writing_portable_code_and_maintaining_ports
- **Cloud security:**
  Security Guidance for Critical Areas of Focus in Cloud Computing 2.1,
  Cloud Security Alliance
  - cloudsecurityalliance.org/csaguide.pdf
  - wiki.cloudsecurityalliance.org/guidance/index.php/Cloud_Computing_Architectural_Framework
- **Database storage:**
  - http://blog.sphereinc.com/2012/03/pros-and-cons-of-using-nosql-solutions/
- **Definition of cloud computing:**
  National Institute of Standards and Technology (NIST)
- **TEI interactive content and related technologies:**
  - www.technologyreview.com/view/426083/html5-triumphant-silverlight-flash-discontinuing/
  - www.techrepublic.com/blog/webmaster/how-to-replace-flash-and-silverlight-with-html5/995
  - edge.adobe.com/whatisedge.html
  - www.eweek.com/c/a/Application-Development/IBM-Launches-Magetta-HTML5-Tool-as-OpenSource-Answer-to-Flash-Silverlight-669762/
- **Definition of idempotence:**
  en.wikipedia.org/wiki/Idempotence
- **FURPS:**
  Grady, Robert; Caswell, Deborah (1987). *Software Metrics: Establishing a Company-wide Program*. Prentice Hall.
- **J2EE:**
  - java.sun.com/j2ee
- **Monitoring and management tools:**
  - www.monitortools.com/servicelevel/
  - www.slideshare.net/tomdc/open-source-monitoring-tools-shootout

- docs.oracle.com/javase/1.5.0/docs/guide/management/agent.html
- cee.mitre.org/docs/Common_Event_Expression_White_Paper_June_2008.pdf

- **Network requirements:**
  - en.wikipedia.org/wiki/Bandwidth_%28computing%29
  - technet.microsoft.com/en-us/library/cc785130%28v=ws.10%29.aspx
  - en.wikipedia.org/wiki/Bit_rate#Multimedia
  - www.adobe.com/devnet/flash/apps/flv_bitrate_calculator.html
- **Security:**
  - Guide to Security for Full Virtualization Technologies,
  - National Institute of Standards and Technology (NIST), January 2011.
- **Web services, REST and SOAP:**
  - http://www.infoq.com/articles/rest-soap-when-to-use-each
  - http://nordsc.com/ext/classification_of_http_based_apis.html#uri-rpc
  - Roy Fielding's dissertation (where REST is defined):
    http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
  - http://wanderingbarque.com/nonintersecting/2006/11/15/the-s-stands-for-simple/
  - http://duncan-cragg.org/blog/post/setting-data-rest-dialogues/
  - http://en.wikipedia.org/wiki/SOAP
  - http://uddi.org/pubs/uddi-tech-wp.pdf
- **Software architecture:**
  - Draft Technical Standard SOA Reference Architecture,
  - The Open Group:
    www.opengroup.org/projects/soa-ref.../soa-ra-public-050609.pdf

6. TERMS AND ACRONYMS

Table 7 – Definition of Terms and Acronyms provides definitions for the terms and acronyms used in this document.

Table 7 – Definition of Terms and Acronyms

Term/Acronym	Definition	Description
.NET	.NET	A set of Microsoft-developed, Windows-based, technologies that provide enterprise-scale IT capabilities similar to J2EE.
ACID	Atomicity, Consistency, Isolation, Durability	A set of properties which define the level of reliability of database transactions.
ADSL	Asymmetric Digital Subscriber Line	A type of digital subscriber line (DSL) technology that enables faster data transmission over copper telephone lines.
API	Application Programming Interface	A list of operations for a component or sub-system that can be used by other components or sub-systems.
APIP	Accessible Portable Item Protocol	A standard data file format with a focus on accessibility used for exchanging digital test items between assessment systems and item/test banks.
BPEL	Business Process Execution Language	A language used to define the details of a business process.
BPM	Business Process Management	A management approach which promotes business efficiency, customer focus, and tight integration with technology to achieve continuous process improvement.
BPMN	Business Process Model and Notation	A graphical notation for describing the steps in business processes and their interaction with each other.

Term/Acronym	Definition	Description
C#	C#	A Microsoft-developed programming language similar to Java.
C++	C++	A general purpose programming language with object-oriented features.
CEDS	Common Education Data Standards	A data standard which specifies a set of the most commonly used education data elements used to exchange data within and across states, as well as for federal reporting.
CEE	Common Event Expression	An initiative by a community of vendors, researchers. and end users with the goal of standardizing the representation and exchange of logs from electronics systems.
COPPA	Children's Online Privacy Protection Act	A U.S. federal law that places parents in control of what information is collected online from children under 13 years of age.
COTS	Commercial off-the-shelf	A Federal Acquisition Regulation (FAR) term defining a non-developmental item (NDI) of supply that is both commercial and sold in substantial quantities in the commercial marketplace, and that can be procured or utilized under government contract in the same precise form as available to the general public.
CP	Content Packaging	A standard for distribution of distributed digital learning content and resources.
CPU	Central Processing Unit	The main chip in a computer that executes computing instructions utilizing data in memory.
CSS	Cascading Style Sheets	A language used for describing the presentation features of a document.

Term/Acronym	Definition	Description
CSV	Comma-separated values	A data format consisting of rows of data, each containing values separated by commas.
DAS	Direct Attached Storage	A system of multiple physical disk drives directly connected to the computer.
DS1/T1	Digital Signal 1/T-carrier 1	A high-speed connection capable of transmitting data up to 1.5 Mbps.
DS3/T3	Digital Signal 3/T-carrier 3	A high-speed connection capable of transmitting data up to 45 Mbps.
DSL	Digital Subscriber Line	A family of technologies that provide internet access by transmitting digital data over the wires of a local telephone network.
DW	Data Warehouse	A type of database used to store reporting and other (typically multi-dimensional) data.
DW/R	Data Warehouse/Reporting	A combination of two types of non-transactional data stores.
EIS	Enterprise Information System	An information system offering high quality of service in a large data volume environment typically supporting the business needs of a large organization.
EJB	Enterprise Java Beans	A server-side component architecture, part of the Java EE specification.
ESB	Enterprise Service Bus	A technology that provides connectivity between components in the enterprise.
ETL	Extract, Transform, Load	A type of technology used to transfer disparate data among systems.
FC	Fibre Channel	A very fast type of network technology used for storage networking.

Term/Acronym	Definition	Description
FERPA	Family Educational Rights and Privacy Act	A U.S. federal law that protects the privacy of student education records.
FTP	File Transfer Protocol	Standard network protocol used to transfer files across the network from one machine to another machine.
FURPS	Functionality, Usability, Reliability, Performance, and Supportability	A framework for categorizing system requirements into five categories: Functionality, Usability, Reliability, Performance, and Supportability.
HTML5	HyperText Markup Language Revision 5	The 5th revision of the HTML standard still under development which aims to provide more standard and streamlined support for multimedia and complex Web applications across a variety of platforms and devices.
HTTP	HyperText Transfer Protocol	An application protocol for distributed information systems communicating over the World Wide Web.
HTTPS	HyperText Transfer Protocol Secure	A secure version of the HTTP protocol. It uses SSL/TLS protocol on top of the HTTP protocol to provide secure communication between a Web site and a Web server.
I/O	Input/Output	The communication between an information processing system (such as a computer) and the outside world.
IaaS	Infrastructure-as-a-service	A type of deployment service in a cloud infrastructure wherein the user has the most control over the application deployment as well as many aspects of the hardware infrastructure.
ICMP	Internet Control Message Protocol	One of the core Internet protocols used to send error messages indicating a service is not available or a host could not be reached.

Term/Acronym	Definition	Description
IDE	Integrated Development Environment	A software application that developers use to write code in different languages.
IOPS	Input/Output operations per second	A performance measurement used to benchmark computer storage devices like hard disk drives, solid state drives, and storage area networks.
IPMI	Intelligent Platform Management Interface	A standard computer systems interface used for computer monitoring and management. It is a message-based, hardware-level specification used to monitor hardware characteristics like system temperature, voltage levels, fan operation, power supplies, etc.
IPSEC	Internet Protocol Security	A protocol for securing Internet Protocol (IP) communications by authenticating and encrypting each packet of a communication session.
iSCSI	Internet Small Computer System Interface	A storage networking standard for linking data storage devices. It uses the IP protocol to carry SCSI protocol commands over IP networks.
ISDN	Integrated Services Digital Network	A set of communications standards for simultaneous digital transmission of voice, video and data over the traditional public switched telephone network.
ITN	Invitation to Negotiate	The Florida Department of Education's solicitation for the PARCC Assessment System Architecture program.
J2EE	Java 2 Enterprise Edition	A set of Java-based technologies providing enterprise-scale IT capabilities like server-side business components development, messaging, transactions, database connectivity, and many others.

Term/Acronym	Definition	Description
Java EE	Java Enterprise Edition	An equivalent term for J2EE. Sun Microsystems changed the term J2EE to Java EE with version 5 of the platform.
JMS	Java Message Service	A Java-based messaging platform which is part of Java EE. It allows J2EE application components to create, send, and receive messages, enabling the components to be loosely-coupled and asynchronous.
JMX	Java Management Extensions	A Java technology and API for providing JVM management and monitoring status.
JSON	JavaScript Object Notation	A data format used to describe structured data that is readable by both humans and machines. Its syntax is "lighter" than the XML format which fulfills the same purpose.
JSP	Java Server Pages	A Java technology for creating dynamically generated Web pages.
JVM	Java Virtual Machine	The runtime environment wherein Java programs execute.
LAN	Local Area Network	A computer network that interconnects computers in a limited area such as a home, school, computer laboratory, or office building.
MIB	Management Information Base	A set of data elements describing management information available for network devices. It is used to manage the entities in an SNMP-managed network.
MVC	Model-view-controller	A design pattern for constructing systems that separates the implementation into three layers.
NAS	Network Attached Storage	File-level computer data storage accessible over the network using file-sharing protocols.

Term/Acronym	Definition	Description
NE	Network Element	A device in an SNMP-managed system.
NIST	National Institute of Standards and Technology	The U.S. federal organization responsible for creating national science and technology measurements and standards.
NMS	Network Management System	A combination of hardware and software used to monitor and administer a computer network or system.
NNTP	Network News Transfer Protocol	An early Internet application protocol used to exchange data between news servers.
NoSQL	No Structured Query Language	A type of database which does not use the Structured Query Language for manipulating its data, and which often does not implement the full set of the ACID properties. These reduced capabilities are offset by big gains in scalability and performance.
OASIS	Organization for the Advancement of Structured Information Standards	A global consortium that drives the development, convergence, and adoption of e-business and web service standards.
OC	Optical Carrier	A designation used to specify the speed of fiber-optic networks (i.e., transmission speed): • **OC-1.** Up to 51.85 Mbps • **OC-3.** Up to 155.52 Mbps • **OC-12.** Up to 622.08 Mbps
ODS	Operational Data Store	A database used to store transient data during an assessment.
OFM	Oracle Fusion Middleware	A set of middleware products from Oracle Corporation.
OLTP	Online Transaction Processing	A type of processing between transaction-oriented applications across networks.

Term/Acronym	Definition	Description
OMG	Object Management Group	A consortium dedicated to setting standards for distributed object-oriented systems.
OOD	Object-oriented Database	A type of database which attempts to represent and store data in a manner which is very close to how Object-Oriented Languages represent and store data in memory.
OOP	Object-oriented Programming	A type of programming wherein data in computer memory is represented in a hierarchical fashion using techniques such as inheritance and encapsulation.
OPS	Operations per Second	A measure of a computer's performance, usually its processing speed.
ORM	Object-relational Mapper	A software layer that sits between an object-oriented language and a relational database and acts as a data adapter.
OS	Operating System	The main (i.e., kernel) software that drives a computer.
OW2	ObjectWeb 2	A consortium devoted to producing open-source middleware.
PaaS	Platform-as-a-service	A type of deployment for service in a cloud infrastructure wherein the user has the most control over the application deployment.
POP3	Post Office Protocol 3	An application-layer Internet protocol used by client email programs to retrieve email messages from an email server over a TCP/IP connection.
PPP	Point-to-point Protocol	A data link protocol used to establish direct connection between two network nodes.

Term/Acronym	Definition	Description
QTI	Question and Test Interoperability	A standard format for representing assessment content and results created by IMS Global Learning Consortium.
RAID	Redundant Array of Independent Disks	A storage technology that combines multiple disk drives into a single logical unit, providing greater reliability through increased redundancy.
RBAC	Role-based Access Control	An access control mechanism that defines the functionality that an end user can access based on an assigned role.
RDB	Relational Database	A type of database wherein data is represented and stored as tables of rows and columns.
REST	REpresentational State Transfer	A protocol used to invoke Web services that is simpler to use than SOAP.
RPM	Revolutions per minute	A measure of the frequency of rotation.
SaaS	Software-as-a-service	A type of deployment for service in a cloud infrastructure wherein the user does not have much control over application deployment.
SAML	Security Assertion Markup Language	A language used to express the interactions between several systems during a single sign-on authentication.
SAN	Storage Area Network	A storage infrastructure containing multiple, physical disk drives, processors, network, and other components serving as a single storage device accessible through the network.
SAS	Serial Attached SCSI	A point-to-point communication protocol for moving data between storage devices like hard drives and tape drives.
SATA	Serial AT Attachment	A computer interface used to connect a host machine to mass storage devices.

Term/Acronym	Definition	Description
SCSI	Small Computer System Interface	A set of standards for connecting and transferring data between computers and peripheral devices, including storage devices.
SIF	Schools Interoperability Framework	A specification facilitating data sharing and reporting between applications in K through 12 instructional and administrative environments.
SIS	Student Information System	A state-level system that stores and maintains student data.
SLA	Service Level Agreement	A contract where the level of service is formally defined between a customer and a service provider.
SLIP	Serial Line Internet Protocol	A version of the Internet protocol designed to work over serial ports and modems.
SMTP	Simple Mail Transfer Protocol	Internet standard for transmission of electronic mail.
SNMP	Simple Network Management Protocol	A network management and monitoring protocol.
SOA	Service-oriented architecture	A technology for connecting disparate IT components and systems.
SOAP	Simple Object Access Protocol	An XML-based protocol used to invoke Web services.
SQL	Structured Query Language	The main language used to manipulate data in a relational database.
SSD	Solid-state Drive	A disk drive technology that eliminates the disadvantages of classical electro-mechanical drives.

Term/Acronym	Definition	Description
SSH	Secure Shell	A network protocol for secure data communications, remote services, and command execution between two networked computers.
SSL	Secure Sockets Layer	A cryptographic protocol providing communication security over the Internet. This protocol has been superseded by Transport Layer Security (TLS).
SSO	Single Sign-on	An authentication technology that enables a user to log into one system and automatically be recognized in other systems without the need to log in again.
TCP/IP	Transmission Control Protocol/Internet Protocol	The two most important low-level protocols of the Internet protocol suite. They specify how data should be formatted, addressed, transmitted, routed and received at the destination point.
TEI	Technology-enhanced Item	A new type of test item featuring interactive content enabled by certain underlying technology platforms (e.g., Adobe Flash, Microsoft Silverlight, Oracle Java Applets, and HTML5).
TLS	Transport Layer Security	The cryptographic protocol that replaced SSL.
UDDI	Universal Description, Discovery, and Integration	A standard registry used to discover services in an SOA Web services-based architecture.
UI	User Interface	The way a human interacts with a computer.
URL	Uniform Resource Locator	A character-string reference to an Internet resource.
VM	Virtual Machine	A simulation of a physical computer that runs as software inside another physical computer.

Term/Acronym	Definition	Description
WAN	Wide Area Network	A telecommunications network covering a broad area with links across regional or national boundaries.
WDSL	Web Service Definition Language	A language used to describe the functionality of a service.
WLAN	Wireless Local Area Network	A network linking two or more devices using s wireless distribution method.
WMI	Windows Management Instrumentation	The infrastructure for the management of data and operations in the Windows operating system.
WS-Security	Web Services Security	An extension to the SOAP protocol providing security to Web services published by the OASIS security group.
XML	Extensible Markup Language	A textual data format that is used to describe complex data structures using tags, elements, and attributes.
XMPP	Extensible Messaging and Presence Protocol	An XML-based open communications protocol for message-oriented middleware. This protocol was originally named Jabber and used for instant messaging.