

Architecture and System Design Document

Version 1.0

*<Project or Solution Name>*

**U.S. Department of Housing and Urban Development**

*<Month Year>*

# Document History

<Provide information on how the development and distribution of the Architecture and System Design document is controlled and tracked. Use the table below to provide the version number, date, author, and a brief description of the reason for creating the revised version.>

|  |  |  |  |
| --- | --- | --- | --- |
| **Version No.** | **Date** | **Author** | **Revision Description** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Document History ii](#_Toc450730285)

[1. Introduction 2](#_Toc450730286)

[1.1. System Overview 2](#_Toc450730287)

[2. Application Architecture 2](#_Toc450730288)

[2.1. Context View 2](#_Toc450730289)

[2.2. Logical Composition Views 3](#_Toc450730290)

[2.3. Physical Composition Views 3](#_Toc450730291)

[3. Design Rationale 4](#_Toc450730292)

[3.1. Design Assumptions and Decisions 4](#_Toc450730293)

[3.2. Design Constraints 4](#_Toc450730294)

[4. Application Engineering 5](#_Toc450730295)

[4.1. Structure Views 5](#_Toc450730296)

[4.2. Information Views 5](#_Toc450730297)

[4.3. Interaction Views 5](#_Toc450730298)

[4.4. Process Flows/Diagrams 5](#_Toc450730299)

[4.5. Workflow Diagrams 5](#_Toc450730300)

[4.6. Common Service Specifications 5](#_Toc450730301)

[5. System Architecture 5](#_Toc450730302)

[5.1. System Hardware 5](#_Toc450730303)

[5.2. System Hardware Architecture 6](#_Toc450730304)

[5.3. Software Architecture 6](#_Toc450730305)

[5.3.1. COTS Software Components 6](#_Toc450730306)

[5.3.2. Contractor Developed Software 6](#_Toc450730307)

[5.4. System Software Detailed Design 7](#_Toc450730308)

[5.5. Communications Architecture 7](#_Toc450730309)

[5.6. Internal Communications Detailed Design 7](#_Toc450730310)

[5.7. Network Detailed Design 8](#_Toc450730311)

[6. Data Architecture 8](#_Toc450730312)

[6.1. Data Flow Diagrams 9](#_Toc450730313)

[6.2. Logical Data Model 9](#_Toc450730314)

[6.3. Authoritative Data Sources 9](#_Toc450730315)

[6.4. Standard Data Elements 9](#_Toc450730316)

[6.5. XML/JSON Schemas 9](#_Toc450730317)

[6.6. Data Migration Approach 9](#_Toc450730318)

[6.7. Electronic Records Management 9](#_Toc450730319)

[6.8. Data Dictionary and File Structure 9](#_Toc450730320)

[6.8.1. Database Management System (DBMS) 10](#_Toc450730321)

[6.8.2. Non-Database Management System Files 10](#_Toc450730322)

[6.9. Database Detailed Design 10](#_Toc450730323)

[7. User Interfaces 11](#_Toc450730324)

[7.1. Inputs 11](#_Toc450730325)

[7.2. Outputs 11](#_Toc450730326)

[8. System Interfaces 11](#_Toc450730327)

[9. Security Architecture 12](#_Toc450730328)

[9.1. Security Architecture Goals and Constraints 12](#_Toc450730329)

[9.2. System Integrity Controls 12](#_Toc450730330)

[9.3. Access Management and Control 12](#_Toc450730331)

[10. Infrastructure 12](#_Toc450730332)

[10.1. Deployment Models 13](#_Toc450730333)

[11. Section 508 Compliance 13](#_Toc450730334)

[Appendix A: References 15](#_Toc450730335)

[Appendix B: Key Terms 15](#_Toc450730336)

[Appendix C: Interface Control Document 15](#_Toc450730337)

Exhibit List

[**Exhibit 1 Context View Diagram Elements** 3](#_Toc450730338)

[**Exhibit 2 Logical Composition View elements** 3](#_Toc450730339)

[**Exhibit 3 Physical Composition View Elements** 4](#_Toc450730340)

[**Exhibit 4 Design Decisions** 4](#_Toc450730341)

*Note: The information reported in this Architecture and System Design document should align with the architectural and design requirements and recommendations as documented in the Service Layered Architecture Profile (SLAP). The EA team designed the SLAP document specifically for this project during the Program Validation phase of the PPM lifecycle. The SLAP provides project-specific architectural guidance in alignment with HUD’s Enterprise Technical Architecture (ETA). The ETA establishes a consistent, vendor-agnostic, and standards-based architecture to be used department-wide in the development of custom-built, open source applications. The ETA prescribes architectural standards for application layers including:*

* *Presentation Layer – manages user interaction*
* *Business Logic Layer – manages business logic and rules*
* *Data Access Layer – manages access to the data (relational database, flat files, or repositories)*
* *Java Coding Standards – conventions and best practices for maintaining Java code*
* *Service Layer – guidance for Service Oriented Architecture (SOA) implementation*

*Project teams should reference and adopt the guidance provided in the SLAP when developing solutions. If the SLAP document prescribes architectural requirements or recommendations that hinder the solution development, the project team should contact the Enterprise Architecture team and initiate the waiver process to adjust architectural compliance as appropriate.*

# Introduction

This document is a concise, effective representation of the current design state for the *<System-of-Interest>* at a specific release cycle. It includes the definition and description of the operations, interfaces, performance, and quality assurance requirements of the software and databases being developed. Initially, this document serves as the method for recording the preliminary design plans. This is a living document during the design and development phases of the life cycle. Once the system is fully implemented, this documents should represent the system in production.

<To satisfy the content requirements for the various sections within this document, the project team is encouraged to include and provide links / references to models, diagrams, and other relevant artifacts. At a minimum, all source documents should be posted to at the TRC SharePoint site for this project. Both source and derived documents are requested to enable all project stakeholders to review the artifacts without the need for specific tool access.>

## System Overview

<This subsection should describe the business mission, goals, purpose and functionality of the system in non-technical terms. Include high-level context diagrams that show each subsystem and the interfaces to external systems. Provide an explanation on how the functional requirements will be addressed during development.>

# Application Architecture

The Application Architecture level provides an abstract representation of the solution design. It serves as both a launch pad and a roadmap for Application Engineering activities. The Application Architecture level comprises a Context view, Logical Composition views and Physical Composition views, as well as additional optional views when needed for clarity or completeness.

More specifically, the Application Architecture provides an abstract representation of the solution design, which can serve as both a launch pad and a roadmap for engineering activities. Multiple views should be considered to promote clarity and understanding.

## Context View

<Include a Context View diagram representing a high-level view of the System of Interest with supporting narrative text as appropriate. The Context View diagram defines the boundary between the system and its environment, showing the entities that interact with it. The Context view of a system defines the relationships, dependencies, and interactions between the system and its environment—the people, systems, and external entities with which it interacts. It defines what the system does and does not do; where the boundaries are between it and the outside world; and how the system interacts with other systems, organizations, and people across these boundaries. Ensure that the diagram(s) depict the major components of the solution and the relationships between the components, input and output data flows, major processes, functions, and system tasks.

Context diagrams include:

* Entities – labeled boxes; one representing the system, and multiple boxes representing external entities
* Relationships: labeled lines between the entities and system; and
* Other pictures, stick- figures, and shapes that are used to convey meaning.>

The Context View diagram should include the elements in Exhibit 1:

|  |  |
| --- | --- |
| Element | Description |
| <<boundary>> | A delineation of subsystems, modules, and components.  |
| <<data>> | Category of data flowing through an interface. |
| <<grouping>> | A bounded arrangement of closely related elements, such as all application tier processing nodes and all Web tier processing nodes. |
| <<subsystem>> | A collection of people, processes, and technologies that are organized to accomplish a specific set of functions within a System-of-Interest.  |
| <<user>> | Human actor in a human/machine interface. |

**Exhibit 1 Context View Diagram Elements**

<insert and provide links/references to the Context View and define and describe diagram elements >

## Logical Composition Views

The Logical Composition Views are concerned with the functionality that the system provides to end-users. This views includes the perspectives of the principal subsystems on the System-of-Interest. In a logical composition view, the relationships among subsystem components are represented by the interfaces that connect them.

The Logical Composition Views should include the elements in Exhibit 2:

|  |  |
| --- | --- |
| Element | Description |
| <<boundary>> | A delineation of subsystems, modules, and components. Two boundaries are required in a Logical Composition diagram: USCIS and Design Subject. |
| <<component>> | A unit of composition (source or binary), accessible via a well-defined interface, that provides runtime functionality within an execution environment. |
| <<grouping>> | A bounded arrangement of closely related elements, such as all application tier processing nodes and all Web tier processing nodes. |
| <<module>> | A bounded collection of components, accessible via a well-defined interface, which is organized to accomplish a specific function or set of functions within a subsystem. |
| <<protocol>> | A set of conventions governing the connection, communication, and transfer of data among interfaces with subsystems, modules, and components. |
| <<subsystem>> | A collection of people, processes, and technologies that are organized to accomplish a specific set of functions within a System-of-Interest.  |

**Exhibit 2 Logical Composition View elements**

 <insert and provide links/references to Logical Composition Views and define and describe diagram elements>

## Physical Composition Views

The Physical Composition Views provide a detailed model representing how components will be deployed across the system infrastructure. It details network capabilities, server specifications, hardware requirements and other information related to deploying the proposed system.

The purpose of the Physical Composition View is to provide a device-agnostic description of the system deployment requirements for the System-of-Interest. This view includes the resource specifications including the operating systems, CPU cores, memory allocations, storage space, and other resources required in any other deployment environment.

The Physical Composition Views should include the elements in Exhibit 3:

|  |  |
| --- | --- |
| Element | Description |
| <<artifact>> | A directory path specification to contain one or more units of composition that define or provide the runtime structure or behavior for a component or execution environment.  |
| <<component>> | A unit of composition (source or binary), accessible via a well-defined interface, that provides runtime functionality within an execution environment. |
| <<execution environment>> | Compiled or interpreted application-level software providing machine instructions to a processing node. One or more execution environments may reside on a processing node.  |
| <<grouping>> | A bounded arrangement of closely related elements, such as all application tier processing nodes and all Web tier processing nodes. |
| <<processing node>> | The provisioning of hardware and system-level software necessary to process machine instructions provided by the execution environment. A processing node is described in terms of its operating system, CPU architecture, number of cores, memory, and storage. Where a processing node represents the combined resources of two or more identical servers in the current deployment, values attributed to the individual node comprise the aggregate values of the like servers. |
| <<schema>> | Any database schema file. |

**Exhibit 3 Physical Composition View Elements**

< Insert and provide links/references to Physical Composition Views and define and describe diagram elements>

# Design Rationale

<This section contains assumptions, constraints, and considerations affecting the system design.>

## Design Assumptions and Decisions

<Using Exhibit 4, list and describe the impact of any significant design assumptions and critical decisions that the team has made throughout development. Where applicable, provide the history of decisions and identify how and when a decision was made in context of governance. Focus on decisions that are not obvious which had other reasonable alternatives. The goal is to avoid spending time in the future rehashing a previously made decision.>

|  |  |
| --- | --- |
| **Design Decision** | **Rationale** |
| <add rows as necessary> |  |
|  |  |

**Exhibit 4 Design Decisions**

##  Design Constraints

<This subsection describes any constraints that will influence the system design. Reference any trade-off analyses conducted such as possible conflicts with other systems. Also note the assumptions made in developing the system design to overcome these constraints.

Examples of architectural constraints could include: safety, security, or privacy restrictions, required use of existing systems, products, legacy components, or infrastructure, as well as the need for portability, distribution, and reuse.

Examples of project constraints could include: limitations on design and implementation strategies, lack of development tools, understaffed team, or a compressed schedule.>

# Application Engineering

This subsection provides an implementation-level representation of the solution design, which is traceable to the application architecture. This section includes Structure views, Information views, and Interaction views, as well as additional optional views when needed for clarity or completeness, but at a minimum Process Flows / Diagrams are required.

## Structure Views

<A Structure View depicts static structures and transient data aspects of the principal subsystem. Where appropriate, refer to the code package structure.>

## Information Views

<An Information View depicts persistent data aspects of the principal subsystem.>

## Interaction Views

<An Interaction View depicts human-machine and machine-machine process flows between systems/subsystems.>

## Process Flows/Diagrams

<Process Flows illustrate human-human, human-machine and machine-machine interactions required to deliver the capability.>

## Workflow Diagrams

<Describe the main modes of communication between processes, such as message passing, human intervention, etc. Workflow diagrams should be included to clarify complex interactions. >

## Common Service Specifications

<Provide service specifications to describe and define how all common services are integrated into the solution. This section is only required for solutions integrating common services as identified by the solution architecture common service attributes.>

# System Architecture

This subsection section provides an overview of the hardware, software, communication, and network architectures and detailed design for the *<System-of-Interest>.*

## System Hardware

<Describe the overall system hardware and organization. Include a list of hardware components with a brief description of each item and diagrams showing the connectivity between the components. If appropriate, use subsections to address each subsystem.>

## System Hardware Architecture

<A hardware component is the lowest level of design granularity in the system. Depending on the design requirements, there may be one or more components per system. Provide enough detailed information about individual component requirements to correctly build and/or procure all the hardware for the system (or integrate COTS items).

If there are many components or if the component documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each component and its functions, adequately. Follow industry standard component specification practices. For COTS procurements, if a specific vendor has been identified, include appropriate item names. Include the following information in the detailed component designs (as applicable):

* Power input requirements for each component
* Signal impedances and logic states
* Connector specifications (serial/parallel, 11-pin, male/female, etc.)
* Memory and/or storage space requirements
* Processor requirements (speed and functionality)
* Graphical representation depicting the number of hardware items (for example, monitors, printers, servers, I/O devices), and the relative positioning of the components to each other
* Cable type(s) and length(s)
* User interfaces (buttons, toggle switches, etc.)
* Hard drive/floppy drive/CD-ROM requirements
* Monitor resolution>

## Software Architecture

<Describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), database platform, computer languages, and programming computer-aided software engineering tools, commercial off-shelf (COTS) software, open source frameworks (with a brief description of the function of each item and identifying information such as manufacturer, version number, number and types of licenses needed as appropriate).

### COTS Software Components

<Information on COTS software components is provided in Section X.X>

### Contractor Developed Software

<This section describes the contractor-developed software including individual components, applications, database stored procedures, and interactive Web server pages, which may be distributed across various hardware servers and virtual server instances.

List the contractor-developed components and the design/development method. Include interfaces, web components, stored procedures, communication software, reports, data management and data layer components, etc.>

## System Software Detailed Design

<A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. Provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Follow industry standard module specification practices. Include the following information in the detailed module designs:

* A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc. Explain any algorithms used by the module in detail.
* For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (e.g. dynamic link libraries)
* Data elements, record structures, and file structures associated with module input and output and a CRUD matrix of modules to data entities. Create, Read, Update and Delete (CRUD) Matrix Business Process to Data Entities.
* Graphical representation of the module processing, logic, flow of control, and algorithms, using an industry standard notation and diagramming approach (e.g., structure charts action diagrams, flowcharts).
* Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section.
* Report layout (e.g. type of report and report design >

## Communications Architecture

<Describe the overall communications within the system (e.g., how each layer communicates via protocols (http, https, REST, etc.)). Include the communications architecture(s) being implemented and how the system components are linked (e.g., communication between the current application and external systems). Provide a diagram depicting the communication flow between the system and subsystem components. If appropriate, use subsections to address each architecture being employed. Insert any related communication architecture documents or provide a reference to where they are stored.>

## Internal Communications Detailed Design

<If the system includes more than one component there may be a requirement for internal communications to exchange information, provide commands, or support input/output functions. Provide enough detailed information about the communication requirements to correctly build and/or procure the communications components for the system. Include the following information in the detailed designs (as appropriate):

* The number of servers and clients to be included on each area network
* Specifications for bus timing requirements and bus control
* Format(s) for data being exchanged between components
* Graphical representation of the connectivity between components, showing the direction of data flow (if applicable), and approximate distances between components; information should provide enough detail to support the procurement of hardware to complete the installation at a given location
* Local Area Network (LAN) topology>

## Network Detailed Design

<This subsection provides detailed information about the requirements to correctly build and/or procure the network components for the system. If the system includes more than one network there may be a requirement for to exchange information, provide commands, or support input/output functions. Include the following information in the detailed designs (as appropriate):

* The number of servers and clients to be accessed by each network
* Specifications for bus timing requirements and bus control
* Format(s) for data being exchanged between components
* Diagrams showing connectivity, the direction of data flow (if applicable), and locations of network interfaces. The information should provide enough detail to support the procurement of hardware to complete the installation at a given location
* LAN/WAN topology>

# Data Architecture

<All projects that are updating or designing a new data system must follow all Federal Government and HUD data requirements and standards. HUD has developed the Common Application Relational Schema (CARS) enterprise logical data model. CARS is a high quality, documented data model that provides a common definition and structure of HUD’s core data requirements. Alignment to the CARS model and its methodology and design approach will ensure data quality, adhere to consistent business rules, and institutionalize HUD data standards. In conjunction with the EA Data Architecture team, all new development projects are required to use the CARS model to: Establish project data model baseline

1. Apply data modeling standards to incorporate additional business data model requirements.
2. Evolve CARS with project-specific business data.

The Project Lead or Lead Data Architect for the project contacts the OCIO Enterprise Architecture team to review their data requirements and high level data architecture design. The Enterprise Data Architecture team will work with the project team to provide guidance in utilizing the CARS model for new development. This includes working together to build out the project’s data model as the project evolves.

HUD’s Enterprise Data Warehouse (EDW) will become the unified source that stores all business information for the agency. All data in the EDW will be made available for analyzing planning, and reporting purposes, providing reliable and consistent data agency-wide. The EDW grows with HUD’s business needs through the expansion of data definitions and addition of classifications as they emerge in the business model. The EDW should be used by all project teams to meet data warehousing needs. The Enterprise Data Warehouse team will implement any reporting and data warehousing requirements>

<This section should include and/or reference data models and diagrams to promote the understanding of the data architecture. The Logical Data Model is required.>

## Data Flow Diagrams

<Provide data flow diagrams showing data flows between a generalized application within the domain and the other entities and abstractions with which it communicates. A data flow diagram is a graphical representation of the "flow" of data both internal to the system and with external systems. >

## Logical Data Model

< HUD has adopted the CARS model as its enterprise standard logical data model. Project teams must leverage this model for all new development projects. The Enterprise Architecture team will work with the project team to create a baseline project data model from the CARS data model.

Provide a link or reference to the logical data model for the System-of-Interest and make it available to the Enterprise Architecture team for review. Describe the logical grouping of the basic data building blocks of the solution and the major processes and data requirements of the business.>

## Authoritative Data Sources

<Identify the authoritative data sources required for access during this project. This enables rapid, trusted transactions in HUD core business functions.>

## Standard Data Elements

<Identify standard data element (SDE) usage and/or propose new SDEs. This promotes a consistent data standard that makes information understandable and reusable enterprise wide.>

## XML/JSON Schemas

<Provide link or reference to schema which should be developed in accordance with the Enterprise Technical Architecture guidance.>

## Data Migration Approach

<Indicate data migration sequencing requirements in relationship to the transition from current baseline architecture to the solution’s target architecture.

This section provides architectural approach for any significant data migration effort required by the solution. The Data Conversion Plan documents specifics for this area.>

## Electronic Records Management

<Provide data architecture attributes necessary to meet solution requirements for electronic records management and related requirements specified by the E-Government Act of 2002 & 2007. Explain the approach in place to address archiving, record retention, the impacted data elements, etc.>

##  Data Dictionary and File Structure

<This section describes the overall file structure of the system, including both database files and non-database files. Additional information may be added as required for the particular project. The goal of this section is to produce a comprehensive data dictionary that shows the data element by name, type, length, source, validation rules, maintenance (create, read, update, delete (CRUD) capability), data stores, outputs, aliases, and description. A link/reference to the data dictionary can either be entered here or it can be included as an appendix. Consult with the Database Administrator (DBA) when preparing this section.>

### Database Management System (DBMS)

<Provide the final design of the DBMS files and include the following information, as appropriate (refer to the data dictionary):

* Refined logical model; provide normalized table layouts, entity relationship diagrams, and other logical design information (reference or update information from the Solution Architecture document)
* Separate from the logical model, provide a physical description of the DBMS schemas, sub-schemas, records, sets, tables, storage page sizes, etc.
* Access methods (e.g. ,indexed, via set, sequential, random access, sorted pointer array)
* Estimate of the DBMS file size or volume of data within the file, and data pages, including overhead resulting from access methods and free space

Definition of the update frequency of the database tables, views, files, areas, records, sets, and data pages; estimate the number of transactions if the database is an online transaction-based system>

### Non-Database Management System Files

<Provide the detailed description of all non-DBMS files and include a narrative description of the usage of each file—including if the file is used for input, output, or both; if this file is a temporary file; an indication of which modules read and write the file; and file structures (refer to the data dictionary). As appropriate, the file structure information should:

* Identify record structures, record keys or indexes, and reference data elements within the records
* Define record length (fixed or maximum variable length) and blocking factors
* Define file access method, e.g. index sequential, virtual sequential, random access
* Estimate the file size or volume of data within the file, including overhead resulting from file access methods

Define the update frequency of the file; if the file is part of an online transaction-based system, provide the estimated number of transactions per unit time, and the statistical mean, mode, and distribution of those transactions>

## Database Detailed Design

<This subsection provides detailed information about the logical, functional, and data structures needed to create the system’s database.

* Describe each data field to be developed and its characteristics (size, type, etc.)
* Describe each table to be developed and the fields each table will contain. State any specific restrictions on use of each table, if applicable.
* Describe the relationships among the tables
* Identify key fields, foreign keys, etc., in sufficient detail to enable a final logical design to be developed.
* Include Entity-Relationship diagrams or other depictions to clarify explanations.>

# User Interfaces

<In the subsections below, describe the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Depending on the particular nature of the project, it may be appropriate to repeat these sections at both the subsystem and design module levels. Provide additional information in the subsections if the suggested lists are inadequate to describe the system’s inputs and outputs.>

## Inputs

<Describe the input media used by the operator for providing information to the system (e.g. data entry screens, optical character readers, bar scanners). If appropriate, reference the input record types, file structures, and database structures provided in Section 6 - Data Design. Include reference to the data dictionary.

Provide the layout of all input data screens or graphical user interfaces (GUIs) (e.g., Microsoft Windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary.

This section should reference the HUD Standard Data Dictionary for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also address data entry controls to prevent edit bypassing.

Discuss the miscellaneous messages associated with operator inputs, including the following:

* Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
* Description of any access restrictions or security considerations
* Each transaction name, code, and definition, if the system is a transaction-based processing system>

## Outputs

<Describe the system output design relative to the user/operator; show a mapping to the high level data flows described in Section 2.2. - Software Detailed Design. System outputs include reports, data display screens and GUIs, query results, etc. The following should be provided, if appropriate:

* Identification of codes and names for reports and data display screens
* Description of report and screen contents (provide a graphic representation of each layout and reference the data dictionary)
* Description of the purpose of the output, including identification of the primary users
* Report distribution requirements, if any (include frequency for periodic reports)
* Description of any access restrictions or security consideration>

# System Interfaces

<See Appendix C - Interface Control Document>

# Security Architecture

<Insert any related security documents, including PPM System Security Plan, integrity controls, or provide a reference to where they are stored. Provide a high level overview of the security architecture. Identify and describe how the security architecture meets the solution’s security requirements.>

## Security Architecture Goals and Constraints

<Identify and describe the significant and central security goals and constraints of the solution’s security architecture.

Goals: Security Architecture goals are to provide integrity, confidentiality, availability, non-repudiation, and privacy, transparently to the user, and seamlessly to all business conducted via the HUD network.

Constraints: Security architecture constraints include all Federal mandated security requirements from the Office of Management and Budget (OMB) and the National Institute of Standards and Technology (NIST) necessary for Federal agencies to comply with Title III of the E-Government Act of 2002 (Public Law 107-347 December 2002) entitled the Federal Information Security Management Act (FISMA). FISMA requires each federal agency to develop, document, and implement an agency-wide information security program.

For additional information, please refer to the “Comprehensive Access Control Model”.>

## System Integrity Controls

<Sensitive systems use information for which the loss, misuse, modification of, or unauthorized access to that information could affect the conduct of Federal and HUD programs, or the privacy to which individuals are entitled.

If this is a sensitive system, provide specifications for the following minimum levels of control:

* Internal security to restrict access of critical data items to only those access types required by users
* Audit procedures to meet control, reporting, and retention period requirements for operational and management reports
* Application audit trails to dynamically audit retrieval access to designated critical data
* Standard tables to be used or requested for validating data fields
* Verification processes for additions, deletions, or updates of critical data
* Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed>

## Access Management and Control

*<Describe controls to provide access to system capabilities, application capabilities, and data, based upon User IDs, passwords, and user roles.>*

# Infrastructure

<Map application architecture deployment models to hardware and software infrastructure specifications including memory and central processing unit specifications required to meet volume and performance requirements. Include in this section infrastructure architecture guidance and specifications for:

1. Software < The programs, routines, and symbolic languages that control the functioning of the hardware and direct its operation.>
2. Hardware< A computer and the associated physical equipment directly involved in the performance of data-processing or communications functions. Machines and other physical equipment directly involved in performing an industrial, technological, or military function>.
3. Network < network is a group of two or more computer systems linked together.>
4. Middleware < Software that serves as an intermediary between systems software and an application.>

Provide infrastructure architecture guidance and specifications for all environments required for developing, testing, deploying, and operating the solution.>

## Deployment Models

<Describe how the application architecture is deployed into one or more physical network (hardware) configurations.

Replace the example web application middleware deployment and Oracle RAC deployment diagrams below with one or more diagrams to illustrate significant and central components of the infrastructure architecture, these diagrams should be easily cross-referenced with the infrastructure architecture overview diagram.

Provide diagrams as required to develop the level of detail required to provide unambiguous high-level architectural specifications and guidance to architects and designers developing the solution detailed design.

Annotate all diagrams with supporting narrative to define all objects and relationships depicted by the diagrams.>

# Section 508 Compliance

<Discuss how the system and hardware architecture design will address and integrate accessibility features that align with best practices for Section 508 compliance. Section 508 compliance Best Practices information is available on the HUD web site as follows:

The HUD Section 508 Policy website <http://hudatwork.hud.gov/po/i/508/overview.cfm> provides a link to the HUD level policy. The Section 508 Coordinators, Officials, and Other Public Contacts web site <http://hudatwork.hud.gov/po/i/508/roles.cfm> lists functional area coordinators and other officials who are directly responsible for Section 508 activities across HUD. The HUD Section 508 Voluntary Template Product Accessibility Template (VPAT) web site <http://hudatwork.hud.gov/po/i/508/forms.cfm> provides information on the requirements for vendors involved in Section 508-related products and services and provides information on completing the product assessment templates and links in consideration of Section 508 of the Rehabilitation Act of 1973. The HUD Tools web site <http://hudatwork.hud.gov/po/i/it/sd/guide/section508.cfm> includes contact information that identifies the functional requirements of a project including the intended users and how the application will be accessed. The GSA-sponsored Section 508 web site (<http://section508.gov>) provides information at the Federal level on Section 508 laws and applicable EIT standards developed by the U.S. Access Board.

For HUD IT projects, the following Section 508 compliance practice activities are appropriate:

Ensure projects include high level compliance requirements in the design and use of the IT solution and how provisions for Section 508 compliance will be made in:

1. **Testing** – Develop and execute a testing plan utilizing available tools.
2. **Implementation** – Implement any required changes.
3. **Review** – Re-evaluate compliance whenever changes or updates are made.>

# Appendix A: References

<Insert the name, version number, description, and physical location of any documents referenced in this document. Add rows to the table as necessary.>

The following table summarizes the documents referenced in this document.

|  |  |  |
| --- | --- | --- |
| <Document Name and Version Number> | <Document description> | <URL to where document is located> |
|  |  |  |
|  |  |  |

# Appendix B: Key Terms

The table below provides definitions and explanations for terms and acronyms relevant to the content presented within this document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| [Insert Term] | <Provide definition of term and acronyms used in this document> |
| Architectural attributes | Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements". Qualities, that are non-functional requirements, can be divided into two main categories:Execution qualities, such as security and usability, which are observable at run time.Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system. |
| Common service attributes | A service is defined by a set of attributes. Some attributes are common to all service instances created from one service definition, and are therefore set during service definition time. Other attributes are specific to a service instance and must be set in that instance. Some attributes can be set either in the service definition or in the instance; in such cases it is up to the service designer to determine when the attribute will be set. |

# Appendix C: Interface Control Document

|  |  |
| --- | --- |
| <Document Name and Version Number> | <URL to where Interface Control Document for the System of Interest is located> |
| PPM Template | <http://portal.hud.gov/hudportal/HUD?src=/program_offices/cio/ppm/PPMV20Artifacts>  |
|  |  |