<OIDV Logo>

PRACTICES GUIDE

INTERFACE CONTROL

Issue Date: Error! Unknown document property name.
Revision Date: Error! Unknown document property name.

Purpose

This Practices Guide is provides an overview describing the best practices, activities, attributes, and related templates, tools, information, and key terminology of industry-leading project management practices and their accompanying project management templates. The purpose of this document is to provide guidance on the Interface Control and to describe the practice overview, requirements, best practices, activities, and key terms related to these requirements.

Background

The Department of Health and Human Services (HHS) Enterprise Performance Life Cycle (EPLC) is a framework to enhance Information Technology (IT) governance through rigorous application of sound investment and project management principles and industry's best practices. The EPLC provides the context for the governance process and describes interdependencies between its project management, investment management, and capital planning components. The EPLC framework establishes an environment in which HHS IT investments and projects consistently achieve successful outcomes that align with Department and Operating Division goals and objectives.

The interface control document is one of the components of the Design Document. It is used to describe the inputs and outputs of a single system, the interface between two systems or the interface protocol between physical components.

Practice Overview

An interface, from the perspective of system development, can be identified as any point where a system and something, or someone, meet. This interface point may include other systems, internal hardware, circuitry, external peripherals, networks, system users, etc. For example, common interfaces for computer peripherals may include USB, serial, parallel ports, etc; common interfaces for system users may include monitors, keyboards, mice, etc.

Interfaces are documented using interface control documents (ICD) that describe the system's interfaces including any rules for communicating with them. ICDs help ensure compatibility between system segments and components and are often considered key elements of effective system design and development. The purpose of the ICD is to clearly communicate all possible inputs and outputs from a system for all potential actions whether they are internal to the system or transparent to system users. An ICD may describe the system interfaces to the lowest physical elements (circuits, voltage, watts, etc) to the user interface or any subset thereof. The level of detail included in any ICD is dependant upon the requirements of the stakeholders to successfully deliver on project requirements.

ICDs define for the development team system inputs, outputs, internal interfaces, and interfaces between other systems, subsystems and/or users. An ICD should only describe the interface itself and not any characteristics of the systems using it. An ICD should also not include anything about the meaning or intended use of the data. Any features, functions, or logic supporting the system interface should be outlined within separate design and/or specification documents. Defining an ICD in this way allows other teams to develop connecting systems without concern of how the data is treated by the other system. This allows development teams to work without the requirement of knowing the business logic or technical aspects behind the system and allows for modularity that leads to easy maintenance and extensibility of systems.

Depending on the nature of the project it is possible that a number of different ICDs will be required, one for each different interface type. For example, network interface control document, graphical user interface control document, application program interface document, etc. Regardless of the interface type, format, or name, an effective ICD would be used in conjunction with relevant system design and specifications documents, models, and drawings, to communicate how the overall system functions and interacts. An effective ICD may include:

- Description of interface and interface types. For system development, this may include information such as data type, size, format, measures, etc
- Description of commands and parameters that pass between systems
- Description of how systems communicate with subsystems
- Description of how systems communicate with external systems
- Description of how systems communicate with its users
- Description of how systems interprets external commands and what responses/behaviors are expected

The Federal Enterprise Architecture (FEA) Reference Models contain components that reference many of the items that should be considered when defining ICDs. Some of these items include:

- Interfaces between systems and their users
- Levels of access required by the application and other systems
- End-to-end management of communication sessions between systems and system components
- Hardware, operating system, other software, network protocols, etc that provide or request information
- Data storage, modification, and extraction of information from databases and any required interfaces
- Physical devices that provide the computing and networking within and between systems
- Security methods for protecting information and systems and how such policies/mandates may affect system interfaces
- Connections between the system, users, other systems, and peripherals
- Method for enforcing business rules
- Structure, elements, objects, and properties of data to be exchanged
- Methods in which data is transferred and represented in and between systems and how data is discovered and shared
- Communicating, transporting, and exchanging information through common dialogs or methods

An ICD also allows project teams to leverage other systems by providing developers with the information necessary to build add-on or complementary applications or to simply interface with existing systems. Some examples of interfaces where an ICD may be required include:

Examples based on the Open Systems Interconnection (OSI) 7-layer model		
OSI Layer	**Description**	**Interface Example**
Application Layer	The application layer interfaces directly to and performs common application services for the application processes; it also issues requests to the presentation layer.	Database Schema, Data Dictionary, Health Level 7 (HL7), Session Initiation Protocol (SIP)
Presentation Layer	The Presentation layer establishes a context between application layer entities, in which the higher-layer entities can use different syntax and semantics, as long as the Presentation Service understands both and the mapping between them.	American Standard Code for Information Interchange (ASCII), Moving Picture Experts Group (MPEG)
Session Layer	The Session layer controls the sessions between computers. It establishes, manages and terminates the connections between the local and remote application.	NetBIOS, Point-to-point tunneling protocol (PPTP), Remote procedure call (RPC)
Transport Layer	The Transport layer provides transparent transfer of data between end users, providing reliable data transfer services to the upper layers. The transport layer controls the reliability of a given link through flow control, segmentation, and error control.	Transmission Control Protocol (TCP), Small Computer System Interface (SCSI), User Datagram Protocol (UDP)
Network Layer	The Network layer provides the functional and procedural means of transferring variable length	Internet Protocol (IP),

Examples based on the Open Systems Interconnection (OSI) 7-layer model		
OSI Layer	**Description**	**Interface Example**
	data sequences from a source to a destination via one or more networks while maintaining the quality of service requested by the Transport layer. The Network layer performs network routing functions, and might also perform fragmentation and reassembly, and report delivery errors.	
Data Link Layer	The Data Link layer provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the Physical layer.	Point-to-Point Protocol (PPP), IEEE 802.11x
Physical Layer	The Physical layer defines all the electrical and physical specifications for the device. In particular, it defines the relationship between the device and a physical medium. This includes the layout of pins, voltages, cable specifications, Hubs, repeaters, network adapters, Host Bus Adapters (HBAs used in Storage Area Networks) and more.	10-Base-T, DSL, V.34

Best Practices

The following best practices are recommended for **Interface Control**:

- **Active Stakeholder Participation -** Access is needed for users that have the authority and ability to provide information regarding the system and interfaces being built.
- **Review and Approve -** Review the interface and design documents for precision, completeness, and usability.
- **Standardize -** Whenever possible a project team should consider using an industry recognized standard for system interfaces.
- **Clearly Documented** - Interface details should be clearly documented in the form of an interface control document, and supporting system specifications should be documented in the form of a system design document.
- **Establish Agreements** – Memorandum of Understanding or System Interface Agreements should be in established to document the interface expectations.
- **Rationale** - Document rationale behind design decisions and tradeoffs.
- **Other Interfaces -** Systems and devices have varying interfaces. Understand any required system interfaces prior to design. This may include hardware, software, network, devices, etc.
- **Verify -** Verify all models to ensure they fulfill the functional requirements.
- **Tradeoffs -** In most instances there are a number of ways to accomplish the same action. Be aware of the tradeoffs of selecting one approach as opposed to another and how the effects of that decision may propagate throughout the system.

Practice Activities

For software development projects the following practice activities are appropriate:

- **Guidelines -** Define and document modeling guidelines such as recording of assumptions, constraints, issues, approaches, documentation, templates, and notations.
- **Business Process Specification -** Define the intended future state of business processes.
- **Functional Specification -** Define the functional part(s) of the system being developed, external relationships, interaction with the user and other system elements.
- **Logical Specification -** Define the internal logic of the system explaining how it operates.
- **Conceptual Data Model -** Models the underlying business irrelevant of technology.
- **Logical Data Model -** Translates the conceptual model into a structure that can be implemented, by describing the data in as much detail as possible, without regard to how it may be physically implemented.
- **Physical Data Model -** Specifies how to physically implement the logical design by specifying items such as tables, columns, formal relationships between data and tables, etc.

- **Information Exchanges** – Model's the OPDIV's high-level information exchanges between IT systems.
- **Evaluation** - Validates the effectiveness of the models that were developed. This is usually accomplished via user feedback.
- **Verification** - Verify each step within each model to ensure traceability to requirements that delivers the expected system functionality.
- **Agreement** – Document and agree on the interface expectations.