Issue Date: <mm/dd/yyyy>
Revision Date: <mm/dd/yyyy>

Document Purpose

This Practices Guide is a brief document that 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 related to the Release Strategy.

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 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.

Practice Overview

System development decisions often have deep strategic, business, and cost implications. If not planned correctly, ramifications of incorrect decisions may be felt long after the decisions have been made. On the periphery of system development one very important aspect addressing this issue may often be under emphasized or overlooked all together. This is the implementation, execution, and management of an effective software release strategy. This becomes exponentially important when building a custom application because of the many challenges facing development teams:

- Managing development activities
- Managing integration of new development into the existing application
- Managing application dependencies
- Managing new feature/function requests
- Managing compliance with departmental/federal regulations, mandates, and processes
- Managing security requirements (vulnerability assessment testing [VA])

To illustrate how complex a development environment can be, and how important an effective release strategy is, consider the following example. A development team is building a custom application and might be required to release a major new feature or function multiple times a year to meet the demand of its clients. This environment is further complicated by the management of defects, issues, risks, change requests, etc. resulting in the distribution of multiple minor patch or emergency releases. The team uses an iterative approach to development and performs a test build at least once a day. Additionally, the application may have dependencies and interdependencies on other applications. Managing software in this type of environment is often done using a source control management software application, for example (Microsoft's Visual Source Safe [Visual Studio], IBM's Rational Clear Case, Borland's StarTeam, CVS, etc).

Deciding to release an application is often a tradeoff between early release and deferred release. Each alternative has its own sets of pros and cons that must be weighed against each other to determine maximum value for stakeholders. For example, rushing delivery benefits stakeholders with earlier release. However, this may require reducing functionality and may decrease overall quality. As a result, future costs may rise in order to fix bugs and distribute patches. Deferring a release allows time for enhanced functionality and improved quality. However, this approach may incur additional development cost and may result in missed opportunities.

Release Strategy

It is sometimes difficult to accurately determine the most appropriate product release date, feature functionality, associated development costs, quality concerns, etc are all challenges needing to be considered. Proper development and implementation of a release strategy may alleviate some of these and other challenges related to scope management, quality management, communications, risk management, etc. A formal release strategy makes distributing software easier and more consistent for the performing organization and also outlines how and when product will be made available to the client. A release strategy may include information on topics such as:

Producing the software - Activities that outline how the product will be designed, developed, and built, defining items such as

- Development approach (Iterative, waterfall, spiral, etc)
- Functionality defined for each planned release
- Operating systems supported
- Programming languages used
- Requirements around application hosting, security, etc

Testing the software - Activities that outline how the product will be tested, defining items such as:

- Quality measurements used (bug counts, performance, user feedback, etc)
- Quality requirements and acceptable tolerances
- Testing approach (automated, beta, sampling, etc)

Documentation - Activities that outline how the product will be documented, defining items such as:

- Product guides
- Release notes
- Training material

Packaging, distributing, and installing the software - Activities that outline how the product will be packaged and distributed, defining items such as:

- Box package distribution approach
- Electronic distribution approaches
- CD distribution/copying authorities and guidelines

Migrating data

Hosting the software

- Requirements for internally facing systems regarding hardware, software, security, etc.
- Requirements for externally facing systems regarding hardware, software, security, etc.

Providing training to end-users

Release Management

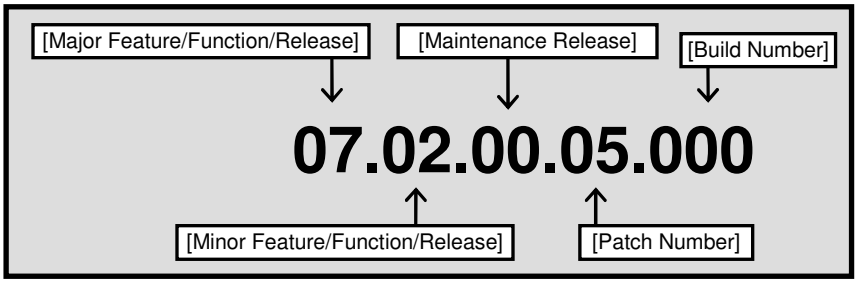
Release management approaches may vary from organization to organization. However, regardless of which approach is used a best practice approach to release management is comprised of activities that effectively manage the planning, organization, development, testing, and implementation of new features and functions, defects, change requests, etc. into the application being developed. Some of the stages that a software release may go through as it works through the release process may include:

- *Pre-Alpha* - A pre-alpha release does not necessarily contain completed feature/function. This is often an interim product build, prior to testing, often to validate a particular piece of work or that development to this point hasn't broken the build process.
- *Alpha* - An alpha release does not necessarily contain all completed features/functions but does satisfy release requirements. An alpha release is often the first internal product build delivered to the testing group. It is often a preliminary build that is only partially complete and typically contains temporary code, comments, product breaks, etc.
- *Beta* - A beta release is the first product version released outside the performing organization for the purposes of real-world testing. A beta release includes all features/functions but often still contains known issues and bugs.
- *Release Candidate* - A release candidate contains all completed features/functions and is a product version that has the potential to be a final product. A release is called code complete when it is agreed that no additional "new" source code will be added to the release. However, there may still be changes to fix defects.
- *General Availability (Gold)* - A general availability release is the final version of a particular product. A gold release is stable and relatively bug-free with a quality exceeding the client's expectations.

Release Numbering

Software release numbering may appear trivial but is critical to the overall success of any effective release strategy. A typical example of a numbering and naming releases scheme is shown below. This

type of a software release management naming convention is flexible enough to handle most software delivery situations and can be modified if needed to apply to almost any project.



Release decisions are ultimately affected by how much testing is needed to verify that both functional and non-functional requirements have been correctly built into the system and quality, as defined by project stakeholders, has been met. How much testing is needed depends on the level of difficulty to verify requirements and quality (testing/verification). The optimal level is difficult, if not impossible, to find. In practice, cost and time will constrain this activity to a level that the performing organization can tolerate.

Release Checklist

A release checklist is one approach that can help identify when a product is ready for release that can be used to help the project team identify when the product is ready for release to the client. This type of checklist also enables the project team to validate client requirements and expectations and can be used as a communication vehicle to validate this for the client as well. The process for creating a release checklist may include items such as:

- *Product Defects* – Quantify an amount of defects that the client finds acceptable. Obviously, the optimal level of defects would be zero. However, in practice, achieving zero defects would be extremely cost and time prohibitive. Constrained by cost and time, a reasonable level of quality should be identified and agreed upon by stakeholders.
- *Coding Errors* – If errors in code are discovered resolve them before continuing development. The cost of correcting errors increases exponentially as the project matures. For example, to correct a requirement error in the operation stage could cost a multiple of 100-times or more than if that same error was fixed earlier in the project's life.
- *Product Documentation* – Documentation should be a reflection of the code. Clarity, completeness, and consistency are better achieved if the individuals who developed the product also create the documentation.

Release Roles

Some of the individuals involved in a typical release process may include:

- *Software Architects* are responsible for capturing and understand the physical execution environment of the system and related issues.
- *Designers* are responsible for understand the distribution of processing and data in the system.
- *System Managers* are responsible for understanding the physical environment in which the system executes.
- *Project Manager* is responsible for estimating costs and schedules and for monitoring and controlling project activities.
- *Configuration Manager* is responsible for the assembly of product builds and releases and for maintaining the organization of development units, history, and access to product files.
- *Technical Control Board (TCB) or Change Control Board (CCB)* is responsible to evaluate the impact of the release on the operational environment.

The ultimate decision on determining if a product release is ready for distribution should be made based on an objective analysis of factors such as:

- *Completeness* – Completion of milestones/deliverables that deliver functionality required by the client.
- *Performance* – Product performance and server load is within acceptable margins defined in the requirements gathering and design phases of the project.
- *Defects* – Defects have been reduced to a level acceptable by the client.
- *Security* – Compliance with Certification and Accreditation (C&A) requirements.

- *Sign-off* – Final sign-off by the appropriate stakeholders and/or authorizing individuals to confirm that all product expectations have been met and that the product is officially approved for general release. A sign-off checklist may include items such as:
  - Annotate the correct release number as defined by an agreed upon standard
  - Confirm legal, license, and copyright elements
  - Remove any testing and debugging code
  - Check that documentation is complete and up to date
  - Ensure compliance with regulations and mandates.

Practice Best Practices

- **Release Numbering** - Use a standard for numbering and naming releases that is flexible enough to handle the organizations delivery situations
- **Checklist** - Use of a release checklist can be used to help the project team identify when the product is ready for release to the client
- **Validate** - Validate client requirements/expectations are built into the release. This is an activity that should be performed throughout the entire project life cycle to avoid errors and rework
- **Documentation** - Documentation should be a reflection of the code.
- **Software** - Release management can become extremely complex almost always requiring a source control management system application to control effectively.
- **Assessment** - Allow adequate time throughout the release cycle/schedule to perform any necessary assessments and validations for the product to be certified for release.
- **Compliance** - Allow adequate time throughout the release cycle/schedule to perform any actions required to comply with Federal and departmental regulations and mandates.
- **Authority to Operate** - After completing any assessments and compliance related items obtain formal sign-off validating authorization to operate (ATO).

Practice Activities

- **Stakeholders** - Identify and communicate with application stakeholder.
- **Plan Early** - Plan enough time to comply with processes and regulations.
- **Guidelines** - Outline how and when the product will be available to the client. Include information that defines development and testing processes, documentation, packaging, distribution, installation, data migration, training, etc.
- **Configuration Management** - Define the organizations configuration management approach.
- **Strategy** - Define a release strategy for the organization which may include definitions for pre-alpha, alpha, beta, candidate, and gold releases.
- **Numbering** - Define a release numbering approach that meets the needs of the organization and product being developed.
- **Source Control System** - Product source code is often controlled using a source control management system to assist in managing different version of product code (Microsoft's Visual Source Safe [Visual Studio], IBM's Rational Clear Case, Borland's StarTeam, CVS, etc).
- **Checklist** - Outline a release checklist to help identify when a product is ready for release.