AASHTO

AASHTOWare™

SOLUTIONS

# Standards and Guidelines Notebook

**April 1, 2026**

This page is intentionally blank.

AASHTO AASHTOWare™ SOLUTIONS

**To:** Members of the Special Committee on AASHTOWare and Product Task Force Chairpersons

**From:** Technical & Application Architecture Task Force

**Subject:** AASHTOWare Standards & Guidelines Notebook

Enclosed for your use is the AASHTOWare Standards & Guidelines Notebook. This Notebook contains all currently approved standards and guidelines that are effective April 1, 2026. Please refer to the notebook's *Overview* section for explanations of its content, organization, scope, maintenance, and compliance requirements. The latter of these, because of its importance, bears restatement.

**Compliance with the approved AASHTOWare standards is required from their effective date. Any exception to the application of approved standards requires the approval of the Special Committee on AASHTOWare (SCOA).**

All new contracts should include the approved standards and guidelines in this notebook.

These standards are living documents. They should be expected to change along with AASHTOWare development practices and technology. User input would be appreciated to ensure that these documents always reflect these changing circumstances.

Refer to the *Summary of Changes* for a list of the changes made to the notebook since the previous approved release of the notebook.

Questions concerning application for exceptions should be directed to your SCOA or AASHTO staff liaison. Technical questions about the notebook and its contents may be directed to the members of the T&AA Task Force.

cc: Contractors, AASHTO Staff, and T&AA Task Force members

This page is intentionally blank.

# Standards and Guidelines Notebook
# Summary of Changes

April 1, 2026

The following summarizes changes made to the Standards and Guidelines Notebook.

- The effective date was changed to April 1, 2026.
- The Software Development and Maintenance Process Standard, Hybrid Agile Development and Maintenance Standard, and Common Artifacts Standard were updated to better identify the VPAT form. The term “VPAT” technically refers to the blank assessment form. After the form is completed, the correct term is “Accessibility Conformance Report, or “ACR.” No other details about the VPAT were updated.
- The Hosted Services Standard was updated with new requirements for minimum necessary resources and for providing a minimum of two environments. The bold font for new requirements introduced in the last update was removed.
- The AASHTOWare OpenAPI Standard is new. Portions of it were included in the API Lifecycle Management Standard in the previous S&G, which has been removed from this version of the S&G. In addition to the previous content, technical API requirements for AASHTOWare OpenAPI were added.
- The Security Standard was updated to remove bold font for new requirements introduced in the last update.

This page is intentionally blank.

# AASHTOWare Standards and Guidelines
# Table of Contents

This page is intentionally blank.

# Standards & Guidelines Notebook Overview

## 1. Introduction

The Special Committee on AASHTOWare (SCOA) formed the Technical & Application Architecture (T&AA) Task Force to provide standards and technical guidance for the development of AASHTOWare software products. The Standards and Guidelines Notebook, also referred to as the S&G Notebook, is the published repository that contains all current AASHTOWare standards and guidelines. These standards and guidelines apply to all software analysis, design, development, maintenance, testing, implementation, and related work performed for AASHTOWare projects and for annual product maintenance, support, and enhancement (MSE) work. Projects and MSE work are defined and described in the *Software Development and Maintenance Process Standard* of the S&G Notebook. The purpose of these standards and guidelines was and is to maximize the return on investment, improve the quality, and increase the usefulness of the products. The following principles are emphasized.

- Standards are created and implemented to ensure a consistent approach is used to develop, change, maintain, and deliver software products.
- Guidelines are created to communicate suggestions, recommendations, and best practices that are considered useful or beneficial, but are not binding.
- Standards should be adaptable to changing technological and procedural circumstances so as not to hamper product growth or viability. They should not be viewed as static, but rather as dynamic specifications which can be easily revised whenever circumstances change and be retired whenever they no longer achieve their objectives.
- Standards should not be developed or implemented for their own sake, but only where there are apparent opportunities for benefiting AASHTOWare.
- Standards should be designed to avoid, as far as possible, increasing the administrative burdens of the project and product task forces.
- The development and implementation of standards should be a cooperative effort. All participants in the AASHTOWare development process should be included in the formulation, review, and implementation of standards and their perspectives and requirements should be respected.
- Standards include an effective date and should not ordinarily be applied retroactively. Their application should be coordinated with product contracts to avoid any unnecessary disruptions in service or inefficient use of resources.

The T&AA Task Force is responsible for the development and maintenance of the S&G Notebook and the standards, guidelines, and other documents contained in the notebook. The T&AA Task Force also provides guidance to the project and product task forces in the use and application the standards and guidelines.

Only approved standards and guidelines are included in the notebook. Each standard must be approved by SCOA. The S&G Notebook is reviewed each year, standards and guidelines are created and/or updated as needed, and a new version is normally published each year.

## 2. Notebook Organization

The following includes the organization and order of the S&G Notebooks and provides a brief description of the sections, documents, standards and guideline.

- *Cover Letter* – S&G Notebook cover letter.

- *Summary of Changes* – Summarizes all changes that have been made to this version of S&G Notebook since the previous approved release of the notebook.
- *Table of Contents* – S&G Notebook table of contents with hyperlinks to each section, standard, guideline, and document.
- *S&G Notebook Overview* – The introduction and overview of the S&G Notebook.
- *1 - Project Management and Software Engineering*

  The standards in this section define the required methods, practices, deliverables, and artifacts for project management and software engineering.

  - *Software Development and Maintenance Process (SDMP) Standard* – Defines the approach for primarily waterfall processes for both AASHTOWare software projects and for maintenance, enhancement, and support (MSE) efforts of an existing product.
  - *Hybrid Agile Development and Maintenance Process Standard* – Defines the hybrid Agile approach and an associated alternate work intake process for software products and for MSE efforts of an existing product.
  - *Common Artifacts Standard* – Defines the non-code related artifacts required to develop, enhance, and maintain software products.
  - *Quality Assurance (QA) Standard* – Defines the process for AASHTOWare quality assurance, including those QA practices that must be performed by the task forces and contractors.
- *2 – Technical Standards and Guidelines*

  The standards in this section define the required methods, practices, technologies, deliverables, and artifacts for various certain technical areas.

  - *Security Standard* – Defines the security requirements and responsibilities that shall be met when developing AASHTOWare products.
  - *Critical Application Infrastructure Currency Standard* – Describes the requirements needed to ensure AASHTOWare products maintain compatibility with updated technology and drop support for outdated technology.
  - *Database Selection and Use Standard* – Defines requirements and best practices for the use of databases in AASHTOWare product development.
  - *Spatial Standard* – Defines requirements and best practices to assure proper capture of location and its effective use in AASHTOWare products in both the mobile and office environments.
  - *Product Naming Conventions Standard* – Assists AASHTOWare contractors and users in proper use of AASHTOWare terminology for product nomenclature and identification.
  - *Backup and Disaster Recovery Standard* – Defines the actions that AASHTOWare contractors shall take to safeguard AASHTO's development investment in a project or product should a disaster occur.
  - *Mobile Application Development Guideline* – Defines an initial set of practices and technologies that should be used when developing AASHTOWare mobile applications.
  - *Web Application Development Guideline and Architecture Goals* – This guideline promotes approaches and practices for developing web-based applications (i.e. - web application) for AASHTOWare. The establishes a consistent high-level approach for contractors such that existing AASHTOWare software products evolve in a consistent and recognizable fashion which will help to align product architecture over time.

- *Web & Mobile Data Exchange Guideline* – Promotes approaches and practices for developing web and mobile device data exchanges for AASHTOWare.
- *Hosted Services Standard* – Provides details for service providers to deliver applications as a hosted service to customers. The standard requires service providers to develop a technical architecture document for the hosted solution and to establish a service level agreement with customers.
- *AASHTOWare OpenAPI Standard* – Describes the requirements and recommendations for using AASHTOWare OpenAPI and the responsibilities and requirements for AASHTOWare product application programming interface lifecycles.

- *3 – Appendices*

  This section includes documents that support the standards and guidelines included the sections 2 and 3.

  - *AASHTOWare Standards and Guidelines Definition Standard* – Defines AASHTOWare's process for developing and maintaining standards and guidelines and the S&G Notebook. This standard applies to the T&AA Task Force and is not used by the product contractors and task forces
  - *Standards and Guidelines Glossary* – Includes a glossary of all terms used throughout the S&G Notebook.

The S&G Notebook is formatted for both printing and electronic use; however, recent improvements have been added to improve viewing and navigating the electronic version.

## 3. Numbering System

The standards and guidelines are numbered to correspond to the sections to which they belong and are ordered sequentially by number in their respective sections followed by a version number. Standards include an "S" suffix following the S&G number, where guidelines include "G" suffix. Reference or informational documents, such as the glossary, include an "R" suffix. Templates and forms include a "T" suffix. For a more detailed description of the numerical format refer to the *AASHTOWare Standards and Guidelines Definition Standard* in this notebook.

## 4. Format

Each standard and guideline use a consistent style and format, beginning with a cover page, table of contents and standard sections. The cover page includes the title, S&G number, version, date, and document history. For standards, the date on the cover page is the effective date for the standard.

## 5. Requirements & Exemptions

Each standard includes requirements that must be met to comply with the standard. *These requirements are shown in red italicized text and include procedures that must be followed, deliverables and artifacts that must be produced, and required submittals and approvals.* ***New requirements that were not in the previous version of the S&G Notebook are shown in red bold italicized text.***

*All standards are in force from their effective date, which is normally the same as the effective date of the S&G Notebook.* Since guidelines are not binding, they do not include an effective date.

*If a requirement of the standard cannot be complied with, a request for an exception, including a justification, must be sent SCOA.*

Approval of exceptions to the standards is under the purview of the SCOA. The most common method of requesting an exception is to include the exception request and justification in the work plan for which the exception applies. In such a case, approval of the work plan by SCOA also includes approval of the exception request.


# 1 – Project Management and Software Engineering

This page is intentionally blank.

AASHTO AASHTOWare™ SOLUTIONS

# SOFTWARE DEVELOPMENT AND MAINTENANCE PROCESS STANDARD

## S&G Number: 1.005.02.11S

## Effective Date: April 1, 2026

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 2.8 | 5/06/2022 | Added the release review gate and allowed requirements documented externally to be referenced. | 7/13/2022 Approved by SCOA |
| 2.9 | 9/30/2023 | Updated information about data dictionaries and corrected links. | 10/02/2023 Approved by SCOA |
| 2.10 | 8/07/2024 | Removed bold font from new requirements from last year. | 12/11/2024 Approved by SCOA |
| 2.11 | 5/12/2025 | Updated information about the VPAT/ACR. | 3/12/2026 Approved by SCOA |

# Table of Contents

# 1. Overview

## 1.1. Introduction

The Software Development and Maintenance Process Standard (SDMP) primarily applies to desktop application development, enhancement, and maintenance or other efforts for which the hybrid Agile approach defined in the *Hybrid Agile Development and Maintenance Standard* does not serve needs. The default approach for development and maintenance for mobile, web-based, and software-as-a-service applications should be the hybrid Agile approach as defined in the Hybrid Agile Development and Maintenance Standard.

The SDMP defines the software development and management processes that shall be used by task forces, contractors, and other AASHTOWare stakeholders when planning and executing an AASHTOWare project or an annual product maintenance, support, and enhancement (MSE) work effort for an existing product. These processes describe the following types of activities:

- Planning and preparing the work plan for an AASHTOWare project or MSE work effort;
- Executing, managing, monitoring, and controlling the project or Product MSE work effort;
- Performing the system analysis, design, construction, and testing for the project's or MSE work effort's desired product; and
- Delivering, supporting, and maintaining the product.

The SDMP includes separate processes for project development and MSE work, which are organized around the lifecycle of a project and the lifecycle of an MSE work effort. Those activities that are unique to a single phase of the lifecycle are documented in a section that describes the activities for that phase and the deliverables and artifacts that shall be produced during that phase. Other standards used during the phase are noted with a hyperlink to that standard.

Duplicate activities that are used in a specific phase of both the project and MSE process are normally listed or summarized in the MSE process and include a hyperlink to a more detailed description in the project process. In addition, common features in both processes and common activities that are used in multiple phases are described later in this Overview chapter. When referenced in the project and MSE processes, a hyperlink is included to the description of the feature or activity. Additional information on SDMP's organization is described in the *Document Organization* section of this chapter.

### 1.1.1. Applicability and Requirements

The Software Development and Maintenance Process (SDMP) was initially published as a guideline on July 1, 2012. Effective July 1, 2013, the SDMP became a standard and the following standards were eliminated.

- Deliverable Planning and Acceptance Standard
- Requirements Standard
- Design and Construction Standard
- Testing Standard
- Implementation and Closeout Standard

*The SDMP standard applies to all AASHTOWare projects and all annual MSE work efforts. All requirements for compliance with this standard are shown in red italicized text.* ***New requirements that were not in the previous version of the Standards and Guidelines are shown in red bold italicized text.***

Refer to the Requirements & Exceptions section in the *Standards and Guidelines Notebook Overview* for a description of the procedure used to request an exception to this standard.

Most of the non-required portions of the SDMP are considered best practices and should be followed, when applicable, to ensure that AASHTOWare product development uses quality processes that can be measured and subsequently improved.

The project development and MSE process defined in the SDMP are both standard processes; however, the SDMP allows certain customizations to be made these processes. In addition, the SDMP includes two predefined adaptions to standard project process.

## 1.1.2. Project/Product Determination

As noted in the prior section, the SDMP standard includes separate processes for project development and for annual MSE work. In most cases, it is clear when to use each process; however, there are times when using the project development process for work on an existing product may be preferred. This section provides guidelines for making that determination.

### 1.1.2.1. Projects

In the context of the AASHTOWare technical service program, a project refers to work that meets one or more of the following:

- Performed under the auspice of a solicitation with funds collected one-time up-front;
- Performed to develop a new AASHTOWare product or a new module for an existing product;
- Performed to develop a redesigned or re-architected version of an existing product;
- Performed to develop one or more enhancements for an existing product where the cost, effort, timeframe, complexity, methodology, and/or beta testing meets the criteria shown in the table below; and/or
- Performed to develop requirements and/or design specifications for future development work where the cost, effort, timeframe meets the criteria shown in the table below.

### 1.1.2.2. MSE Work Effort

An MSE work effort refers to the annual maintenance, support, and enhancement work performed for an existing AASHTOWare product during a single fiscal year. The following definitions and characteristics apply to MSE work:

- The work performed under an MSE work effort is funded by annual license fee revenue.
- Maintenance is the technical activity to correct errors and other problems that cause an existing product to operate incorrectly.
- An enhancement is a development effort to add new features to an existing AASHTOWare product that is licensed annually; or an effort to modify an existing product.
- Enhancements are typically classified as small, medium or large, where
  - A small enhancement is not complex; there are minimal opportunities for confusion or misinterpretation of what needs to be built. It requires minimum funding, effort and/or resources to implement, and planning, analysis, and design.
  - A medium enhancement is more complex than a small enhancement; requires a moderate amount of funding, effort and/or resources; and requires more planning, analysis and design than a small enhancement.

- A large enhancement is complex; requires significant funding, effort and/or resources to implement; and requires significant planning, analysis, and design.

- An MSE work effort should normally not include the development of very large enhancements or specifications or major changes to the existing product's application or technical architecture.

Previous versions of the S&G Notebook referred to enhancements as minor and major, where a minor enhancement was small, and a major enhancement included both medium and large. Small, medium and large was determined to be more straight-forward and is now used is lieu of minor and major.

### 1.1.2.3. Work Plans

Separate work plans are created for each project and MSE work effort. The work plan is the formal document that describes the scope and objectives of the work to be performed by the contractor during a specific contract period, requirements or specifications to be met, tasks to be performed, deliverables to be produced, schedule to be met, cost of the effort, required staffing and resources, the technical approach for accomplishing the work, and the approach for managing, monitoring, and controlling the work.

A project work plan should be created for each development effort that meets the criteria for a project; where a product work plan (or MSE work plan) should be created for the annual MSE work effort for an existing product. The term "work plan" by itself refers to the either type of work plan (project or MSE) and is used in discussions that apply to both types of work plan.

The following table is used to help determine when a project work plan should be used and when a product work plan should be used. It also shows when enhancement and specification work for an existing product should be performed under a project work plan. If one or more of the criteria in the "Project Work Plan" column is met, it is recommended, but not required, that a project work plan be used.

**Project/Product Determination Table**

| Project/Product Criteria | Project Work Plan | Product (MSE) Work Plan |
|---|---|---|
| Solicitation of funds | Work performed with funds from a one-time solicitation is performed under a project work plan. | Not applicable |
| New product | New products are developed, tested, and completed under a project work plan. | Not applicable |
| Redesigned or re-architected version of product | A redesigned product is normally developed, tested, and completed under a project work plan. Major changes to a product's application or technical architecture are also normally performed under a project. | Not recommend due to the complexity, cost, effort and resources. |
| New module | Most new modules are normally developed, tested, and completed under a project work plan. | A new module may be developed under a product (MSE) work plan when supported by the criteria below for enhancements. |

| Project/Product Criteria | Project Work Plan | Product (MSE) Work Plan |
|---|---|---|
| Duration of time estimated to complete all work described in the work plan | The development of an enhancement for an existing product or the development of a set of specifications should be performed under a project work plan; if all work on the enhancement or specifications cannot be completed in a single fiscal year MSE work plan. | A product (MSE) work plan is used when it is estimated that all work can be completed within a single fiscal year. |
| The estimated cost to complete all work on an enhancement or specification | Work requiring $250,000 or more to complete should be performed under a project work plan. | Work requiring less than $250,000 to complete may be performed under a product work plan. |
| The amount of effort (hours) estimated to complete all work on an enhancement or specification | Work requiring 2000 or more person-hours of effort to complete should be performed under a project work plan. | Work requiring less than 2000 person-hours of effort to complete may be performed under a product work plan. |
| The development work warrants a more rigorous development or project management methodology | If yes, the work should be performed under a project work plan. | If no, the work may be performed under a product work plan. |
| Level of beta testing for the enhancement or component | The software created or revised in projects will normally require multiple beta test sites and a rigorous beta testing effort. | If fewer sites or a shorter timeframe is required for beta testing, the work may be performed under a product work plan. |
| The complexity of an enhancement's development or implementation work | The contractor should estimate the complexity of each enhancement. Enhancements with a high complexity introduce more risk in not being completed on time or on budget. It is recommended that highly complex enhancements be performed under a project work plan when the estimate for cost or effort approaches the above maximums. | Enhancements that are not estimated to be highly complex introduce less risk and may be performed under a product work plan. |
| Other project/product criteria | As defined by task force and/or SCOA. | As defined by task force and/or SCOA. |

The contractor should estimate the cost, effort, duration, and complexity of each enhancement or specification and provide to the task force. These estimates should cover all work required to complete an enhancement; including the analysis, development, integration, testing, documentation, presentation, and approval work. For those efforts limited to the development of specifications, the estimates should cover all work required to complete, document, present, and approve the specifications.

The T&AA liaison, SCOA liaison, and contractor should provide guidance in making the determination regarding the use of a project work plan or MSE work plan for an enhancement. The task force and AASHTO Project Manager (PM) will make the final decision.

### 1.1.3. Document Organization

The SDMP standard is organized differently from the other standards in the S&G Notebook. Where the other standards include a common set of sections, the SDMP standard is divided into the following Chapters:

- *1. Overview* – Includes an introduction to standard, provides guidance in the use of projects versus MSE work plans, and includes *Common Features and Activities* that apply to both projects and MSE work efforts. The processes in Chapters 2-4 include hyperlinks to items in Chapter 1 rather than repeating definitions and descriptions, such those for lifecycle phases, review gates, approvals, and status reporting

- *2. Project Development Process* – Defines the standard process for planning an AASHTOWare project and for developing and delivering the project's desired product. The standard process is described as a waterfall development methodology; however, Chapter 4 provides options for adapting the process to other methodologies.

- *3. Maintenance, Support and Enhancement Process* – Defines the standard process for planning an MSE work effort, performing maintenance work and developing enhancements for the existing product, and for delivering the modified product. This process is written as a companion process to the Project Development Process and primarily focuses on those areas were the two processes differ. When an activity in the MSE process is the same or very similar to an activity in the project process, the duplicate activity will normally include a hyperlink to the more detailed description in Chapter 2.

- *4. Adapting the Lifecycle and Process* – Defines methods to adapt the standard project development process to work with projects that use other development methodologies and defines methods to adjust the project or MSE process to work with small projects/MSE efforts. Standard adaptions are included for the following variations of the project development process.
  - *Iterative Project Development Process* – for projects using an iterative development approach.
  - *Requirements/Design Development Process* – for projects that are limited to the development of requirements and/or design specifications.

- *5. Deliverable and Artifact Definitions*– Describes each required deliverable and artifact defined in this guideline and defines the required content for each

- *6. Forms and Templates*– Includes forms and templates referenced in the body of the standard.

## 1.2. Common Features and Activities

This section describes features that are included in both the project and MSE processes, such as lifecycles, phases, review gates, and deliverables. This section also describes common activities that occur in multiple phases of both processes.

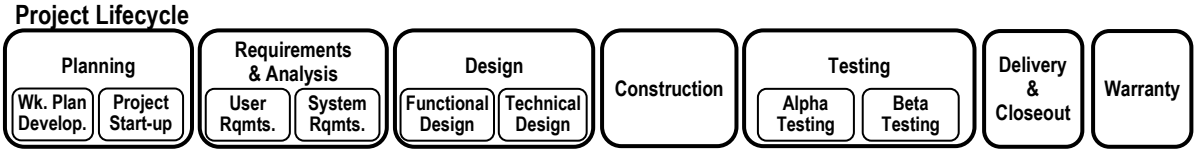
### 1.2.1. Lifecycle Model

The life cycle model partitions a project or an MSE work effort into major phases. Each phase of the lifecycle is normally divided into activities and tasks; with milestones at the completion of a group of key activities or tasks or the completion or approval of key deliverable(s). Some phases in the standard AASHTOWare lifecycles are also partitioned into sub-phases or segments between the phase and activity levels.

As described earlier in this document, projects and MSE work efforts use different work plans and development processes. The lifecycle models for projects and MSE work are also different a described below.

#### 1.2.1.1. Project Lifecycle

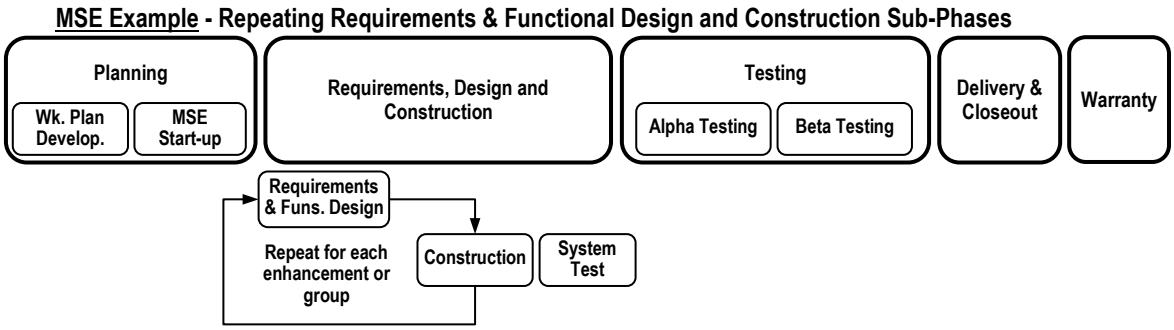
The standard project lifecycle is shown below with six phases in a waterfall sequence, with four of the phases partitioned into sub-phases. Each phase and sub-phase shown in the diagram is described in *Chapter 2*. The lifecycle is shown in a waterfall sequence; however, variations to this lifecycle allow several phases and sub-phases to overlap. For example, the Functional Design sub-phase typically begins before the end of the System Requirements sub-phase. Also, the Technical Design sub-phase typically overlaps with the Construction Phase.




The SDMP standard also includes two standard adaptions of the project lifecycle that were created to address types of projects that occur frequently within AASHTOWare. Where other adaptions to the lifecycle require an exception to standards to be approved, the *Iterative Project Lifecycle* and the *Requirements/Design Development Project Lifecycle* may be used without requesting an exception. Both lifecycles also allow flexibility for further adaption to fit the scope of each project and/or the development methodology being used.

#### 1.2.1.2. MSE Lifecycle

Since maintenance work and enhancement development do not normally require the same level of analysis and design as the development of new software, the standard MSE lifecycle combines the Requirements & Analysis, Design, and Construction activities into a single phase with three sub-phases. The Requirements & Functional Design and Construction sub-phases are normally repeated for each enhancement or each group of related enhancements as shown in the example below. These sub-phases may also be performed in a waterfall sequence where the Requirements & Functional Design is completed for all enhancements before beginning Construction. Both approaches end with a successful system test of all enhancements. Either method is considered standard and can be used without requesting an exception. The Testing and Delivery & Closeout Phases are performed for the complete scope of the MSE effort (inclusive of all enhancements, maintenance work, and upgrades). The specific development approach for each MSE effort is defined in the work plan.

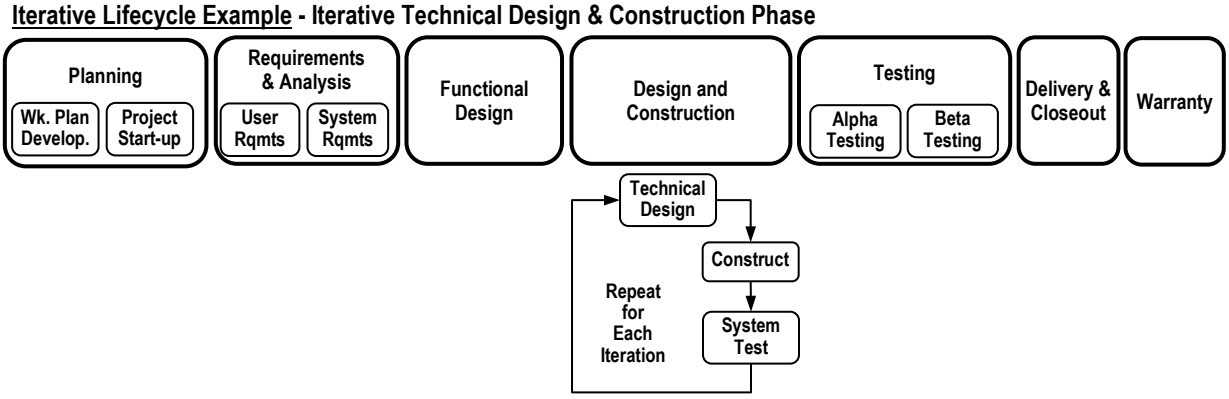



Refer to *Chapter 3* for additional details on the use and requirements of the MSE lifecycle.

### 1.2.1.3. Iterative Project Lifecycle

In this adaption of the standard project lifecycle, the project is divided into functional segments or software development timebox segments; and the key activities in the Requirements & Analysis, Design, and/or Construction Phases are completed by repeating these activities in cycles (iterations) for each segment. The phases and sub-phases from the standard project lifecycle are typically combined to form one or more iterative phases. The Iterative Project Lifecycle allows flexibility to customize the Requirements & Analysis, Design, and/or Construction Phases of the standard lifecycle (above) to fit most iterative development methodologies. The work plan should define the specific development approach and lifecycle that will be used for each project.

In the example shown below, the Technical Design, Construction, and System Testing activities for each segment are completed in iterations during a Design and Construction Phase. The User Requirements, Systems Requirements and Functional Design are prepared for the complete scope of the project (inclusive of all segments) prior to beginning the Technical Design, Construction and System Test iterations. After the iterations are completed, the lifecycle returns to the standard project lifecycle.

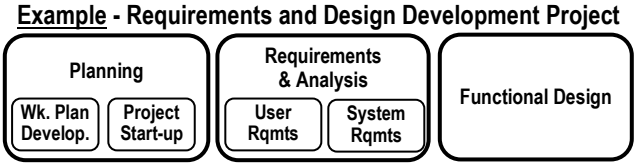



The iterative lifecycle shown in the above example plus two other variations of the iterative lifecycle are described in the *Iterative Project Development Process* section of Chapter 4 with the requirements for using these lifecycle and options for additional adaptions.

### 1.2.1.4. Requirements/Design Development Project Lifecycle

This adaption of the standard lifecycle is used for projects that are limited to the development of requirements and/or design specifications. Since this type of project includes no software development, testing and implementation activities; the phases for those activities have been eliminated. Depending on the objectives of the project, the

lifecycle model may include both the Requirements & Analysis and the Functional Design Phases as shown in the example below or include a single phase/sub-phase or combination of sub-phases.




Other examples of this lifecycle and the requirements for using and adapting the lifecycle are described in the *Requirements/Design Development Process* section in Chapter 4. As with other projects, the work plan should define the specific development approach and lifecycle that will be used for each project.

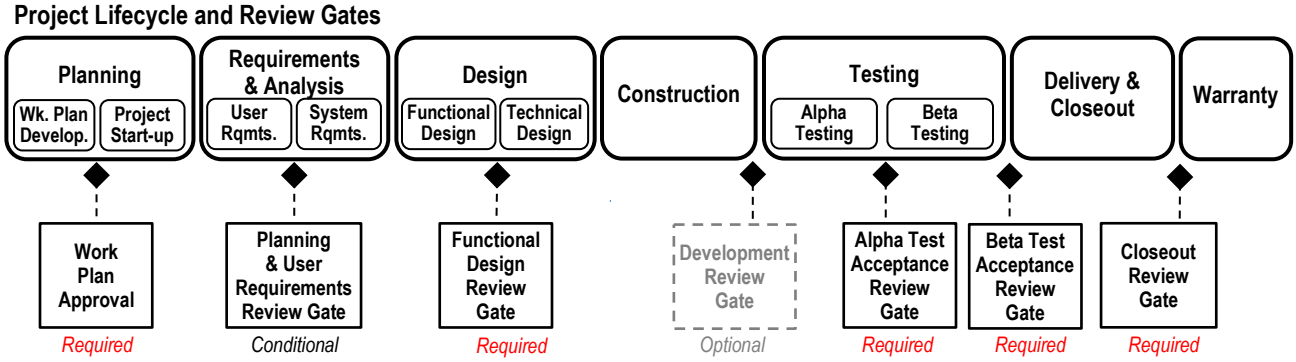
#### 1.2.1.5. Other Adaptions of the Lifecycle

In addition to the standard adaptions described above for the project and MSE lifecycle, other adaptions may be made when approved. The options for adapting the lifecycle and other components of the project or MSE process are described in *Chapter 4*.

### 1.2.2. Review Gates

Review gates are specific milestones in the lifecycle of both projects and MSE work that are achieved when the task force and contractor reach a common agreement on the completion and status of the key activities and deliverables that were planned in the work plan. When all work activities and deliverables for a review gate are completed, the contractor submits an approval request that acknowledges the completion, status of open issues, level of compliance with standards, and implementation of user requirements. Task force approval of the review gate request authorizes the contractor to proceed with the next phase or sub-phase of the project or MSE work effort.

The diagram below shows the standard project review gates relative to the phases of the project lifecycle. Work Plan Approval, which is not a review gate but represents an important approval point for a project, is also shown in the diagram. Most of the review gates are required; however, the Planning and User Requirements review gate is labelled "conditional" since it is only required under certain conditions. Also, the Development review gate is optional. Each of the standard review gates are described below in the *Review Gate Descriptions* section.




- The *Iterative Project Development Process* includes the same review gates as the standard project lifecycle. The location of the Functional Design Review varies based on the iterative lifecycle used. Additional approval points are included at the end of each iteration and/or for certain iteration deliverables. See *Additional Iterative Approval Points* for more information.

- The *Maintenance, Support and Enhancement Process* includes the same review gates except for the Functional Design review gate. When enhancements are designed and construction iteratively, additional approval points are included to approve each functional design. See *Additional Iterative Approval Points* for more information. A single approval point is used to approve the functional design when a using a waterfall process.
- The Requirements/Design Specifications lifecycle does not include the Functional Design, Development, Alpha Test Acceptance and Beta Test Acceptance review gates.

### 1.2.3. Required Deliverables and Artifacts

The S&G Notebook describes two types of output, artifacts and deliverables, where

- An artifact is defined as a tangible by-product of a software development, maintenance, or project management activity; and
- A deliverable is an artifact that shall be delivered to the task force and approved.

*Each deliverable described in this standard, as well as any other standard, is required for compliance with the standard. In addition, there are some artifacts that are not deliverables but are also required by a standard.*

The following rules and guidelines apply to preparation of each required deliverable and artifact.

- *Each required deliverable and artifact is associated with a specific review gate and shall be prepared and completed by the end of its review gate period.* A review gate period refers to the time period the between review gates.
- *Each required deliverable shall be reviewed and approved by the task force prior to its associated review gate or approved with the review gate.*
- When resources are available, each required deliverable should be reviewed by a stakeholder group made up of subject matter experts, such as a Technical Advisory Group (TAG) or Technical Review Team (TRT).
- *Each required deliverable shall be planned in the work plan, tracked and monitored.* Required artifacts should also be tracked and monitored.
- *Each required deliverable and artifact shall include the required content defined in the Deliverable and Artifact Definitions chapter (Chapter 5) of this standard or the same named section of a referenced standard.*

General activities for preparing most of the required deliverables and artifacts are described in Chapters 2-4 or another standard referenced in these chapters.

A summary view of required deliverables and artifacts for the standard project lifecycle is shown in the *Project Review Gate Diagram* in Chapter 2. This diagram shows the standard project review gates and the required deliverables and artifacts associated with each review gate in relationship to the standard project lifecycle.

Chapter 3 includes an MSE summary view in the *MSE Review Gate Diagram* and Chapter 4 includes an iterative project summary view in the *Iterative Review Gate Diagram*.

### 1.2.4. Review Gate Descriptions

The standard review gates are described in this section. Other review gates that may occur are also described. The deliverables and artifacts associated with each review gate are referenced but are not described in this section. Refer to *Chapter 5* for a description of each deliverable and artifact. Additional information is provided in Chapters 2-4 during the description of the phase where each deliverable and artifact is prepared and approved.

#### 1.2.4.1. Work Plan Approval

Work plan approval is not a standard review gate; however, *it is an important approval point that is required for each project or MSE work effort*. This approval occurs at the end of the Work Plan Development sub-phase of the Planning phase when both the task force and SCOA have approved the work plan. The approved work plan is the primary deliverable that is produced at this approval point.

Approval of the work plan authorizes the contractor to formally start-up the project or MSE work effort as defined in the work plan and contract.

#### 1.2.4.2. Planning & User Requirements Review Gate

This is a conditional review gate, which is required when any of the following components are not included or referenced in the work plan, and/or are not complete when the work plan is approved. In this case the update or the initial definition of the missing or incomplete component(s) should be planned during the execution of the work plan. A valid justification (exception) must be submitted for any of these components to be excluded from the work plan with no plan to complete them during the execution of the work plan.

- Application Infrastructure Upgrade Services (normally for MSE work)
- Technical Process and Technologies
- Management, Monitoring, and Control Procedures/Plans (The Project/Product Test Plan may be approved with the next review gate; however, it is recommended that it be approved with this review gate.)
- Backup and Recovery Plan
- User Requirements Specification (URS)
- List of Enhancements (normally for MSE work)

The Planning & User Requirements Review Gate should be planned and scheduled as described below:

- The review gate should be scheduled when significant changes are made to any of the above items other than the URS during the Project/MSE Start sub-phase. The review gate is also scheduled when the URS is revised/defined during the User Requirements sub-phase. For example:
  - If the list of enhancements to be implemented for an MSE effort is revised, this review gate is required to approve the updated list of enhancements.
  - If significant changes are made to the planned Application Infrastructure Upgrade Services in the work plan, this review gate is required to approve these changes.
  - If no Disaster Recovery (DR) Plan is included in the work plan, a DR must be developed during Project/MSE Start-Up and approved with this review gate.
  - If the existing user requirements are revised and/or new requirements or added during the User Requirements sub-phase, this review gate is required to approve the updated URS.
- The review gate is not required for minor changes that do not change the intent of these items.
- When the completion of the items is planned to occur during the execution the work plan; the work plan should include this review gate and each incomplete item should be planned as a deliverable for this review gate.
- The review gate occurs after all work on these items is completed.

- Approval of this review gate authorizes the contractor to proceed with the project or MSE work and begin developing the system requirements and functional design.

If the above items have been finalized in the work plan and no additional work is planned:

- This review gate is not scheduled; and
- The Functional Design, Development, or Alpha Test Acceptance Review Gate will be the first review gate scheduled depending on the type of project or MSE effort.

### 1.2.4.3. Functional Design Review Gate

*This is a required review gate for waterfall development projects that occurs after the Functional Design sub-phase has been completed and the following deliverables and artifacts have been completed.*

- System Requirements Specification (SRS)
- Functional Design Specification (FDS)
- Preliminary Requirements Traceability Matrix (RTM)
- Project Test Plan (May be approved with this review gate or the Planning & User Requirements Review Gate)

Approval of this review gate for waterfall development projects authorizes the contractor to proceed with the technical design and construction activities.

*This is also a required review gate for iterative development when the SRS and/or FDS are developed prior to the beginning of the iterative Design and Construction Phase. In this case, the same deliverables and artifacts listed above for waterfall (SRS, FDS, Preliminary RTM, and Project Test Plan) are approved with this review gate; and approval authorizes the contractor to proceed with Design and Construction phase.*

*The review gate is also required when preliminary (high level) SRS and/or FDS deliverables are prepared prior to beginning the iterative Design and Construction Phase. In this case, these deliverables, the Preliminary RTM and Project Test Plan are approved with this review gate; and approval authorizes the contractor to proceed with the Design and Construction Phase.*

*The review gate is also required for when a separate iterative phase is used to create the SRS and FDS prior to beginning the iterative Design and Construction phase. In this case, an Iteration FDS (with the iteration system requirements, functional design and test procedures) is created for each iteration and must be approved as described in the Additional Iterative Approval Points section. The Functional Design Review Gate is submitted after the last iteration, and, as with the above cases, approval authorizes the contractor to proceed with the Design and Construction Phase.*

The Functional Design Review Gate is not used for MSE work efforts where a separate functional design deliverable (FDS or SRDS) is created for each enhancement. *These deliverables are approved as completed by the MSE's deliverable approval procedure as described in the Additional MSE Approval Points section.*

The Functional Design Review Gate is recommended as the approval method for MSE efforts using a waterfall approach where all FDS and SRDS deliverables are approved at the same time.

This review gate is not normally used for requirements/design specification projects. The projects normally end with the Closeout Review Gate after completing the development and review of specifications.

### 1.2.4.4. Development Review Gate

This is an optional review gate for waterfall development projects which occurs at the end of the Construction Phase after a successful system test has been completed. The review gate is scheduled when the task force requests the contractor to formally acknowledge that a system test has been completed successfully, meeting all user requirements in the URS, and the product is ready for Alpha Testing. The review gate may also be scheduled to review and approve the Alpha Test Plan. In addition to a successful system test, the following deliverables have been completed at this review gate.

- Alpha Test Plan
- Preliminary RTM - At this point the Preliminary RTM has been updated since submitted with the prior review gates and includes references to test scripts.

This an optional review gate for MSE work and, if scheduled, would occur after the Requirements, Design, and Construction Phase.

The Development Review Gate is also optional for iterative projects; however, it is the recommended method for approving the last iteration in the iterative Design and Construction phase.

This review gate is not applicable for requirements/design specification projects.

### 1.2.4.5. Alpha Test Acceptance Review Gate

*This is a required review gate for software development projects (both waterfall and iterative) and for MSE work that occurs at the end of the Alpha Testing sub-phase.* By this review gate; alpha testing has been completed, issues from alpha test have been resolved, and the following deliverables and artifacts have been completed.

- Alpha Test Results Report
- Beta Test Materials (Not included when beta testing will be omitted, due to an approved exception or in those cases where the work plan includes no enhancements).
- Final Requirements Traceability Matrix (RTM) – not required for MSE work

For MSE work, the formal submission of the Product Test Plan may be delayed until this review gate; however, it should be completed and reviewed by the task force prior beginning construction of the first enhancement.

Approval of this review gate represents the acceptance of alpha testing and authorizes the contractor to proceed with the project and begin beta testing.

This review gate is not applicable for requirements/design specification projects.

### 1.2.4.6. Beta Test Acceptance Review Gate

*This is a required review gate for software development projects (both waterfall and iterative) and MSE work that occurs at the end of the Beta Testing sub-phase.* By this review gate; beta testing has been completed, issues from beta test have been resolved, and the following deliverables and artifacts have been completed.

- Beta Test Results Report
- Product Installation Package

Approval of this review gate represents the acceptance of beta testing and authorizes the contractor to proceed with the project and begin product distribution.

This review gate is not applicable for requirements/design specification projects.

This review gate is also not applicable to MSE efforts where no beta testing is performed due to an approved exception or in those cases where the work plan includes no enhancements.

### 1.2.4.7. Release Review Gate

This is an optional review gate for MSE work. It is used when multiple software releases are made for the same product in an MSE work plan. All deliverables and artifacts for the software release shall be completed at this time, including the list identified in Section 1.2.4.8. Approval of this review gate represents an agreement between the contractor and the task force to formally close work on the software release but continue working on the MSE work plan.

### 1.2.4.8. Closeout Review Gate

*This is a required review gate for all projects (including requirements/design specification projects) and all MSE work that occurs at the formal closeout of the project/MSE work effort.* All deliverables and artifacts shall be completed at this time, including the following.

- Project or MSE Archive Package
- Technical Design Specification (TDS) – not required for MSE work.
- Development and Maintenance Document - not required for MSE work.
- Updated or New Accessibility Conformance Report (ACR)
- Updated or New Application Infrastructure Component List

The Project/MSE Archive Package is the only deliverable that is applicable to requirements/design specifications projects.

Approval of this review gate represents an agreement between the contractor and task force to formally close the project or MSE.

### 1.2.4.9. Other Review Gates

The task force may also request the contractor to add other review gates to the work plan in addition to the standard review gates. If additional review gates are used, the work plan should define when the review gates occur and what deliverables are associated with each review gate.

### 1.2.4.10. Additional Iterative Approval Points

In most cases, iterative development projects include additional task force approval points that align with the incremental nature of the development methodology used as described below.

- At the conclusion of each iteration in the Design and Construction Phase.
  - *An Iteration Test Results Report is prepared after the system testing of each iteration.*
  - *Each Iteration Test Results Report must be approved by the project's Review Gate Approval Procedure or Deliverable Review and Approval procedure.*
- At the conclusion of each iteration in the Requirements & Functional Design phase:
  - *URS, SRS, and/or FDS deliverables (or a combined deliverable) are prepared for each iteration.*
  - *The iteration deliverable(s) must be approved by the project's Review Gate Approval Procedure or Deliverable Review and Approval procedure.*

The iteration approvals and review gates should be planned and documented in the approved work plan. Refer to the *Iterative Project Development Process* section in Chapter 4 for additional details on iterative development.

#### 1.2.4.11. Additional MSE Approval Points

If enhancements are designed and constructed by repeating the requirements, design and construction activities in an iterative, non-waterfall approach; deliverable approval points occur between the design and contraction of each enhancement.

- *SRS and FDS deliverables or a combined SRDS deliverable is prepared for each enhancement or for a group of related enhancements.*
- *The deliverable(s) must be approved by the project's Review Gate Approval Procedure or Deliverable Review and Approval procedure.*

Refer to the *Maintenance, Support and Enhancement Process* chapter for additional details on MSE work efforts.

### 1.2.5. Identify Stakeholders to Review Deliverables

As previously noted, each deliverable shall be approved by the task force. It is also recommended that a stakeholder group, such as a Technical Review Team (TRT) or Technical Advisory Group (TAG), review and comment on each deliverable prior to the task force review. If the appropriate resources are available for stakeholder review:

- The task force should review each planned deliverable and identify a stakeholder group to review each deliverable.
- The task force should also determine if they want an approval recommendation from the stakeholder group for each deliverable.
- The stakeholders to review and approve deliverables should be identified early in the project/MSE lifecycle, documented, and communicated to the contractor.

### 1.2.6. Deliverable Review and Approval

After each deliverable is completed during the execution of the project or MSE work effort, the contractor should provide the deliverable to the task force for review, comment, and approval. As described above, stakeholder review is also recommended. This section describes a typical procedure used for deliverable review and approval beginning with stakeholder review and ending with task force approval. *The specific Deliverable Approval Procedure used in a project or MSE work effort is defined in the work plan.*

#### 1.2.6.1. Obtain Stakeholder Review and Approval

If a stakeholder group has been identified to review the deliverable, the contractor should:

- Provide the completed deliverable to the stakeholder group and solicit comments and issues;
- If needed, schedule a meeting, conference call, or online conference for the contractor to walk through the deliverable with the stakeholder group;
- Ensure that all stakeholder comments and issues are documented;
- Resolve important issues prior to submitting the deliverable for task force review and approval;
- Maintain a log of issues and actions taken;
- If needed, repeat the review; and

- Copy the task force chair and the AASHTO PM on all correspondence to the stakeholder group regarding this review.

If the task force requests an approval recommendation from the stakeholder group, the contractor should solicit an approval recommendation for the deliverable after the stakeholder review and modification process has been completed. The recommendation should be documented and communicated to the contractor, AASHTO PM and task force.

#### 1.2.6.2. Obtain Task Force Review and Approval

After stakeholder review on the deliverable is completed, the contractor should:

- Provide the deliverable to the task force for review and solicit comments and issues;
- Provide the approval recommendation from the stakeholders (if available) along with any unresolved issues regarding the deliverable;
- Include the AASHTO PM in the review process;
- If needed, schedule a walkthrough of the deliverable with the task force;
- Ensure that task force comments and issues are collected and documented;
- Make corrections to the deliverable as directed by the task force;
- Update the log of issues and actions taken; and
- If needed, repeat the review.

After the task force review and follow-up corrections have been completed, the task force may elect to approve the deliverable prior to the review gate. If an approval decision is made at this time, the decision should be documented and communicated to the contractor and the AASHTO PM. The approval method used should be consistent with the Deliverable Approval Procedure in the work plan.

*If the deliverable is not approved at this time; it shall be submitted and approved with its' associated review gate.*

### 1.2.7. Review Gate Approval Procedure

*After completing the stakeholder and task force reviews, making needed corrections, and completing all other activities and tasks planned for the current review gate period; the contractor shall prepare a review gate approval request and submit it to the task force for approval. Task force approval is required to proceed to the next review gate period.*

This section describes the typical procedure that should be used for preparing, submitting, and approving review gates for both projects and MSE work efforts. *The specific Review Gate Approval Procedure used in a project or MSE work effort is defined in the work plan.* The method used for signatures should also be defined in this procedure, as should the responsibilities of designees (refer to *Review Gate Signatures and Designees)*.

The order and schedule of the review gates should also be defined in the work plan.

#### 1.2.7.1. Review Issues

*Prior to submitting each review gate request for task force approval, the contractor shall review all open issues associated with the deliverables, artifacts, and activities for the current review gate period. All unresolved issues shall be reported with the review gate approval request along with the plan for resolution of each open issue.*

#### 1.2.7.2. Review for Compliance with Standards

*The contractor shall also review the deliverables and required artifacts and determine if each deliverable/artifact complies with the applicable AASHTOWare standard(s). Any*

*area of noncompliance shall be reported with the review gate approval request with the justification for the noncompliance.*

### 1.2.7.3. Review User Requirements Implementation

*In addition to the above review for issues and compliance, the contractor shall determine if each deliverable implements or supports all user requirements and enhancements in the URS. Each requirement and enhancement that is not implemented or supported in one of the deliverables shall be reported with the review gate approval request.*

### 1.2.7.4. Prepare and Submit Review Gate Approval Request

*The contractor shall prepare and submit a Review Gate Approval Request as follows:*

- *The Review Gate Approval Request form, which may be found in the Common Artifacts Standard, or an equivalent document with the same content is used for preparing review gate approval requests.*
- *The completed form is signed by the contractor project manager and submitted to the task force chair and the AASHTO PM.*
- *Any unapproved deliverables for the review gate are submitted with and referenced on the form.*
- *If the task force has already approved a deliverable prior to the review gate, the prior approval is documented on the review gate approval request along with the location of the deliverable.*
- *Answer the checklist questions on the form regarding compliance with standards, open issues, and implemented user requirements.*
- *If the answer to any of the checklist questions is "no", additional information shall be provided to support the "no" answers. The checklist answers shall address all deliverables for the review gate period regardless of prior approval.*
- Any other useful information, such as documentation of stakeholder approval recommendations should be submitted with the request, as deemed appropriate.

### 1.2.7.5. Approve Review Gate

*After receiving each review gate approval request, the task force shall review the request and make an approval decision as follows:*

- *Review the deliverables submitted, the answers to each question with the request, and the additional information provided to support the deliverables and/or questions.*
- *Make an approval decision regarding the request.*
- *If approved, the task force chair signs the review gate approval request, includes the approval decision, and returns the document to the contractor project manager.*
- If the task force decides not to approve the review gate request:
  - The task force chair may decide to request additional information from the contractor regarding the request; or
  - The chair may sign and return the request to the contractor and provide the reason for the denial and directions to the contractor regarding correction and resubmission of the request.
- The task force may also provide directions to the contractor with an approval decision. This could include conditions regarding the approval, next steps, or other information the task force needs to communicate regarding the approval decision.

After the review gate approval request is approved, the AASHTO PM also signs the review gate approval request.

The approved request with signatures by the contractor project manager, the task force chair and the AASHTO PM represents a common agreement between all parties that:

- All planned activities, deliverables, and artifacts for the review gate period have been completed;
- All user requirements have been implemented and/or tested in each deliverable, or an acceptable justification has been provided for requirements that have not been implemented/tested;
- AASHTOWare standards have been complied with or acceptable justification has been provided for noncompliance with the standards;
- All issues have been resolved or acceptable plans have been provided for resolving open issues; and
- The project is ready to proceed to the next review gate period (next phase or sub-phase in the project lifecycle following the review gate).

### 1.2.8. Review Gate Signatures and Designees

Signatures on the review gate approval request may be in any form agreed upon by both the task force and contractor, such as written signatures, electronic images of signatures, a note in signature block referencing an email approval, or some other method. The method of signature should be documented in the *Review Gate Approval Procedure* or another document and communicated to all parties involved.

The task force chair and/or the contractor project manager may also assign designees to sign the review gate approval request. For example, the task force chair may assign another task force member to sign the request; and the contractor project manager may assign a lead for a specific product or module within the family of products to sign a request that is associated with that product or module. The task force chair and/or contractor project manager may also assign a designee as sender or recipient of the review gate approval.

When designees are assigned, their responsibility should be documented in the *Review Gate Approval Procedure* or another document, communicated to all parties involved, and agreed to by both the task force chair and contractor project manager.

Designees may also be assigned responsibilities regarding the procedures for *Deliverable Review and Approval*. These responsibilities should also be documented, communicated, and agreed to by the task force chair and contractor project manager.

### 1.2.9. Status Reporting

*As described in the Common Artifacts Standard, the contractor shall prepare a status report and provide to the task force at least once a month for projects and once a quarter for each MSE efforts. The status reporting process for each project and MSE work effort shall be defined in the work plan or shall be defined later in the project or MSE effort during Project Start-up. The process shall describe the frequency of status reports, the distribution of the status reports, and the content that will be provided in each status report or reference example report.*

*The status reports shall include but are not limited to the following content: date of report, dates of reporting period, summary view, accomplishments for this period, planned activities for next reporting period, budget status, milestones/deliverables, changes requests, risks, and issues.*

The intent of the Summary View is to provide a quick view of the status of key areas such as schedule, scope, budget, deliverables, changes, risks and issues. Green, Yellow, and Red or another similar method should be used in the Summary View.

The *Common Artifacts Standard* includes an example status report that meets the content requirements.

## 1.2.10. Project Repository

*A project repository shall be established and maintained for each project and MSE work effort as described below.*

### 1.2.10.1. Establish Repository

*The project repository shall be established before or during the Project Start-up sub-phase of the Planning Phase for both projects and MSE work.* Microsoft SharePoint is the preferred tool for creating, maintaining, and accessing the repository; however, other tools may be used if approved by the task force and SCOA.

AASHTO Staff is responsible for creating all SharePoint workspaces. *The content stored in the repository shall be accessible to the contractor, task force, AASHTO PM, SCOA and T&AA liaisons, and other stakeholders designated by the task force.*

The organization of the repository is left up to the task force, the AASHTO PM, and the contractor. *The technology used for the project repository and the procedures for naming, versioning, storing and revising deliverables and artifacts shall be defined in the work plan or shall be defined later during Project Start-up.*

### 1.2.10.2. Store and Update Files in Repository

*Each document-based deliverable and artifact prepared or updated during the project/MSE shall be stored in the repository. All documentation related to the review, feedback, submission, approval, rejection, or changes of review gates and deliverables shall also be stored in a project repository.* The repository should also be used to store other project documentation that will need to be accessed by the project participants.

After a deliverable is approved by the task force, TRT, or TAG, it should be stored in the project repository. Each time a deliverable is changed and reapproved, the revised deliverable should be stored in the repository as a new file with a new version and date. Required artifacts should also be stored and updated using similar conventions.

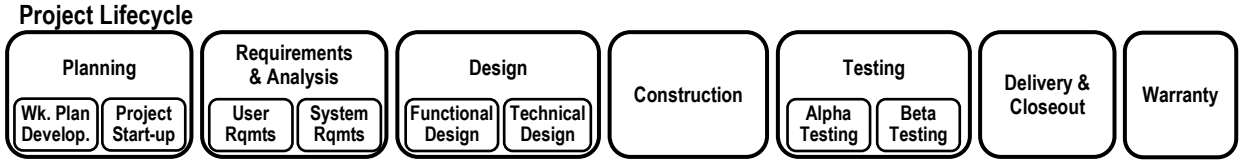
# 2. Project Development Process

## 2.1. Introduction

The Project Development Process defines the standard process that shall be used by task forces, contractors, and other AASHTOWare stakeholders when planning and executing an AASHTOWare software development project. The process is described for projects using a waterfall development methodology or a variation of waterfall; however, it may be adapted to other software development lifecycle methodologies and limited scope projects as described in the *Adapting the Lifecycle and Process* chapter.

### 2.1.1. Chapter Organization

This chapter is organized around the standard project lifecycle model shown below.



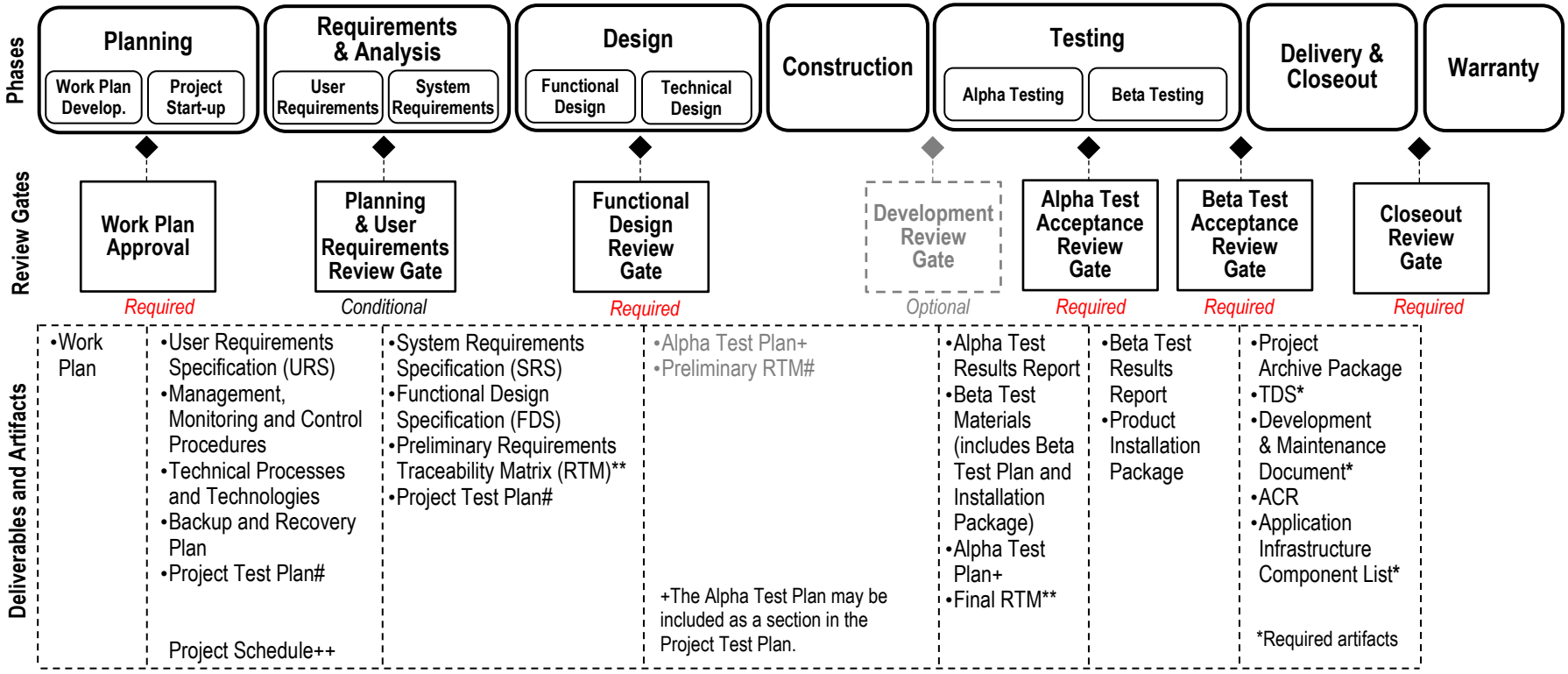

Following this Introduction section is a section for each phase of the project lifecycle model. Each of the phase sections incudes:

- The deliverables and artifacts that are prepared during the phase;
- The deliverables and review gates that are approved during the phase;
- The other standards that are used during this phase; and
- The procedures to be followed during the phase.

### 2.1.2. Review Gates, Deliverables and Artifacts

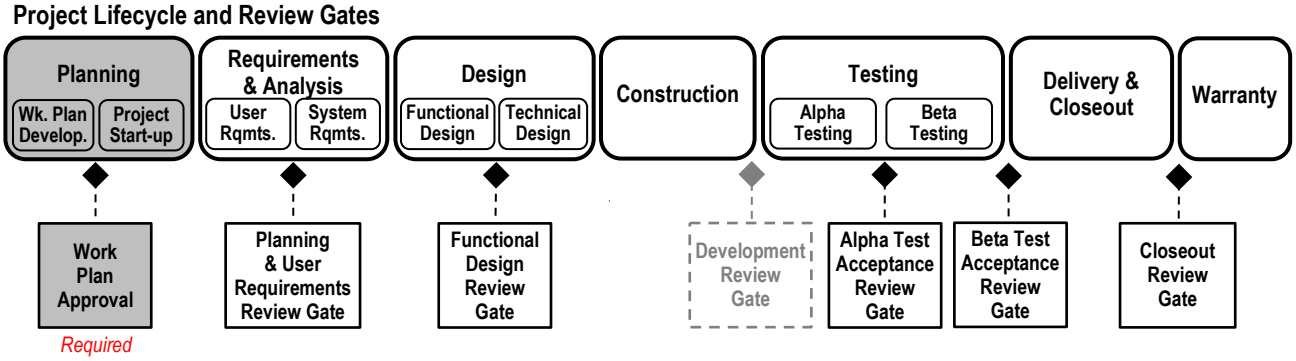
The following diagram provides a summary view of the standard project review gates and the required deliverables and artifacts associated with each review gate in relationship to the standard project lifecycle.

## Project Lifecycle Phases and Review Gates with Required Deliverables and Artifacts




- Each review gate is required, with the exception of the Planning & User Requirements Review Gate and the Development Review Gate.
- The Planning & User Requirements Review Gate is conditional and must be planned when any of the items listed below the review gate are not included or not completed in the work plan. ++This does not apply to the project schedule; however, an initial project schedule is created during Project Start-up as shown above.
- The Development Review Gate is optional and is planned when the task force requests the contractor to formally acknowledge that a system test has been completed successfully or the task force wants to review the Alpha Test Plan prior to Alpha Testing.
- If not submitted with the Development Review Gate, the Alpha Test Plan is submitted with the Alpha Test Acceptance Review Gate.
- **The RTM is developed in stages. A Preliminary RTM is submitted with the Functional Design Review Gate and the optional Development Review Gate. The Final RTM is submitted is submitted with the Alpha Test Acceptance Review Gate.
- #The Project Test Plan describes the testing methodology, test phases and test deliverables. It is recommended to complete the Project Test Plan during Project Start-up, but it may be delayed until the Functional Design Review Gate.
- The deliverables with each review gates are required and must be approved prior to, or with, their review gate.
- *The artifacts with each review gate are also required; however, approval is not required.
- Additional review gates may be used as agreed upon by the task force and contractor.

# 2.2. Planning Phase




## 2.2.1. Phase Overview

The Planning Phase is the first phase in the lifecycle of an AASHTOWare project and is divided two segments or sub-phases, Work Plan Development and Project Start-Up.

- During the Work Plan Development sub-phase, the project is planned, the work plan is prepared by the contractor, the work plan is reviewed and refined, and the final work plan is approved by the task force and SCOA.
- During Project Start-Up, the project is formally started, and all planning and mobilization activities needed to move forward with the development activities are completed. These activities include, but are not limited to, reviewing the work plan, refining the components of the work plan and obtaining task force approval, and establishing the procedures and technologies for the project.

The SDMP standard addresses the content of the work plan, the preparation and content of the project user requirements, and the planning activities that occur during the Project Start-Up sub-phase. The procedures to review and approve the work plan are briefly described but are outside the scope of this document. Pre-work plan development activities, such as preparing solicitations and RFPs, selecting a contractor, and forming the task force are also outside the scope of this standard. These out of scope activities are accomplished using internal AASHTOWare procedures.

## 2.2.2. Input to the Planning Phase

The primary deliverables and artifacts that will be used or referenced in this phase are:

- AASHTOWare Project Work Plan Template;
- Request for Proposal (RPF)
- Vendor response to the RPF
- User Requirements

Other key items used in this phase are the AASHTOWare Policies, Guidelines, and Procedures (PG&P), AASHTOWare Project/Product Task Force Handbook, and internal AASHTOWare procedures. The PG&P and Task Force Handbook are available for download on the AASHTOWare web server at:

*https://www.aashtoware.org/wp-content/uploads/2021/07/Policies_Guidelines_Procedures-June-2021.pdf*

*https://www.aashtoware.org/wp-content/uploads/2018/03/TaskForceHandbook-October2009.pdf*

In some cases, the System Requirement Specifications (SRS) and/or Functional Design Specifications (FDS) are created during a prior project or MSE effort and will be included as input to the current project in the User Requirements and Specifications section of the work plan.

Also, there may be cases where one or more large or complex enhancements are implemented in a project in lieu of an MSE effort. The description of each enhancement will be included as input to the project in the User Requirements and Specifications section of the work plan along with any previously defined requirements and design specifications.

### 2.2.3. Output from the Planning Phase

The following artifacts and deliverables are created or updated during this phase of the project.

- *Project Work Plan*
- Work Plan Components – The work plan includes sections and sub-sections that shall be completed in the approved work plan, and it also includes other parts that may be completed after the project is formally started. *If any of the following work plan components are not included or are not complete in the work plan, the work plan shall define the plan to prepare or revise and approve the incomplete components as deliverables during the execution of the work plan.*
  - User Requirements and Specifications
  - Technical Process and Technologies
  - Project Management, Monitoring and Control Procedures
  - Communication Management Approach
  - Configuration Management and Project Repository Approach
  - Risk Management Approach
  - Backup and Disaster Recovery
  - Planned Deliverables, Review Gates and Milestones
- *Project Schedule/Work Breakdown Structure*
- *Project Repository*
- If a project includes enhancements to an existing product, any changes to the planned enhancements should be made during Project Start-Up.

### 2.2.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Common Artifacts Standard*
- *Quality Assurance Standard* – Used when developing the Quality Assurance Reviews section of the work plan and planning QA work activities, including participating in the annual QA meeting.
- *Backup and Disaster Recovery Standard* – Used when developing the Backup and Disaster Recovery Plans for the work plan and planning backup and disaster recovery activities.

### 2.2.5. Procedures

This section defines the project planning and project management activities that are to be followed by the task force and/or contractor during the Planning Phase and the results of those activities.

### 2.2.5.1. Develop and Approve Work Plan

The Work Plan Development sub-phase includes those planning activities associated with the work plan that occur prior to the formal start-up of the project. Those planning activities that occur prior to work plan development, such as preparing a solicitation or RFP or the initial development of the user requirements, are not addressed.

#### 2.2.5.1.1. Review and Approve User Requirements for Inclusion in Work Plan

The user requirements for a software development project are normally developed prior to the beginning the project and will likely have been developed in one of the following scenarios.

- Developed as a deliverable from a previous AASHTOWare project, as described in the *Requirements/Design Development Process* section of Chapter 4.
- Developed as a deliverable from a previous MSE work effort; or
- Developed as part of a work effort to prepare a project solicitation and the subsequent Request for Proposal.

This process assumes that an initial set of user requirements exist when the work plan is prepared; therefore, when developing work plan, the user requirements work activities are normally limited to the following:

- Review of the previously defined user requirements by the current project organization (task force, contractor, AASHTO PM, and/or others).
- Agreeing on the final set of user requirements to be included in the work plan.
- *Documenting the requirements in the form of a User Requirements Specification (URS), as described in Chapter 5.*
- Planning and estimating additional work activities regarding the user requirements that will occur after the project has formally started. This could include a more detailed review and validation, potential revisions or additions to the user requirements, and task force approval of the revised URS.

Similar planning and review activities occur for those projects that implement enhancements for an existing product.

If the project is to define the user requirements, the planning is limited to the initial estimation of those work activities required to solicit, document, review, validate and approve the user requirements.

#### 2.2.5.1.2. Review and Approve Other Specifications for Inclusion in Work Plan

If the work plan includes a set of system requirements and/or functional design specifications that shall be met by the project's proposed product or product component, these should be reviewed at this point. As with the user requirements, these specifications should be reviewed and agreed upon before they are included in a new project work plan. Also, if any additional work on these specifications will be performed after the project is started, this work should be planned and estimated.

#### 2.2.5.1.3. Prepare Project Work Plan

*The AASHTOWare Project Work Plan Template is a Microsoft Word template that is used to create a project work plan.* The template includes all the required information that shall be included for each project work plan and contains instructions on how the template is to be used and completed. The URL for downloading the template is included in the *Common Artifacts Standard*.

*Each section of the template shall be completed unless it will be completed after the project is started or the section is not applicable due to the scope of the project.*

If a section is not applicable, this should be noted in the work plan. An explanation as to why the section is not applicable may be needed to obtain work plan approval.

*When estimating the work effort and cost for the project, the contractor shall include the work activities to prepare and approve each required deliverable, artifact and review gate defined in the Project Development Process. This shall include any revisions to the URS and other specifications, as described above. Also, if any other required components of the work plan are not complete or not included in the work plan, these components shall also be included as planned deliverables and the work to revise/develop and approve these as deliverables shall be planned and estimated in the work plan.*

*Any known exceptions to AASHTOWare standards shall be described in the work plan with the justification for each exception.*

#### 2.2.5.1.4. Approve Work Plan

*The completed work plan (with attachments or references) is reviewed and approved by the task force.* Approval of the work plan represents approval of the plan for the project and all methods, tools, technologies, procedures, and plans documented in the work plan. Approval also indicates the commitment to the work plan by both the task force and contractor.

*SCOA also approves the work plan and any planned exceptions to standards.* If exceptions to standards are included in the work plan, these are subsequently approved or rejected with the work plan approval. Exceptions may also be submitted to SCOA for approval in writing from the task force chair to the SCOA chair at a later date.

The specific procedures used to approve the work plan are outside the scope of this standard.

### 2.2.5.2. Perform Project Start-Up Activities

The Project Start-Up sub-phase begins with the formal start-up of the project and includes the planning and mobilization activities that occur prior to beginning the analysis, development, testing, and implementation activities of the project. The project is started as defined in the project contract; executed as defined by the work activities and tasks in the work plan and project schedule; and managed as defined by the project's management, monitoring, and control procedures.

#### 2.2.5.2.1. Review Work Plan

One of the first activities the contractor should perform is a review of the work plan and project schedule to ensure the appropriate level of understanding of the work to be completed during the first month, remainder of this phase, remainder of the first review gate period, and the remainder of the project.

#### 2.2.5.2.2. Plan Work for First Month

After the review is complete, the contractor should define the detailed activities, tasks, deliverables, and milestones, with target dates, projected to be completed during the first month or thirty-day period. The level of detail should be adequate to:

- Determine the amount of effort required;
- Assign the appropriate technical resources to the activities and tasks;
- Determine issues, concerns, and/or risks;
- Track progress; and
- Prepare status reports.

#### 2.2.5.2.3. Plan Work through Current Phase and First Review Gate

After planning the first month of work, the contractor should plan or validate the work for the remainder of the phase and the first review gate period. Where monthly planning should be detailed, this planning should be at a higher level of detail.

*If any of the required work plan components (technical process, technology, or procedure) were not included or completed in the approved work plan, the Project Planning & User Requirements Review Gate will be the first review gate. The Project Planning & User Requirements Review Gate must also be the first review gate when significant revisions are made to the URS/Enhancement List and when a new URS is prepared. If the Project Planning & User Requirements Review Gate is not required, the first review gate will be Functional Design Review Gate.* In either case, the contractor should ensure that the first review gate is planned and verify or update the target date for this review gate.

#### 2.2.5.2.4. Prepare Project Schedule

After the current phase and first review gate are planned, the contractor should estimate the target dates of all planned deliverables and review gates other key milestones for the remainder of the project.

*The contractor shall then prepare an initial project schedule or work breakdown structure that includes the estimated completion/approval date of all deliverables, review gates, and key milestones for the project.* The more detailed activities, tasks, milestones, and deliverables estimated in the previous steps should also be included in the schedule.

The project schedule should be provided to the task force and AASHTO PM for review and approval. Any issues and concerns and new risks found should also be provided.

#### 2.2.5.2.5. Identify Stakeholders

In addition to the above activities, the task force should *Identify Stakeholders to Review Deliverables* and provide to the contractor as discussed in Chapter 1.

#### 2.2.5.2.6. Execute Plan for First Month and Remainder of Project-Start-Up

After the project schedule is updated and approved, the contractor, task force, and AASHTO PM should begin executing the activities and tasks defined for the first month of the project and the remainder of the Project Start-Up sub-phase. These will normally include the following:

- Prepare and/or revise any of the required work plan components that were not included or completed in the approved work plan, with the exception of the URS and Project Test Plan, which are normally prepared or revised in later phases.
- Review and approve the new/revised work plan components as deliverables or as a revision to the work plan. If a component is not approved here, it shall be approved with the Project Planning & User Requirements Review Gate.
- Implement and setup the technologies required for the project, including establishing the project repository.
- Store the work plan, components of the work plan, project schedule, and all documentation created to date in the repository.

#### 2.2.5.2.7. Report Status and Plan Next Month

*As described in the Status Reporting section of Chapter 1, a status report shall be submitted to the task force at least once a month.* At the end of the first month and at the end of each month thereafter, the contractor should:

- Review the project schedule and the progress made during this period.
- After the first month, review the information reported in previous status report.
- Review the status of open issues, high level risks, change requests, and project budget.
- Define the detailed activities, tasks, deliverables, and milestones, with target dates, projected to be completed during the next month in the same manner described above for planning the work for the first month.
- If needed, revise the high-level target dates for deliverables, milestones, and review gates for the remainder of the current phase, the next phase, and the remainder of the project.
- *Prepare and submit the status report as required by the project's Status Reporting procedure.*
- Update the project schedule, as required, and provide to the task force and AASHTO PM with the status report.

If the status reporting period is more frequent than once a month, the above activities should be adjusted appropriately.

### 2.2.5.3. Plan and Execute the Next Phase of the Project

At the end of the Project-Start-Up sub-phase, the contractor should:

- Review the planned work for the next phase of work, Requirements and Analysis Phase, as well as any detailed work planned for the next month.
- Perform additional planning for the next phase and update the project schedule, as required.
- Execute the next phase as defined by the activities, tasks, and milestones in the project schedule and the development methodology.

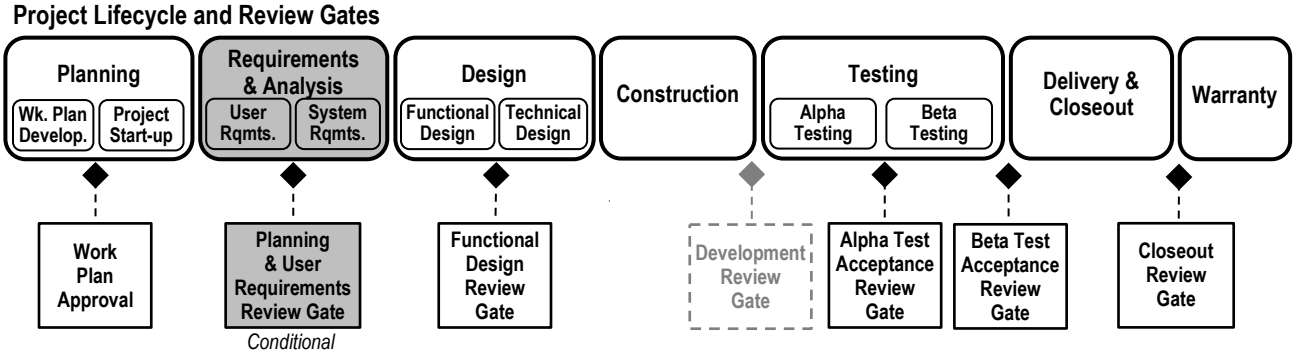
### 2.2.5.4. Manage, Monitor, and Control the Project

In parallel to executing the specific activities and tasks for each phase, there are certain activities that are repeated throughout the lifecycle of the project. The occurrence of these activities and the specific actions and methods used are defined by the management, monitoring, and control procedures that were documented in the work plan or defined or updated during project Start-Up. These activities include:

- Submitting, reviewing and approving deliverables as they are completed.
- Submitting, reviewing and approving review gate approval requests at the end of each review gate period.
- Storing new and revised deliverables, artifacts, and other documentation in the project repository.
- Monitoring and tracking progress.
- Identifying and managing issues as they are encountered.
- Submitting, reviewing, approving, and implementing changes to deliverables, requirements, scope, schedule, budget, etc.
- Identifying and managing potential risks.
- Reporting the project status each month and planning the next month's activities.
- Performing quality assurance activities.
- Performing testing activities.
- Managing configuration items.
- Communicating project information between stakeholders.
- Performing backups and restores of the development and maintenance environment.

- Restoring the development or maintenance environment at an alternate location if an emergency occurs.

# 2.3. Requirements & Analysis Phase




## 2.3.1. Phase Overview

The Requirements and Analysis Phase is divided into two segments or sub-phases, User Requirements and System Requirements.

- The User Requirements sub-phase typically involves the review, validation, update and/or definition of user requirements.
  - User requirements (or user stories) define what the users and other business stakeholders need and expect from the product or product component that will be developed.
  - Most user requirements define functions that the product/component must do or perform, data needs, and known business and technical constraints.
  - The user requirements for the project are documented in the form of a User Requirements Specification (URS) unless the user requirements are documented externally, such as scientific or technical publications. In this case, such requirements may be included in a project or MSE work effort by reference.
  - In most cases, the URS is developed in a prior project or MSE work effort or developed in conjunction with a solicitation and is included or referenced in the current project's work plan.
- During the System Requirements sub-phase, the system requirements are defined, or existing system requirements are reviewed and validated, and, if needed, updated.
  - System requirements are derived by reviewing, analyzing and decomposing the user requirements and normally provide the additional detail and clarification needed for the design of the product.
  - System requirements are typically in the form of functional, data and non-functional requirements.
  - The system requirements are documented in one or more documents that are referred to as the System Requirements Specification (SRS). The SRS may be composed of or include references to sufficiently documented requirements external to the work effort, such as those included in technical or scientific publications.
  - The SRS must be approved by the task force.
  - The approved SRS is considered the final set of requirements and is the basis for developing the Functional Design Specification (FDS) for the proposed product, which is described in the next section, *Design Phase*.

The Requirements and Analysis Phase also includes preparing and submitting the initial versions (preliminary) of the Requirements Traceability Matrix (RTM). The RTM is described later in this chapter.

### 2.3.2. Input to the Requirements and Analysis Phase

- The primary deliverables and artifacts that will be used or referenced in this phase are:
- User Requirements Specification (URS) – except when project will define the URS
- Description of enhancements to be implemented during the project. These may be accompanied by previously defined requirements for the enhancements.
- Previously defined System Requirements Specification (SRS) – if included in work plan
- Other key items used in this phase are the Project Work Plan; Project Schedule; the work plan procedures, processes and technologies; and Section 508/Accessibility web sites.

### 2.3.3. Output from the Requirements and Analysis Phase

The following artifacts and deliverables are created or updated during this phase of the project.

- User Requirements Specification (URS) – if revised or created during this phase.
- The Planning and User Requirements Review Gate Approval Request – If the URS is revised/created during this phase, or if any of the required work plan components (technical process, technology, and procedure) or enhancement list were revised during Project-Start-Up, this review gate is required.
- *System Requirements Specification (SRS) – initial or revised*
- *Preliminary Requirements Traceability Matrix (RTM) - initial*
- *Project Schedule – if revised*
- *Project Repository - updated*

### 2.3.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Security Standard* - Used when developing security requirements for the SRS.

### 2.3.5. Procedures

This section defines the major activities that are to be followed by the task force and/or contractor during the Requirements and Analysis Phase and the results of those activities.

#### 2.3.5.1. Maintain Bi-Directional Traceability

*During this phase of the project, the contractor shall create the initial Requirements Traceability Matrix (RTM).* The RTM is a tool used to assist in maintaining the traceability between user requirements, systems requirements, and elements in other deliverables. *The RTM includes:*

- *All user requirements from the approved URS with the same Requirement ID used in the URS.*
- *Backwards reference from each user requirement to its source or origin. The source of most user requirements will be the URS in the work plan, updated URS, or a change request. Other sources such as an RFP may also be used.*
- *All system requirements from the approved SRS with backwards traceability reference to a source user requirement. System requirements are identified by their ID in the SRS.*

- *Forward traceability from each system requirement to design element in the Functional Design Specification (FDS).*
- *Forward traceability from each system requirement to test procedures in the Alpha Test Plan.*
- *Content listed for the Requirements Traceability Matrix (RTM) in Chapter 5.*

*If a user requirement is of sufficient detail that no further detail and clarification is required, the user requirement should be repeated in the SRS as a system requirement to ensure traceability.* Refer to the *Define System Requirements* section for additional information.

The RTM is created in phases as the project progresses. The initial version of the RTM is created after the user requirements are reviewed and validated. The RTM is updated and maintained as the user and system requirements are defined and revised and as the requirements are implemented in design and testing deliverables.

Until all elements are added to the RTM, it is considered preliminary and is referred to as the Preliminary RTM. The Preliminary RTM is submitted to stakeholders and the task force when reviewing and approving the SRS, FDS, the Functional Design Review Gate, and the optional Development Review Gate. The final version of the RTM is submitted for approval with the Alpha Test Acceptance Review Gate. The RTM may be created as a document, spread sheet, or another type of digital file or repository.

### 2.3.5.2. Review, Analyze and Validate User Requirements

In most cases, an initial set of user requirements is included in the work plan, so the initial user requirements activity described below addresses the review, analysis and validation of the existing URS.

If no user requirements exist, the activities in *Define and Approve User Requirements* in the *Requirements/Design Development Process* section should be used to define requirements and prepare the URS. After defining the URS, the following review, analysis and validation activity is performed on the new URS.

The contractor staff and the task force should both review the URS and ensure that:

- All parties have a common understanding of the intent of each user requirement,
- Each user requirement is uniquely-identified, clear and complete, needed, appropriate to implement, and capable of being testing and accepted.
- There are no conflicts between user requirements.
- For existing applications, the analysis should also:
- Ensure that new requirements do not conflict with or “undo” previous requirements or enhancements, and
- Determine if the requirements would have an adverse impact on the application’s current processes and logic; and determine if any existing interfaces will require modifications based on the proposed requirements.

The AASHTO PM and a stakeholder group (TAG or TRT) normally assist in the review as described in *Deliverable Review and Approval* section of Chapter 1.

### 2.3.5.3. Revise URS

If the review and analysis of the URS reveals any conflicts or problems, the contractor and task force should make the appropriate resolutions; which may include the following.

- Modifying the description of certain requirements to clarify the meaning of those requirements and/or to remove conflicts;

- Removing requirements from the URS that are not needed, introduce a serious impact, or cannot be justified;
- Adding requirements to the URS to help resolve a conflict or clarify an issue; and
- Justifying requirements that were added or revised to the URS since the work plan.

### 2.3.5.4. Review and Approve Final URS

*If the URS is revised, the contractor determines the impact of the revisions and both the task force and the contractor shall approve the revisions using a process similar to that described below.*

- The contractor reviews the revised URS and determines if any of the changes will have an impact on the cost, effort, or schedule of the work plan.
- Any significant impact is documented by the contractor and submitted to the task force as a change request.
- The task force approves the change request prior to, or with the approval of the URS.
- If needed, a contract change is submitted and approved by the task force and SCOA.
- The revised URS is reviewed and approved by the task force and, if needed, by a stakeholder group. Refer to the *Deliverable Review and Approval* section in Chapter 1.
- The task force documents and communicates the approval decision of the change request and URS to the contractor.
- The task force may also decide to wait and approve the URS with the Planning and User Requirements Review Gate, which is described next.

The approval of the URS acknowledges:

- That task force or its designee has reviewed, analyzed, and accepted each requirement in the URS, and
- The commitment of both the task force and contractor to implementing all requirements in the URS.

### 2.3.5.5. Create Initial Preliminary Requirements Traceability Matrix

*After the task force and contractor have approved the revised URS, the contractor shall create the initial version of the Preliminary Requirements Traceability Matrix (RTM). All user requirements from the URS are included with backwards references to a source as described above.*

### 2.3.5.6. Submit Planning and User Requirements Review Gate

*If the URS was revised during this phase or if any required components of the work plan were revised or defined during the Project Start-Up sub-phase, the Planning and User Requirements Review Gate is scheduled and initiated as described below:*

- The review gate is scheduled after revisions to the work plan components and URS have been completed.
- The review gate is initiated by the contractor by preparing a Review Gate Approval Request form, which may be found in the *Common Artifacts Standard,* and submitting the form to the task force chair (or designee) and AASHTO PM.

- If the URS and work plan components have not been approved by the task force prior to the review gate, these are submitted and approved with the review gate approval request.
- If this review gate is not scheduled, the contractor continues the project with the development of the system requirements.

### 2.3.5.7. Approve Planning and User Requirements Review Gate

The task force reviews the review gate approval request and the deliverables submitted and determines if they are satisfied that:

- All required work, deliverables and artifacts for this review gate period are complete and; if not, acceptable plans have been provided for incomplete items and for resolving open issues;
- Acceptable justification has been provided for noncompliance with standards; and
- The contractor may proceed with developing the system requirements and functional design.

If satisfied with the above, the task force approves the review gate and the contractor is authorized to proceed with developing the system requirements. If not approved, the contractor shall address task force issues, and resubmit the review gate approval request and the unapproved deliverables.

Refer to the *Review Gate Approval Procedure* section in Chapter 1 for additional information on submitting and approving review gates.

### 2.3.5.8. Define System Requirements

*After the URS is revised and approved, the contractor begins or continues to define the system requirements. The system requirements are documented in one or more documents that are referred to as the System Requirements Specification (SRS). The completed SRS shall be submitted to and approved by the task force.* In a strict waterfall environment, work begins on the systems requirements at this point; however, in many cases, work on the system requirements has already begun.

While the user requirements should define the requirements from a business perspective, the system requirements should define the requirements from a software perspective. The system requirements are derived from the user requirements and define what the proposed product must do to fulfil the user requirements. A system analyst, software developer or integrator typically prepares the system requirements by reviewing, analyzing and decomposing the user requirements into additional requirements that provide additional detail and clarification needed for the design of the proposed product.

Each system requirement must be traceable to a source user requirement. In many cases a single user requirement is broken down into multiple system requirements to better understand and/or estimate the implementation of the original requirement. Some user requirements may define software requirements and/or may be of sufficient detail that no further clarification or expansion is required. In this case, a system requirement should be defined in the SRS that is identical or near identical to its source user requirement in the URS to ensure traceability. The Requirements Traceability Matrix (RTM) documents the traceability between the original (source) user requirements and the derived system requirements.

The SRS should be considered as the final set of requirements and the source of requirements for the functional and technical design.

*As with user requirements, each system requirement shall be understandable, clear, concise, testable; include a unique ID, and have no conflicts with other requirements. Each system requirement shall also be traceable to one or more user requirements in the approved URS.*

*The SRS shall include functional, non-functional, technical architecture, preliminary data, and interface requirements for the proposed system in sufficient detail to support the design of the proposed product. The non-functional requirements of the SRS shall include security, accessibility, user interface, and performance requirements.* Other types of non-functional requirements may also be included in the SRS. The SRS also includes system roles which are normally defined in conjunction with the security requirements.

The approach for preparing and completing the SRS varies based on the type of development methodology used. For projects using a waterfall methodology or a variation of waterfall; the SRS should be detailed, cover the full scope of the project, and support all user requirements in the URS. The *Iterative Project Development Process* describes a Preliminary SRS where an initial set of broad, high level requirements are defined as the first step in developing the system requirements incrementally.

The following subsections describe typical activities used to define the required components of the SRS.

If the SRS was developed in a prior project or MSE effort, these activities are skipped or replaced by a more limited set of activities where the SRS is reviewed and validated, and, if needed, updated using the appropriate activities listed below. If this activity is skipped, the project continues with the update of the preliminary RTM and the review and approval of the SRS. These activities are both described below beginning with the *Update Preliminary RTM* section.

#### 2.3.5.8.1. Define Functional Requirements

The functional requirements for the proposed product should answer the following questions.

- How are inputs transformed into outputs?
- Who initiates and receives specific information?
- What information must be available for each function to be performed?

Functional requirements are defined for all functions (tasks, calculations, services, data manipulations, etc.) that will be automated the system (proposed product). A function is described as a set of inputs, the behavior, and outputs. A functional model or domain model is normally developed to depict each function that needs to be included in the product. The goal of this model is to represent a complete top-down picture of the product.

The behavior of the functions is normally described by use cases. Use cases describe how actors (users, other systems, or devices) and the system interact. The development of functional requirements, the functional model, and use cases will normally overlap. The functional or domain model is a component of the FDS and is described in the *Design Phase* section.

#### 2.3.5.8.2. Define Preliminary Data Requirements

Preliminary data requirements are defined by identifying input and output requirements. All manual and automated input requirements for the product such as data entry from source documents and data extracts from other applications should be identified. In addition, all output requirements for the product are identified such as printed reports, display screens, files, and other applications. Where the inputs are obtained and who or what is to receive the output should also be identified.

Data requirements identify the data elements and logical data groupings that will be stored and processed by the product. The identification and grouping of data begins during this activity and is expanded in subsequent phases as more information about the data is known.

#### 2.3.5.8.3. Define System Interface Requirements

System interface requirements specify hardware and software interfaces required to support the implementation or operation of the proposed or revised product. When defining the system interface requirements, the contractor should consider:

- Existing or planned software that will provide data to or accept data from the product.
- Access needed by the user organizations to the product or by their business partners.
- Common users, data elements, reports, and sources for forms/events/outputs.
- Timing considerations that will influence sharing of data, direction of data exchange, and security constraints.
- Constraints imposed by the proposed development, implementation, and/or operational environments.

#### 2.3.5.8.4. Define User Interface Requirements

The system requirements shall include user interface requirements that describe how the user will access and interact with the product, and how information will flow between the user and the product. The following are some of the items that should be considered when identifying user interface requirements.

- User requirements or project objectives that address the look and feel, navigation, and/or help information.
- Industry standards for user interfaces, existing user interface standards for the specific AASHTOWare product, the user interface used by another product overseen by the same task force, or the user interface used by a product overseen by another task force.
- The types of the users who will access and use the product and the range of work that the users will be performing with the product.

#### 2.3.5.8.5. Define Security Requirements

Security requirements define the extent to which access to the product or data is provided or restricted to various groups of users. These requirements support access requirements defined in the user requirements or those derived during analysis. System roles are defined in conjunction with the security requirements. *The proposed or revised product shall also meet the security requirements defined in the Security Standard*.

#### 2.3.5.8.6. Define Accessibility Requirements

Accessibility requirements define the requirements for access to the product for users with physical disabilities, such vision and hearing disabilities. *The accessibility and compliance reporting requirements the product must meet are defined in the Common Artifacts Standard.*

#### 2.3.5.8.7. Define Performance Requirements

Performance requirements define the required time to perform a specific process or the required volumes of various items that the product must be able to input, output, process, use, etc. Examples are response time, number of concurrent users, amount of data, and hours of operation.

#### 2.3.5.8.8. Define Technical Architecture Requirements

The technical architecture requirements define specific technical requirements or constraints that the proposed or revised product must conform to during development, implementation or operation. These technologies include, but are not limited to, integrated development environments, development languages, run time environments, configuration management and version control software, browser, desktop and server operating systems, testing software, database engines, networking software, and utilities.

#### 2.3.5.8.9. Define Other Non-Functional Requirements

The contractor should define other non-functional requirements as required to support the functional requirements of the proposed or revised product. Non-functional requirements such as communications, disaster recovery, maintainability, portability, reliability, and scalability impose constraints on the design or implementation.

### 2.3.5.9. Prepare SRS

After the initial definition of the system requirements, the contractor should prepare a draft of the SRS deliverable by compiling the functional, preliminary data, system interface, user interface, security, accessibility, performance, technical architecture, and other non-functional requirements. *The required content of the System Requirements Specification (SRS) is described in Chapter 5 of this document.* The SRS may be created as a document, spreadsheet, or other type of digital file as long as the required content is included. The contractor may also choose to combine the content of the SRS with the *Functional Design Specification (FDS)*.

### 2.3.5.10. Update Preliminary RTM

*After preparing the draft SRS, the contractor shall update the preliminary RTM. Each system requirement is entered into the RTM and referenced to its source user requirement.* Multiple system requirements may trace to a single user requirement.

### 2.3.5.11. Review and Approve System Requirements

This section describes the procedure used to review, analyze, and validate the SRS. The SRS may be reviewed separately, as described here, or may be reviewed with the FDS after both deliverables have been completed. Except in the case of a strict waterfall methodology, the development of the SRS and FDS will normally overlap.

#### 2.3.5.11.1. Review and Analyze System Requirements

Before the SRS is completed, the task force and contractor should review and analyze the SRS in the same manner that was used for the URS. As with the URS, the task force may assign a stakeholder group to assist in the review. The current version of the preliminary RTM should also be provided to the task force and stakeholders when reviewing the SRS.

The contractor should conduct facilitated reviews of the system requirements with the task force and/or stakeholder groups to help validate the system requirements. Product representations, such as prototypes, mock-ups, simulations, or storyboards, should be used to assist in the analysis or validation of the system requirements.

Any issues or new requirements discovered during the analysis or validation of the SRS should be documented and reviewed by the task force or stakeholder group and contractor. If the SRS is reworked and resubmitted, the analysis and validation procedures should be repeated. The RTM should be updated to reflect any changes to the SRS.

#### 2.3.5.11.2. Approve SRS

After the task force and stakeholder reviews are completed, the contractor should analyze the impact of the system requirements against the work plan, tasks, deliverables, and other planned artifacts; and report the impact to the task force. The task force may elect to approve the SRS at this time as described in the *Deliverable Review and Approval* section in Chapter 1; or wait to approve the SRS with the *Submit Functional Design Review Gate*.

### 2.3.5.12. Manage Changes to Requirements

Any addition, modification or deletion of a requirement after approval of the URS and SRS should be accomplished and approved following the project's change control procedure. This should include analyzing and reporting the impact of the changes against other requirements, work plan, tasks, deliverables, and other planned artifacts. The Project Work Plan Template describes the requirements of the change control procedure. The template link may be found in the *Common Artifacts Standard*.

### 2.3.5.13. Identify Inconsistencies

The contractor and task force should review the work plan and planned deliverables at key points during the project lifecycle and ensure that there are no inconsistencies with the user or system requirements. Review for inconsistencies should occur with the following occurrences:

- Approval of the User Requirements Specification (URS).
- Approval of the System Requirements Specification (SRS).
- Approval of a change request that adds, modifies, or deletes requirements.
- Work plan changes.

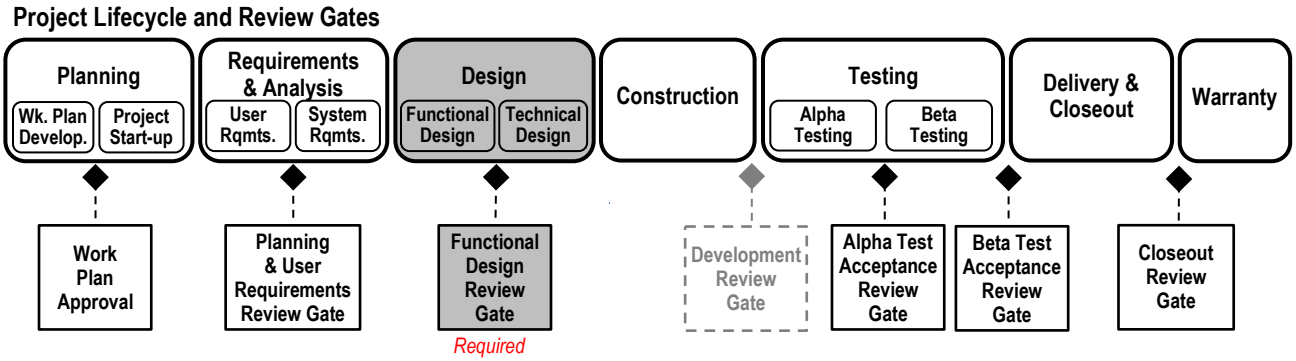
If inconsistencies are found, proposed changes to the work plan, URS, or SRS should be submitted to the task force to address the inconsistencies.

### 2.3.5.14. Continue to Plan, Manage, Monitor and Control

As the Requirements and Analysis Phase is executed and completed; the contractor, task force and AASHTO PM should continue to *Manage, Monitor, and Control the Project*. Key activities involve status reporting, planning activities for next reporting period, issue management, and storing deliverables and artifacts in the project repository.

At the end of this phase, the contractor should review the planned work for the next phase; perform additional planning and revise the project schedule, as required, and begin executing the Design Phase.

# 2.4. Design Phase




## 2.4.1. Phase Overview

The primary objective of the Design Phase is to translate the requirements in the User Requirements Specification (URS) and System Requirements Specification (SRS) into design specifications for the proposed product or product component. Where the requirements define "what the product will do", the design of the product defines "how it will be done."

The Design Phase is divided into two segments or sub-phases; Functional Design and Technical Design.

- During the Functional Design sub-phase, the user and system requirements are translated into a functional design that describes the design of the proposed product or product component using terminology that can be readily reviewed and understood by the task force, technical review teams (TRTs), technical advisory groups (TAGs), and other stakeholders. The functional design is documented in one or more documents referred to as the Functional Design Specification (FDS). In addition, the SRS is normally revised during this sub-phase.
- During the second sub-phase, the design specifications in the FDS are expanded and finalized and the Technical Design Specification (TDS) is created. The TDS represents the final system design and includes precise descriptions of the components, interfaces, and data necessary before coding and testing can begin.

## 2.4.2. Input to the Design Phase

The primary deliverables and artifacts that will be used or referenced in this phase are:

- User Requirements Specification (URS)
- System Requirements Specification (SRS)
- Preliminary Requirements Traceability Matrix (RTM)
- Previously defined Functional Design Specification (FDS) – if included in work plan

Other key items used in this phase are the Project Work Plan; Project Schedule; the work plan procedures, processes and technologies; and Section 508/Accessibility web sites.

## 2.4.3. Output from the Design Phase

The following deliverables and artifacts shall be planned, prepared or updated, submitted, and approved to comply with this standard or the referenced standard.

- System Requirements Specification (SRS) – if revised
- *Preliminary Requirements Traceability Matrix (RTM) - revised*

- Project Test Plan – If not already completed, the Project Test Plan shall be completed by the end of this phase.
- *Functional Design Specification (FDS) – initial or revised*
- *Technical Design Specification (TDS) - initial*
- *Function Design Review Gate Approval Request*
- *Project Schedule – if revised*
- *Project Repository - updated*

### 2.4.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Security Standard* - Used when designing security controls.
- *Critical Application Infrastructure Currency Standard* – Used when upgrading technologies included in an existing product.

### 2.4.5. Procedures

This section defines the major activities that are to be followed by the task force and/or contractor during the Design Phase and the results of those activities.

#### 2.4.5.1. Update and Refine the SRS

The development of the SRS normally continues as the contractor develops the functional design. As the requirements are analyzed and functional design decisions are made, the contractor normally defines additional system requirements and/or modifies existing system requirements. Activities such as preparing and demonstrating prototypes, screen mock-ups, and process diagrams typically lead to new or revised system requirements.

Since the development and refinement of the SRS normally overlaps with the development of the FDS, both are typically completed in the same general time frame. *The SRS and FDS shall both be approved prior to, or with, the Functional Design Review Gate.*

The URS is normally not modified at this point in the lifecycle; however, *if user requirements are added or revised; these shall be included and approved in a revised URS or change request*, and a contract modification may be required.

#### 2.4.5.2. Develop the Functional Design

*During the functional design sub-phase, the user and system requirements are translated into design specifications that define "how the requirements will be implemented" from a user or business perspective. The end result of functional design is the Functional Design Specification (FDS).*

The FDS documents the design of the proposed product using terminology that can be readily reviewed and understood by the task force, technical review teams (TRTs), technical advisory groups (TAGs), and other stakeholders; and should clearly demonstrate that all user and system requirements will be implemented.

The FDS does not need to define the design at the level of detail required for construction and coding of the proposed product. A separate Technical Design Specification (TDS) is created after the FDS is approved. The TDS represents the final system design of the proposed product and includes design specifications that are used by the programmers and system integrators to construct the product.

The next group of subsections describes typical activities performed during functional design. The FDS is prepared by compiling the results of these activities.

If the FDS was developed in a prior project or MSE effort, these activities are skipped or replaced by a more limited set of activities where the FDS is reviewed and validated, and, if needed, the FDS is updated using the appropriate activities listed below. If this activity is skipped, the project continues with the update of the preliminary RTM and the review and approval of the FDS. These activities are described below beginning with the *Update the Requirements Traceability Matrix (RTM)* section.

#### 2.4.5.2.1. Determine the System Structure

The contractor normally begins the design process by performing functional analysis to transform the system requirements into a description of entities or objects. A hierarchical approach is useful for determining the structure and components of the product. System decomposition is one hierarchical approach that divides the system into different levels of abstraction. Decomposition is an iterative process that continues until single purpose components (design entities or objects) can be identified. Decomposition is used to understand how the product will be structured, and the purpose and function of each entity or object.

Several reliable methods exist for performing system decomposition. A method that enables the design of simple, independent entities should be selected. Functional and object-oriented designs are two common approaches to decomposition.

#### 2.4.5.2.2. Identify Design Entities

As described above, design entities result from the decomposition of the system requirements. A design entity is an element or object of the design that is structurally and functionally distinct from other elements and is separately named and referenced. The number and type of entities required to partition the design are dependent on a number of factors, such as the complexity of the product, the design method used, and the development environment. The objective of design entities is to divide the product into separate components that can be coded, implemented, changed, and tested with minimal effect on other entities.

The contractor should perform decomposition activities and identify and document each design entity. Each entity should be described by attributes that define characteristics of the entity such as name, purpose, type, input, output, and transformation rules.

#### 2.4.5.2.3. Identify Design Dependencies

Design dependencies describe the relationships or interactions between design entities at the module, process, and data levels. These interactions may involve the initiation, order of execution, data sharing, creation, duplication, use, storage, or destruction of entities.

The contractor should identify the dependent entities, describe their coupling, and identify the resources required for the entities to perform their function. In addition, the strategies for interactions among design entities should be defined and the information needed to determine how, why, where, and at what level actions occur should be defined.

Dependency descriptions should provide an overall picture of how the product will work. Graphical representations such as data flow diagrams and structure charts are useful for showing the relationship among design entities. The dependency descriptions and diagrams should be useful in planning system integration by identifying the entities that are needed by other entities and that must be developed first.

#### 2.4.5.2.4. Design Content of System Inputs and Outputs

This activity involves identifying and documenting all input data that will be accepted by the proposed product and all output that will be produced. The contractor should involve the task force, TRTs, TAGs and/or other stakeholders in this activity to ensure that their needs and expectations are met.

All types of input that will be accepted by the product should be identified, such as data entered manually into the product, data from documents, records and files, and data that will be imported from other systems. All types of electronic and printed output that will be produced by the product should also be identified; such as records, files, screen displays, printed reports, and data that will be exported to other systems.

In addition, each input and output data element should be documented in the data dictionary described below.

#### 2.4.5.2.5. Design User Interface and Reports

During this activity the contractor should define and document the application user interface and report design. The design should include application menus/User Interface (UI) navigation, input and output screens, reports, system messages, and online help. The contractor should work closely with the task force and/or stakeholders while designing the user interface and reports and should consider the use of prototypes and mock-ups to help communicate the design.

##### 2.4.5.2.5.1. Design Application Menus/UI Navigation

The menu/UI navigation design should describe the look and feel of the application menus or equivalent method used for the user interface navigation; and describe the hierarchy and the navigation through menus and display screens. The design documentation should include tables, diagrams and/or charts to describe the menu hierarchy and the navigation.

##### 2.4.5.2.5.2. Design Display Screens

The display screen design should describe the look and feel and content for the user interface screens for the proposed product. The screen design documentation should also reference the data elements from the data dictionary that are input and output to each display screen. The design should include additional information, as required, to help communicate the screen design to the task force and stakeholders.

##### 2.4.5.2.5.3. Design Reports

Report design should describe the layout and content of each report, data elements in the report, format of the data elements, and any other information needed to help communicate the report design to the task force and stakeholders.

##### 2.4.5.2.5.4. Design System Messages

System messages are the various types of messages that are displayed to users during the use of the proposed system, such as error messages, status messages, and prompt messages. The contractor should define and document the messages for the proposed product, the type and text of each message, and the condition that triggers each message to be displayed.

##### 2.4.5.2.5.5. Design Online Help

The contractor should define and document the online help design to explain the concepts, procedures, messages, menu choices, commands, words, function keys, formats, and other information, as needed. Effective online help information communicates to the users what the product is doing, where they are

in the sequence of screens, what options they have selected, and what options are available.

#### 2.4.5.2.6. Design System Interfaces

The interface design describes how the product will interface with other systems based on the system interface requirements identified in the SRS. The contractor should define and document each interface considering the following issues:

- System inputs and outputs
- Method of interface
- Volume and frequency of data
- Platform of interfacing system
- Format of data
- Automatic or manual initiation of interface
- Need for polling device(s)
- Verification of data exchange
- Validation of data

The interface design should be coordinated with the data identified for import and export in the design of system inputs and outputs and reference data elements in the data dictionary.

#### 2.4.5.2.7. Design System Security Controls

This activity involves designing security controls for the proposed product that support the security and access requirements identified in the SRS. The contractor should perform the following or similar activities when designing the security controls and document the results.

- Identify the types of users that will have access to the product and define the access restrictions for each type of user.
- Identify controls for the product, such as the user identification code for system access and the network access code for the network on which the product will reside.
- Identify whether access restrictions will be applied at the system, subsystem, transaction, record, or data element levels.
- Identify physical safeguards required to protect hardware, software, or information from natural hazards and malicious acts.
- Identify communications security requirements.

*The design of the proposed or revised product shall also meet the security requirements defined in the Security Standard.*

#### 2.4.5.2.8. Develop Logical Process Model

This activity involves the development of a logical process model that describes the flow of data through the proposed system and determines a logically consistent structure for the system. Each module that defines a function is identified, interfaces between modules are established, and design constraints and limitations are described.

A logical process model has the following characteristics:

- Describes the final sources and destinations of data and control flows crossing the system boundary rather than intermediate handlers of the flows.
- Describes the net transfer of data across the system boundary rather than the details of the data transfer.

- Provides for data stores only when required by an externally imposed time delay.

The logical process model should be documented in user terminology and contain sufficient detail to ensure that it is understood by the task force and stakeholders. The contractor should use data flow diagrams or another type of diagram to show the levels of detail necessary to reach a clear, complete picture of the product processes, data flow, and data stores.

#### 2.4.5.2.9. Develop Data Model and Data Dictionary

During this activity the contractor should develop the data model and data dictionary. The data model is a representation of a collection of data objects and the relationships among these objects and is used to provide the following functions:

- Transform the business entities into data entities.
- Transform the business rules into data relationships.
- Resolve the many-to-many relationships as intersecting data entities.
- Determine a unique identifier for each data entity.
- Add the attributes for each data entity.
- Document the integrity rules required in the model.
- Determine the data accesses (navigation) of the model.

The data dictionary should include all data elements in the logical process model and data model and any other data input, created, or output by the proposed system.

The data model and data dictionary should be updated and finalized during the technical design sub-phase.

### 2.4.5.3. Select and Document Initial Technical Architecture

During the development of the functional design, the contractor should begin to identify and analyze alternative technical architectural solutions for developing and implementing the proposed product. When analyzing alternatives, the contractor should review technical architecture solutions that satisfy the technical architecture requirements and constraints in the SRS. These requirements and constraints are typically based on current and planned customer technical environments.

The contractor should also consider current or emerging technologies that may benefit the development, implementation, operation, and/or maintenance of the proposed product. Technologies that could impact the development, implementation, operation or maintenance should also be considered as additional constraints. When considering technologies to include in the technical architecture, the contractor should consider new versions of technology components and be aware of technology that will soon be outdated, as described in the *Critical Application Infrastructure Currency Standard.* In addition, the existing skills of the development team and the availability of reusable components and open source tools should be considered when analyzing technical architecture solutions.

Based on the above analysis, the contractor should recommend the most cost effective technical architectural solution that best satisfies the technical requirements and constraints, supports the user requirements and other system requirements, and satisfies the additional criteria defined by the contractor.

The contractor should document the technical architecture solutions considered, the recommended solution, and the rationale for why the recommended solution was selected. Diagrams are normally used to depict the overall technical architecture, including the system components (software, hardware, networks, databases, operating systems, etc.) that support the proposed product. Interfaces between components should also be shown in the diagrams.

The recommended development tools and other technical tools used for analysis, design, construction and implementation should be identified early in the project lifecycle. These tools should be included in the technical architecture documentation. Applicable development standards should also be referenced in the technical architecture documentation.

The technical architecture is updated and finalized during the Technical Design sub-phase and Construction Phase.

### 2.4.5.4. Prepare Functional Design Specification (FDS)

*After the above functional design activities are completed and the initial technical architecture recommendations are made, the contractor shall prepare the Functional Design Specification (FDS).* The draft FDS is prepared using the functional design information developed for input and output, user interface and reports, system interfaces, security controls, system structure, process model, data dictionary, and data model. The FDS also includes the initial technical architecture recommendations, or the contractor may choose to document the initial technical architecture in a separate document.

*The FDS is developed for the full scope of the project addressing all requirements in the URS and SRS. The level of detail for the FDS shall be sufficient to provide the task force and stakeholders with a clear understanding of how the proposed system will work and ensure that all user and system requirements will be implemented. The required content for the Functional Design Specification (FDS), including the initial technical architecture, is defined in Chapter 5.*

### 2.4.5.5. Update the Requirements Traceability Matrix (RTM)

*After the draft FDS is prepared, the contractor shall update the Requirements Traceability Matrix (RTM) and add references to the design elements in the FDS. Each system requirement shall reference the design element in the FDS that implements the requirement, such as, function, sub function, screen, or report.*

*At this point the RTM shall include (1) all user requirements from the URS with a reference to the source of each requirement; (2) all system requirements from the SRS with a reference to the source user requirement; and (3) references from each system requirement to the design element that implements the requirement. If no system requirements are defined for a user requirement, the user requirement shall reference a design element*.

### 2.4.5.6. Obtain Stakeholder/Task Force Review & Approval of FDS

The FDS should be reviewed by both a stakeholder group (TRT or TAG) and the task force. The Preliminary RTM should be provided with the FDS. Comments and approval recommendations from the reviews should be documented; corrections should be made as required; and, if needed, additional reviews should be scheduled. After the task force completes the review of the FDS, the task force may elect to approve the FDS at this time or wait to approve with the Functional Design Review Gate.

Refer to the *Deliverable Review and Approval* section in Chapter 1 for additional information.

### 2.4.5.7. Submit Functional Design Review Gate

*After the SRS and FDS have been reviewed and completed, the contractor shall prepare the Functional Design Review Gate approval request. The request form shall be completed as described in the Prepare and Submit Review Gate Approval Request section of Chapter 1.*

*If the SRS or FDS have not been approved by the task force prior to the review gate, these shall be submitted and approved with the review gate approval request along with stakeholder recommendation documentation. If already approved, the approval documentation for these deliverables shall be submitted along with the location of the approved deliverables. The Preliminary RTM shall also be provided with this request.*

*The completed and signed review gate approval request and attachments are submitted to the task force chair (or designee) and copied to the AASHTO PM.*

### 2.4.5.8. Approve Functional Design Review Gate

The task force reviews the information provided with the review gate approval request and the deliverables submitted, and determines if they are satisfied that:

- All work activities needed to proceed with the technical design and construction have been completed; and the content in the SRS, FDS preliminary RTM deliverables is complete;
- Each system requirement supports one or more user requirements;
- All user and system requirements have been implemented in the FDS; or an acceptable justification has been provided for requirements that have not been implemented;
- Acceptable justification has been provided for noncompliance with standards; and
- Acceptable plans have been provided for resolving open issues, incomplete tasks or incomplete deliverables.

The preliminary RTM is used to assist in the approval decision. The RTM should show that each system requirement has a source user requirement, and that each user and system requirement is implemented in one or more functional design elements.

*After the review is complete, the task force shall approve or reject the review gate. If the review gate is approved, the contractor is authorized to proceed with the technical design. If rejected, the contractor shall address task force issues, and resubmit the review gate approval request and the unapproved deliverables.*

Refer to the *Review Gate Approval Procedure* section in Chapter 1 for additional information on submitting and approving review gates.

### 2.4.5.9. Develop and Document Technical Design

*During the Technical Design sub-phase, a final set of design specifications are prepared, that are referred to as the Technical Design Specification (TDS). Where the FDS is documented in a functional level of detail that is appropriate to obtain stakeholder and task force understanding and approval of the design for the proposed product, the TDS is documented at the appropriate detail and terminology for the contractor's development staff to construct the proposed product.* The goal should be to finalize the system design and produce an end product of design specifications that represent the blueprint for the Construction Phase.

#### 2.4.5.9.1. Prepare Technical Design Specification (TDS)

The primary users of the TDS are the contractor's project manager, designers, integrators, developers, and testers; therefore, the TDS should describe and document the design in the adequate level of detail and terminology to code, configure, build, integrate, and test the proposed product and all components, programs, databases, files, interfaces, security controls, screens, and reports.

The TDS should also be defined in appropriate detail and terminology for use by technical personnel outside the development team. These uses include future

maintenance and enhancement activities and product customization and integration activities at customer sites.

The format for the TDS is generally left up to the contractor. *The required content for the Technical Design Specification (TDS) is defined in Chapter 5, and the required content for the Data Dictionary is defined in the Common Artifacts Standard.*

As noted above, the TDS may be produced and packaged in any format acceptable to the contractor, as long as the required content is included. For example, the TDS may be created by:

- Updating the FDS with the required TDS content;
- Creating a supplemental set of design specifications that is used in conjunction with the existing FDS; or
- Creating a complete set of design specifications that is independent of the FDS.

*Regardless of the format and the number of documents, the end product shall include all required content for the TDS and all other design documentation that was created by the contractor after the approval of the FDS.*

The contractor may also choose to document and maintain some of the final specifications in the Development and Maintenance Documentation or the Systems Documentation. This documentation is described in the *Delivery and Closeout Phase* section.

#### 2.4.5.9.2. Complete and Submit TDS

The TDS is not a deliverable and does not require task force approval*; however, the TDS is a required artifact*. The TDS should normally be completed by the end of the Design Phase; however, updates may occur during the Construction Phase. The progress and completion of the TDS should be reported in status reports to the task force.

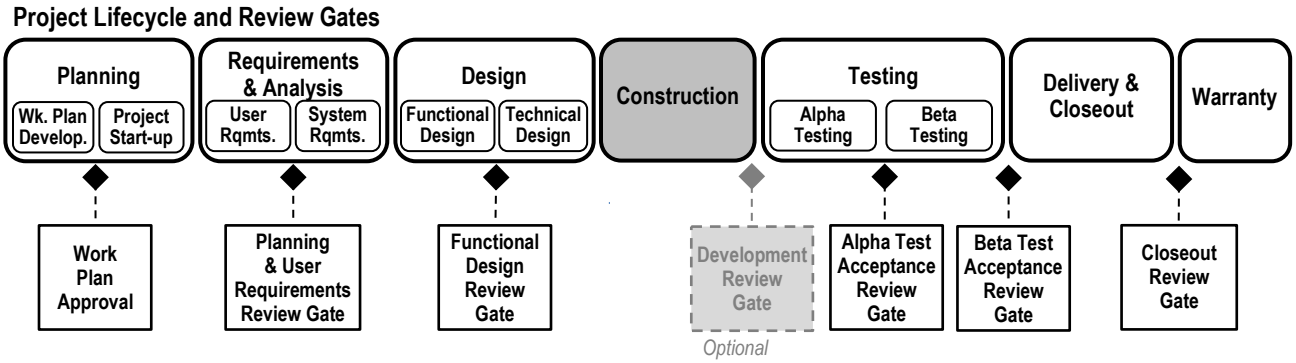
*The TDS shall also be included in the Product Archive Package that is sent to AASHTO at the conclusion of the project.* The Product Archive Package is described in the *Delivery and Closeout Phase* section.

### 2.4.5.10. Continue to Plan, Manage, Monitor and Control

As the Design Phase is executed and completed; the contractor, task force and AASHTO PM should continue to *Manage, Monitor, and Control the Project*. Key activities involve status reporting, planning activities for next reporting period, issue management, and storing deliverables and artifacts in the project repository.

At the end of this phase, the contractor should review the planned work for the next phase (Construction); perform additional planning and revise the project schedule, as required, and begin executing the Construction Phase.

# 2.5. Construction Phase




## 2.5.1. Phase Overview

The goal of the Construction Phase of the project/product release is to develop or build the proposed product using the TDS, supplemented by the FDS, as a blueprint.

Construction involves coding, building data bases and interfaces, validation and unit testing by a developer. Any hardware or software procured to support the construction effort is installed. Plans are developed for the installation of the operating environment hardware and software. In addition, the TDS is normally revised during this phase. No standards have been defined for the construction phase and the process used for construction is left up to the contractor.

The Construction Phase ends after a successful System Test. System testing tests the complete system, ensures that integration is complete, and the system performs as required. System testing ensures the product is complete and ready for formal alpha testing and subsequent acceptance. All test procedures used for alpha testing should be executed to ensure that all user and system requirements are met.

## 2.5.2. Input to the Construction Phase

The primary deliverables and artifacts that will be used or referenced in this phase are:

- User Requirements Specification (SRS)
- System Requirements Specification (SRS)
- Preliminary Requirements Traceability Matrix (RTM)
- Functional Design Specification (FDS)
- Technical Design Specification (TDS)
- Project Test Plan
- Test Procedures

Other key items used in this phase are the Project Work Plan; Project Schedule; and the work plan procedures, processes and technologies.

## 2.5.3. Output from the Construction Phase

The following deliverables and artifacts shall be planned, prepared or updated, submitted, and approved in order to comply with this standard or the referenced standard.

- System Requirements Specification (SRS) – if revised
- Preliminary Requirements Traceability Matrix (RTM) – if revised
- Functional Design Specification (FDS) – if revised
- *Technical Design Specification (TDS) – revised*
- *Test Procedures – revised*

- *Successful System Test*
- *Alpha Test Plan – initial*
- The Development Review Gate Approval (Optional) – Submission of this review gate approval request occurs in those cases when the task force requests the contractor to formally acknowledge that a system test has been completed successfully and the product is ready for Alpha Testing and/or when the task wants to review the Alpha Test plan prior to beginning Alpha Testing.
- *Project Schedule – if revised*
- *Project Repository – updated*

### 2.5.4. Standards Used/Referenced in this Phase

Software Development and Maintenance Process Standard

### 2.5.5. Procedures

This section defines the major activities that are to be followed by the task force and/or contractor during the Construction Phase and the results of those activities.

#### 2.5.5.1. Construct the Product

*The only requirements for construction are completing the construction of the product and completing and verifying a successful system.* Otherwise, the process used for construction is left up to the contractor. Typical construction activities include the following:

- Establish the development technical environment;
- Revise/finalize the TDS;
- If needed, update previously approved deliverables (SRS and TDS) and obtain task force approval;
- Update or complete test procedures;
- Code, validate, and test the individual units of code (routine, function, procedure, etc.);
- Create and test databases;
- If applicable, install, configure, customize, and test off-the-shelf (COTS) applications;
- Build and test interfaces with external systems;
- Integrate developed units, reusable code units, open source programs, COTS applications, etc.; and perform build/integration testing;
- *Build and test the entire system (product);*
- *Evaluate system test results and determine if system is complete and ready for alpha testing;*
- Establish development baselines prior to each system test; and
- Begin developing the Alpha Test Plan.

Unit, build, and system testing are performed and documented as described in the Test Strategy. System testing tests the complete system, ensures integration is complete, and the system performs as required. System testing also ensures the product is complete and ready for formal alpha testing and subsequent acceptance. All test procedures used for alpha testing should be executed to ensure that all user and system requirements are met.

#### 2.5.5.2. Submit and Approve Development Review Gate (Optional)

If the task force and contractor have planned the Development Review Gate, the contractor should submit the review gate approval request after completing a successful system test. This is an optional review gate that is planned when the task force requests the contractor to formally acknowledge that a system test has been completed successfully, meeting all user requirements in the URS, and the product is ready for Alpha Testing. The task force may also schedule the review to review the Alpha Test Plan.

*When the Development Review Gate is planned, the contractor shall complete the Alpha Test Plan as described in the Testing Phase, obtain stakeholder and task force review and approval of the plan, and submit the plan with the review gate approval request. The current version of the Preliminary RTM is also submitted with this review gate.*

Refer to the *Deliverable Review and Approval* section and the *Review Gate Approval Procedure* section in Chapter 1 for additional information on submitting and approving deliverables and review gates.

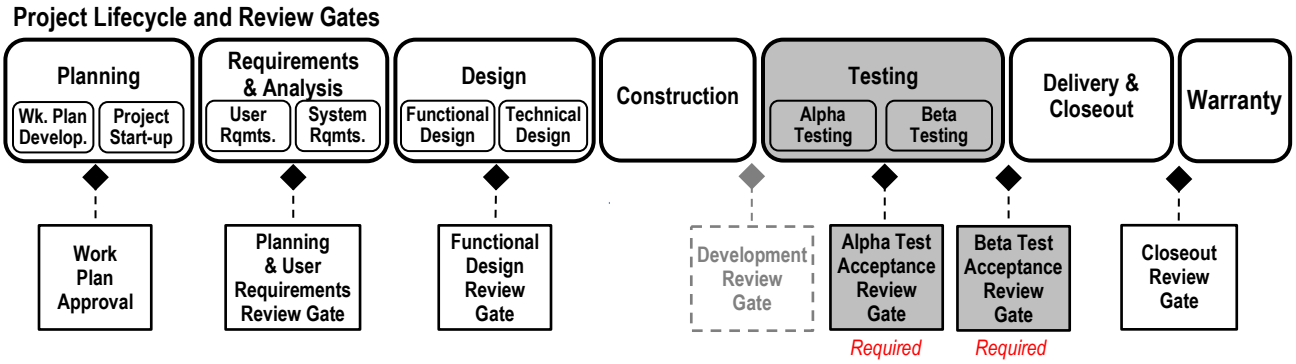
If this review gate is not planned, the Alpha Test Plan is completed during the Testing Phase and submitted with the Alpha Test Acceptance Review Gate

#### 2.5.5.3. Continue to Plan, Manage, Monitor and Control

As the Construction Phase is executed and completed, the contractor, task force and AASHTO PM should continue to *Manage, Monitor, and Control the Project.* Key activities involve status reporting, planning activities for next reporting period, issue management, and storing deliverables and artifacts in the project repository.

At the end of this phase, the contractor should review the planned work for the next phase (Testing); perform additional planning and revise the project schedule, as required, and then begin executing the Testing Phase.

# 2.6. Testing Phase




## 2.6.1. Phase Overview

The Testing Phase is subdivided into two sub-phases, Alpha Testing and Beta Testing.

- During the first sub-phase, alpha testing is performed which covers the same scope as system testing and uses the same test procedures. The emphasis is on breaking the system, checking user and system requirements, and reviewing all documentation for completeness by using the application as if it were in production.
- During the second sub-phase, beta testing is performed. The purpose of beta testing is to confirm to the user/tester that all functionality and operability requirements are satisfied, the system will operate correctly in the user's environment, and the system is ready for delivery and implementation. Beta testing also includes the review and validation of all documentation and procedures.

## 2.6.2. Input to the Testing Phase

The primary deliverables and artifacts that will be used in this phase are:

- System/Alpha Test Procedures
- Alpha Test Plan
- User Requirements Specification (URS)
- Systems Requirements Specification (SRS)
- Preliminary Requirements Traceability Matrix (RTM)

Other key items used in this phase are the Project Work Plan; Project Schedule; and the work plan procedures, processes and technologies.

## 2.6.3. Output from the Testing Phase

The following deliverables and artifacts shall be planned, prepared or updated, submitted, and approved in order to comply with this standard or the referenced standard.

- *Alpha Test Plan - revised*
- *Alpha Test Results*
- *Requirements Traceability Matrix (RTM) - final*
- *Beta Test Materials – Includes the Beta Test Plan and Beta Test Installation Package*
- *Alpha Test Acceptance Review Gate Approval Request*
- *Agency Beta Test Results Report(s)*
- *Beta Test Results Report*
- *Product Installation Package*
- *Beta Test Acceptance Review Gate Approval Request*

- *Project Schedule –if revised*
- *Project Repository - updated*

### 2.6.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Common Artifacts Standard*

### 2.6.5. Procedures

The procedures described below define the activities that are to be followed by the task force and/or contractor during this phase.

#### 2.6.5.1. Prepare Alpha Test Plan

*Prior to beginning alpha testing, the contractor shall complete the Alpha Test Plan. This plan includes:*

- *The test procedures that will be used during Alpha Testing.* Each test procedure should include references to the user and system requirement(s) that are tested by the procedure, and the expected results for the procedure.
- *The format for recording the test results, exceptions discovered, and the resolution to exceptions.*
- *Activities to validate installation at user sites or hardware outside of the contractor development environment.* The intent of this requirement is to include sufficient platform/environment installation testing to minimize the chance of installation errors during beta testing.

*The complete content of the Alpha Test Plan is described in Chapter 5.* The Alpha Test Plan is referred to in this checklist as a distinct document; however, the required content may also be included in another deliverable, such as the Project/Product Test Plan.

*Alpha testing shall be performed in a similar test environment to the technical environment that will be used in production.* If possible, alpha testing should be performed all approved platforms for the product. *Planning for alpha testing includes establishing the technical environment for alpha testing. The contractor is responsible for establishing and maintaining the appropriate testing environment.*

The scope of alpha testing (complete system, specific components, specific enhancements, etc.) and the Alpha Test Plan is defined in the Project Test Plan and/or the “Test Plan” section of the project work plan.

#### 2.6.5.2. Update Requirements Traceability Matrix

*After completing the Alpha Test Plan, the contractor shall update the Requirements Traceability Matrix (RTM) and include a reference to all alpha test procedures. Each system requirement shall reference the test procedure that is used to test and accept the requirement.* Each user requirement should trace forward to one or more system requirements in the RTM; however, *if no system requirements are defined for a user requirement, that user requirement shall reference a test procedure in the RTM.*

#### 2.6.5.3. Obtain Stakeholder/Task Force Review of Alpha Test Plan

The Alpha Test Plan does not require formal task force approval; however, it is recommended that the task force or stakeholder group review the plan and provide informal approval to the contractor prior to alpha testing.

*The Alpha Test Plan shall be included or referenced in the submission of the Alpha Test Acceptance Review Gate. If the Development Review Gate is scheduled, as described*

*in the Construction Phase, the Alpha Test Plan is included with the submission for this review gate.*

### 2.6.5.4. Perform Alpha Testing and Review Test Results

*During alpha testing, the contractor and/or stakeholder group runs each test procedure in the Alpha Test Plan, documents the test results in the Alpha Test Results Report, compares the results with the expected results, and notes the problems found.*

*After completing alpha testing and documenting the results, the contractor analyzes the problems found, determines which problems are valid, determines a recommended resolution for each valid problem, and notes the reason that the other problems are not considered valid or critical. The valid problems are corrected and retested; and the Alpha Test Results Report is updated appropriately.* "To Be Determined" may be used for proposed resolution, if the resolution/action is not known at the time of the report submittal. In this case, a timeframe for resolution should be provided.

*The required content for the Alpha Test Results Report is defined in Chapter 5.*

The Alpha Test Results Report should be reviewed by both a stakeholder group (TRT or TAG) and the task force. The primary focus of the task force and TAG/TRT reviews should be on the problems found during testing, the resolution to the problem, and the problems determined not to require resolution. Follow-up corrections and retesting may result from these reviews.

After reviews and follow-up corrections have been completed, the task force may choose to approve the Alpha Test Results Report at this time or wait to approve the report with the Alpha Test Acceptance Review Gate.

### 2.6.5.5. Prepare and Review Beta Test Materials

*Prior to submitting the Alpha Test Acceptance Review Gate form, the contractor shall prepare for Beta Testing. The contractor begins by preparing the Beta Test Materials, which includes two component deliverables, the Beta Test Plan and the Beta Test Installation Package.*

The Beta Test Plan includes all procedures and instructions needed to plan, prepare, execute, and report progress for beta testing. The Beta Test Installation Package contains all procedures, scripts, executables, and documentation needed to install, implement, and operate the beta product at the beta test site.

The scope of beta testing (complete system, specific components, specific enhancements, etc.) and the Beta Test Materials is defined in the Project Test Plan and/or the "Test Plan" section of the project work plan.

*The required content of the Beta Test Materials is described in Chapter 5.*

The contractor should also recommend potential agency testers for beta testing and prepare an invitation to send to the testing agencies. A primary goal of the selection of test agencies is to validate the system in all intended environments.

After the contractor has completed the Beta Test Materials, the stakeholder group (TAG or TRT) and the task force should review and comment on the materials. The list of candidate agency testers and the invitation should also be provided for review.

After the reviews and follow-up corrections have been completed, the task force may choose to approve the Beta Test Materials at this time or wait to approve the materials with the Alpha Test Acceptance Review Gate.

The task force may also choose to approve and send out the Beta Test Plan earlier than the Beta Test Installation Package to allow the beta test sites to plan and prepare for beta testing.

### 2.6.5.6. Submit Alpha Test Acceptance Review Gate

*After the Alpha Test Results Report and the Beta Test Materials are both reviewed and completed, the contractor shall prepare the Alpha Test Acceptance Review Gate approval request.*

*If the Alpha Test Results Report and Beta Test Materials have not been approved by the task force prior to the review gate, these are submitted and approved with the review gate approval request along with stakeholder recommendation documentation. If already approved, the approval documentation for these deliverables is submitted along with the location of the approved deliverables.*

*The final version of the RTM is also submitted with the review gate approval request. In addition, the beta test invitation and list of candidate beta testers are included or referenced.*

*The completed and signed review gate approval request and attachments are submitted to both the task force chair (or designee) and the AASHTO PM.*

### 2.6.5.7. Approve Alpha Test Acceptance Review Gate

The task force reviews the review gate approval request and the deliverables and information provided, and determines if they are satisfied that:

- Construction has been completed;
- Alpha testing has been completed and the content in the Alpha Test Results Report and RTM are complete;
- Acceptable plans have been provided for resolving open issues, incomplete tasks or incomplete deliverables.
- All requirements have been implemented in the product and tested; or an acceptable justification has been provided. (The RTM should provide a link between each requirement and the test procedure used to validate the requirement.);
- Acceptable justification has been provided for noncompliance with standards;
- The Beta Test Materials have been completed and the product is ready for beta testing.
- The appropriate user platforms will be included in the beta test.

*After the review is complete, the task force approves or rejects the review gate. If the review gate is approved, the contractor is authorized to proceed with the beta testing. If rejected, the contractor addresses task force issues, and resubmits the review gate approval request and the unapproved deliverables.*

Refer to the *Review Gate Approval Procedure* section in Chapter 1 for additional information on submitting and approving review gates.

### 2.6.5.8. Perform Beta Testing and Review Results

Beta Testing occurs during the Beta sub-phase of the Testing Phase. The purpose of beta testing is to confirm to the user/tester that all functionality and operability requirements are satisfied, the system will operate correctly in the user's environment, and the system is ready for delivery and implementation. Beta testing also includes the review and validation of all documentation and procedures.

*To prepare for beta testing, the contractor or task force sends the beta test invitations to the agencies requesting their participation in beta testing.* Each agency tester assesses the invitation and decides whether to commit to beta test participation. The contractor then determines whether all the intended environments are represented. If not, the contractor confers with the task force to determine if additional sites are needed or whether to proceed with the existing sites.

*After the Alpha Test Acceptance Review Gate is approved and the selection of beta test sites is complete, the contractor or task force distributes the Beta Installation Package or hosted beta site authentication credentials to the beta test participants. The Beta Test Plan is sent, if not sent previously or if changes have been made.*

After receiving the Beta Test Plan, which may have occurred earlier, each beta test agency staff should begin by preparing a local beta test plan following the instructions in the documentation. The initial tasks in the local test plan should include planning and establishing the required technical infrastructure; and identifying and obtaining a commitment from the business and technical staff required to setup and support the beta test environment and execute the beta test.

After establishing the technical environment, the beta test agency staff installs the system using the Beta Test Installation Package and reports any installation problems to the contractor. *The contractor shall resolve any installation problems and redistribute all materials if necessary.*

After the successful completion of the installation, the beta tester(s) performs each of the beta test procedures that were included in the Beta Test Plan. The beta tester(s) should then perform additional test procedures defined by the testing agency. These normally include “Day-in-the-Life” tests that simulate normal business activities using the beta system. These test procedures, expected results, and the results of testing should be documented in the Agency Beta Test Results Report.

The beta tester(s) then compare the test results of the performed procedures with the expected results and records exceptions found during testing. Any other results that appear to be errors or inconsistencies should also be noted.

After beta testing is complete, each agency returns the Agency Beta Test Results Report to the contractor. Exceptions and errors may also be reported to the contractor as they are discovered to expedite their correction.

*When the contractor receives the results, exceptions, and errors from an agency beta tester, the following activities are performed:*

- *Validate the problems and errors reported; and remove or note those that are determined not to be problems or errors based on the requirements of the system.*
- *Discover any additional problems missed by the beta tester and record these in the Agency Beta Test Results Report.*
- *Combine all Agency Beta Test Results Reports into a single report, which contains all results and validated problems and errors. The required content of the Beta Test Results Report is described in Chapter 5.*

*After the contractor has reviewed all beta test results and the exceptions and errors are validated, the contractor makes the appropriate corrections and adds a brief description of each resolution to the Beta Test Results Report.* “To Be Determined” may be used, if the resolution/action is not known at the time of the report submittal. In this case, a timeframe for resolution should be provided.

After the Beta Test Results Report has been completed, the stakeholder group (TAG or TRT) and task force should review and comment on the report. The primary focus of these reviews is to determine if:

- The product has been thoroughly tested in the appropriate user environments,
- All problems found have been resolved, and
- The product is ready for implementation.

After the reviews and follow-up corrections have been completed, the task force may choose to approve the Beta Test Results Report at this time or wait to approve the materials with the Beta Test Acceptance Review Gate.

### 2.6.5.9. Finalize Product Installation Package

*Prior to submitting the Beta Test Acceptance Review Gate approval request, the contractor completes the Product Installation Package. The Product Installation Package contains all procedures, executables, and documentation needed to install, implement, and operate the product at the customer site. Refer to the Product Installation Package in Chapter 5 for the required content of this deliverable.*

The Product Installation Package is initially created as the Beta Test Installation Package and is normally corrected or refined during and/or after beta testing. After beta testing is completed, the contractor makes the final adjustments to the Product Installation Package. The task force group may choose to approve this deliverable at this time or wait to approve the deliverable with the Beta Test Acceptance Review Gate.

### 2.6.5.10. Submit Beta Test Acceptance Review Gate

*After the Beta Test Results Report and Product Installation Package have been reviewed and completed, the contractor prepares the Beta Test Acceptance Review Gate Approval Request. At this point, the product has been successfully beta tested and the product is ready for distribution to the user organizations.*

*If the Beta Test Results Report and Product Installation Package have not been approved by the task force prior to the review gate, these are submitted and approved with the review gate approval request along with stakeholder recommendation documentation. If already approved, the approval documentation for these deliverables is submitted along with the location of the approved deliverables.*

*The contractor also drafts the cover letter that will be used to transmit the package to all licensees and provide the letter to the task force with the review gate approval. The completed and signed review gate approval request and attachments are submitted to both the task force chair (or designee) and the AASHTO PM.*

### 2.6.5.11. Approve Beta Test Acceptance Review Gate

The task force reviews the information provided with the review gate form and the deliverables submitted, and determines if they are satisfied that:

- All beta testing activities have been completed;
- The Beta Test Results Report and Product Installation Package have been completed;
- All requirements have been tested and implemented in the final product; or an acceptable justification has been provided for requirements that have not been tested or implemented;
- Acceptable justifications have been provided for areas of noncompliance with standards;
- Acceptable plans have been provided for resolving open issues; and
- The product is ready for delivery to the licensees and no further beta testing is needed.

*After the review is complete, the task force approves or rejects the review gate. If the review gate is approved, the contractor is authorized to proceed with the distribution of the product. If rejected, the contractor addresses task force issues, and resubmits the review gate approval request and the unapproved deliverables.*

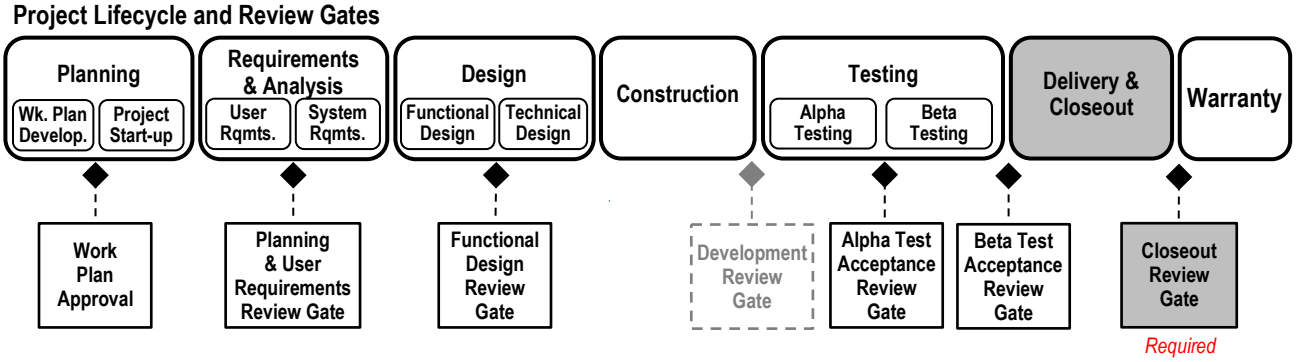
Refer to the *Review Gate Approval Procedure* section in Chapter 1 for additional information on submitting and approving review gates.

### 2.6.5.12. Continue to Plan, Manage, Monitor and Control

As the Testing Phase is executed and completed, the contractor, task force and AASHTO PM should *Manage, Monitor, and Control the Project*. Key activities involve status reporting, planning activities for next reporting period, issue management, and storing deliverables and artifacts in the project repository.

At the end of this phase, the contractor should review the planned work for the next phase; perform additional planning and revise the project schedule, as required, and then begin executing the Delivery and Closeout Phase.

# 2.7. Delivery and Closeout Phase




## 2.7.1. Phase Overview

The goal of this phase is to deliver the product to the customer sites for implementation and to formally close out the project.

## 2.7.2. Input to the Delivery and Closeout Phase

The primary deliverables and artifacts that will be used in this phase are:

- Product Installation Package (or Beta Installation Package)
- ACR (if existing)
- Application Infrastructure Component List (if exists)
- All deliverables and artifacts prepared during the project

Other key items used in this phase are the Project Work Plan; Project Schedule; and the work plan procedures, processes and technologies.

## 2.7.3. Output from the Delivery and Closeout Phase

The following deliverables and artifacts shall be planned, prepared or updated, submitted, and approved to comply with this standard or the referenced standard.

- *Completed product - final*
- *Product Installation Package – final*
- *Accessibility Conformance Report (ACR) - final*
- *Application Infrastructure Component List - final*
- Development and Maintenance Document (if created updated)
- *Technical Design Specification (TDS) - final*
- *Project Archive Package*
- *Closeout Review Gate Approval Request*
- *Project Repository - updated*

## 2.7.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Common Artifacts Standard*
- *Critical Application Infrastructure Currency Standard* – Used when preparing or updating the Application Infrastructure Component List.
- *Product Naming Conventions Standard* – Used to ensure that product names, terminology, branding, icons, release numbers, and splash screens are correct.

## 2.7.5. Procedures

The procedures described below define the activities that are to be followed by the task force and/or contractor during this phase.

### 2.7.5.1. Distribute the Product and Provide Support

*After the Beta Test Acceptance Review Gate is approved, the contractor begins distributing the Product Installation Package* or hosted site authentication credentials *to all licensed customer sites and begins providing production support.*

After receiving the installation package, each customer site should install the product and report to the contractor any problems encountered. The contractor and representatives from the customer organizations should work to resolve these problems.

*The contractor provides routine status reports to the task force regarding the status of customer implementations, any installation problems reported, and the efforts taken by the contractor to resolve the problems. Any other problems reported are also included in the status reports. Serious problems shall be reported to the task force immediately.*

### 2.7.5.2. Prepare/Update ACR

The Voluntary Product Accessibility Template®, or VPAT® is a tool used to document a product's conformance with the accessibility standards under Section 508 of the Rehabilitation Act and the Web Content Accessibility Guidelines (WCAG). The completed VPAT template results in the Accessibility Conformance Report (ACR). Refer to the Accessibility Conformance Report (ACR) section of the *Common Artifacts Standard* for compliance and reporting requirements.

After the ACR has been completed, contractor should provide the ACR to the stakeholder group and/or task force for review and comment.

### 2.7.5.3. Prepare/Update Application Infrastructure Component List

The Application Infrastructure Component List includes the name of each application infrastructure component, the version of the component, and the owner/vendor of the component. Refer to the *Critical Application Infrastructure Currency Standard* for the required content of this list.

*For a new product or a redeveloped product, the contractor prepares the Application Infrastructure Component List before closing the project. For existing products, the contractor reviews the existing Application Infrastructure Component List and makes the appropriate revisions to the list.*

After the list has been completed, the contractor should provide the list to the stakeholder group and/or task force for review and comment.

### 2.7.5.4. Prepare/Update Development and Maintenance Document

*For projects that develop a new product or redevelop an existing product, the contractor shall prepare Development and Maintenance Documentation. This documentation, supplemented by the Technical Design Specification, represents the internal documentation for the product, and describes the logic used in developing the product and the system flow to help the development and maintenance staffs understand how the programs fit together. The documentation also provides instructions for establishing the development environment and should enable a developer to determine which programs or data may need to be modified to change a system function or to fix an error.*

*If the project revises an existing product, and Development and Maintenance Documentation exists, the existing documentation shall be updated.*

#### 2.7.5.5. Prepare Project Archive Package

After the product is distributed, the above deliverables and artifacts are completed, and all installation problems have been resolved satisfactorily; the task force notifies the contractor to begin closeout.

*After the contractor is notified to begin closeout, the contractor shall prepare the Project Archive Package. This deliverable is an archive of the final product, project materials, and development artifacts. The required content of the archive is listed with the Project/MSE Archive Package in Chapter 5.*

#### 2.7.5.6. Submit Closeout Review Gate

*After the Project Archive Package has been completed, the contractor prepares the Closeout Review Gate approval request. At this point, all activities associated with the project have been completed (other than warranty work) and the Project Archive Package and ACR are ready to send to AASHTO. New and updated versions of the ACR and Application Infrastructure Component List are submitted with the request or the location is included in the request. The request also includes the location of the Product Archive Package and TDS. Areas where these deliverables and artifacts do not comply with AASHTOWare standards, planned resolutions to open issues, and any requirement not implemented in the final product are also reported with the approval request.*

#### 2.7.5.7. Approve Closeout Review Gate

The task force reviews the information provided with the review gate form and determines if they are satisfied that:

- The Product Archive Package, Accessibility Conformance Report (ACR), Application Infrastructure Component List, and TDS have been completed;
- All other required deliverables and artifacts have for the project are complete;
- All requirements in the URS have been implemented in the final product; or an acceptable justification has been provided for requirements that have not been implemented;
- Acceptable justifications have been provided for areas of noncompliance with standards; and
- Acceptable plans have been provided for resolving open issues.
- All project activities other than warranty work have been completed; and project is ready to be closed.

*After the review is complete, the task force approves or rejects the review gate. If the review gate is approved, the contractor ships the archive package and the approved review gate approval request to the AASHTO PM and terminates all project activities. If rejected, the contractor addresses the reasons and task force directions and resubmits the review gate approval request and the unapproved deliverables.*

Refer to the *Review Gate Approval Procedure* section in Chapter 1 for additional information on submitting and approving review gates.

With the approval of the Closeout Review Gate, the product warranty period will begin. The product released at closeout is the initial release and may be replaced later by a warranty release following the completion of warranty work. Other than warranty work, all other revisions and additions to the product will occur under annual MSE work or a new project.

# 3. Maintenance, Support and Enhancement Process

## 3.1. Introduction

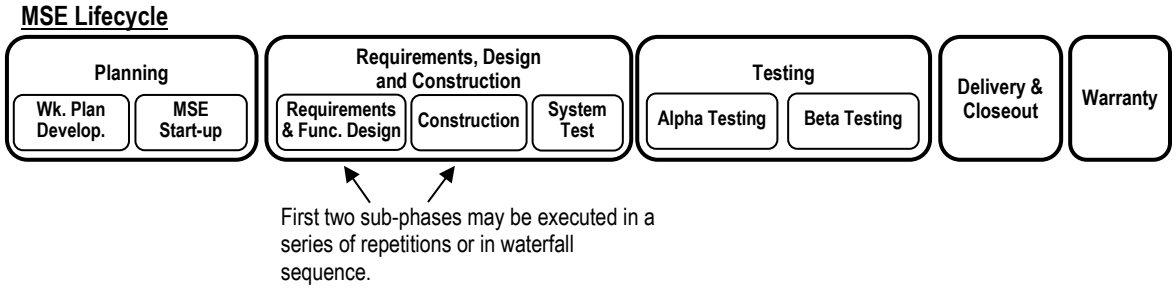
The Maintenance, Support and Enhancement (MSE) Process defines the standard process that shall be used by task forces, contractors, and other AASHTOWare stakeholders when planning and executing the annual Maintenance, Support and Enhancement (MSE) work on an existing AASHTOWare product.

The volume and complexity of the work effort in an MSE work plan should not be so great or complex that all work cannot be completed within the applicable fiscal year. As a guideline, the majority of the enhancements in a work plan should be of a small or medium size and the number of large enhancements should be limited. In addition, major changes to the existing product's application or technical architecture should not normally occur during an MSE work effort. The *Project/Product Determination* section in Chapter 1 provides guidance for determining when a project should be used for work on an existing product.

### 3.1.1. Chapter Organization

This process only applies to MSE work and is written as a companion process to the *Project Development Process* described in Chapter 2. To minimize duplication, this chapter primarily focuses on the differences between MSE work and projects. As with the project development process, this chapter is organized around the phases of the lifecycle model and includes a section for each phase.

Since maintenance work and enhancement development do not normally require the same level of analysis and design as the development of new software, the standard MSE lifecycle combines the Requirements & Analysis, Design, and Construction activities into a single Requirements, Design and Construction Phase with three sub-phases. The first two sub-phases may be executed in a series of repetitions (iterations) or in waterfall sequence.




Following this Introduction section is a section for each phase of the MSE lifecycle model. Each of the phase sections incudes:

- The deliverables and artifacts that are prepared during the phase;
- The deliverables and review gates that are approved during the phase;
- The other standards that are used during this phase; and
- The procedures to be followed during the phase.

### 3.1.2. Review Gates, Deliverables and Artifacts

In addition to the lifecycle model, an MSE work effort also includes differences in the review gates, deliverables, and artifacts. The key differences are summarized below:

- The MSE work plan has much of the same content; however, there is specific content that is unique to MSE work. The work plan differences are summarized below:
- The work plan is prepared with a different work plan template.

- The URS for the work plan is normally defined as a list of requested enhancements.
- Maintenance services and minor technology upgrade services that will be performed to address defects, problems, out-of-date components and minor improvements with the existing product(s) are included.
- Planned upgrades and testing of application infrastructure components are included and the current Application Infrastructure Component List is included.
- Sections describing time and materials work, service unit work, and other type of work to be performed on the existing product(s) are included.
- In most cases, the majority of the information in the Technical Process and Technologies, Monitoring and Control, Quality Management, Communication Management, Configuration Management and Project Repository, Risk Management, and Backup and Recovery sections is initially defined in a prior project or MSE work plan. These will normally be referenced in the current work plan or may be included with revisions.
- The system requirements and functional design specifications for MSE work are normally documented together in one or more enhancement system requirements and design deliverables where a deliverable is created for each medium and large enhancement or a group of enhancements, or all enhancements. This type of deliverable is referred to as an Enhancement FDS or Enhancement System Requirements and Design Specification (SRDS).
- The FDS/SRDS deliverables are created during Requirements & Functional Design sub-phase of the Requirements, Design and Construction Phase.
- The Requirements & Functional Design and Construction sub-phases may be executed iteratively in a series of repetitions or as single sub-phases in a waterfall sequence.
- Each FDS or SRDS deliverable must be approved prior to construction of the enhancements covered by the FDS or SRDS by deliverable approval or by review gate approval.
- The options for executing the Requirements, Design and Construction Phase or described in the *Requirements, Design & Construction Phase* section of this chapter.
- The specific development approach for each MSE effort is defined in the work plan.
- No system requirements and design deliverable is required for small enhancements (those requiring little effort) and maintenance work.
- MSE work does not require an RTM, TDS, or new Development and Maintenance Documentation to be prepared.
- An MSE work plan may include multiple software releases. In these instances, a Release Review Gate must be approved to release the software and close work for that release. The MSE lifecycle begins for the next release. If the MSE work plan will close after a software release, the Closeout Review Gate must be approved.

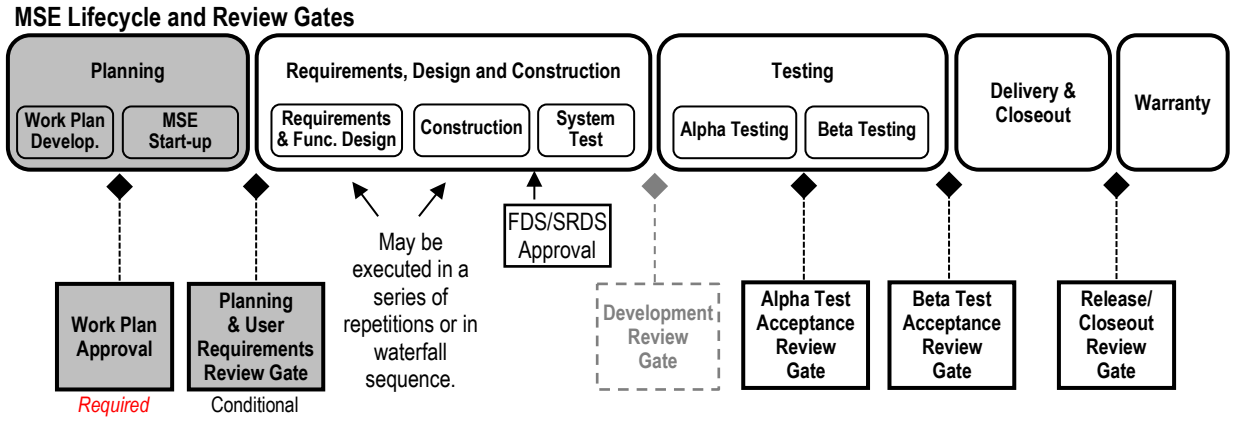
The following diagram provides a summary view of the standard MSE review gates and the required deliverables and artifacts associated with each review gate in relationship to the MSE lifecycle.

## MSE Lifecycle Phases and Review Gates with Required Deliverables and Artifacts




- The Planning & User Requirements, Development, Alpha Test Acceptance Testing, and Beta Test Acceptance, and Closeout review gates are the same as for projects. A Release Review Gate is required if the software release doesn't complete the MSE work plan.
- Each FDS or SRDS contains the system requirements and functional design for one or more of the enhancements (medium and large enhancements).
- @The Function Design review gate is not required but is the recommended approval method for the single waterfall sequence FDS/SRDS approval.
- +The Alpha Test Plan may be included as a section in the Product Test Plan.
- No FDS/SRDS is required for small enhancements or maintenance activities; however, some level of type of design may be required by the task force.
- No RTM, TDS, or Development and Maintenance Documentation are required for MSE work efforts.
- The current Application Infrastructure Component List is included in the work plan and is updated at the end of the work effort.
- #The Product Test Plan describes the testing methodology, test phases and test deliverables. It is recommended to complete the Product Test Plan during MSE Start-up but may be delayed until prior to the first Construction sub-phase and formally submitted the next review gate.

# 3.2. Planning Phase




## 3.2.1. Phase Overview

As with the project lifecycle, the Planning Phase is the first phase in the lifecycle of an MSE work effort and is divided into two sub-phases; with work plan preparation and approval in the first sub-phase; and formal start-up, planning and mobilization in the second sub-phase.

## 3.2.2. Input to the Planning Phase

The primary deliverables and artifacts that will be used or referenced in this phase are:

- AASHTOWare MSE Work Plan Template;
- Prioritized lists of requested enhancements, existing problems, and technology upgrades needed; and
- Existing task force/product management, monitoring and control procedures.

Other key items used in this phase are the AASHTOWare Policies, Guidelines, and Procedures (PG&P), AASHTOWare Project/Product Task Force Handbook, and internal AASHTOWare procedures. The PG&P and Task Force Handbook are available for download on the AASHTOWare web server at:

*https://www.aashtoware.org/wp-content/uploads/2021/07/Policies_Guidelines_Procedures-June-2021.pdf*

*https://www.aashtoware.org/wp-content/uploads/2018/03/TaskForceHandbook-October2009.pdf*

In some cases, there may also be existing user requirements, system requirements, and/or functional design specifications that will be used to develop one or more requested enhancements.

## 3.2.3. Output from the Planning Phase

The following artifacts and deliverables are created or updated during this phase of the MSE work effort.

- *MSE (Product) Work Plan*
- Work Plan Components – The work plan includes sections and sub-sections that shall be completed in the approved work plan, and it also includes other parts that may be completed after the project is formally started. *If any of the following work plan components are not included or not complete in the work plan, the work plan shall define*

*the plan to prepare or revise and approve the incomplete components as deliverables during the execution of the work plan.*

- Enhancements to be implemented, problems to be corrected, and technology upgrades to be performed.
- Requirements and specifications for enhancements
- Application Infrastructure Upgrade Services
- Technical Process and Technologies
- Project Management, Monitoring and Control Procedures
- Communication Management Approach
- Configuration Management and Project Repository Approach
- Risk Management Approach
- Backup and Disaster Recovery
- Planned Deliverables, Review Gates and Milestones

- Planning and User Requirements Review Gate Approval Request – Required when the enhancements or another component of the work plan is revised during Project-Start-Up.
- *MSE Schedule/Work Breakdown Structure - initial*
- *Project Repository*

### 3.2.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Common Artifacts Standard*
- *Critical Application Infrastructure Currency Standard* – Used when developing the Application Infrastructure Upgrade Services section of the work plan, planning upgrade work activities for an existing product, and planning work to develop or update the Application Infrastructure Component List.
- *Quality Assurance Standard* – Used when developing the Quality Assurance Reviews section of the work plan and planning QA work activities, including participating in the annual QA meeting.
- *Backup and Disaster Recovery Standard* – Used when developing the Backup and Disaster Recovery Plans for the work plan and planning backup and disaster recovery activities.

### 3.2.5. Procedures

This section defines the project planning and project management activities that are to be followed by the task force and/or contractor during the Planning Phase and the results of those activities.

#### 3.2.5.1. Develop and Approve Work Plan

During the Work Plan Development sub-phase, the MSE work plan is prepared and approved. *The AASHTOWare MSE Work Plan Template is a Microsoft Word template that is used to create an MSE work plan.* The template includes all the required information that shall be included for each MSE work plan and contains instructions on how the template is to be used and completed. The URL for downloading both the template is included in the *Common Artifacts Standard*.

As described above, the MSE work plan has several sections that are specific to MSE work efforts. The template includes instructions on how the MSE-specific sections are to be completed, as well as, those sections which are the same as those for projects.

*After the contractor prepares all sections of the work plan, the completed work plan is reviewed and approved by both the task force and SCOA.* The work plan is approved in the same manner as described in the *Approve Work Plan* section in Chapter 2.

### 3.2.5.2. Perform MSE Start-Up Activities

The Project Start-Up sub-phase begins with the formal start-up of the contract work for the MSE work effort; and includes the planning and mobilization activities that occur prior to beginning the analysis, development, testing, and implementation activities for the MSE work. The contractor performs the following activities in the same manner as described in the *Perform Project Start-Up Activities* section projects in Chapter 2.

- Review work plan
- Plan work for first month
- Plan work through current phase and first review gate
- *Prepare MSE schedule*
- Execute plan for first month and remainder of Project-Start-Up – This step is the execution of the work planned in the prior activities which should include:
  - Prepare or revise any of the required work plan components that were not included or completed in the approved work plan. The Product Test Plan is recommended to be completed during this phase; however, it may also be completed in the next phase prior to construction.
  - Review and approve the new/revised work plan components as deliverables or as a revision to the work plan.
  - Implement and set up the technologies and procedures. Establish the project repository and store work plan and all documentation created to date.
  - Review and validate the existing list of enhancements to be implemented planned maintenance activities, and planned application infrastructure upgrade services. Make needed clarifications.
  - If needed and agreed to by both the task force and contractor, update the list of enhancements to be implemented, maintenance activities and application infrastructure upgrade services. These changes will require additional planning and update to the schedule during the next activity.
- *Report Status and Plan Next Month*

In addition to the above, the task force should *Identify Stakeholders to Review Deliverables* and provide to the contractor as discussed in Chapter 1.

### 3.2.5.3. Plan the Next Phase of the MSE Work Effort

At the end of the Project-Start-Up sub-phase, the contractor should review the planned work for the next phase; perform additional planning and update the MSE schedule, as required; and begin executing the Requirements, Design and Construction Phase.

This process of reviewing the work for the next phase should be repeated at the end of each phase; except for the Delivery and Closeout Phase.

### 3.2.5.4. Submit and Approve Planning & User Requirements Gate

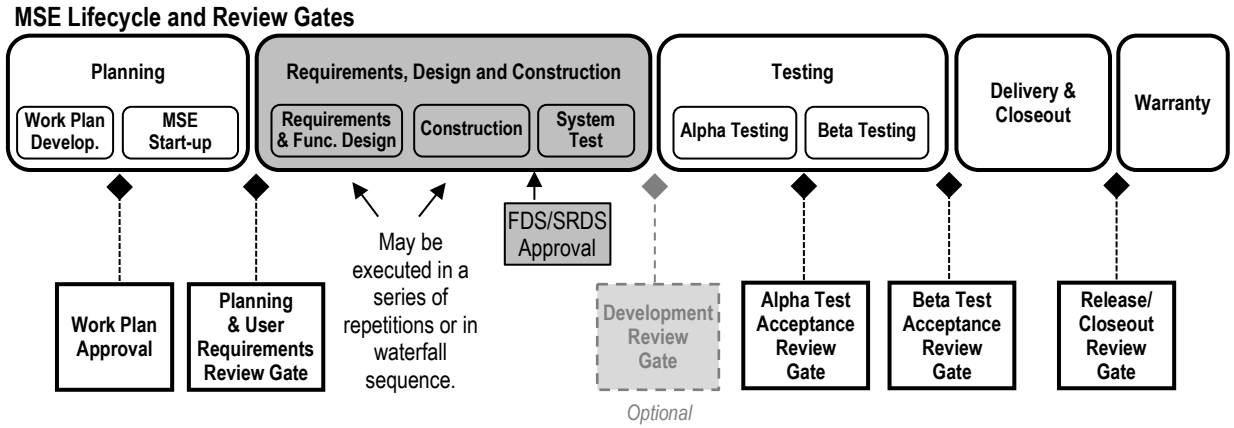
*If the enhancements, technology upgrades or other components of the work plan were revised or defined during the Project Start-Up sub-phase, the Planning and User Requirements Review Gate is scheduled, initiated, and approved at the end of the Project-Start-Up phase.* The review gate approval request is submitted and approved as described in the *Submit Planning and User Requirements Review Gate* and *Approve Planning and User Requirements Review Gate* sections in Chapter 2.

If this review gate is not scheduled, the contractor should begin executing the next phase.

#### 3.2.5.5. Manage, Monitor, and Control MSE Work Effort

In parallel to executing the planned activities and tasks, the MSE work effort should be managed, monitored, and controlled as defined by the procedures from the work plan or those revised during project Start-Up. These activities continue with each phase throughout the lifecycle of the MSE work effort. Refer to the *Manage, Monitor, and Control the Project* section in Chapter 2 for additional information.

## 3.3. Requirements, Design & Construction Phase




### 3.3.1. Phase Overview

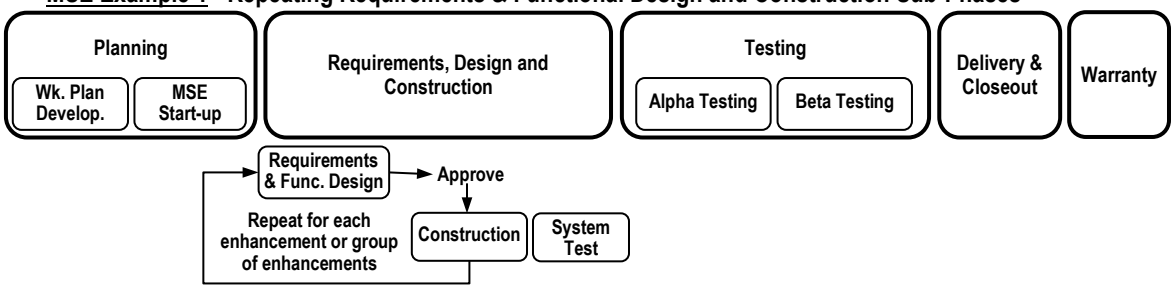
Where the project lifecycle includes three phases (Requirements & Analysis, Design, and Construction) between the Planning and Testing Phases; the MSE lifecycle combines the functions of these phases into a single phase with three sub-phases.

- During first sub-phase (Requirements & Functional Design), each enhancement is analyzed and the system requirements and design specifications for each enhancement are developed, documented, and approved.
- During the second sub-phase, each enhancement is constructed and tested (unit/build).
- During the third sub-phase all enhancements are system tested.

Two options are provided for executing the sub-phases as described below.

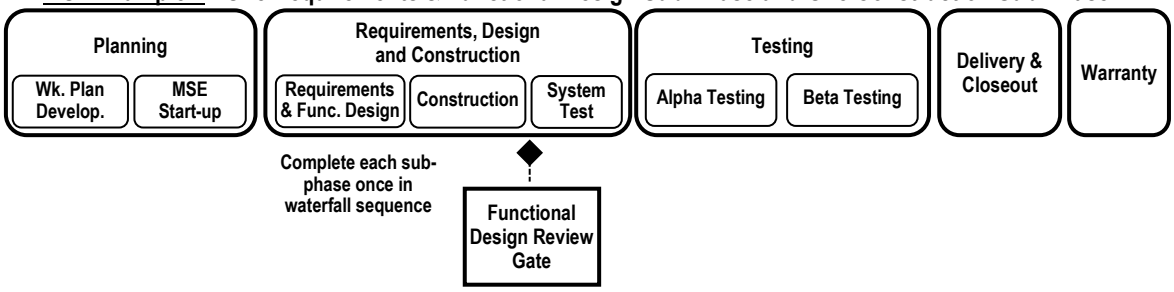
- In most cases an iterative approach is used, where the Requirements & Functional Design and Construction sub-phases are repeated for each enhancement or each group of related enhancements as shown in the example 1 diagram and described below.
  - An FDS or SRDS deliverable is created for each repetition of the Requirements & Functional Design sub-phase.
  - *Each FDS/SRDS is approved prior to the beginning the construction sub-phase for the applicable enhancement(s) by deliverable approval or by review gate.*
  - The repetitions of the Requirements & Functional Design and Construction sub-phases may be executed sequentially, overlapping, or concurrently.
  - *The last Construction sub-phase is followed by a single System Test sub-phase with all enhancements included in the system test.*

**MSE Example 1 - Repeating Requirements & Functional Design and Construction Sub-Phases**




- The Requirements & Functional Design and Construction sub-phases may also be performed in a waterfall sequence as shown in the example 2 diagram and described below.
  - The FDS/SRDS deliverables for all enhancements are completed in a single Requirements & Functional Design sub-phase.
  - The FDS/SRDS may be prepared as a single deliverable for all enhancements or as multiple deliverables like that used in the iterative approach.
  - *All FDS/SRDS deliverable(s) must be approved prior to beginning the development of all enhancements in a single Construction sub-phase by deliverable approval or by review gate.* The Functional Design Review Gate is the recommend method for approval when using a waterfall sequence as shown in the following diagram.
  - *The Construction sub-phase is followed by a single System Test sub-phase with all enhancements included in the system test.*

**MSE Example 2 - One Requirements & Functional Design Sub-Phase and One Construction Sub-Phase**




- *In both of the above options, the Requirements, Design and Construction Phase ends with a successful system test of all enhancements.*
- *The Testing and Delivery & Closeout Phases are performed for the complete scope of the MSE effort (inclusive of all enhancements, maintenance work, and upgrades).*
- Either option is considered standard and can be used without requesting an exception.
- *Regardless of what method is used, the approach should be agreed upon by the both contractor and task force and documented in the MSE work plan and reflected in the MSE schedule.*

This standard does not specifically address the analysis, design and construction work associated maintenance activities and technology upgrades; however, it is assumed that these activities are completed during this phase and the results are included in the product that will be alpha tested in the next phase.

### 3.3.2. Input to the Requirements, Design & Construction Phase

The primary deliverables and artifacts that will be used or referenced in this phase are:

- The enhancements to be implemented, the problems to be corrected, and the technology upgrades to be performed.
- In some cases, there may also be existing user requirements, system requirements, and/or functional design specifications that will be used to develop one or more requested enhancements.

Other key items used in this phase are the MSE Work Plan; MSE Schedule; the work plan procedures, processes and technologies; and Section 508/Accessibility web sites.

### 3.3.3. Output from the Requirements, Design & Construction Phase

The following artifacts and deliverables are created or updated during this phase of the MSE work effort.

- Lists of enhancements to be implemented and problems to be corrected – if revised during this phase
- *Enhancement system requirements and design specifications in the form of one or more FDS or SRDS deliverable(s).*
- *Approval documentation for SRDS deliverables.*
- Product Test Plan - if revised during this phase
- *Test procedures for enhancements – initial or revised*
- *Successful System Test*
- *Alpha Test Plan – initial or revised*
- Development Review Gate Approval Request (Optional) – Submission of this review gate approval request occurs in those cases when the task force requests the contractor to formally acknowledge that a system test has been completed successfully and the product is ready for Alpha Testing and/or when the task wants to review the Alpha Test plan prior to beginning Alpha Testing.
- *MSE Schedule – if revised*
- *Project Repository - updated*

### 3.3.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Security Standard* - Used when developing the requirements and design for the system security.

### 3.3.5. Procedures

This section defines the analysis, design, and construction activities that are to be followed by the task force and/or contractor during this phase and the results of those activities.

#### 3.3.5.1. Develop System Requirements

During this phase, the contractor analyzes each enhancement and determines if the description of each enhancement is clear and concise. At this point, the contractor should also determine which enhancements (if any) should be grouped together for system requirements, design, documentation, and construction purposes.

The contractor develops and documents system requirements to expand and clarify what is needed to meet the intent of each enhancement. A system requirement may also define what needs to be done to implement an enhancement.

The number of system requirements and the level of detail will vary based on the size and complexity of each enhancement. Typically, system requirements are only created for the medium and large size enhancements in an MSE work effort and are not normally needed for small enhancements or maintenance work. When analyzing enhancements, each of the following type of system requirements should be considered during this activity.

- Functional Requirements
- Preliminary Data Requirements
- System Interface Requirements
- User Interface Requirements
- Security Requirements
- Accessibility Requirements
- Performance Requirements
- Technical Architecture Requirements
- Other Non-Functional Requirements (such as communications, disaster recovery, maintainability, portability, reliability, and scalability impose constraints on the design or implementation)

Each of the above types of system requirements is described in the *Define System Requirements* section in Chapter 2.

In some cases, the contractor may have developed partial system requirements prior to beginning the project and included these in the work plan. In these cases, the number of system requirements developed in this phase will be limited. The goal should be to have the appropriate detail in the system requirements to design each enhancement and to adequately test and accept each completed enhancement.

The system requirements for an enhancement or group of enhancements are normally documented with the functional design specifications for the enhancement(s) in FDS or SRDS (System Requirements and Design Specifications) deliverables, as described below. The system requirements may also be documented separately in a spread sheet or other type of document as long as the deliverable has the required content and is reviewed and approved by the task force. The content requirements for an enhancement SRS are defined in the *System Requirements Specification (SRS)* section in Chapter 5.

### 3.3.5.2. Develop the Functional Design

*During this activity, the enhancements and their system requirements are translated into functional design specifications that define "how the requirements will be implemented" from a user or business perspective.* The functional design specifications should be documented using terminology that can be readily reviewed and understood by the task force, technical review teams (TRTs), technical advisory groups (TAGs), and other stakeholders; and should demonstrate that the enhancement(s) and associated system requirements will be implemented.

As with the system requirements; the level of detail of the functional design will vary based on the size and complexity of each enhancement. Technical design specifications (TDS) are not required for MSE work; however, if needed, the contractor should develop additional specifications at the appropriate level of detail required to construct the enhancements. This may include updating the TDS from a prior project or MSE effort,

When designing the enhancements, each of the following type of design elements should be considered for inclusion in the functional design specifications.

- System Structure Diagram

- Logical Process Model
- Data Dictionary
- User Interface and Report Design
- System Interface Design
- Security Design
- Technical Architecture

In many cases, the above design elements will normally be documented as revisions or additions to the design elements that were developed during the initial development of the existing product or prior enhancements. Each of the above types of design elements is described in *Develop the Functional Design* section in Chapter 2.

As with the system requirements, contractor may have developed partial functional design specifications prior to beginning the project and included these in the work plan. In these cases, functional design activities will normally include revisions or additions to the design specifications provided in the work plan.

As described above, the design specifications for each enhancement or group of related enhancements are normally documented in an enhancement FDS or SRDS deliverable along with the systems requirements. The content requirement for these deliverables is defined in the *Functional Design Specification (FDS)* section of Chapter 5. As noted in this chapter, these deliverables are not required to be named enhancement FDS or SRDS.

Design specifications are not required for small enhancements unless requested by the task force. Also, no design specifications are required for maintenance work.

### 3.3.5.3. Review and Approve the FDS/SRDS Deliverables

*Regardless of the development approach used, each FDS or SRDS deliverable must be approved by the task force prior to the construction of the enhancements covered by the FDS/SRDS.* These deliverables will normally be submitted to a stakeholder group (TAG or TRT) for review prior to task force review.

- Iterative Approach (Repeating Requirements & Functional Design and Construction Sub-Phases)
  - *After both the system requirements and functional design activities are completed and documented for an enhancement or a group of related enhancements, the contractor submits the FDS/SRDS deliverable for the enhancement(s) to the task force for review and approval.*
  - *When approved, the contractor is authorized to begin construction of the enhancement(s) covered by the approved FDS/SRDS deliverable.*
  - *Each FDS/SRDS deliverable shall be approved by the MSE's Deliverable Review and Approval procedure or the Review Gate Approval Procedure.*
  - The contractor and task force may choose to submit and approve each FDS/SRDS deliverable as completed or submit and approve multiple FDS/SRDS deliverables in a single submission.
- Waterfall Approach (One Requirements & Functional Design Sub-Phase and One Construction Sub-Phase)
  - *After the system requirements and functional design activities are completed and documented for all enhancements, the contractor submits all FDS/SRDS deliverables to the task force for approval.*
  - *When approved, the contractor is authorized to begin construction of all enhancements.*

- *As with the iterative approach, each FDS/SRDS deliverable shall be approved by the MSE's Deliverable Review and Approval procedure or the Review Gate Approval Procedure*. Submission of the Functional Design Review Gate approval request is the recommended method for approval.
-

### 3.3.5.4. Develop Test Procedures

*Prior to or with the development of each enhancement, the contractor shall develop the appropriate test procedures for unit testing, build and system testing.* In most cases, new test procedures are created to support the enhancements and are added to existing test procedures created for the product during a previous project or MSE work effort. Existing test procedures may also be revised to support the enhancements for the MSE work.

### 3.3.5.5. Construct Enhancements

As described in the Project Development Process; construction involves coding, creating databases, integration, unit testing and build testing. All hardware or software needed to support the construction effort should be installed and setup by the time construction begins.

The design specifications for the current enhancements (FDS/SRDS) and existing FDS, SRDS, and TDS deliverables from prior projects and MSE efforts are used as a design blueprint. Each enhancement or group of enhancements is constructed after the system requirements and design specifications are approved and the test procedures for the enhancement(s) are completed.

The process used for construction is left up to the contractor. Construction of the various enhancements for the MSE work effort should normally occur in the order defined in the MSE work effort's schedule. The contractor is authorized to begin construction of an enhancement, after the task force approves the FDS/SRDS for the enhancement. Since all FDS/SRDS deliverables are approved together when using a waterfall sequence, all enhancements are authorized for construction after a single approval.

Once approved, some enhancements may be developed concurrently while others may be developed sequentially. With the iterative approach, one or more enhancements may be in design while other enhancements are being constructed and tested, and while additional enhancements may be completed or not started. Construction and testing for small enhancements and maintenance work are also completed during this phase.

### 3.3.5.6. Perform System Test

*After all construction is completed for all enhancements, the contractor shall perform a system test.* System testing tests all system components constructed during the MSE effort and ensures that integration is complete with existing components and the system performs as required. All test procedures that will be used for alpha testing should be executed to ensure that all user and system requirements are met. The scope of system testing is the same as that of alpha testing (complete system, specific components, specific enhancements, etc.) and is defined the Product Test Plan or the "Test Plan" section of work plan.

*The Requirements, Design and Construction phase ends with a successful system test and with the product ready for alpha testing.*

#### 3.3.5.7. Submit and Approve Development Review Gate (Optional)

If the task force and contractor have planned the optional Development Review Gate, the contractor should submit the review gate approval request after completing a successful system test. In this case, the Alpha Test Plan, which is described in the Testing Phase, is completed and approved prior to, or with, this review gate.

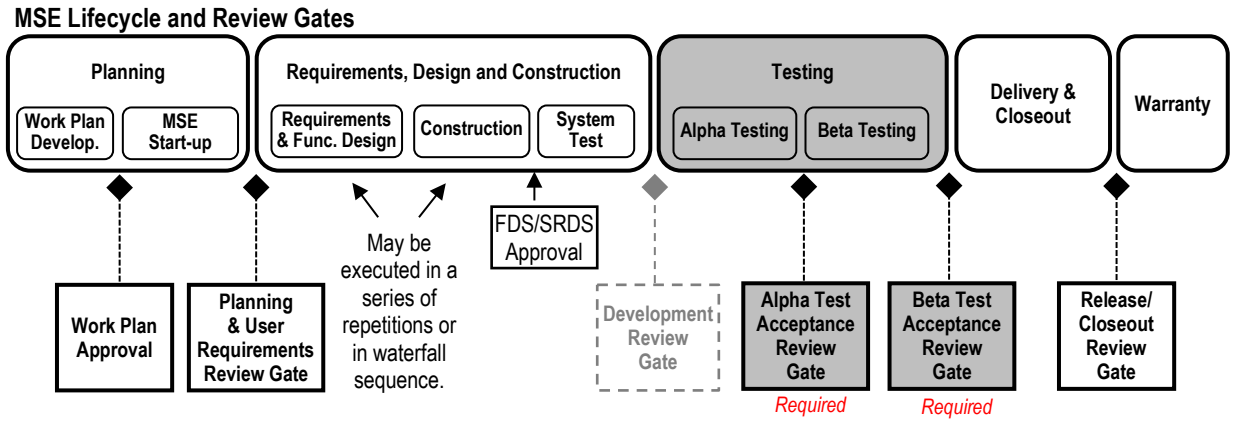
If this review gate is not planned, the Alpha Test Plan is completed prior to beginning Alpha Testing in next phase and is submitted prior to, or with, the Alpha Test Acceptance Review Gate.

#### 3.3.5.8. Continue to Plan, Manage, Monitor and Control

As the Requirements, Design and Construction Phase is executed and completed, the contractor, task force and AASHTO PM should continue to *Manage, Monitor, and Control the Project*.

At the end of this phase, the contractor should review the planned work for the next phase (Testing); perform additional planning and update the MSE schedule, as required; and then begin executing the Testing Phase.

# 3.4. Testing Phase




## 3.4.1. Phase Overview

The Testing phase for MSE work is the same as that for projects and is subdivided into two sub-phases, Alpha Testing and Beta Testing. The Testing Phase ends after approval of the Beta Test Acceptance Review Gate.

There are certain cases where beta testing is not performed during an MSE work effort. Exclusion of beta testing may occur in MSE work plans with only minor or limited enhancements; and in work plans where an exception to omit beta testing is included in the approved work plan or approved by SCOA in a separate request. In both cases, the "Scope" and/or "Test Plan" sections of the work plan should state that no beta testing will be performed.

## 3.4.2. Input to the Testing Phase

The primary deliverables and artifacts that will be used in this phase are:

- The enhancements to be implemented, user requirements, the problems to be corrected, and the technology upgrades to be performed
- Enhancement system requirements and functional design specifications (SRDS)
- Product Test Plan
- Test procedures for enhancements
- Alpha Test Plan (if completed in previous phase) or Existing Alpha Test from previous project or MSE
- Existing Beta Test Plan and Product Installation Package from previous project or MSE

Other key items used in this phase are the MSE Work Plan; MSE Schedule; and the work plan procedures, processes and technologies.

## 3.4.3. Output from the Testing Phase

The following deliverables and artifacts shall be planned, prepared or updated, submitted, and approved to comply with this standard or the referenced standard.

- *Alpha Test Plan - revised*
- *Alpha Test Results*
- *Beta Test Materials* (Beta Test Plan and Beta Installation Package) initial or revised*
- *Alpha Test Acceptance Review Gate Approval Request*
- *Agency Beta Test Results Report(s)**
- *Beta Test Results Report**

- *Product Installation Package*
- *Beta Test Acceptance Review Gate Request**
- *MSE Schedule - if revised*
- *Project Repository – updated*

*The beta testing deliverables listed above are not produced in those cases where beta testing is excluded in the approved MSE work plan or an exception to exclude beta testing has been approved.

### 3.4.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard
- *Common Artifacts Standard*

### 3.4.5. Procedures

This section defines the alpha and beta testing activities that are to be followed by the task force and/or contractor during this phase and the results of those activities.

#### 3.4.5.1. Perform Alpha and Beta Testing

Alpha and Beta Testing are planned, performed, documented, and approved executing the same activities as those used for projects. In most cases the Alpha Test Plan and the Beta Test Materials (Beta Test Plan and Beta Installation Package) are prepared by updating existing deliverables that were prepared during a previous project or MSE work effort.

The following list summarizes the key testing activities. Each of these is described in the *Procedures* section of the Testing Phase section in Chapter 2.

- Prepare/update Alpha Test Plan
- Review Alpha Test Plan
- Perform Alpha Testing and prepare Alpha Test Results Report
- Review and approve Alpha Test Results Report
- Prepare/update Beta Test Materials* (Beta Test Plan and Beta Test Installation Package)
- Review Beta Test Materials*
- Submit Alpha Test Acceptance Review Gate
- Approve Alpha Test Acceptance Review Gate
- Perform Beta Testing and prepare the Beta Test Results Report*
- Review Beta Test Results Report*
- Repeat Beta Testing as required*
- Finalize Product Installation Package
- Draft the cover letter for Product Installation Package distribution
- Submit Beta Test Acceptance Review Gate*
- Approve the Beta Test Acceptance Review Gate

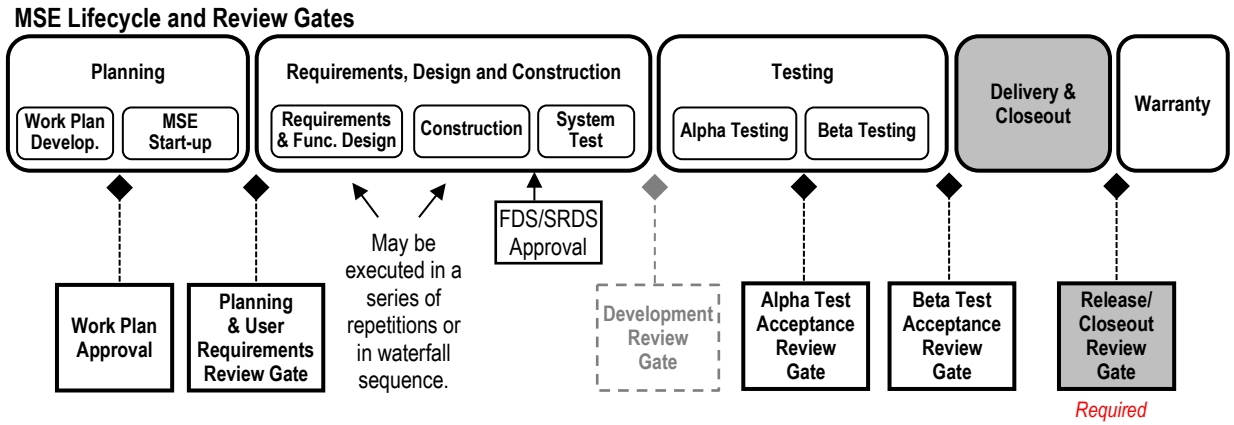
*The beta testing activities listed above will not occur in those cases where beta testing is excluded in the approved MSE work plan or an exception to exclude beta testing has been approved.

#### 3.4.5.2. Continue to Plan, Manage, Monitor and Control

As the Testing is executed and completed, the contractor, task force and AASHTO PM should continue to *Manage, Monitor, and Control the Project*.

At the end of this phase, the contractor should review the planned work for the next phase; perform additional planning and update MSE schedule, as required; and then begin executing the Delivery and Closeout Phase.

# 3.5. Delivery and Closeout Phase




## 3.5.1. Phase Overview

The Delivery and Closeout phase for MSE work is the same as that for project with the primary activities of distributing the completed product to the customer sites and formally closing the MSE work effort unless multiple software versions are released during an MSE period. In that case, this phase may close the work for the version being released but not close the work for the MSE.

## 3.5.2. Input to the Delivery and Closeout Phase

The primary deliverables and artifacts that will be used in this phase are:

- Product Installation Package (or Beta Installation Package)
- Existing ACR
- Existing Application Infrastructure Component List
- All deliverables and artifacts completed during the MSE effort for the software version being released.

Other key items used in this phase are the MSE Work Plan; MSE Schedule; and the work plan procedures, processes and technologies.

## 3.5.3. Output from the Delivery and Closeout Phase

The following deliverables and artifacts shall be planned, prepared or updated, submitted, and approved to comply with this standard or the referenced standard.

- *Completed product - final*
- *Product Installation Package - final*
- *Accessibility Conformance Report (ACR) – final (if updated)*
- *Application Infrastructure Component List – final (if updated)*
- *Data Dictionary – final (if updated)*
- Development and Maintenance Document (if updated)
- *MSE Archive Package (includes the artifacts for the software version(s) being released)*
- *If the MSE is being closed, Closeout Review Gate Approval Request*
- *If the MSE is not being closed, Release Review Gate Approval Request*
- *Project Repository - updated*

## 3.5.4. Standards Used/Referenced in this Phase

- Software Development and Maintenance Process Standard

- *Common Artifacts Standard*
- *Critical Application Infrastructure Currency Standard* – Used when preparing or updating the Application Infrastructure Component List.
- *Product Naming Conventions Standard* – Used to ensure that product names, terminology, branding, icons, release numbers, and splash screens are correct.

## 3.5.5. Procedures

The procedures described below define the activities that are to be followed by the task force and/or contractor during this phase.

### 3.5.5.1. Distribute Product and Complete Work for the Release

The product is distributed and associated work is completed, performing the same activities as those used for projects. In most cases these activities involve updating deliverables and artifacts that were originally prepared during a previous project or MSE work effort.

The following list summarizes the key activities of the Delivery and Closeout Phase. Each of these is described in the *Procedures* section of the Delivery and Closeout Phase of Chapter 2.

- *Distribute the product and begin providing support*
- *Update the Accessibility Conformance Report (ACR) – If the accessibility functions of the product are modified, the contractor shall determine if the existing ACR needs to be updated and make the appropriate modifications.*
- *Update the Application Infrastructure Component List – The contractor shall update the existing Application Infrastructure Component List for the product(s) when a new version of an existing component is implemented, a new component is implemented, and/or an existing component deleted.* Refer to the *Critical Application Infrastructure Currency Standard*.
- Update Development and Maintenance Documentation – If this documentation exists, it should be updated prior to closing the MSE work effort. If there is no existing documentation, no new documentation is required.
- *Prepare MSE Archive Package*
- Review above deliverables and artifacts
- *Resolve installation and other critical problems*
- *Update and redistribute Product Installation Package or login credentials for hosted applications as needed*
- *Report installation status to the task force.*
- *If the MSE is being closed, submit and approve Closeout Review Gate.*
- *If the MSE is not being closed, submit and approve Release Review Gate*.

### 3.5.5.2. Closeout Work Effort

*After the Release Review Gate is approved, the contractor sends the ACR and MSE Archive to the AASHTO PM.*

*After the Closeout Review Gate is approved, the contractor sends the ACR and MSE Archive to the AASHTO PM. Other than warranty work, all other work on the MSE work effort shall be terminated.*

# 4. Adapting the Lifecycle and Process

## 4.1. Introduction

Although the project lifecycle has been described and depicted in Chapter 2 as a waterfall process, where the phases and sub-phases are accomplished sequentially; the lifecycle may also be adapted to accomplish the phases and sub-phases concurrently or cyclically. The lifecycle and the development process are meant to be flexible to address the scope, size, and complexity of specific projects. The MSE lifecycle and process are also meant to be flexible.

The phases and sub-phases of the project and MSE lifecycles are meant to provide a uniform approach to AASHTOWare software development and maintenance; however, there are cases where modification is required. For example, a project that is limited to the development of requirements and/or design specifications will require fewer phases; and a project using an iterative development methodology will typically repeat certain phases. Other modifications may be needed for a large complex project, a small MSE effort with limited enhancements, or for a contractor's specific development methodology.

The required review gates, deliverables and artifacts described in both *Chapter 2* and *Chapter 3* are the least flexible part of the development process; however, there will be cases when certain items may be eliminated or accomplished in a different manner. For example, a project limited to the development of requirements and/or design specifications will not include the Construction or Testing Phases and the deliverables and review gates associated with these phases. An iterative development project normally combines the Design and Construction Phases into a single phase which includes a series of design, construction, and system testing sub-phases for each development iteration.

Since iterative development projects and projects that develop requirements and design specifications occur regularly, a standard adaption has been created for each of these types of projects. These standard adaptions, which are included in the next two sections, provide modifications that may be made to applicable projects without an exception to this standard.

The last section in this chapter includes some other adaptions that may be made to any project or MSE work effort. Some of these adaptions will require an exception which must be approved by SCOA. All adaptations to the standard lifecycle and process should be described in the work plan.

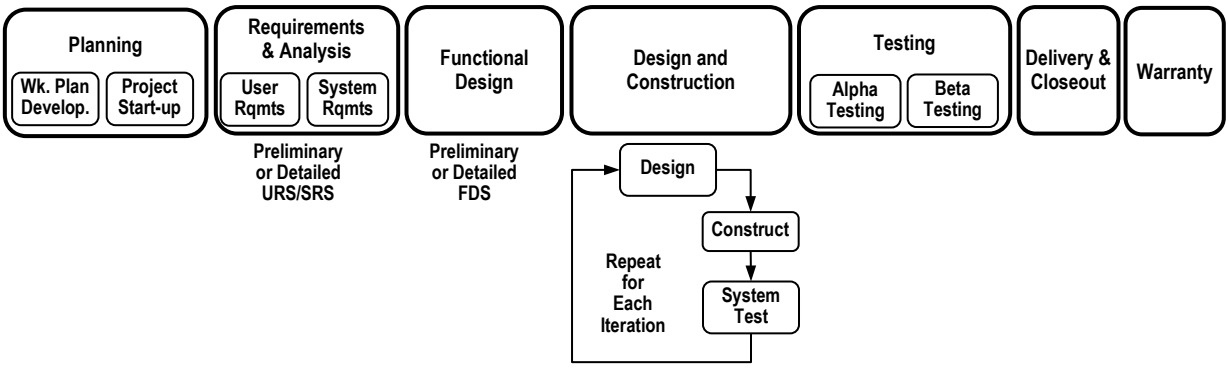
## 4.2. Iterative Project Development Process

This section describes a standard adaption of the project lifecycle and development process for projects using an iterative development methodology. An iterative development project breaks the proposed application into smaller chunks or segments and the design, construction and testing activities are completed in repeated cycles or iterations. For AASHTOWare, the project is typically divided into functional segments or software development time box segments. Any of the key activities (sub-phases) in the Requirements & Analysis, Design, and/or Construction Phases may be completed in iterations. One or more iterative phases are created by combining sub-phases from the standard project lifecycle.

The iterative project development process described in this section allows flexibility to customize the standard project lifecycle (refer to *Project Development Process)* to fit most iterative development methodologies. Three iterative lifecycle example adaptions of the standard project development process are described in this section. Each adaption is briefly discussed below followed by a lifecycle diagram. Additional details on each lifecycle are provided later in this section. If these lifecycles are used as described in this section, no exception is required.

Iterative Lifecycle 1 – In this lifecycle, the Technical Design, Construction, and System Testing activities for each segment are completed in iterations during a Design and Construction Phase. Prior to beginning the iterations; the user requirements, system requirements and functional design specifications are developed in two non-iterative phases. This lifecycle provides the flexibility to define the pre-iteration requirements and functional design specifications at either a broad, preliminary level (high level) or a more granular, detailed level; depending on the project objectives. The lifecycle also allows the requirements to be defined at a detailed level and the functional design to be defined at a high level, as well as, allowing the user requirements to be detailed, while the system requirements and functional design are defined at a high level. When the requirements and/or functional design are defined at a high level, these will normally be refined during the Design and Construction iterations. After the iterations are completed, the project lifecycle returns to the standard lifecycle with the Testing and Delivery & Closeout Phases completed for the complete scope of the project.

**Iterative Lifecycle 1 - Iterative Technical Design & Construction Phase**

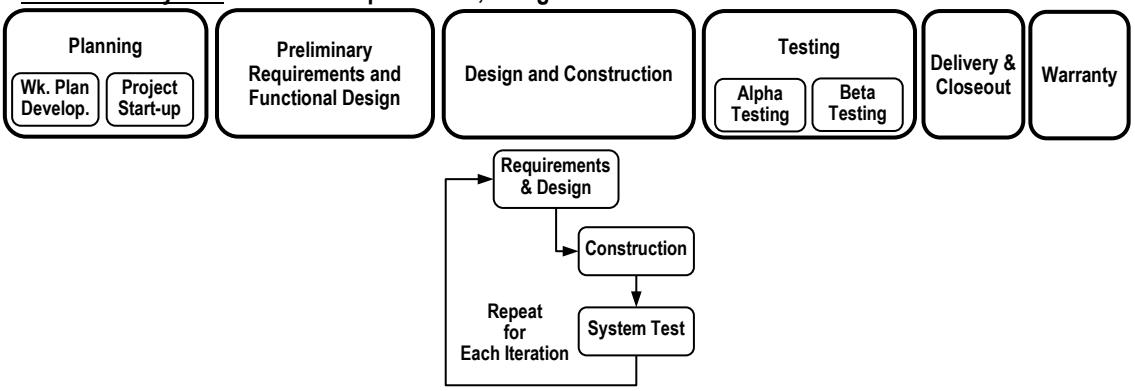


Iterative Lifecycle 2 – In this lifecycle, preliminary (high level) User Requirements, System Requirements and Functional Design specifications are defined in a single Preliminary Requirements and Functional Design phase prior to beginning an iterative Design and Construction Phase. This lifecycle uses a more incremental approach for developing the requirements and function design than that of Lifecycle 1.

- Only a limited number of broad, high level, preliminary requirements and functional design specifications are defined before beginning the iterative Design and Construction phase;
- The preliminary user and system requirements will be refined, and additional requirements will be discovered while iteratively developing the detailed Functional Design specifications for each segment; and
- Additional refinement may occur again during the Technical Design, Construction and System Test activities for each segment.

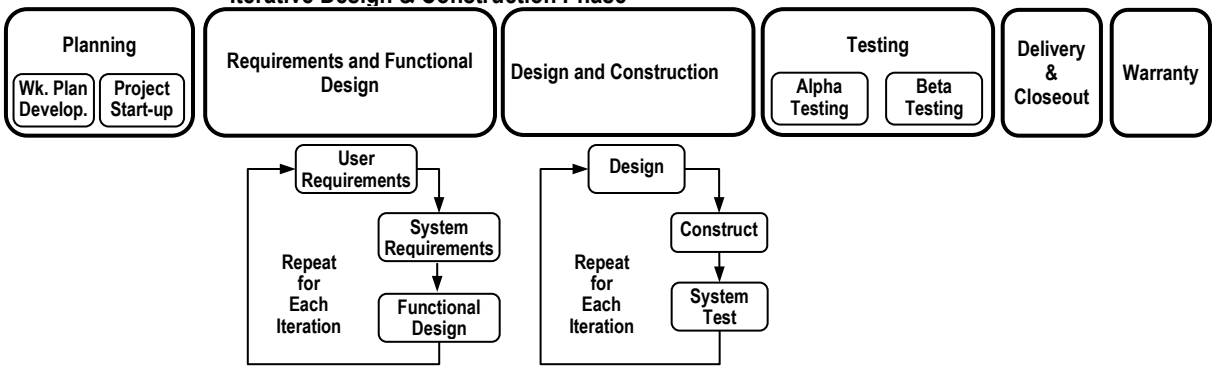
As with Lifecycle 1, the project returns to the standard lifecycle with the Testing and Delivery & Closeout Phases completed for the complete scope of the project.

**Iterative Lifecycle 2 - Iterative Requirements, Design & Construction Phase**




Iterative Lifecycle 3 – This lifecycle includes two iterative phases. The User Requirements, Systems Requirements and Functional Design activities for each segment are completed iteratively during the Requirements and Functional Design Phase. The first phase produces the same deliverables as the first two phases in Lifecycle 2 using an iterative approach is lieu of a waterfall approach. The second iterative phase (Design and Construction) is the same as that used in Lifecycle 2. As with Lifecycles 1 and 2, the project returns to the standard lifecycle with the Testing and Delivery & Closeout Phases completed for the complete scope of the project.

**Iterative Lifecycle 3 - Iterative Requirements & Functional Design Phase and Iterative Design & Construction Phase**




This section is written similar to Chapter 3 and primarily focuses on the differences between projects using the standard project development process (waterfall) and projects using an iterative development methodology. When an activity in the Iterative Project Development

Process is the same or very similar to an activity in the standard project process, the duplicate activity will normally include a hyperlink to the more detailed description in Chapter 2. Each iterative lifecycle includes a section organized by lifecycle phase. After Iterative Lifecycle1 is described, Iterative Lifecycle 2 primarily focuses on the differences with Lifecycle 2. Lifecycle 3 then focuses on the differences with Lifecycle 2 and 3.

The next section summarizes the differences between the iterative and waterfall project development processes.

### 4.2.1. Phases, Review Gates, Deliverables and Artifacts

The section summarizes the differences in phases, review gates, deliverable and artifacts between an iterative development project and a waterfall development project; and the allowable customizations.

- The work plan for an iterative development project has the same sections and uses the same template as a standard project. The work plan should define the typical information in each section plus the specific iterative development and testing approaches, lifecycle, review gates, and deliverables that are unique to the project. Refer to the *Planning Phase* for addition information.
- *One or more iterative phases are created by combining the key activities (sub-phases) shown below of the following standard project lifecycle phases.*
  - Requirements & Analysis Phase
    - User Requirements
    - System Requirements
  - Design Phase
    - Functional Design
    - Technical Design
  - Construction Phase
    - Construct (or Build)
    - System Testing
- Examples of iterative phases are provided in the three iterative lifecycles in the previous section.
  - Lifecycle 1, 2 and 3 – The Technical Design, Construction, and System Testing sub-phases are combined to form an iterative Design and Construction Phase.
  - Lifecycle 3 - The User Requirements, Systems Requirements and Functional Design sub-phases are combined to form an iterative Requirements and Functional Design Phase.
- Examples of other possible iterative phases not shown in the three iterative lifecycles are listed below.
  - An iterative phase may combine the Construction and System Testing sub-phase to form an iterative Construction Phase.
  - An iterative phase may combine the System Requirements and Functional Design sub-phases to form an iterative System Requirements and Functional Design Phase.
- *All other standard project phases (Planning, Testing and Delivery and Closeout) are executed in waterfall sequence as described in the Project Development Process section*.

- Required deliverables, review gates and other deliverable approvals will normally influence the number of iterative phases and the execution of those phases, as described below.
- *An Iteration FDS shall be prepared for each iteration in an iterative phase that includes a sub-phase for developing or significant refining the functional design.*
  - *An Iteration FDS includes the system requirements, functional design specifications and test procedures for the iteration. Refer to Functional Design Specification (FDS) for the content of the Iteration FDS.*
  - *Each iteration FDS shall be approved by the task force using the project's Deliverable Review and Approval procedure* or *Review Gate Approval Procedure.*
  - *Approval authorizes the contractor to begin the construction of the iteration.*
- *An Iteration Test Results Report shall be prepared after the system test for each iteration in an iterative Design and Construction phase or an iterative Construction phase.*
  - *An Iteration Test Results Report Iteration Test Results Report includes similar content to that of an Alpha Test Results Report.*
  - *Each iteration Test Results Report shall be approved by the task force using the project's Deliverable Review and Approval procedure* or *Review Gate Approval Procedure.*
  - *Approval represents completion and acceptance of the iteration.*
- As with waterfall projects, the Planning and User Requirements review gate is conditional, *and the Alpha Test Acceptance, Beta Test Acceptance, and Closeout review gates are required. The required deliverables and artifacts for these review gates are the same as that for waterfall projects*.
- *The Functional Design Review Gate is required in the following cases for iterative projects.*
  - *Following an iterative Functional Design phase or sub-phase.*
    - *The Functional Design Review Gate is required after the last iteration of an iterative functional design phase or sub-phase.* For example, in Lifecycle 3, the Functional Design Review Gate occurs after the last iteration in the Requirements and Functional Design phase.
    - This review gate will normally replace the Iteration FDS approval for the last iteration.
    - *The current versions of the URS, RTM, SRS and FDS shall be submitted with the review gate.*
  - *Following a non-iterative Functional Design phase or sub-phase.*
    - *The Functional Design Review Gate is required after all functional design activities for a non-iterative phase or sub-phase are completed.*
      - In Lifecycle 1, the Functional Design Review Gate occurs after the Functional Design phase.
      - In Lifecycle 2, the Functional Design Review Gate occurs after the Preliminary Requirements and Functional Design Phase.
    - *The current versions of the RTM, SRS and FDS (preliminary or detailed) shall be submitted with, or prior to the review gate.*

- Refer to the *Functional Design Review Gate*, *Submit Functional Design Review Gate* and *Approve Functional Design Review Gate* sections for additional information on this review gate.

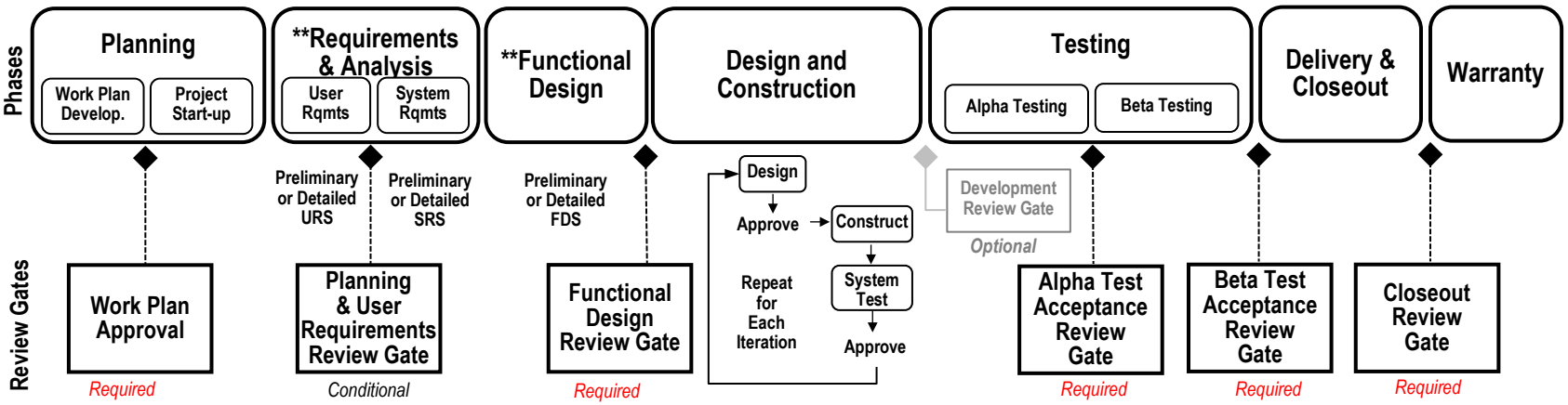
- The Development Review Gate is optional, as with waterfall development projects; however, it is recommended that this review gate be scheduled at the completion of an iterative Design and Construction phase or an iterative Construction phase.
  - For example, in Lifecycle 1, the Development Review Gate would occur after the last iteration in the Design and Construction phase.
  - When scheduled, this review gate will normally replace the Iteration Test Results Report approval for the last iteration.
  - Refer to the *Development Review Gate* and *Submit and Approve Development Review Gate (Optional)* sections for additional information on this review gate.
- As with waterfall projects, the User Requirements Specification (URS) is normally defined in the work plan and is reviewed, validated, and, if needed, revised during the execution project. If no URS exists, the project will normally define the URS during the execution of the project.
  - A URS may be defined at the normal, detailed level as used with waterfall projects.
  - A Preliminary URS with high level user requirements is defined as the initial URS when the user requirements are developed incrementally as the lifecycle of the project progresses. A Preliminary URS is defined during the Preliminary Requirements and Functional Design Phase of Lifecycle 2. A Preliminary URS may be optionally defined during the Requirements and Analysis Phase of Lifecycle 1.
  - The User requirements may be defined iteratively as in in Lifecycle 3.
  - *If the URS (preliminary or detailed) is defined or updated in a non-iterative phase, the URS is approved with the Planning and User Requirements Review Gate.*
  - *If the URS is defined or updated in an iterative phase, the required content of the URS is included in an iteration deliverable (such as an Iteration FDS) and approved with that deliverable.*
- As with waterfall projects, both the System Requirements Specification (SRS) and the Functional Design Specification (FDS) are normally defined during the execution of the project. An existing SRS or FDS may also be included in the work plan and reviewed, validated, and revised (if needed) during the execution of the project.
  - If the SRS is developed or updated prior to an iterative phase, this normally occurs in a System Requirements sub-phase that follows or overlaps with the User Requirements sub-phase as in Lifecycle 1.
  - If the FDS is developed or updated prior to an iterative phase, this normally occurs in a Functional Design Phase or sub-phase which follows or overlaps with the System Requirements sub-phase as in Lifecycle 1.
  - A Preliminary SRS is defined as the initial SRS when the system requirements are developed incrementally as the lifecycle of the project progresses. A Preliminary FDS is defined when the functional specifications are developed incrementally. A Preliminary SRS and FDS are refined during the Preliminary Requirements and Functional Design Phase of Lifecycle 2. A Preliminary SRS and/or FDS may be optionally defined during the Requirements and Analysis Phase of Lifecycle 1.
  - The SRS and/or FDS are defined iteratively as shown in Lifecycle 3.

- *If the SRS and/or FDS (preliminary or detailed) are defined or updated in a non-iterative phase, both are approved with the Functional Design Review Gate.*
- *If SRS and FDS are defined or updated in an iterative phase, both are documented in an Iteration FDS and approved with that Iteration FDS.*

- *The RTM is created and updated as the projects progresses during the various User Requirements, System Requirements, Functional Design, and Construction phases/sub-phases of an iterative project.*
- Any phase that defines preliminary, high level versions of deliverables (URS, SRS and/or FSD) should be prefixed with “Preliminary”. The deliverables should also be prefixed. This strictly a recommendation to help clarify the scope and content of these deliverables.
- The TDS is created incrementally as the iterations are designed and constructed.

The following diagram provides a summary view of the iterative project review gates and required deliverables and artifacts in relationship to iterative project development lifecycle approach shown in Lifecycle 1.

## Iterative Lifecycle 1 - Iterative Project Lifecycle Phases and Review Gates with Required Deliverables and Artifacts



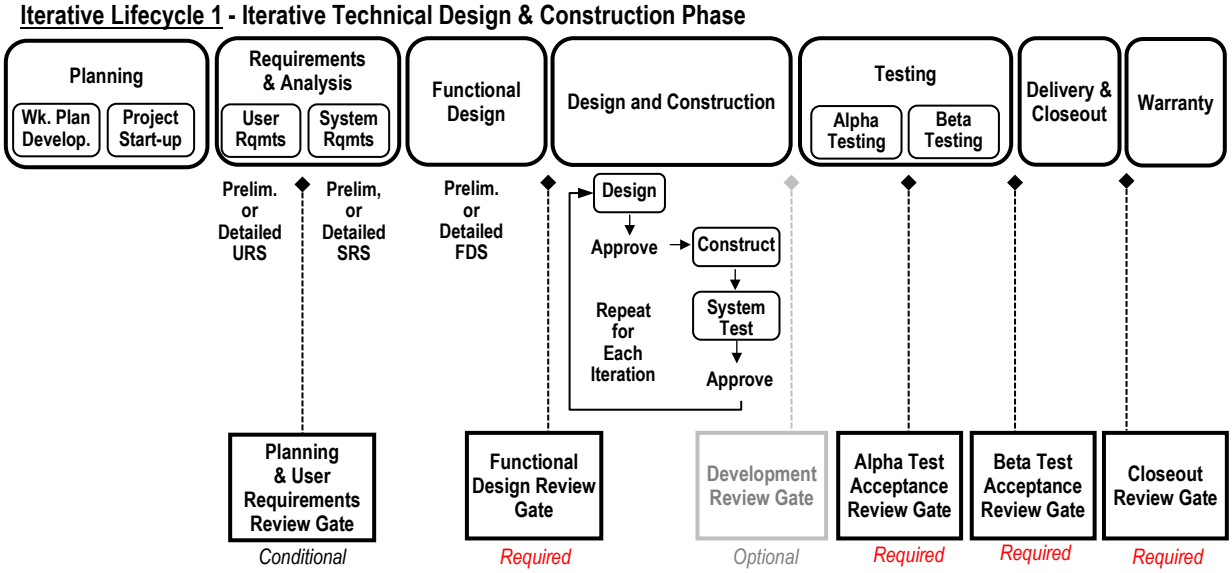

**Deliverables and Artifacts**

| | | | | | | |
|---|---|---|---|---|---|---|
| •Work Plan | •User Requirements Specification (URS)<br>•Technical Processes and Technologies<br>•Management, Monitoring and Control Procedures<br>•Backup and Recovery Plan<br>•##Project Test Plan<br>•Project Schedule | •Preliminary SRS or detailed SRS<br>•Preliminary FDS or detailed FDS<br>•Preliminary RTM#<br>•##Project Test Plan | The following are prepared and approved for each iteration:<br>•Iteration FDS<br>•Iteration Test Results Report<br><br>•SRS and Preliminary RTM# are updated with each iteration<br>•TDS is developed incrementally | The following are submitted with/before Alpha Test Acceptance Gate:<br>•Alpha Test Results Report<br>•Beta Test Materials (includes Beta Test Plan and Installation Package)<br>•Alpha Test Plan<br>•Final RTM# | •Beta Test Results Report<br>•Product Installation Package | •Project Archive Package<br>• TDS*<br>• Development & Maintenance Document*<br>•ACR<br>•Application Infrastructure Component List*<br><br>*Required artifacts |

- The Planning& User Requirements, Alpha Test Acceptance, Beta Test Acceptance, and Closeout review gates are the same as that of the standard waterfall project for all iterative projects.
- The following apply to Iterative Lifecyle1 shown in the above diagram.
  - The Functional Design Review Gate occurs after the Functional Design Phase.
  - The URS, SRS and FDS may be developed at a broad, preliminary level or a detailed level prior to beginning the iterations. In most cases, the FDS is developed at a preliminary level.
  - An Iteration FDS is prepared and approved for each iteration.
  - An Iteration Test Results Report is prepared to document the test results of each iteration.
  - Each iteration is accepted by review gate approval or by the project's deliverable approval procedure.
  - @The Development Review Gate is not required but is recommended after an iterative construction phase.
  - #The Preliminary RTM is updated with each iteration. The Final RTM is submitted with the Alpha Test Acceptance Review Gate.
  - **The Requirements & Analysis and Functional Design Phases or the sub-phases should be prefixed with "Preliminary" when the URS, SRS or FDS is created with a broad level of detail.

## 4.2.2. Iterative Lifecycle 1

The development approach used in this lifecycle is a hybrid type of iterative development approach that uses both waterfall and iterative methods across the Requirements, Design, Construction, and System Testing activities. The lifecycle includes a non-iterative Requirements & Analysis Phase and a non-iterative Functional Design Phase followed by an iterative Design and Construction Phase. The lifecycle allows the requirements and functional design to be defined at preliminary level or a detailed level prior to beginning the Design and Construction iterations.




### 4.2.2.1. Planning Phase

As with a water fall development project, the Planning Phase is divided into two sub-phases; with work plan preparation and approval in the Work Plan Development sub-phase; and formal start-up, planning and mobilization in the Project Start-Up sub-phase.

*The standard AASHTOWare Project Work Plan Template is used to develop the work plan. The work plan should describe the specific iterative development approach, lifecycle with iterative phases, development activities within each iterative phase, number segments/iterations that will be used, the scope of each segment, and test approach that will be used for the project. The type and scope of deliverables that will be created prior to the iterations and the type of deliverables created during the design, development and testing of the iterations is also defined in the work plan. The method for approving the iteration deliverables is also defined.*

The *Procedures* described in the Planning Phase section of Chapter 2 should be used to prepare and approve the work plan; and to plan and execute the project start-up activities, including the update and approval of work plan components, setup activities, and begin management, monitoring and control activities.

### 4.2.2.2. Requirements and Analysis Phase

Similar to the waterfall project development process, the Requirements and Analysis Phase for this iterative lifecycle is divided into two sub-phases.

#### 4.2.2.2.1. User Requirements Sub-Phase

During this phase, the URS is developed or revised at the normal, detailed level used with waterfall projects or at a preliminary, high level of detail. The preliminary, high

level URS is developed in those cases, where the user requirements are developed incrementally as the project progresses.

If a detailed level of URS is defined or revised, the following activities are performed during the User Requirements sub-phase as described in Chapter 2.

- *Review, Analyze and Validate User Requirements*
- *Revise URS*
- *Review and Approve Final URS*
- *Create Initial Preliminary Requirements Traceability Matrix*

If no user requirements exist, the project will include activities to define and document the URS. In this case, refer to the user requirement activities in the *Define and Approve User Requirements* section of the *Requirements/Design Development Process*.

Although the URS is created at a detailed level, the iterative process may still introduce revisions and/or additions during the following phases, including the iterative Design and Construction phase.

If a preliminary, less detailed version of the URS is defined, then the activities in this sub-phase are considered as the first steps in the incremental development of the project's user requirements. This type of URS is referred to as a Preliminary URS to help clarify the scope and content; however, this naming convention is not required. If needed, the sub-phase may also be referred to as the Preliminary User Requirements sub-phase.

- The Preliminary URS normally has a limited number of high level or broad user requirements.
- The requirements in the Preliminary URS should provide sufficient detail to proceed with the System Requirements sub-phase and the Functional Design Phase.
- When a Preliminary URS is defined, the assumption is that the user requirements will be revised and additional user requirements discovered during the following phases.

The above activities may also be performed when refining a Preliminary URS that was included in the work plan.

#### 4.2.2.2.2. Submit and Approve Planning and User Requirements Review Gate

If the URS (or Preliminary URS) is revised or defined during the User Requirements sub-phase phase or if work plan components were modified during the Project Start-Up sub-phase, the Planning and User Requirements Review Gate shall be scheduled, initiated, and approved. This review gate is scheduled at the end of the User Requirements sub-phase when the URS is defined or revised during the execution of the project. The review gate approval request is submitted and approved as described in the *Submit Planning and User Requirements Review Gate* and *Approve Planning and User Requirements Review Gate* sections in the Requirements and Analysis Phase of Chapter 2.

If this review gate is not scheduled, the contractor should begin executing the System Requirements sub-phase.

#### 4.2.2.2.3. System Requirements Sub-Phase

During this phase, the SRS is developed or revised at the normal, detailed level used with waterfall projects or at a preliminary, high level of detail. As with the URS, the preliminary, high level SRS is developed in those cases where the system requirements are developed incrementally as the project progresses.

If a detailed level of SRS is defined or revised, the following activities are performed during the System Requirements sub-phase as described in Chapter 2.

- *Define System Requirements*
- *Prepare SRS*
- *Update Preliminary RTM*
- *Review and Approve System Requirements*

Although the SRS is created at a detailed level, the iterative process may still introduce revisions and/or additions during the following phases, including the iterative Design and Construction phase.

If a Preliminary SRS is defined during this sub-phase, then the activities in this sub-phase are considered the first steps in the incremental development of the project's system requirements. As with the Preliminary URS, the "Preliminary" prefix is added for clarification and may also be added to the sub-phase name.

- The requirements in the Preliminary SRS normally focus on high level functionality and on broad requirements that apply to the overall proposed product, such as user interface, security, accessibility, technical architecture, and internal/external software interface requirements.
- *Although the level of detail will be less in a Preliminary SRS, each type of system requirement component described for a System Requirements Specification (SRS) in Chapter 5 shall be considered when developing the Preliminary SRS.*
- The incremental development of the SRS will continue with revisions and the discovery of additional user requirements during the following Functional Design phase and the Iterative Design and Construction phase.
- A limited version of the above activities may also be performed when refining a Preliminary SRS that was included in the work plan.
- *Each system requirement in the Preliminary SRS is entered into the RTM and referenced to its source user requirement.*

*Regardless if the SRS is developed at a preliminary or at a detailed level, it shall be approved prior to, or with, the Functional Design Review Gate; and prior to beginning the first iteration.*

#### 4.2.2.2.4. Functional Design Phase

During this phase, the FDS is developed or revised at the normal detailed level used with waterfall projects or at a preliminary, high level of detail. As with the URS and FDS, the preliminary, high level FDS is developed in those cases where the functional design is developed incrementally as the project progresses.

If a detailed level of FDS is defined or revised, the following activities are performed in the same manner as those in the Functional Design sub-phase described in Chapter 2.

- *Update and Refine the SRS*
- *Develop the Functional Design*
- *Select and Document Initial Technical Architecture*
- *Prepare Functional Design Specification (FDS)*
- *Update the Requirements Traceability Matrix (RTM)*
- *Obtain Stakeholder/Task Force Review & Approval of FDS*

Although the FDS is created at a detailed level, the iterative process may still introduce revisions and/or additions during the iterative Design and Construction phase.

If a Preliminary FDS is defined during this sub-phase, then the activities in this sub-phase are considered the first steps in the incremental development of the project's functional design. As with the Preliminary URS and SRS, the "Preliminary" prefix is added for clarification and may also be added to the sub-phase name.

- The Preliminary FDS should address the overall proposed product and only provide limited details on areas of the design that are specific to a single iteration.
- The level of detail in the Preliminary FDS should be sufficient to provide the task force and stakeholders with a preliminary understanding of how the system will work.
- *Although the level of detail is less than that of the normal FDS, the contractor shall consider each of the required FDS content areas listed with the Functional Design Specification (FDS) in Chapter 5 when developing the Preliminary FDS.*
- Example content of a Preliminary FDS includes a preliminary domain diagram; process diagrams that describe the proposed system down to the level of the iterations, internal and external interfaces, data stores; an initial data dictionary; example screens, menus/UI navigation and reports; high level descriptions of the security and interface designs; and high level technical architecture diagrams.
- If a Preliminary SRS is created, the Preliminary FDS will primarily focus on design specifications that support the system requirements in the Preliminary SRS.
- The incremental development of the FDS will continue with revisions and additions to functional design specifications during the Iterative Design and Construction phase.
- A limited version of the above activities may also be performed when refining a Preliminary FDS that was included in the work plan.
- *References to each design element in the Preliminary FDS are added RTM with traceability to the source system requirement.*

*Regardless if the FDS is developed at a preliminary or at a detailed level, it shall be approved prior to, or with, the Functional Design Review Gate; and prior to beginning the first iteration.* In most cases, a stakeholder group (TAG/TRT) will review the FDS prior to the task force review.

#### 4.2.2.2.5. Submit and Approve Functional Design Review Gate

*The Functional Design Review Gate approval request is submitted after the SRS (preliminary or detailed) and FDS (preliminary or detailed) are completed and prior to beginning the iterative Design and Construction phase.* The review gate approval request is submitted and approved as described in the *Submit Functional Design Review Gate* and *Approve Functional Design Review Gate* sections in the Design Phase of Chapter 2.

*As with waterfall projects, the Preliminary RTM and Project Test Plan are submitted with this review gate.*

#### 4.2.2.2.6. Design and Construction Phase

After the Functional Design Review Gate has been approved, the contractor begins the iterative Design and Construction Phase, which normally includes the detailed design, construction, and system testing of each iteration.

#### 4.2.2.2.6.1. Design Iteration

The design of each iteration begins by reviewing and analyzing the existing requirements in the URS and SRS (preliminary or detailed), and the existing design specifications in the FDS (preliminary or detailed). As each iteration is designed, existing requirements are revised, and new requirements are discovered. The new and revised requirements may be defined specific to the iteration, as well as those that apply the overall system. If detailed requirements were defined prior to the iterations in lieu of preliminary versions, the updates to the requirements will normally be more limited.

*The design of each iteration supports the requirements discovered and revised during the design process, as well as, the requirements and functional design specifications that were defined in prior phases.* Where the Preliminary FDS is broad or applies to the entire project, the design specifications created during the iterations shall be detailed and specifically addresses the scope, objectives, and requirements for that iteration, as well addressing gaps in the design of the overall system. If a detailed FDS was prepared prior to the iterations in lieu of a Preliminary FDS, the updates to the functional design specifications will normally be more limited.

*The test procedures that will be used as the basis for accepting each iteration, are also developed during the design of the iteration. The Preliminary RTM is updated with the iteration test procedures, system requirements, and design elements.* In addition, the overall SRS (preliminary or detailed) should be updated with the new or revised iteration system requirements.

*The detailed iteration functional design specifications are documented in an Iteration FDS document. The system requirements and test procedures for the iteration are also included in or attached to the Iteration FDS. The required content for the Iteration FDS is defined with the Functional Design Specification (FDS) in Chapter 5.* As noted in this section, the deliverable is not required to be titled or named Iteration FDS.

#### 4.2.2.2.6.2. Review and Approve Iteration Design

As each Iteration FDS is completed, it should be reviewed by the task force, and if available, by a stakeholder group (TAG or TRT). *After these reviews, each Iteration FDS is approved by the task force using the project's Deliverable Review and Approval procedure* or *Review Gate Approval Procedure. The approval authorizes the contractor to proceed with the construction and testing of the iteration.*

*The TDS is also developed during this phase and is normally created incrementally as the iterations are designed and constructed. The required content of the Technical Design Specification (TDS) is defined in Chapter 5, and the required content for the Data Dictionary is defined in the Common Artifacts Standard.*

#### 4.2.2.2.6.3. Construct and System Test Iteration

After each Iteration FDS is approved, the contractor proceeds with the construction of the iteration. *Following the construction, the contractor performs a system test using the test procedures in the Iteration FDS. The process used for iteration testing is documented in the Project Test Plan.*

*The results of the iteration test are documented in an Iteration Test Results Report. The required content for the Iteration Test Results Report is defined in Chapter 5.* As noted in this section, the deliverable is not required to be titled or named Iteration Test Results Report.

4.2.2.2.6.4. Approve Iteration

After each iteration is completed and tested, the Iteration Test Results Report should be reviewed by the task force, and if available, by a stakeholder group (TAG or TRT). *After these reviews, the iteration is approved by the task force using the project's Deliverable Review and Approval procedure* or *Review Gate Approval Procedure*.

**4.2.2.2.7. Submit and Approve Development Review Gate (Optional)**

The Development Review Gate is optional as with waterfall development projects; however, it is recommended that this review gate be scheduled at the completion of an iterative Design and Construction phase or iterative Construction phase to acknowledge the completion of all iterations and a successful system test. In this case, the Development Review Gate would occur after the last iteration in the Design and Construction phase. Approval authorizes the contractor to begin Alpha Testing.

When this review gate is scheduled, the *Approve Iteration* activity for the last iteration may not be needed if agreed to by both the contractor and task force.

Refer to the *Development Review Gate* and *Submit and Approve Development Review Gate (Optional)* sections for additional information on this review gate.

### 4.2.2.3. Testing Phase

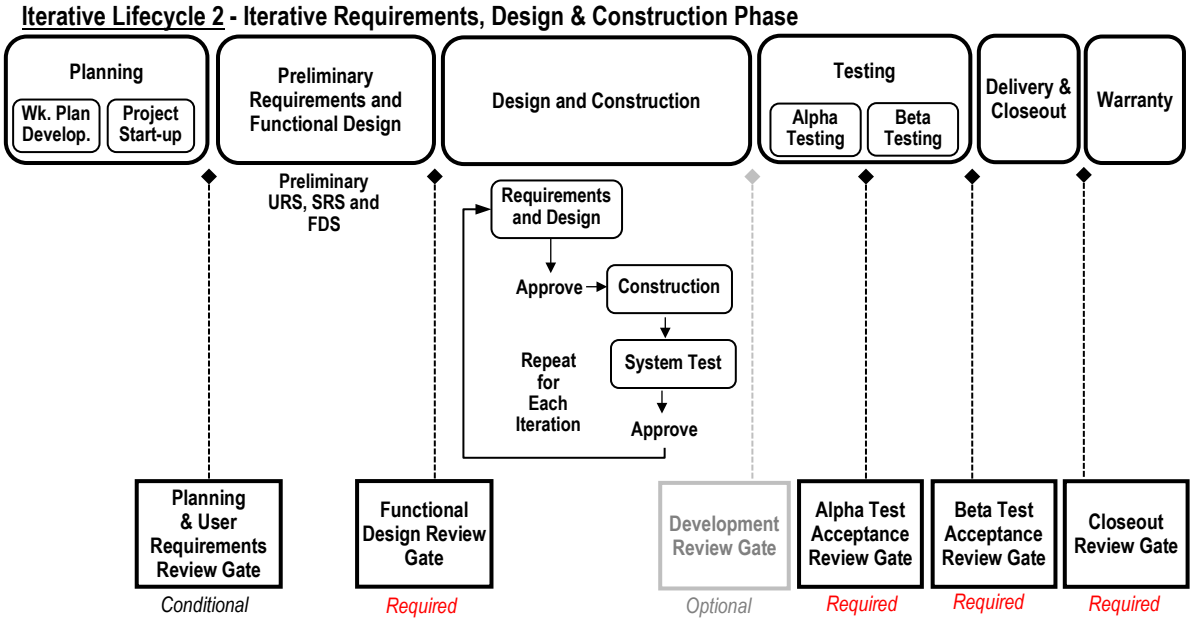
*After all iterations are tested and approved, the remainder of the project lifecycle, beginning with the Testing Phase, is the same as a waterfall development project. An alpha test shall be performed on the complete product from the test procedures in the Alpha Test Plan. The Alpha Test Plan includes a composite of all test procedures that were used during the testing of the iterations. The Procedures described in the Testing Phase section of Chapter 2 are used to complete the Testing Phase, including preparing the Beta Test Materials, approving the Alpha Test Acceptance Review Gate, performing Beta Testing, preparing the Beta Test Results Report, and approving the Beta Test Acceptance Review Gate.*

### 4.2.2.4. Delivery and Closeout Phase

*The Delivery and Closeout Phase is the same for projects using an iterative development methodology as those using a waterfall development methodology. The Procedures described in the Delivery and Closeout Phase section of Chapter 2 are used to complete this phase and closeout the project, including distributing the Product Installation Package, preparing or updating the ACR and Application Infrastructure Component List, preparing the Project Archive Package, approving the Closeout Review Gate, and sending the ACR and Project Archive Package to AASHTO.*

## 4.2.3. Iterative Lifecycle 2

The development approach used is this lifecycle follows a more traditional approach to iterative development than that of Lifecycle 1, where only a limited number of broad, high level, preliminary requirements and functional design specifications are defined before beginning the iterative Design and Construction Phase.




### 4.2.3.1. Planning Phase

The Planning Phase for Iterative Lifecycle 2 is the same as the *Planning Phase* that for Iterative Lifecycle 1.

### 4.2.3.2. Preliminary Requirements and Functional Design Phase

This phase is similar to the Requirements & Analysis and Functional Design Phases in Iterative Lifecycle 1; however, in this phase the URS, SRS and FDS are always created at preliminary level.

- Only a limited number of broad, high level, preliminary requirements and functional design specifications are defined before beginning the iterative phase;
- The preliminary user and system requirements will be refined, and additional requirements will be discovered while iteratively developing the detailed Functional Design specifications for each iteration; and
- Additional refinement may also occur again during the Technical Design, Construction and System Test activities for each segment.

The same basic activities should be followed for defining the Preliminary URS, SRS and FDS as those used in Lifecycle 1. The preliminary requirements and functional design development activities are described in the following sections in Lifecycle 1.

- *User Requirements Sub-Phase*
- *System Requirements Sub-Phase*
- *Functional Design Phase*

*The Preliminary URS, SRS and FDS is be documented in the same manner as the detailed versions described in Chapter 2*; however, the content should be adjusted appropriately for the scope and type of requirements and functional design specifications defined. The contractor may also choose to combine these three preliminary deliverables into a single deliverable.

*The RTM is defined and updated in the same manner as other projects with entries for each user requirement, system requirement and FDS design element.* These activities are defined in the following sections of Chapter 2.

- *Create Initial Preliminary Requirements Traceability Matrix*
- *Update Preliminary RTM*
- *Update the Requirements Traceability Matrix (RTM)*

The Planning and User Requirements Phase is not needed for approving the Preliminary URS, however, it should be scheduled when needed to approve work plan elements that were defined or revised during Project Start-Up.

*The Functional Design Review Gate approval request is required and should be submitted after the Preliminary URS, SRS and FDS are all completed and the SRS and prior to beginning the iterative Design and Construction phase.* The review gate approval request is submitted and approved as described in the *Submit Functional Design Review Gate* and *Approve Functional Design Review Gate* sections in the Design Phase of Chapter 2.

#### 4.2.3.3. Design and Construction Phase

The Design and Construction phase is executed in a similar manner to that described in Lifecycle 1. Since no detailed requirements are created prior to this phase, the first sub-phase is referred to as the Requirements and Design sub-phase, where Lifecycle 1 refers to this sub-phase as the Design sub-phase.

The Requirements and Design sub-phase in Lifecycle 2 focuses on revising and refining the preliminary requirements and discovering new requirements as each iteration is designed. Also, since the FDS is also only defined at a preliminary level prior to the iteration, this sub-phase also focuses on the detailed functional design of each iteration. The same basic activities listed below from Lifecycle 1 are performed to *produce a detailed functional design for each iteration using the Preliminary URS, SRS, and FDS, develop an Iteration FDS for each iteration, and obtain task force approval of each iteration functional design.*

- *Design Iteration*
- *Review and Approve Iteration Design*

While the Requirements and Design sub-phase for each iteration are executed in more detail that that of Lifecycle 1, the Construction and System Testing sub-phase for each iteration are executed the same as those in Lifecycle 1 following the activities listed below.

- *Construct and System Test Iteration*
- *Approve Iteration*
- *Submit and Approve Development Review Gate (Optional)*

*As a result of the above activities, each iteration is constructed and system tested, an Iteration Test Results Report is created for the iteration, and the constructed and tested iteration is approved by the task force.*

#### 4.2.3.4. Testing Phase

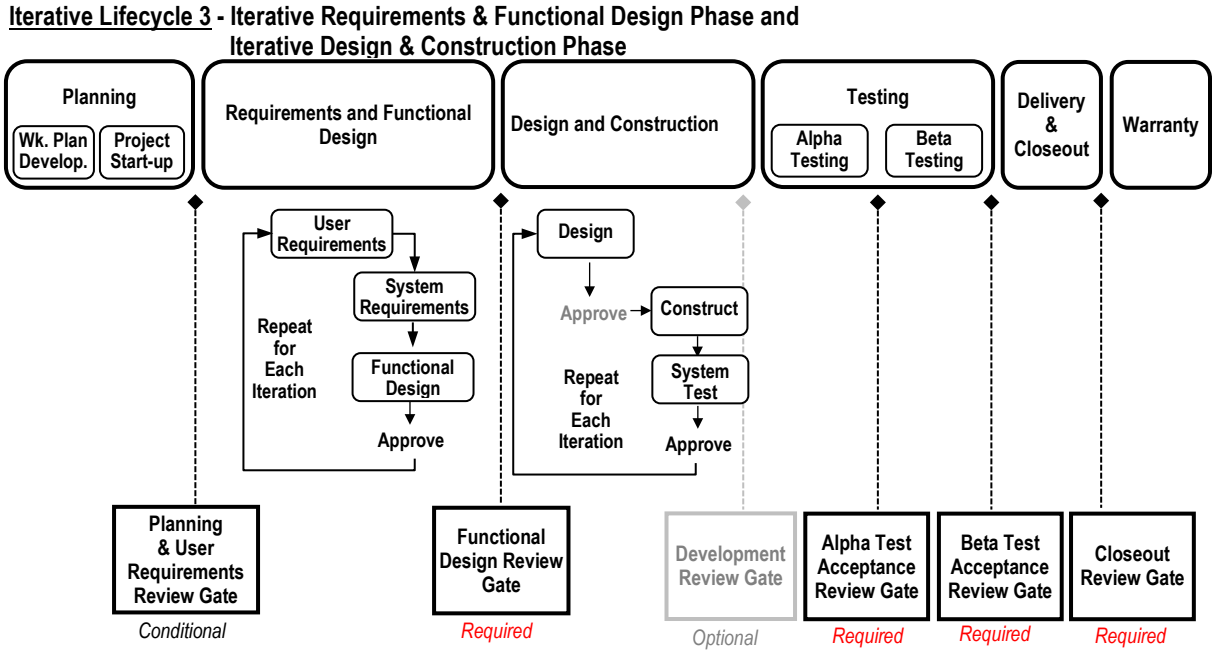
The Testing Phase for Iterative Lifecycle 2 is the same as the *Testing Phase* for Lifecycle 2.

#### 4.2.3.5. Delivery and Closeout Phase

The Delivery and Closeout Phase for Iterative Lifecycle 2 is the same as the *Delivery and Closeout Phase* for Lifecycle 1.

### 4.2.4. Iterative Lifecycle 3

The development approach used is this lifecycle uses iterative development for all User Requirement, System Requirement, Functional Design, Technical Design, Construction and System Testing activities. The lifecycle includes an iterative Requirements and Functional Design Phase followed by an iterative Design and Construction Phase.




#### 4.2.4.1. Planning Phase

The Planning Phase for Iterative Lifecycle 2 is the same as the *Planning Phase* that for Iterative Lifecycle 1.

#### 4.2.4.2. Requirements and Functional Design Phase

This phase is different from the Requirements and Functional Design Phases used in Lifecycle 1 and 2. During this phase the user requirements, system requirements and functional design specification are all created incrementally by iteration using the same basic activities described in Lifecycle 1.

- *User Requirements Sub-Phase*
- *System Requirements Sub-Phase*
- *Functional Design Phase*

*An Iteration FDS is created with each iteration which includes the system requirements, functional design specifications and test procedures for the iteration.* Refer to *Functional Design Specification (FDS)* for the content requirements of the Iteration FDS.

*The URS and Preliminary RTM are to be updated incrementally as each iteration is completed.* Refer to the *User Requirements Specification (URS)* and the *Requirements Traceability Matrix (RTM)* for the content requirements of these deliverables. The contractor and task force may also choose to document the user requirements in the Iteration FDS.

*After each iteration is completed, the Iteration FDS is approved by the task force using the project's Deliverable Review and Approval procedure* or *Review Gate Approval Procedure. The current version of the URS and Preliminary RTM are included or referenced with the approval request.* This approval is not needed with the last iteration since Functional Design Review Gate approval is required after the last iteration as described below.

*The Functional Design Review Gate is required after the last iteration of this phase. The contractor must submit the Functional Design Review Gate approval request and include or reference the URS, Preliminary RTM, and the Iteration FDS for the last iteration. If not previously submitted, the Project Test Plan is also submitted of referenced with the review gate request.* The review gate approval request is submitted and approved as described in the *Submit Functional Design Review Gate* and *Approve Functional Design Review Gate* sections in the Design Phase of Chapter 2.

#### 4.2.4.3. Design and Construction Phase

The Design and Construction Phase for Iterative Lifecycle 2 is generally the same as the *Design and Construction Phase* for Lifecycle 1.

- *An Iteration FDS is created for each iteration and approved by the task force.*
- *An Iteration Test Results Report is created for each iteration, and the constructed and tested iteration is approved by the task force.*

#### 4.2.4.4. Testing Phase

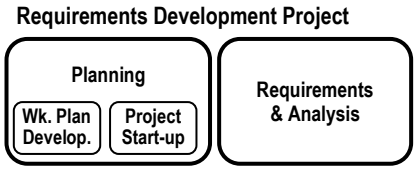
The Testing Phase for Iterative Lifecycle 3 is the same as the *Testing Phase* for Lifecycle 1.

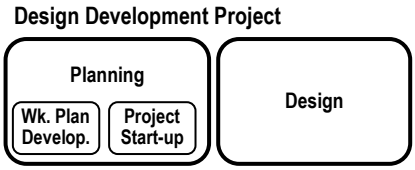
#### 4.2.4.5. Delivery and Closeout Phase

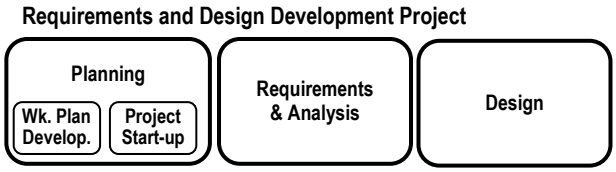
The Delivery and Closeout Phase for Iterative Lifecycle 3 is the same as the *Delivery and Closeout Phase* for Lifecycle 1.

## 4.3. Requirements/Design Development Process

This section describes a standard adaption of the project lifecycle and development process for projects that are limited to the development of requirements and/or design specifications. This type of project includes no software development and, subsequently, includes no testing and implementation activities. The lifecycle includes the Planning Phase and, depending on the objectives of the project, the Requirements & Analysis Phase or the Design Phase, or both phases, as depicted in the diagrams below.

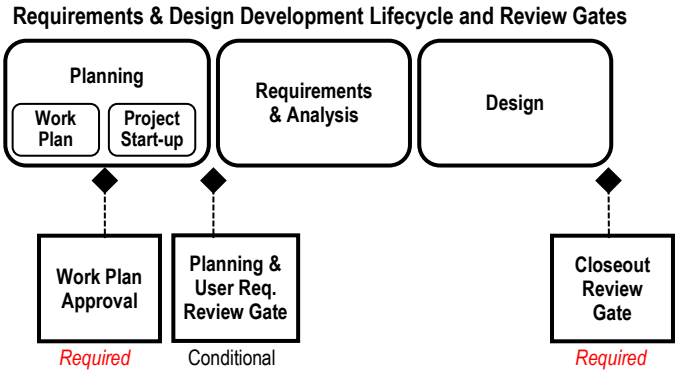



### 4.3.1. Review Gates, Deliverables and Artifacts

In most cases, there will be no review gates other than the conditional Planning and User Requirements Review Gate and the Closeout Review Gate as shown below.




The deliverables will also be limited by the project objectives, to a URS, SRS, and/or FDS, plus any work plan components modified during Project-Start-up. In addition, the work plan will not include the components normally associated with software development.

### 4.3.2. Planning Phase

*The Planning Phase is similar to other projects with work plan preparation and approval in the first sub-phase; and formal start-up, planning and mobilization in the second sub-phase.* Since this type of project does not include construction, testing, and implementation; the scope of the work plan is limited when compared to that of a software development work plan. The work plan for this type of project should be condensed and adjusted to fit the specific goals and objectives of the project. For example:

- If the goal of the work plan is to develop a set of user requirements for a future project; no user requirements will be included in the work plan and the URS may be the only deliverable planned.

- If the goal of the work plan is to develop a set of system requirements for a future project; user requirements would be defined in the work plan and the SRS may be the only deliverable planned.
- If the goal of the project is to develop functional design specifications; user requirements and system requirements would be defined in the work plan, and the FDS may be the only deliverables planned.
- If the goal of the project is to develop system requirements and functional design specifications; user requirements would be defined in the work plan and the SRS and FDS may be the only deliverables planned.

*In all the above scenarios, the Closeout is the only review gate required; and the work plan will only need to define those technologies and procedures applicable to the scope of the project and the amount of project management, monitoring, and control required.* The AASHTO PM will provide guidance on the level of project management, monitoring, and control needed for this type of project.

As with all other projects, the Project Start-Up sub-phase may include planned activities to define or revise a component of the work plan. *In this case, the Planning and User Requirements Review Gate is scheduled, initiated, and approved.*

## 4.3.3. Requirements & Analysis Phase

Depending on the goals of the project; this phase may not exist; it may exist with both the User Requirements and System Requirements sub-phases, or it may exist with only one of these sub-phases.

### 4.3.3.1. Define and Approve User Requirements

If the project is to define user requirements, the AASHTO PM, the contractor project manager, or a task force member should facilitate the collection, analysis, documentation, and approval of user requirements. Since the previous chapters have only defined activities for reviewing and updating previously defined user requirements, the following activities are provided as a guide.

#### 4.3.3.1.1. Elicit Business and User Needs and Expectations

Normally the first step in developing user requirements is to collect a list of prioritized business/user needs and expectations for the proposed product. High level security/access requirements, known constraints, reporting requirements, and required interfaces with other business processes or systems should be also be identified and documented.

These items should be collected by direct requests to the user and business participants of the project, as well as, using eliciting techniques to proactively identify additional needs and other items not explicitly provided by user/business participants. Many of these items may also be captured by reviewing and analyzing existing business processes and systems of the user/business participants.

#### 4.3.3.1.2. Prepare URS

After documenting the initial list of needs, expectations, constraints, security requirements, and report and interface requirements, these items should be compiled into a set of user requirements into the form of a *User Requirements Specification (URS)* as defined in *Chapter 5*.

#### 4.3.3.1.3. Review, Analyze and Validate URS

After the initial URS is created, all project participants should review and analyze the requirements and determine if any conflicts exist and verify the need and priority of each requirement. As a result of this above analysis, all conflicts and invalid

requirements should be eliminated, existing requirements may need to be revised, new requirements may need to be added, and the priority of each requirement should be verified or updated. All participants should also have common understanding of the intent of each user requirement. Refer to the *Review, Analyze and Validate User Requirements* and *Revise URS* sections for additional details.

#### 4.3.3.2. Review and Approve URS

After the above activities are completed the completed URS should be reviewed and approved by the project task force. If a group of stakeholders, such as a TRT or TAG, is available, it is recommended that this group participate in the above review and analysis of the URS prior to the task force review. Refer to the *Review and Approve Final URS* section for additional details on these activities.

#### 4.3.3.3. Define and Approve System Requirements

If the project is to define system requirements, the contractor staff should define and document the System Requirements Specification (SRS) and facilitate the review and approval of the SRS deliverable using the activities described in the *Requirements & Analysis Phase* section of Chapter 2.

#### 4.3.3.4. Develop and Approve Functional Design

If the project is to develop the functional design, the contractor staff should develop and document the Functional Design Specification (FDS) and facilitate the review and approval of the FDS using the activities described the *Design Phase* section of Chapter 2.

### 4.3.4. Closeout

*After the planned deliverables have been completed and approved, the project is closed with the submission and approval of the Closeout Review Gate Approval Request. The requirements and design deliverables (URS, SRS, and/or FDS) are submitted with this review gate if not approved during the previous phase(s). Although limited in comparison to a software development project or MSE work, a Project Archive Package is prepared and submitted to AASHTO. The Project Archive Package includes all deliverables, artifacts, and documentation produced during the project.*

## 4.4. Other Adaptions

*Other adaptions to the process, lifecycle, review gates, deliverable and artifacts may be made if an exception is documented in the project or MSE work plan and approved by the task force and SCOA. The work plan shall describe the elements of the planned work that will not be compliant with this standard and should include a justification for the exception.* Some possible exceptions to this standard are listed below:

- Eliminating required deliverable or artifact
- Eliminating required content on a deliverable or artifact
- Eliminating or combining required approvals or review gates

*There may also be adaptions that do not require an exception to be approved, but still need to be clearly documented in the approved work plan.* This could include using a lifecycle model or software development methodology that does not fit the development processes defined in this standard; yet still includes the required review gates, deliverables and artifacts.

There also may be minor adaptions such as planned overlapping of lifecycle phases or sub-phases. This type of adaption does not need an exception and may not need to be documented in the work plan; however, the overlaps should be depicted in the project schedule.

# 5. Deliverable and Artifact Definitions

## 5.1. Introduction

This chapter includes a description and the required content for each required deliverable and artifact defined in the Software Development and Maintenance Standard. The deliverables and artifacts listed are in the order they are prepared during the project and MSE lifecycles.

*In addition to the content listed below, each deliverable and artifact shall include the appropriate document identification information including the Project/Product Name, Contract Period, Version Number, and Date.* If needed, an introduction section that explains the purpose of the deliverable or artifact should be included.

## 5.2. Work Plan

See the *Common Artifacts Standard*.

## 5.3. Review Gate Approval Request

See the *Common Artifacts Standard*.

## 5.4. User Requirements Specification (URS)

### 5.4.1. Description:

*The User Requirements Specification (URS) is a required deliverable that contains all the user requirements that are approved by the task force to be accomplished in a specified contract period.* The user requirements define what the users and other business stakeholders expect from the proposed product and primary acceptance criteria for the project. For MSE work efforts, the URS normally contains descriptions of enhancements to an existing product. In most cases, the URS is included or referenced in the work plan; however, there are also cases where the URS is revised or fully created as a deliverable of a project or MSE work effort. The URS may also be composed of or include references to requirements documented external to the work effort, such as those included in technical or scientific publications.

### 5.4.2. Content

The primary content of the URS should be the information that describes the user requirements and/or enhancements. *Each requirement or enhancement in the URS shall include the content listed below.* Additional information may be included as required.

- *Requirement ID: The number or tag that uniquely identifies each requirement or enhancement.*
- *Description: The full description of the requirement or enhancement.*
- Short Description: An optional short description which describes the content of the requirement or enhancement but is short enough to use in tables and listings.
- Priority: An optional field for defining the business priority for implementing the requirement or enhancement (example - Critical, Urgent, High, Medium, Low).
- Cost: An optional field for defining the estimated cost to implement the requirement or an enhancement or a group of related requirements or enhancements. For enhancements, the task force will typically request a cost to be defined for each enhancement or group of related enhancements. When the cost applies to a group of requirements or enhancements, a method for grouping should also be included.

# 5.5. System Requirements Specification (SRS)

## 5.5.1. Description

*The System Requirements Specification (SRS) is a required deliverable that contains the system requirements for software development projects and for the medium and large size enhancements in an MSE work effort.* The system requirements describe what the proposed product must do to support or implement the user requirements. System requirements may describe functionality or impose constraints on the design or implementation (such as accessibility, performance, interface, security, and user interface requirements). The SRS may be composed of or include references to externally documented requirements, such as those that are published in technical or scientific publications.

*Each system requirement shall be traceable to one or more user requirements.* Multiple system requirements may be created to support and/or clarify each user requirement. There also may be some cases where a system requirement is identical or near identical to its source user requirement.

*The SRS is created differently for waterfall projects, iterative development projects and MSE work. In some of these cases the system requirements are documented with the functional design specifications in an FDS or SRDS deliverable. Each type of system requirements deliverable/documentation is described below. The names used for each type describe the purpose of the deliverable and are not required; however, in most cases the titles and file names should include "SRS" or "Systems Requirements".*

*SRS (detailed, full scope) – This is the basic SRS that is created for projects that use the standard Project Development Process. The SRS includes detailed system requirements that address the full scope of the project and support all the project's user requirements. This type of SRS shall include all the required content in the Content section below.* This is also the normal type of SRS created for a Requirements and Design Development Project.

In some cases, a detailed, full scale SRS is created for iterative projects as described below.

- Preliminary SRS – This type of SRS may be created for an iterative development project prior to the design and construction of the development iterations. When created, the Preliminary SRS is the first set of system requirements created in the iterative development process. These requirements are normally high-level and are not specific to a single iteration. The detailed, iteration specific, requirements are created during the design and construction of the iterations. *This deliverable includes the applicable SRS content listed below.*
- *Iteration SRS – This type of SRS documentation is created for each iteration of an iterative development project.* The system requirements for an iteration should be detailed and should focus on the specific scope, objectives, and user requirements of the proposed iteration. In most cases, the system requirements for an iteration are documented in the Iteration FDS deliverable with the functional design specifications and test procedures for the iteration. The system requirements may also be documented independent of the Iteration FDS as a separate SRS deliverable. *The system requirements for each iteration includes the applicable SRS content listed below.*
- *Enhancement SRS - This type of SRS documentation is created for each medium and large size enhancement in an MSE work effort or for a group of related enhancements.*

*These system requirements expand and clarify what is needed to meet the intent of each enhancement and/or define what needs to be done to implement an enhancement. The system requirements are normally documented with the functional design in an Enhancement FDS or SRDS, as described with the Functional Design Specification (FDS) deliverable; however, they may also be documented as a separate SRS deliverable. Enhancement system requirements include the applicable SRS content listed below.*

### 5.5.2. Content

*A detailed, full scope SRS shall include all the following content.* The other types of SRS should include the type of content that is applicable and adjust the detail for the purpose of the deliverable.

- *Requirement ID: Each requirement included in the SRS shall be identified by a unique number or tag.*
- *Description: The full description of the requirement.*
- Short Description: An optional short description which describes the content of the requirement but is short enough to use in tables and listings.
- *Technical Requirements: The SRS shall contain requirements that define what technical environment are to be supported by the proposed product. These typically address technical constraints. (Examples are requirements which define platforms, databases, etc.).*
- *Functional Requirements: The SRS shall contain functional requirements that describe what the proposed product must do to fulfil the user requirements. Functional requirements should be identified for all functions that will be automated by the proposed product. A function is described as a set of inputs, the behavior, and outputs. The behavior of the functions is normally described by use cases.*
- *Preliminary Data Requirements: The preliminary data requirements include the input and output data requirements for the proposed product.*
- *System Interface Requirements: The system interface requirements describe hardware and software interfaces required to support the implementation or operation of the proposed product.*
- Non-functional requirements should be broken down into types such as reliability, accuracy, performance, scalability, testability, maintainability, security, usability, interface, user interface, design constraints, and implementation constraints. *Security, accessibility, user interface, and performance requirements shall always be included in the SRS.*
  - *Refer to the Security Standard for additional information regarding security requirements. The roles of the various stakeholders that use and support the system are defined in conjunction with security requirements.*
  - *Refer to the Common Artifacts Standard for additional information regarding accessibility and reporting requirements.*

# 5.6. Requirements Traceability Matrix (RTM) (Preliminary and Final RTM)

## 5.6.1. Description

*The Requirements Traceability Matrix (RTM) is a required deliverable for all projects. The RTM describes the backward traceability and forward traceability of the requirements in the URS and SRS. The RTM also includes traceability of each requirement to elements in the functional design and to testing procedures.*

The RTM is created in phases as the project progresses. Until all required elements are added to the RTM, it is referred to as the Preliminary RTM. When completed, the RTM is referred to as the Final RTM.

Note: An RTM is not required for MSE work efforts.

## 5.6.2. Content

*The RTM shall contain the following content.*

- *User Requirement ID: The number or tag that uniquely identifies a user requirement. All requirements from the approved URS shall be included in the RTM and use the same IDs used in the URS.*
- *User Requirement Source: A reference or link to the source of the user requirement, such as the work plan, RFP, or user group enhancement list.*
- *System Requirement ID: The number or tag that uniquely identifies a system requirement. Each system requirement in the approved SRS shall be entered in the RTM, and each system requirement shall trace to a source user requirement.*
- *Design Element Reference: A reference or link to an element in the FDS that was derived from a system requirement. Multiple design elements may be traced to a source system requirement.*
- *Test Reference: A reference or link to the alpha or beta test procedure used to test and accept a user or system requirement. Multiple tests references may be traced to a source requirement.*

The RTM is normally created as a grid with one or more rows for each user requirement and columns for each of the other items. Other items may be added as required.

The reference of system requirements to design elements and test procedures may be documented in other artifacts that are reviewed and approved by the task force. *In this case, a document shall be prepared that describes where the components of the RTM are located and how they are used to define traceability. Each document shall use the same Requirement IDs that are used in the URS, SRS, and RTM.*

# 5.7. Functional Design Specification (FDS)

## 5.7.1. Description

*The Functional Design Specification (FDS) is a required deliverable for projects and MSE work. The FDS documents the design of the proposed product using terminology that can be readily reviewed and understood by the task force, technical review teams (TRTs), technical advisory groups (TAGs), and other business stakeholders.*

The FDS is created differently for waterfall projects, iterative development projects and MSE work. Certain types FDS deliverables include other information in addition to the design specifications. Each type of FDS deliverable is described below. The names used for each

type describe the purpose of the deliverable and are not required; however, in most cases the titles and file names should include “FDS”, “Functional Design” or “Design”.

- *FDS (detailed, full-scope) – This type is the basic FDS that is created for projects that use the standard Project Development Process. The FDS includes detailed design specifications that address the full scope of the project and support all the project’s user and system requirements. This type of FDS shall include all the required content in the Content section below.*
- Preliminary FDS – This type of FDS is created for an iterative development project prior to the design and construction of the development iterations. *The preliminary FDS is created in lieu of the detailed, full scope FDS.* The design specifications in this FDS address the overall proposed product and only provide limited details on areas of the design that are specific to a single iteration. *This deliverable includes the applicable FDS content listed below; however, the specifications are created at a higher level of detail and/or for a broader scope.*
- *Iteration FDS – A FDS deliverable of this type is created for each iteration of an iterative development project. In addition to the functional design specifications, this type of FDS deliverable contains the system requirements and test procedures for the iteration.*
  - *Each iteration FDS focuses on the specific scope, objectives, and user requirements for the proposed iteration. The iteration FDS refines and expands the design specifications included in the Preliminary FDS that apply to the iteration.*
  - *Each Iteration FDS includes the applicable content in the Content section below, excluding the technical architecture requirements which are normally not applicable to a single iteration.* Some of the other FDS content may also not be applicable to specific iterations and should be skipped or noted as "not applicable".
  - *The system requirements for each iteration include the applicable content listed with the System Requirements Specification (SRS) deliverable.*
  - *The test procedures include the same content listed above for the test procedures in the Alpha Test Plan deliverable.*
  - *If system requirements and test procedures are documented in separate documents, these shall be referenced and attached to the iteration FDS.*
- *Enhancement FDS (or SRDS) – This type of functional design document is created for the medium and large enhancements in an MSE work effort. Each enhancement FDS contains the functional design specifications for one or more enhancements and includes the system requirements for the enhancement(s).* Since both the system requirements and functional design are normally included in the same deliverable, this deliverable may also be referred to as a System Requirements and Design Specification (SRDS).
  - *The amount of detail in each enhancement FDS/SRDS will vary based on the size and complexity of the applicable enhancement(s).* The detail should be adequate to construct the enhancements that are addressed by the FDS/SRDS, since a TDS is not required for MSE work.
  - *Each enhancement FDS/SRDS includes the applicable FDS content listed below in the Content section.*
  - *The system requirements in the enhancement FDS/SRDS include the applicable content list with the System Requirements Specification (SRS)deliverable.*
  - If the system requirements are documented in a separate document, this document should be referenced and attached to the enhancement FDS.

### 5.7.2. Content

*A detailed, full scope FDS includes all the following content.* The other types of FDS should include the type of content that is applicable and adjust the detail for the purpose of the deliverable.

- *System Structure Diagram – Include a preliminary diagram of the system structure, such as a domain diagram or functional diagram.*
- *Logical Process Model –Include a process diagram, data flow diagram or another equivalent model that shows the levels of detail necessary to detail a clear, complete picture of the product processes, data flow, workflow, input/output sources, relationships between processes, and interfaces.*
- *Preliminary Data dictionary – Define all known data elements that will be input to and output by the system through user input, displays, reports, imports, exports, and application interfaces.*
- If directed by the task force, the data model shall also be included with the FDS.
- *User Interface and Report Design – Include definitions of the standard format and methods that will be used for the user interface, menus, reports, system messages, and online help throughout the proposed system. In addition, the design includes sufficient mock-ups of display screens, menus, reports, messages, and online help to provide the task force with an understanding of how the user interface/report standards will be applied for implementing the project's user and system requirements.*
- *Preliminary System Interface Design – Describe how the product will interface with other systems.*
- *Preliminary Security Design – Describe the types of users that will have access to the product, the access restrictions for each type of user, and other preliminary security design information.*
- *Preliminary Technical Architecture – Describe the preliminary technical architecture solution selected for the production environment. Diagrams are normally used to depict the overall technical architecture, including the system components (software, hardware, networks, databases, operating systems, etc.) that support the proposed product. Interfaces between components are also shown in the diagrams. In addition, the recommended development tools and standards are described.*

Any other information useful to the task force or contractor may also be added to the FDS. The FDS may be created as a single document or as multiple documents, where a master document references the other documents. For existing systems, required FDS content that has not changed, such as the technical architecture, may be referenced.

Some of the required content may be satisfied by including references to existing standards or documentation from existing products. The referenced information shall be readily available and its use in the design shall be clear and easy to understand by the task force, TRTs, TAGs, AASHTO PM, and other reviewers.

## 5.8. Technical Design Specification (TDS)

### 5.8.1. Description

The Technical Design Specification (TDS) is the final set of design specifications that are used by the contractor's technical staff to code, configure, build, integrate, and test the proposed product and all components, programs, databases, files, interfaces, security controls, screens, and reports. Since the TDS is used by the contractor staff, the TDS may

be produced and packaged in any format acceptable to the contractor, as long as the required content is included. For example, the TDS may be created by:

- Updating the FDS with the required TDS content;
- Creating a supplemental set of design specifications that is used in conjunction with the existing FDS; or
- Creating a complete set of design specifications that is independent of the FDS.

*When considering technologies to include in the technical architecture, the contractor shall consider new versions of technology components and be aware of technology that will soon be outdated, as described in the Critical Application Infrastructure Currency Standard.*

*The TDS is created for all projects but is not required for MSE work. For project work plans that modify or extend an existing product, the minimum requirements are the applicable updates to the TDS content. However, an updated Data Dictionary is required anytime project or MSE work makes an update necessary.*

### 5.8.2. Content

The following is the required content of the TDS:

- *Physical File and Database Structures*
- *Final Data Dictionary* (The Final Data Dictionary is a component of the TDS; the Data Dictionary also has separate requirements. See the *Common Artifacts Standard*.)
- *Final Technical Architecture*

*The TDS also includes or references all other design specifications or updated specifications for program, modules, interfaces, etc. that were created by the contractor staff and were not included in the approved FDS.*

The contractor may also choose to document and maintain some of the final specifications in the *Development and Maintenance Documentation* or the *System Documentation* portion of the Product Installation Package.

## 5.9. Project/Product Test Plan

### 5.9.1. Description

*The Project/Product Test Plan is a required deliverable that defines the test approach that will be used to test the product during construction and to accept the product during alpha and beta testing. The test deliverables that will be produced during the project and the target schedule for the deliverables are also defined.* The Project/Product Test Plan may be incorporated or referenced in the work plan or it may be prepared as a deliverable during the execution of the project or MSE work effort.

### 5.9.2. Content

*The Project/Product Test Plan contains the following content.*

- *System and system components to be tested.*
- *Description of the testing methods to be used, including the testing activities (unit, build, system, alpha, and beta).*
- *List and description of each testing deliverable*
- *Schedule (end date and duration) of each testing activity.*
- *Schedule for submission of each testing deliverable.*

## 5.10. Alpha Test Plan

### 5.10.1. Description

*The Alpha Test Plan is a required deliverable that includes all the materials needed to perform alpha testing and to document the results of alpha testing. The materials identify the system and system components that will be tested; include the requirements that will be tested; and the test procedures and expected results used to perform and measure the test. A format to document the results of the test, discovered exceptions, proposed resolutions, and actual resolutions is provided.*

The test procedures are the primary components of the Alpha Test Plan. A test procedure, also called a test script or test scenario, is a sequence of steps that are executed to test a component or major function of the system for compliance with one or more requirements in the URS and SRS. More complex test procedures may be divided in sub-procedures each with a sequence of steps to be executed. Successful testing of all test procedures against the product is a minimum requirement before a product can be released for beta testing and production. The test procedures used for Alpha Testing are also used in system testing and be used in Beta Testing.

The Alpha Test Plan is referred to as a distinct document; however, the required content may also be included in another deliverable, and in many cases is included as a section of the Project/Product Test Plan.

### 5.10.2. Content

A form or spread sheet is normally used for documenting testing procedures, expected results, and the results of testing. *Any method for documenting the test procedures and capturing the results, which includes all the following information, is acceptable.*

- *Test Procedures – The test materials contain all test procedures that will be used to test the complete system. Each test procedure contains the following items:*
  - *Test Procedure ID: Unique ID of the test procedure.*
  - *Test Procedure Description: Brief description of the test procedure.*
  - *Steps: Description of the steps of each test procedure; or describe sub-procedures for each test procedure and the steps for each sub-procedure.*
  - *Test Data: Description of the files, databases, input data, or other data needed to execute the test procedure, sub-procedure or a step within the procedure or sub-procedure.*
  - *Requirement IDs: IDs of the requirements in the URS and SRS to be tested by the test procedure, sub-procedure, or step within the procedure or sub-procedure.*
  - *Requirement Short Descriptions: An optional item that provides the short descriptions, from the URS and SRS, of the requirements to be tested.*
  - *Expected Results: Description of the expected results from the test procedure, sub-procedure or step with in the procedure or sub-procedure.*
  - *Setup/Instructions: Description of the setup and initialization information for hardware, software, and/or tools that support testing; or special instructions for the tester.*
  - *Activities to validate installation at user sites or hardware outside of the contractor development environment.* The intent is to include sufficient platform/environment installation testing to minimize the chance of installation errors during testing.
- *Alpha Test Results Report Format - The Alpha Test Plan provides a format or form for recording the test results of each test procedure. The following content is included:*
  - Tester Name for each test
  - Test Date of each test

- *Test Procedure Reference: Include or reference the test procedure, sub-procedure, step, and/or expected results to guide the tester through the test procedures.*
- *Results of each test procedure, sub-procedure or step.*
- *Exceptions found for each test: This item is provided when the test results conflict with the expected results.*
- *Resolutions for each exception: The developer records the following items when an exception is found.*
  - *Proposed Resolution: The proposed resolution or resolutions recommended following the analysis of the exception.*
  - *Actual Resolution Performed: Describes the actual resolution or correction made to address the exception.*

## 5.11. Iteration Test Results Report

### 5.11.1. Description

*When an iterative development methodology is used, an Iteration Test Results Report is created after the testing of each development iteration. The report includes the results recorded during the execution of testing.*

As with the Iteration FDS deliverable, this name describes the purpose of the deliverable. The deliverables produced may be titled and named differently.

### 5.11.2. Content

*The Iteration Test Results Report includes the same content as the Alpha Test Results Report listed below.*

## 5.12. Alpha Test Results Report

### 5.12.1. Description

*The Alpha Test Results Report is a required deliverable that is prepared after all alpha testing and exception resolution is complete. The report includes all alpha test results, validated exceptions, and the resolutions performed to correct the exceptions; and references the test procedures and related deliverables.*

### 5.12.2. Content

*The Alpha Test Results Report includes the following content:*

- *All completed information from the Alpha Test Results Report Format (refer to Alpha Test Plan above)*
- *Reference to the test procedures and expected results.*
- *Reference to the URS, SRS, and RTM.*
- "To Be Determined" may be used for proposed resolution, if the resolution/action is not known at the time of the report submittal. In this case, a timeframe for resolution should be provided.

## 5.13. Beta Test Materials

**(Beta Test Plan and Beta Installation Package)**

### 5.13.1. Description

*The Beta Test Materials is a required deliverable that is comprised of two component deliverables, the Beta Test Plan and the Beta Test Installation Package. The Beta Test Plan includes all procedures and instructions needed to plan, prepare, execute, and report progress for beta testing. The Beta Test Installation Package contains all procedures, scripts, executables, and documentation needed to install, implement, and operate the beta product at the beta test site.*

### 5.13.2. Content

*The Beta Test Plan includes the following content:*

- *Test procedures (see Alpha Test Plan).*
- *Test Instructions*
  - *Purpose of the test.*
  - *Description and use of test materials.*
  - *Method for reporting problems and getting help with the test.*
  - *Test schedule.*
  - *Method for reporting product acceptance.*
  - *Technical infrastructure requirements*
  - *Typical business and technical staffing requirements*
  - *Typical issues or barriers that may impact beta testing*
  - *Example work breakdown structure/schedule including tasks for:*
    - *Planning the technical infrastructure*
    - *Establishing the infrastructure*
    - *Identifying and obtaining commitments from business and technical staff*
    - *Planning testing tasks*
    - *Performing beta testing and recording results*
    - *Analyzing test results*
    - *Returning test results to contractor*

*The Beta Test Installation Package includes the following content:*

- *The product and all components and utilities.*
- *Procedures for the installation, implementation, and operation of the product in the beta test environment. Since there may be disparate environments for which the system is to be tested, there may be a need for multiple versions of the Beta Test Installation Package.*
- *The complete content is described below with the Product Installation Package.*

## 5.14. Beta Test Results Report

### 5.14.1. Description

*The Beta Test Results Report is a required deliverable that documents the combined beta testing results of all testing agencies, exceptions discovered, and resolutions to the exceptions.*

### 5.14.2. Content

*The Beta Test Results Report includes the following content:*

- *The combined test results of all beta testing agencies.*

- *Exceptions found during beta testing.*
- *Resolutions for each exception.* "To Be Determined" may be used, if the resolution/action is not known at the time of the report submittal. In this case, a timeframe for resolution should be provided.

## 5.15. Product Installation Package

See the *Common Artifacts Standard*.

## 5.16. Application Infrastructure Component List

See the *Common Artifacts Standard*.

## 5.17. Accessibility Conformance Report (ACR)

See the *Common Artifacts Standard*.

## 5.18. Project/MSE Archive Package

See the *Common Artifacts Standard*.

## 5.19. Data Dictionary

See the *Common Artifacts Standard*.

## 5.20. Development and Maintenance Documentation

### 5.20.1. Description

*The Development and Maintenance Documentation is a required artifact for new development projects.* This documentation, supplemented by the Technical Design Specification, represents the internal documentation for the product, and should describe the logic used in developing the product and the system flow to help the development and maintenance staffs understand how the programs fit together. The documentation should provide instructions for establishing the development environment and should enable a developer to determine which programs or data may need to be modified to change a system function or to fix an error.

Once created, the Development and Maintenance Documentation should be updated, as required, when the existing product is revised by a project or MSE work effort.

### 5.20.2. Content

The content of the Development and Maintenance Documentation is left up to the contractor.

AASHTO AASHTOWare™ SOLUTIONS

# HYBRID AGILE DEVELOPMENT AND MAINTENANCE STANDARD

## S&G Number: 1.006.01.4S

## Effective Date: April 1, 2026

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.1 | 7/19/2022 | Removed the bold font from last year's new requirements. | 7/19/2022 Approved by T&AA |
| 1.2 | 7/12/2023 | Clarification of 10% increase for SCOA review and UAT/regression testing and access to DevOps tools and corrected links. | 10/02/2023 Approved by SCOA |
| 1.3 | 8/07/2024 | Removed the bold font from last year's new requirements | 12/11/2024 Approved by SCOA |
| 1.4 | 5/13/2025 | Updated information about the VPAT/ACR. | 3/12/2026 Approved by SCOA |

# Table of Contents

## Table of Figures

# 1. Purpose

The primary purpose of the Hybrid Agile Development and Maintenance Standard (HADM) is to reduce the time and cost of web, cloud, and mobile software development, ensure that prioritized features are available earlier, reduce the time and cost of software maintenance support, and improve the quality of software produced. These changes will ultimately improve AASHTOWare's competitive position within DOTs and the transportation domain. Beyond establishing a framework for conducting Agile development, this standard identifies the responsibilities of task force members, which are the stakeholders defining software requirements and behaviors, and ultimately the subject matter experts who will define the software products. Additionally, this Agile standard promotes an alternative work input approach for task forces and contractors to sustain the Agile process, delivering new software, software enhancements, and maintenance updates much more quickly than the waterfall-based Software Development and Maintenance Standard (SDMP).

## 1.1. Introduction

The SDMP is very artifact and process heavy because the originating work came from a Capability Maturity Model Integration process-driven approach. Since the development of the SDMP, other development approaches have come into widespread acceptance because they require much fewer artifacts and continuously place program code in front of users to test and provide feedback. This standard supports using a hybrid Agile approach, which reduces significant schedule impacts, artifact burdens, and increased costs the SDMP places on AASHTOWare product contractors and ultimately AASHTOWare.

As product contractors are delivering, or plan to deliver, more updates within a fiscal period, the time and expense needed to meet the SDMP requirements are expected to keep increasing. The current SDMP elevates AASHTOWare's risks of product contractors missing opportunities to be more responsive to AASHTOWare users, sustains an operational climate that causes product contractors not to comply with standards, increases the potential for severe design or development problems to occur, reduces efficiency, and increases costs.

## 1.2. Background

The Software Development and Maintenance Process Guideline was introduced in 2012. In 2013, it became a standard. The new standard replaced the guideline and five then-existing standards: the Deliverable Planning and Acceptance Standard, Requirements Standard, Design and Construction Standard, Testing Standard, and Implementation and Closeout Standard.

The SDMP is heavily waterfall-based. Through the years, three iterative approaches were bolted on to make the standard more flexible. However, while Agile techniques and methods were incorporated, the required deliverables and approval gates remain heavily based on a waterfall approach. The result was a Frankenstein of sorts. The product contractors have some process flexibility but are required to produce deliverables mostly on a waterfall approach.

Another key aspect of the current SDMP is the legacy desktop software testing approach of alpha and beta testing. This delayed testing is due to the desktop nature of the software, and a key impediment to Agile development approaches.

This standard, a hybrid Agile variation, is intended to establish the key requirements that must be followed when developing and maintaining web, mobile, and hosted AASHTOWare software products, describe the process and how all participants (contractors, AASHTOWare, and task force members) will interact, and provide a baseline for future evolution of this Agile standard.

### 1.2.1. The Agile Manifesto

The manifesto was authored by Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew

Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. https://agilemanifesto.org/

Their manifesto states: "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

> *"Individuals and interactions over processes and tools,*
> *Working software over comprehensive documentation,*
> *Customer collaboration over contract negotiation, and*
> *Responding to change over following a plan*

"That is, while there is value in the items on the right, we value the items on the left more."

### 1.2.2. The 12 Principals of the Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

### 1.2.3. Manifesto Background

The seventeen individuals listed at the beginning of this section met on February 11, 2001, and after two days produced a Software Development Manifesto. A brief explanation of that meeting and the outcome is found at https://agilemanifesto.org/history.html.

The significance of the manifesto is that it results partly from the pressures of the new and ever-accelerating pace of the IT marketplace: *"In order to succeed in the new economy, to move aggressively into the era of e-business, e-commerce, and the web…"* AASHTOWare is not immune from market competition and can evolve as other entities have to maintain its place within the transportation domain and adoption by DOTs. The intent of this standard to support that strategy.

## 1.3. Applicability – Web, Mobile, Cloud Hosted Products

Agile software development was and is primarily focused on web, mobile, and cloud-hosted software products. The HADM aligns itself with this type of product as its primary focus and not on desktop-based software products. Additionally, the standard will take advantage of Agile tools' ability to document requirements, design specifications, user involvement, and decisions

made and allow task force participants (product/business stakeholders) to participate fully during development with contractor staff.

Much of the current SDMP standard is based on delivery of, and the legacy artifacts and processes of, desktop software. The SDMP legacy processes and artifacts increase the cost of development, inject delays, and hinder the responsiveness of AASHTOWare and its contractors.

### 1.4. Compare and Contrast Agile Process with the Current Software Development and Maintenance Process

Currently, the SDMP identifies two main types of work. There are different requirements for the two kinds of work. Project work includes developing a new product, a new module for an existing product, or an enhancement or associated requirements or design work that costs $250,000 or more. Maintenance, support, and enhancement (MSE) work includes maintenance and minor enhancements or new features for an existing product.

The HADM promotes an alternate work intake model, relying on level of effort (LOE) based tiers that differentiate which efforts are service and maintenance and which are new development consistent with new products, new modules, or enhancements. Processes and documentation would require more rigor as the number of hours increases through the tiers. See Section *4.1* for more information.

Continuing the comparison, the current SDMP Standard is heavily waterfall-centric and aligns itself to the legacy approach of developing, testing, and delivering *desktop software products*. It requires many review gates and artifacts, all of which add time and cost. Furthermore, the higher cost comes without any measurable decrease in risk, and the code produced has not been proven to be of higher quality.

The current SDMP process (the AASHTOWare SDLC) is represented below:

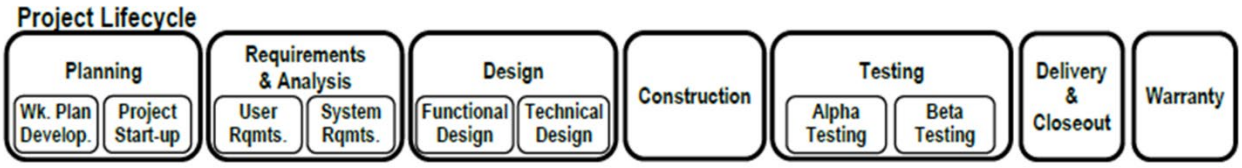



**Figure 1 - Current AASHTOWare SDMP Process**

The process above requires at least nine major documents and up to six formal review gates with their associated documentation. Compared to the Agile model, which still does have approval points, the capabilities of Agile development tools would be used to document participation and approval and to harvest design, development, and testing artifacts. Under this approach, more time would be available for developing and testing the product. *Figure 3* represents the Hybrid Agile Process diagram.

# 2. Task Force Contractor/Responsibilities

### 2.1. Standard Implementation

This standard is written such that the term project is synonymous with a software release and is defined in Sprint 0 and complete after the final sprint. However, a software release may comprise of multiple projects, each with its own Sprint 0 and final sprint. In such cases, some adjustment may be necessary. *Prior to releasing the final product, a final user acceptance test must be completed that covers the scope of the full product.*

## 2.2. Conform to the Hybrid Agile Model

### 2.2.1. Conduct a Sprint 0

Use the outcome of Sprint 0 to facilitate determining the level of effort needed to deliver the software, the acceptable duration for the development effort (timebox the project), and use the time constraint to help establish the most important features/functionality consistent with a minimum viable product deliverable.

### 2.2.2. Conduct a final UAT

Conduct a final user acceptance testing (UAT) sprint two to four weeks before the project's end as defined in Sprint 0.

### 2.2.3. Continuously develop/execute regression tests

Continuously develop regression tests and user tests for features demonstrated on a sprint-by-sprint basis.

### 2.2.4. Use continuous integration (CI)

Use continuous integration or code automation, with consistent regression testing, development of automated user tests, and satisfying the quality gates established by the task force.

### 2.2.5. Adopt code quality analysis tool(s)

Adopt a code quality analysis tool to support code quality objectives, which must be integrated into the CI process such that developed code not meeting quality objectives cannot be promoted as a final product eligible for deployment.

### 2.2.6. Hybrid Agile Model (Scrum) Tasks

Task forces and contractors must execute Scrum tasks that include:

- backlog grooming and sprint planning at least one sprint in advance of the sprint they are going to execute next,
- decomposing user stories and tasks into small work efforts consistent with Scrum (just contractor staff typically),
- holding regular standup meetings,
- providing a sprint demo of code developed at the end of each sprint,
- engaging the task force or its representatives to determine if the product features produced satisfy the requested design goal (user story), and
- providing a mechanism to allow the participating business representatives to test the delivered feature(s) on a sprint by sprint basis.

## 2.3. Agile Development (Scrum)

Contractors and task forces must implement and fully leverage an Agile development tool that supports Scrum (as well as Kanban and other Agile methodologies) for managing and developing software products when following the Hybrid Agile Process. The Agile development tool must provide visibility in the development process and support backlog grooming and normal tracking and performance metrics.

## 2.4. Committed and Engaged Business Stakeholders.

Task forces and contractors must ensure business stakeholders with adequate business knowledge are sufficiently engaged to support Agile development obligations. Agile development methods require active business stakeholder involvement. Without the continuous

and active involvement of business stakeholders, it is difficult to complete the project successfully. This is an extremely critical requirement.

### 2.5. Agile/Scrum Training

Contractors and task forces must ensure appropriate Scrum/Agile training is provided to project participants, both within the contractor staff and the business/task force participants.

### 2.6. Scrum Master

Contractors and task forces must ensure that only technically qualified contractor staff with appropriate Scrum experience and training are placed in the role of scrum lead or scrum master.

## 3. Required Artifacts

The following summarizes the majority of required artifacts that must be created, maintained, and used to support this standard. It should be noted that the Agile development tool adopted by the contractors will be the primary mechanism to create, maintain, and view the artifacts. This tool and its project information need to be accessible to all task force members, participating contractor staff, task force delegates acting as business stakeholders, AASHTOWare staff/contractors, SCOA, and T&AA for all work performed for AASHTOWare.

At the request of the task force, AASHTOWare staff, SCOA, or T&AA, any artifact maintained within the Agile development tool must be accessible to view or be able to extracted either as a consumable file and/or printed as a report to comply with the standard.

- *A project or product maintenance, support, and enhancement (MSE) work plan must be created for each contract period.* See the *Common Artifacts Standard* for more information.
- Review gates, which may be required for all major pay items. See the *Common Artifacts Standard* for more information and the template.
- *Sprint Zero 0 user stories and tasks.*
- *Final Sprint user acceptance tests performed and test outcomes.*
- *All metrics for all sprints, including sprint burndown, epic burndown, release burndown, velocity, and the other typical metrics discussed in Section 5.12.4.*
- *Sprint details, including the user stories in each sprint, tasks, subtasks, story points associated with each user story, the estimated hours for all decomposed tasks and subtasks, and the actual time spent by contractor staff for all work attempted or completed during a sprint.*
- *All backlog content for both the product and projects.*
- *User stories for a specific project, which includes the user stories completed, delivered, and still in the backlog.*
- *Regression tests, as scripted or created in an automated testing product such as Selenium, Katalon, TestCraft, etc.* Test scripts are expected to be created and executed throughout development. Additional scripts are also anticipated to be developed as part of the final sprint, which addresses system-wide testing, including web service dependencies and interface dependencies.

  The Agile development tool will likely not be the source of test scripts or the engine to execute the test scripts. Third-party tools are usually used to perform this function, and the tests created in those platforms should be accessible to all the parties identified in the opening paragraphs of this section.
- *Code quality metrics (used to establish quality gates).*

- *Code defect and defect severity as monitored within the Agile development tool or within an ancillary product for service and maintenance if a separate tool is in use by the contractor.*
- *Project status reports (frequency specified by each task force but not to occur less frequently than once per month)* should include the following items. A status report template is available in Chapter 7 of the *Common Artifacts Standard*.
  - Based on the average velocity attained by the development team and using the total story points for all the user stories in the backlog report, how much time it will require to deliver all features.
  - More importantly, based on the duration established by Sprint 0, the status should report how many story points can be completed within the time estimated for the project and how many story points have been completed.
  - All risks associated with project health, including the participation level of business representatives, technology, contractor performance, and availability.
  - A general description of the features completed, demonstrated, tested, and ready for acceptance.
  - A general statement of what features will be addressed in the next one or two sprints.
  - Other information deemed appropriate by the AASHTOWare project manager and product owners.
- A product installation package is required if the application can be installed by a customer. See the *Common Artifacts Standard* for more information.
- The following required artifacts are defined in the *Common Artifacts Standard*.
  - *Data Dictionary*
  - *Accessibility Conformance Report (ACR)*
  - *Application Infrastructure Component List*
  - *Project/MSE Archive Package*

# 4. Processes and Procedures

## 4.1. Work Intake

An alternate work intake model, relying on level of effort (LOE) based tiers can differentiate which efforts are service and maintenance and which are new development consistent with new products, new modules, or enhancements. The following tiers are promoted:

- All production support, maintenance, and enhancement work under 100 hours;
- All production support, maintenance, enhancement, and new product development work between 100 and 500 hours; and
- All work above 500 hours.

Processes and documentation would require more rigor as the number of hours increases through the tiers. A diagram of the above tiers and associated processes immediately follows.

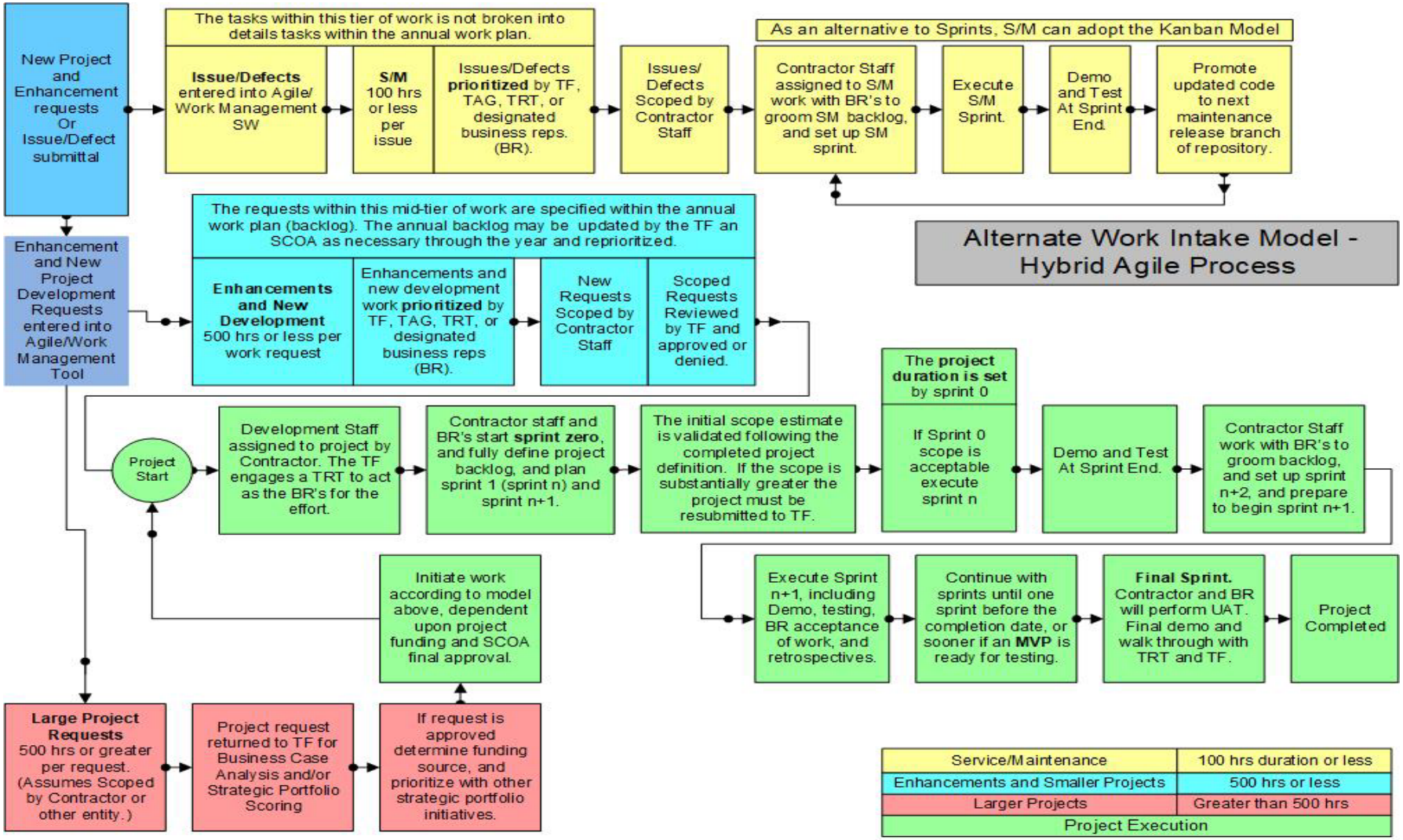



**Figure 2 - Alternative Work Intake Process**

**Example Hybrid Agile Model**

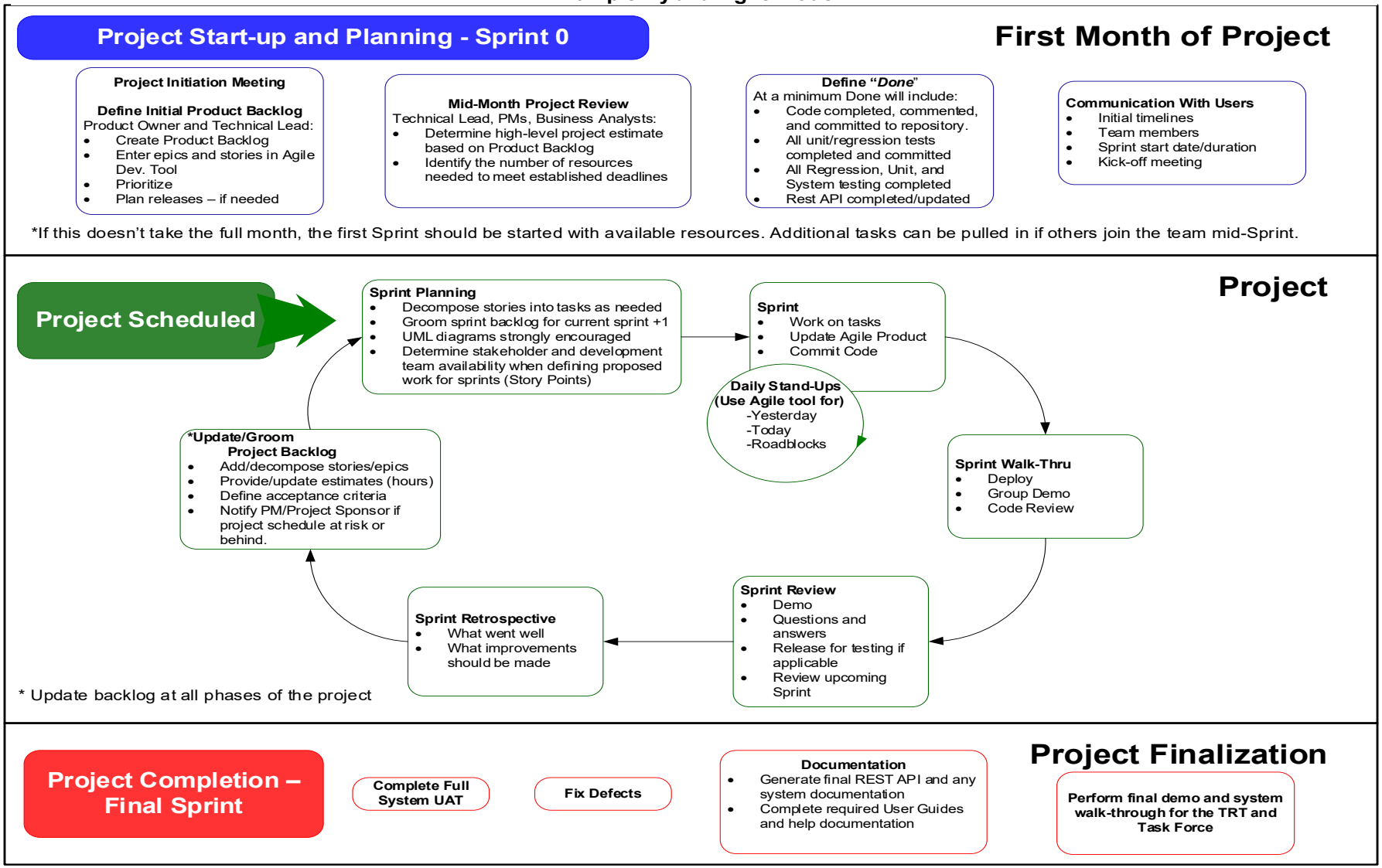



**Figure 3 - Hybrid Agile Development Model**

## 4.2. Development, Enhancement, and Maintenance Work

*The contractor must conform to the Hybrid Agile Development Model in Figure 3. At a minimum, the contractor will complete the following.*

### 4.2.1. Sprint 0 Backlog Development

*Sprint 0 backlog development goals include defining most user stories to produce a minimum viable product and developing work estimates to support a schedule and total project duration.*

- *Validate the initial scoping effort with Sprint 0 estimates.*
- *Plan the first two sprints.*
- *Determine the number of contractor resources needed to meet the desired deadline and define the team.*
- *Conduct a kickoff meeting with business stakeholders and establish a baseline understanding of the Agile process and stakeholder responsibilities.*
- *Establish the target end date based on total work defined and the number of developer resources obligated to the effort. This step defines a maximum time budget (timeboxing) and imposes a constraint on the duration of the project*

## 4.3. The Agile Development Process – Scrum Model

Agile development is an extremely simple development methodology. Simplicity is its strength. The Agile methodology identified and prescribed by this standard is Scrum.

- Identify features/functions of the desired software by descriptive user stories;
- Weight each user story such that larger work efforts can be distinguished from smaller efforts (development team);
- Prioritize the user stories so that the most important/desired features are developed first (product owner and scrum master);
- Decompose the user stories of the sprint into small, discrete tasks that can ideally be completed within a day or two (shorter is better) by the members of the development team;
- Develop software in short work iterations (usually two weeks, but not more than four weeks) called sprints;
- Demonstrate the software to the product owner and business participants at the end of each sprint,
- Business participants test features and validate the delivered design following each sprint demo;
- Developers perform a retrospective after each sprint to discuss what worked well and what needs to improve;
- During each sprint, the development team conducts very short stand up meetings to state what they completed the day before, what they are working on today, and any issues encountered (blockers);
- The product owners and scrum master prioritize the remaining user stories and tasks following the completed sprint and identify what work will be in the next sprint for the development team to complete.

This process repeats until the product owner has the most important features requested but usually not all the features. This model delivers the most value in the least amount of time.

### 4.4. Execute the Agile Development Process (Scrum Model)

- *Conduct each sprint, consistent with accepted Agile practices (and aligned with this standard), including continuous development of unit and regression tests.*
- *Plan sprints a minimum of one sprint ahead of the current (N) sprint. In other words, when sprint N is completed, begin the N+1 sprint and plan sprint N+2. A preference is to plan multiple sprints ahead to address issues earlier, establish better communication of intent with stakeholders, sustain or improve velocity, and deliver a minimum viable product within the schedule.*
- *Web applications developed under this standard will not be promoted for user acceptance testing until all blocking, critical, and major severity defects are corrected.*

### 4.5. Execute Final Sprint

*Execute a final sprint before the project deadline that will include all unit, regression, functional, system, and API tests. Also, complete a full UAT and address all remaining blocking, critical, and major defects before promoting for release. Deliver all documentation, including the REST API, and perform a final system walk-through and final demo with the task force and the technical review team (TRT).*

## 5. Technical Requirements and Recommendations

This standard specifies what task forces and contractors must do and what they are to produce if they choose to develop under this model. The following is a succinct listing of the requirements of this standard, and what is recommended or required is meant to be consistent with the spirit of Agile development.

### 5.1. Hybrid Agile and Engaged Stakeholders

The hybrid Agile development approach requires active business stakeholder involvement and participation with the development team. A major consideration of whether to use this standard's hybrid Agile approach for a project is stakeholder availability through the development and testing period.

Continuous stakeholder involvement is critical for a successful project using a hybrid Agile approach. If stakeholder involvement cannot be counted on throughout the entire project development life cycle, selecting the waterfall approach, while more costly to complete, may be considered more desirable.

### 5.2. Software Development and Maintenance Under the Hybrid Agile Model

This standard promotes integrating Agile concepts and methods with non-Agile techniques. Thus, the finished process would be a hybrid Agile approach. *Figure 3* represents the hybrid Agile process for AASHTOWare.

The model imposes several non-Agile requirements. The first is Sprint 0, which is used to define what the successful project outcome will be. Sprint 0 requires that the effort be as fully defined as possible by developing user stories to describe all necessary software capabilities. Sprint 0 also validates the initial scoping estimate and imposes a limit on duration. Timeboxing the effort enforces that customers and contractors continually groom the backlog such that the most important requirements are delivered and that the completed work meets budget and time expectations.

After Sprint 0, design and development work occurs in sprints of two- to four-week periods. This is the typical Scrum model of Agile development. Executable code is to be delivered for each sprint while the project is underway and demonstrated to the participating stakeholders.

Stakeholders continually work with the developers to create new user stories, plan and prioritize work for the next sprint, test code, and provide feedback and approval.

Customers continuously work to define a minimum viable product (MVP) during development. Agile development, following the Scrum process, allows the customers to add new stories and/or change the priority of stories planned in sprints and delivered in the final software product. However, under the hybrid model, since the effort is time-constrained, only the most important requirements are delivered, and those that are deemed a lower priority stay in the backlog for consideration at a later date as potential enhancements.

After the preponderance of development work is completed, there is a final sprint to execute a full user acceptance testing effort, resolve any major and critical defects, and ultimately allow the task force to accept the finished product at the end of the final sprint.

## 5.3. Defining the Project – User Stories and the Product Backlog

Agile tools support the customer's ability to change the priority of software features due to changing business needs for a given product release. Agile is the art of developing the most desirable and important features first.

Microsoft promotes an equivalent model to user stories called Behavior Driven Development, or BDD, which is considered interchangeable with user stories for this standard.

To ensure that any proposed work is adequately planned and budgeted, an initial scoping will be performed when the work is originally described to the task force. This effectively timeboxes the development effort and captures the scope and cost. The initial scoping also determines which work intake path will apply. When the work is initiated by the contractor team and stakeholders (task force members or TRT representing the task force), the initial sprint (Sprint 0) will fully describe the capabilities/user stories of the effort and validate the original scoping.

The hybrid Agile model espoused by this standard requires that Sprint 0 also set up the initial two sprints for the project.

## 5.4. Review Gates

The continuation of the review gate model from the existing SDMP will support continuity for AASHTO project managers, task force members, and contractors with minimum overhead, which have projects executing under this standard and the existing SDMP.

Agile development practices do not inherently require a review gate artifact. At a future date, AASHTOWare participants (AASHTOWare staff, SCOA, T&AA, project task forces, and contractors) may elect to remove the review gate requirement (Section *5.18*) to reduce cost, save time, or remove an unnecessary step. (By simply injecting needed review/payment stories into sprints at the completion of an initiative (theme), epic, or agreed upon time interval, the same results can be achieved.)

Note: Alpha testing is not applicable to this standard. This standard imposes continuous regression testing and requires that automated testing tools be used throughout the project. Also, TRT/technical advisory group (TAG) participants must test and accept delivered features and functionality, as well as application useability, from sprint to sprint prior to the review gate for the pay item.

Similarly, beta testing is not applicable to this standard. Beta testing will be replaced by the full coverage regression testing, full coverage automated testing, and by the user acceptance testing that will be performed in the final sprint of the project (which is already designated in this standard as a unique sprint).

## 5.5. Hybrid Agile Improvements: Quality, Speed to Market, Customer Satisfaction

As presented earlier, a primary motivation for Agile is the ever accelerating pace of the IT marketplace: "In order to succeed in the new economy, to move aggressively into the era of e-business, e-commerce, and the web…." These motivations are still present. By supporting the methods of this standard, AASHTOWare is actively engaged in improving delivery speed, software quality, and addressing customer satisfaction.

## 5.6. Product Evolution – Web and REST API Model

A key value add for AASHTOWare is development of a REST API model that supports the OpenAPI Specification and improving the ease of adoption of software by providing business critical data integration points for the portfolio. Agile development methods support this strategic business direction and offer an opportunity to treat creation, enhancement, and maintenance of web services as simple service and maintenance tasks by contractors.

## 5.7. Continuous Integration

Agile processes align well with continuous integration (CI) methods and technologies, and this standard promotes that AASHTOWare contractors adopt and employ CI tools and processes. As part of CI and to elevate code quality further, contractors should adopt and use technologies in their CI chain that identify code smells and help address deeper code issues prior to product releases.

## 5.8. Continuous Testing with Integrated Test Suites

In conjunction with CI, this standard promotes that contractors should adopt and employ automated testing tools that integrate into their CI chain. This approach supports continuous testing through all sprints and ultimately produces a test suite for future regression and acceptance testing. Specifically, using a web browser-based testing product will support executing UAT for the final sprint. Continuous testing via the above mechanism is considered a best practice and should be adopted.

## 5.9. Adoption and Use of Agile Software Development Tools

Contractors and task forces are required to use Agile software development tools under this standard to manage and prioritize issues and project development requests. The selection of the tool is within the control of the contractor; however, if the capability of the Agile development tool cannot meet the needs of this standard and that of AASHTOWare, the contractor and task force may be required to either:

- Select a different product or products, or
- Constrain their development work to the requirements and processes of the current SDMP.

## 5.10. Agile Tools, Agile Reporting

All business stakeholders, this includes all product task force members and associated TAG/TRT members and all SCOA and T&AA members, need to have access to each AASHTOWare contractor's Agile tools to support and participate in delivery of software via this standard. Agile tools in use by the contractors:

- Must accommodate remote access for the business representatives.
- Must provide typical Scrum/Kanban metrics, dashboards, and reporting.
- Must support project artifacts such as sprint reports, which include all user stories, tasks, and subtasks, such that task forces can correlate developed/delivered software capabilities to stories, tasks, and subtasks.

- Must support (and document) business stakeholder acceptance of a software feature/capability as demonstrated and initially tested on a sprint-by-sprint basis.
- Must support backlog grooming by business stakeholders and support continuous sprint planning updates.
- Will be used to describe the project work, direct all work and how the work is accomplished, monitor work outcomes, and be used to support standup meetings, sprint planning, backlog grooming, and acceptance of work by stakeholders.

With task force approval, in lieu of providing access to the Agile tools, the contractor may provide information from the Agile tools via an alternate method, such as via web-based reports, with the update frequency identified by the task force.

## 5.11. Frequent Interaction with Business Stakeholders When Performing Agile Development

The importance of close and regular interaction with the business representative with Agile development cannot be overstated or over-emphasized. Agile development will not succeed without business participation.

## 5.12. Agile Miscellanea

This section addresses training, Sprint 0, and contractor (experience) expectations and Agile work types, issue criticality spectrum, and Agile metrics.

### 5.12.1. Training Expectations

#### *5.12.1.1. General Training*

Agile/Scrum training will be provided to AASHTO staff, task force members, TAG and TRT participants participating in Agile development, and generally any business stakeholders engaged in Agile development with or for AASHTOWare.

As a guiding practice, each TAG or TRT should have at least one member who is a trained developer with Agile development experience that participates in the project representing the business.

The on-demand training model is the preferred method for participants to receive training, not only due to cost but also because of the convenience and better time management opportunities. All task force participants should watch “Agile Product Ownership in a Nutshell,” which is approximately 15 minutes and available on YouTube at *https://www.youtube.com/watch?v=502ILHjX9EE*.

Task force members involved in an Agile/Scrum effort will also need to take an introductory Scrum course from an online provider. These courses may last a few hours up to a day or two. Training courses that provide an introduction to Scrum, Scrum basics, or Scrum fundamentals will be sufficient. Members may use a training catalog from a provider their agency already uses or register for one of the following or other similar courses. AASHTO will reimburse training course registration fees paid for by task force members similar to travel expenses.

- Linked-In Learning – *https://www.linkedin.com/learning/scrum-the-basics*
- SkillSoft - *https://www.skillsoft.com/search?page=1&term=introduction%20to%20scrum*
- PluralSight - *https://www.pluralsight.com/courses/fundamentals-scrum*

#### *5.12.1.2. Scrum Master Training*

All contractor staff engaged in an AASHTOWare Agile project will have either Scrum training or scrum master training. At least one member of the contractor team will be trained as a scrum master and will be assigned to the development project.

#### *5.12.1.3. Agile Development Tool Training – Business Stakeholders*

Prior to any new Agile development engagement, contractor staff must train all involved business participants who are members of the task force, TRT, or TAG how to use and navigate within their Agile tools and provide credentials to log into those tools from their remote locations. The level of training is not intended to make business participants experts in the software, but capable of fulfilling their role as business stakeholders.

Additionally, as part of any Agile development engagement, contractor staff must train all business participants on how to use their web-based testing software. Ideally, this is the same software that is integrated with their CI platform and is easily useable by non-developers.

### 5.12.2. Sprint 0 Expectations

#### *5.12.2.1. Contractor Scrum Master Experience*

The contractor staff participating in Sprint 0, if acting as scrum master, must have multiple project engagements, or at least one year of lead role experience, and have web software development experience and education.

As a guiding practice, this contractor's experience and competence should be very high. The goal of fully defining all work in Sprint 0, as well as scoping the work, is critically dependent on this individual's abilities. Similarly, this person has a critical role in training, guiding, and working with business analysts and business stakeholders.

- ***Contractor Developer Experience***

At least one member of the contractor staff, not including the scrum master, who participates in Agile development efforts for AASHTOWare must be a senior level software engineer that is experienced in web, mobile, cloud, or other relevant to the project application development experience and Agile development experience and training.

- ***Task Force/TAG/TRT Sprint 0 Expectations***

The business stakeholders must be fully informed on the project's goals, be experienced in the unique business the software will support at their respective DOTs, and be fully committed to the development engagement for the anticipated time it takes to complete. Following Sprint 0 definition and scoping, the business participants must re-commit to the work based on the updated level of effort from Sprint 0.

### 5.12.3. Agile Work Types, and Issue (Defect) Criticality

#### *5.12.3.1.* Agile Work Types

The types of work (development work tasks) used for Agile software development are given various titles, but a common approach is to identify user stories, tasks, and subtasks. Subtasks will have a dependency on user stories and tasks.

There are also epics in Agile development, which comprise a group of stories, tasks, and subtasks that typically are related to a common function, deliverable, component, or module. Epics commonly span multiple sprints.

Agile development also identifies a work category called a theme or initiative. The following figure below provides a representation of work types and how they might interrelate.

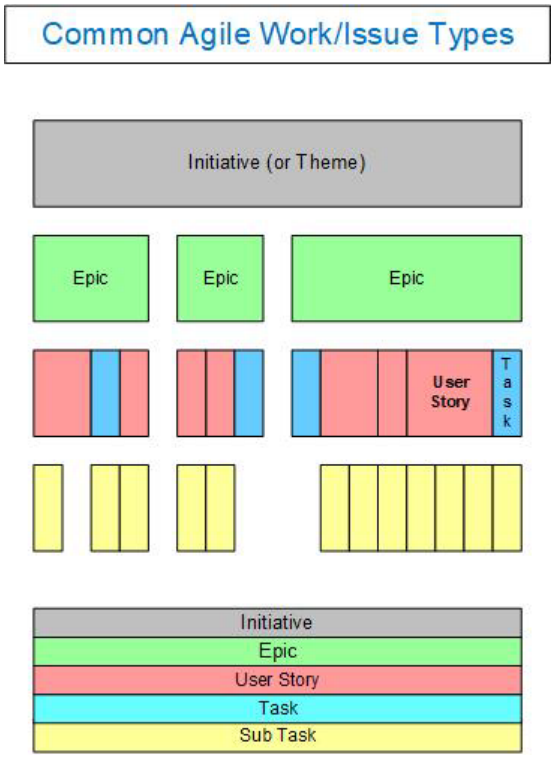


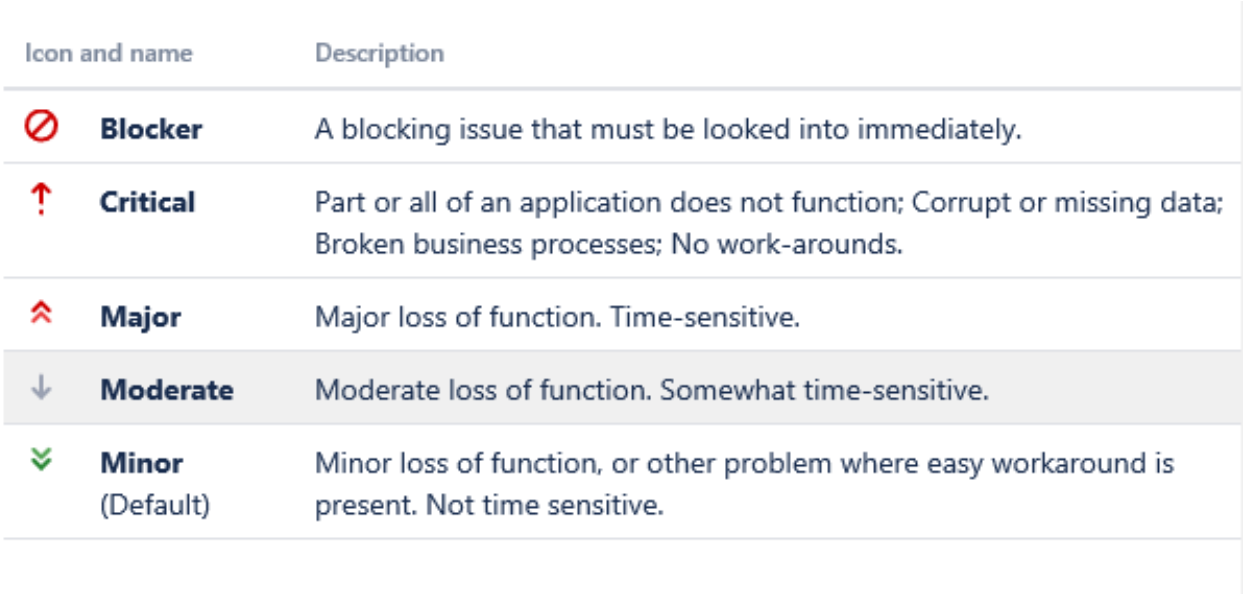

**Figure 4 - Agile Work Types and Hierarchy**

#### *5.12.3.2. Issue / Work Task Priorities*

The spectrum of criticality, and hence priority, defines the order in which defects are addressed, and also what defects have to be resolved before the project is completed and the code is released for production use and distribution.

AASHTOWare will apply a common definition to the below spectrum and require all contractors to adhere to the definitions and the baseline expectations for release of code for use by customers. That is, what is the severity of remaining issues/defects that will block an application and keep it from release/distribution? Typically, all remaining blocking and critical issues must be addressed (and also, to some degree major issues) before software can be released and considered complete.

Below is an example listing of common defects and definitions. (The example is pulled from Jira published by Atlassian Software):



| Icon and name | Description |
|---|---|
| **Blocker** | A blocking issue that must be looked into immediately. |
| **Critical** | Part or all of an application does not function; Corrupt or missing data; Broken business processes; No work-arounds. |
| **Major** | Major loss of function. Time-sensitive. |
| **Moderate** | Moderate loss of function. Somewhat time-sensitive. |
| **Minor** (Default) | Minor loss of function, or other problem where easy workaround is present. Not time sensitive. |

**Figure 5 - Issue Criticality Spectrum and Common Definitions**

### 5.12.4. Agile Metrics

The typical, and minimum for the purposes of this standard, metrics that are to be used and available during an Agile project are as follows. (The following descriptions are from Atlassian Agile Coach – *https://www.atlassian.com/agile/project-management/metrics.*)

#### *5.12.4.1. Sprint Burndown*

"Scrum teams organize development into timeboxed sprints. At the outset of the sprint, the team forecasts how much work they can complete during a sprint. A sprint burndown report then tracks the completion of work throughout the sprint. The x-axis represents time, and the y-axis refers to the amount of work left to complete, measured in either story points or hours. The goal is to have all the forecasted work completed by the end of the sprint."

#### *5.12.4.2. Epic and Release Burndown*

"*Epic and release* (or version) burndown charts track the progress of development over a larger body of work than the sprint burndown, and guide development for both scrum and kanban teams. Since a sprint (for scrum teams) may contain work from several epics and versions, it's important to track both the progress of individual sprints as well as epics and versions."

#### *5.12.4.3. Velocity*

"Velocity is the average amount of work a scrum team completes during a sprint, measured in either story points or hours, and is very useful for forecasting. The product owner can use velocity to predict how quickly a team can work through the backlog, because the report tracks the forecasted and completed work over several iterations–the more iterations, the more accurate the forecast."

#### *5.12.4.4. Control Chart*

"Control charts focus on the cycle time of individual issues–the total time from in progress to done. Teams with shorter cycle times are likely to have higher throughput, and teams with consistent cycle times across many issues are more predictable in delivering work. While cycle time is a primary metric for kanban teams, scrum teams can benefit from optimized cycle time as well."

#### *5.12.4.5. Cumulative Flow Diagram*

"The cumulative flow diagram is a key resource for kanban teams, helping them ensure the flow of work across the team is consistent. With number of issues on the Y axis, time on the X axis, and colors to indicate the various workflow states, it visually points out shortages and bottlenecks…"

#### *5.12.4.6. Other Metrics to Consider to Elevate Product Quality*

"Quality is an important metric for agile teams and there are a number of traditional metrics that can be applied to agile development:

- How many defects are found during development, after release to customers, and by people outside of the team?
- How many defects are deferred to a future release?
- How many customer support requests are coming in?
- What is the percentage of automated test coverage?"

## 5.13. Agile Development and Support Software

*AASHTOWare contractors must adopt, use, and support an Agile development, service and maintenance issue management product. The Agile development support software must accommodate all development projects and their user stories and service and maintenance (S/M) issues and support all normal Scrum and Kanban artifacts. Furthermore, the Agile development support software must allow access to all work subject to this standard by all relevant contractor staff, task force staff, TAG/TRT staff, AASHTO staff, SCOA members, and T&AA members. The Agile development support software will have the capability of publishing/printing relevant development artifacts requested by task force members (and their subordinate teams), SCOA, AASHTO staff, and T&AA. Relevant artifacts would include:*

- *All user stories, tasks, and their decomposed and dependent subtasks (the user stories and tasks by sprint, and the backlog as developed by the end of Sprint 0),*
- *Test outcomes for sprints and the acceptance of sprints/test outcomes by business stakeholders.*
- *All information related to validating the initial scope (the story points or the hours per task), which defines the schedule and scope of the project.*

## 5.14. Task Forces as Business Stakeholders for Executing Agile Development

*Task forces, or their subordinate group, must act as business representatives that function as the key business stakeholders to execute Agile development and testing under this standard. The task force assigned stewardship of each AASHTOWare product must either actively participate as a stakeholder in all Agile development activities, or they must establish a subordinate group such as a TRT, or some other group consistent with AASHTOWare practices, that has close engagement with contractor staff through all project development activities. The subordinate group must have the authority to make decisions on a daily basis, prioritize all work and perform backlog grooming, participate in demos of each sprint, and perform testing and approval of sprint (deliverables) work following successful sprint demos. This group must be accessible to the contractor on an as-needed basis (daily) and will use the Agile support software in use by the contractor to fulfill their role as business representatives and stakeholders.*

### 5.14.1. Agile Activities that Business Stakeholders Must Support

*At a minimum, the Agile development activities that business stakeholders must support are:*

- *Sprint planning,*
- *Maintain the project backlog,*
- *Validate software functions and performance based on new code resulting from sprints,*
- *Test sprint software updates and the overall software product, and*
- *Approve/disapprove the work/deliverables that are demonstrated during each sprint throughout the duration of the project.*

### 5.14.2. Business Stakeholders and the Final Product Definition

*Acting business stakeholders will be required to make and communicate decisions on the acceptance of sprint deliverables, prioritize work, continually reprioritize work as new user stories are created during development, and be fully authorized to make development and approval decisions during the development. Business stakeholders will ultimately determine what the MVP, or minimum viable product, will be and drive the frequent decisions of what is not developed through prioritization decisions.*

### 5.14.3. Agile Reliance on Business Stakeholders and Agile Fallback Option

*If the task force or their designated business stakeholders are not meeting their Agile obligations as specified in Section 5.14.1 such that the contractors cannot progress and the project will not be completed on schedule as determined by Sprint 0 or the initial scoping, the project must be halted, status communicated to SCOA, and the project must revert to the current SDMP Standard requirements including the current SDMP processes, deliverables, review gates, and alpha/beta testing model.*

### 5.14.4. Hybrid Agile Software Development – Captured Schedule/Scope

*This hybrid Agile software development approach timeboxes development such that the effort and duration do not exceed Sprint 0 estimates. The task force must identify the threshold amount that triggers additional task force approval if the cost or schedule estimate increases.*

*If the Sprint 0 estimate exceeds the initial project estimate by the threshold established by the task force, the product owner must obtain task force approval before initiating development. During development, if the project estimate exceeds the Sprint 0 estimate by the threshold established by the task force, the product owner must obtain task force approval before proceeding.*

## 5.15. Status Meetings Between Contractor and Task Force

*The contractor must conduct a status meeting with their task force at a frequency (recommend at least monthly) sufficient to address issues related to schedule, scope, or cost, and provide a demo of the working code produced since the last status meeting/demo for the product under development.*

The meeting is specifically for the benefit of the product task force, not the subordinate group (TRT) acting as the business stakeholders supporting development and addresses project health. It is recommended that both task force and TRT members participate in these status discussions. *If the project schedule is at risk for any reason, such as lack of engagement by business stakeholders, failure of the contractor to hit sprint goals, lack of staffing, new and unforeseen risks, and so forth, this meeting must address and mitigate the condition(s) threatening the schedule.*

## 5.16. Agile Development Metrics, Test Outcomes, Defect Backlog, and Criticality

*The contractor will make common Agile development metrics available to the task force at each status meeting. The information shared will also include test outcomes and the full inventory (backlog) of defects and their criticality.*

### 5.17. Continuous Assessment and Documentation of Tested and Accepted User Stories, Artifacts, Features, and Capabilities of Developed Code

*On a sprint by sprint basis, both the contractor and business stakeholders will assess which user story(ies) have been demonstrated and successfully tested (whether from the current sprint or a previous sprint), and document which user story(ies) are being accepted by the business stakeholders. Acceptance of user stories, any artifact/feature of the project code, or any unique feature or capability must be documented by date and include which business stakeholders were involved.*

### 5.18. Review Gate Sprints

*Task force and contractor representatives will coordinate with designated AASHTO PMs to define major pay items.* Subsequently, if desired by the AASHTO PM or task force, all major pay items will require an associated Agile review gate (a review gate sprint) be incorporated into the project backlog. These review gate sprints must be factored into the overall project schedule and may add to the project duration.

A review gate sprint may be executed concurrently with a normal development sprint, with the consensus of the task force, contractor, and AASHTO PM. If a review gate sprint is executed such that it must be completed prior to the next normal development sprint, the project schedule will be impacted, and the review gate will increase the duration of the project.

### 5.19. Hybrid Agile Process

*The contractor must conform to the Hybrid Agile Model shared in Figure 3. At a minimum, the contractor will complete the following.*

#### 5.19.1. Sprint 0 Backlog Development

*Sprint 0 backlog development has as goals defining the majority of user stories (requirements) to produce a minimum viable product and also development of work estimates to support a schedule and total project duration.*

- *Validation of initial scoping effort with Sprint 0 estimates.*
- *Planning of the first two sprints.*
- *Determine the number of contractor resources needed to meet the desired deadline and define the team.*
- *Conduct a kickoff meeting with business stakeholders and establish a baseline understanding of the Agile process and stakeholder responsibilities.*
- *Establish a target end date based on the total work defined and the number of developer resources obligated to the effort. This step defines a maximum time budget (timeboxing) and imposes a constraint on the duration of the project.*

#### 5.19.2. Execute the Agile Development Process (Scrum Model) for Each Product.

- *Conduct each sprint, consistent with accepted Agile practices (and aligned with this standard) that will also include continuous development of unit and regression tests.*
- *Plan sprints a minimum of one sprint ahead of the current one.* A preference is to plan multiple sprints ahead to address issues earlier, establish better communication of intent with stakeholders, sustain or improve velocity, and deliver a minimum viable product within the schedule.
- *Web applications developed under this standard will not be promoted for UAT until all blocking, critical, and major severity defects are corrected.*

### 5.19.3. Execute Final Sprint

*Execute a final sprint before the project deadline that includes all unit, regression, functional, system, and API tests. Also, complete a full UAT and address all remaining blocking, critical, and major defects before promoting for release. Deliver all documentation, including the REST API, and perform a final system walk-through and final demo with the task force and the TRT.*

# 6. Deliverable and Artifact Definitions

Most deliverables or artifacts required by this standard may be created electronically. If desired, review gates may be required for all major pay items, and those review gates should be executed using the Agile software development tool.

Electronic artifacts are created, used, and consumed by means of the required Agile development tool that contractors will use under this standard. The aforementioned electronic artifacts are identified and defined throughout the previous sections. Even though the electronic artifacts are intended to be viewable and consumable within the contractor's approved Agile development software platform, these artifacts may be exported as files or printed at the prerogative of contractors, task force members, AASHTO staff, SCOA, T&AA, and others.

A concise listing of electronic artifacts associated with Agile development and this standard follows. The reader is directed to the section describing the electronic artifacts for additional description/content of each artifact.

| Electronic Artifact Name | Short Description | Section(s) of Standard |
|---|---|---|
| Major Pay Item Review Gates | An SMDP artifact (based on waterfall processes and desktop software products) | 3, 5.4, 5.18 |
| *Agile Software Development Tool | Supports Scrum, and other Agile methodologies. | 2, 3, 4, 5.9, 5.13 |
| *Software Development & Support Ecosystem | Modern code repository, continuous integration software and technical architecture, software quality analysis tools, automated testing tool (accessible by task force, TRT, and TAG members). | 2.2, 5.7, 5.8 |
| REST API adhering to the latest OpenAPI Specification | The standard interface model adopted by AASHTOWare. Interfaces produced under this standard must meet this specification. | 5.6 |
| Project/Product Backlog | A listing of epics, user stories, tasks, and subtasks defining a project. | 3, 4.2, 5.2, 5.3, 5.19.1 |
| Sprints | Short discrete work efforts following the Scrum methodology. | 3, 4.3, 4.4, 5.13, 5.14, 5.15, 5.19.2 |
| Regression Test | Electronic automated tests. (Manual regression testing will require a test script for users or developers to execute and document results. Manual testing is a less desirable approach.) | 2.2.3, 2.2.4, 3, 4.4, 4.5, 5.4, 5.8, 5.19.2, 5.19.3 |
| Agile Development Metrics/Reporting; | Project reporting/performance tools accessible via the Agile development tool (sprint burndown, release burndown, velocity, control chart, defect classification, etc.) | 3, 5.10, 5.16 |
| Stakeholder acceptance of delivered features. | | 3, 5.10, 5.17 |

*Software development and support assets that must be in use to develop using this standard.

# 7. Glossary

**Agile (Software) Development** – Agile software development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer/end user. Agile development advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.
*https://en.wikipedia.org/wiki/Agile_software_development*

Agile software development is also an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it. *https://www.agilealliance.org/agile101/*

**API** – Otherwise referred to an application program interface (API). An API is a set of routines, protocols, and tools for building software applications, and to enable applications, operating systems, websites, mobile devices, and users to interact.
*https://www.webopedia.com/definitions/api/*

**Backlog** – or the product backlog, or development project backlog, is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product. The product owner (primary business stakeholders) is responsible for the product backlog, including its content, availability, and ordering.
*https://www.scrum.org/resources/what-is-a-product-backlog*.

**Backlog Grooming (Refinement)** – is the agile process whereby the business stakeholder(s) (product owners) work with the scrum master representing the development team to update the backlog with new user stories, and to also reprioritize all work (users stories and tasks) in the backlog to represent current (evolving) needs and importance.

Stated another way, backlog refinement(also referred to as backlog grooming) is when the product owner and scrum master (and potentially some or all of the rest of the team and/or stakeholders) review items on the backlog to ensure the backlog contains the appropriate items, that they are prioritized, and that the items at the top of the backlog are ready for delivery"." *https://www.agilealliance.org/glossary/backlog-refinement/*

**Business Case Analysis** – for large software development efforts (large in terms of cost, hours, duration, or some similar combination of metrics) Software Portfolio owners (and product owners) may require that a business case be prepared. Large projects in Agile are commonly referred to as Initiatives. A business case analysis will attempt to quantify the impacts of recommendations, support findings, provide a cost benefit analysis, and provide a reasonable basis for making software investment decisions.

**Code Smells** – a code smell is any characteristic in the source code of a program that possibly indicates a deeper problem. Determining what is and is not a code smell varies by language and development methodology, among other factors. *https://en.wikipedia.org/wiki/Code_smell*

"Smells are certain structures in the code that indicate violation of fundamental design principles and negatively impact design quality." Smells indicate weaknesses in design that may slow down development or increase the risk of bugs or failures in the future. “Bad code smells can be an indicator of factors that contribute to technical debt.” An example tool which analyses source for smells is SonarQube (*https://www.sonarsource.com/products/sonarqube/*), which is easily integrated with continuous integration platforms to improve code quality.

**Continuous Integration** (or CI) – The practice of merging all developers' working copies to a shared mainline (repository) several times a day. CI was intended to be used in combination with automated unit tests written through the practices of test-driven development. Initially, this was conceived of as running and passing all unit tests in the developer's local environment before committing to the mainline. *https://en.wikipedia.org/wiki/Continuous_integration*

Current definitions and implementations of CI extend the above model and rely upon build servers to analyze code for quality (inject a QA process such as code smells), execute unit tests, build the code for deployment, deploy the code to their assigned application servers, and execute other test suites prior to releasing to users. Sophisticated CI environments allow setting Quality thresholds that will not allow code to be deployed until the desired QA metric is achieved. CI is a foundational piece of the DevOps model.

**Daily Scrum** – The *daily scrum* (daily standup meeting) is a short (usually limited to 15 minutes) discussion where the team coordinates their activities for the following day. The daily scrum is not intended to be a status reporting meeting or a problem solving discussion. What is reported by each person on the team:

- what work was performed yesterday;
- what work is planned for today; and,
- any issues that were encountered (blockers).

**Defect** – Defects are generally perceived as bugs or broken software. From a user or business stakeholder perspective, defects are also conditions where the software does not satisfy a software requirement (specification, user story, etc.) or business stakeholder (customer) expectation. Software developers, looking at a lower level of functionality, consider a defect an "error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results." Both perspectives are correct.

Agile development tools may also refer to defects as issues or by some other naming convention. Defects are typically assigned a criticality or severity. Some examples of criticality are Blocker, Critical, Major, Moderate, Minor.

**Definition of Done** – The definition of done is a team's shared agreement on the criteria that a product backlog Item must meet before it is considered done.

**Development Team** - The *development team* consists of the developers who deliver the product increment inside a sprint. The main responsibility of the development team is to deliver the increment that delivers value every sprint. How the work is divided up to do that is left up to the team to determine based on the conditions at that time.

**Increment** – The increment is the collection of the product backlog items that meet the team's definition of done by the end of the sprint. The product owner may decide to release the increment or build upon it in future sprints.

**Issue** – An issue is an Agile work type. Other Agile work types are initiative (or a theme), epic, user story, task, and subtask. Subtasks have dependencies to user stories and tasks, which have dependencies to epics.

**Kanban** – Kanban is a method for managing the creation of products with an emphasis on continual delivery while not overburdening the development team. Like Scrum, Kanban is a process designed to help teams work together more effectively. *https://kanbanize.com/kanban-resources/getting-started/what-is-kanban*

While Scrum is particularly effective as a software development model, which keeps business stakeholders engaged and actively participating in the design process, Kanban is effective in helping manage work on IT infrastructure (system admins, DBAs), and is also invaluable for software service and maintenance (SM) work activities.

**Minimum Viable Product (MVP)** – A common definition for MVP is: "the smallest possible product that has three critical characteristics: people choose to use it or buy it; people can figure out how to use it; and we can deliver it when we need it with the resources available – also known as valuable, usable and feasible." (Marty Cagan) An alternate definition equates MVP to the minimum feature set of a product that is a candidate for release to users.

**OpenAPI Specification (OAS)** - The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

An OpenAPI definition can then be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases. *https://swagger.io/specification/*

"The OpenAPI Specification was donated to the Linux Foundation under the OpenAPI Initiative in 2015. The specification creates a RESTful interface for easily developing and consuming an API by effectively mapping all the resources and operations associated with it."
*https://swagger.io/resources/open-api/*

**Performance Testing** – As used in this standard, performance testing is a testing practice to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate, or verify other quality attributes of the system, such as availability, reliability, and resource usage. The following are some of the types of performance tests.

- **Load Testing** – Often referred to as the main subset of performance testing, this methodology involves the scrutiny of the website or application, testing how a system behaves with many users and what the response time is under different scenarios. Performance engineers can see how the system behaves with varying numbers of users and what kinds of bottlenecks are created in varying usage scenarios.
- **Capacity Testing** – This is another type of performance testing that helps identify the maximum capacity of users the system can support while not exceeding a maximum response time. In other words, capacity testing exercises the system to check if the application and infrastructure can take on the amount of traffic they were designed to handle without compromising user experience.
- **Stress Testing** – This kind of testing involves testing performance under extreme conditions.

**Product Backlog** – See definition of backlog in this glossary. The product owner (primary business stakeholder) *maintains the product backlog* on an ongoing basis including its content, availability, and ordering.

**Product Owner** - The *product owner* is responsible for managing the product backlog in order to achieve the desired outcome that the team seeks to accomplish. The product owner role exists in Scrum to address challenges that product development teams had with multiple, conflicting goals from stakeholders, or no direction at all with respect to what to build. It is recommended that the task force appoint one or more product owners. Ideally, the product owners would be task force members.

**Regression Testing** – As used in this standard, regression testing is a type of testing in the Agile software development cycle where functional and non-functional tests are performed to ensure that previously developed and tested software performs as expected after a change. The intent is to ensure that changes introduce no unintended breaks. Regression means retesting the application's unchanged parts and addresses a common issue developers face: the emergence of bugs in unchanged code with the introduction of software updates.

**Repository** (or Source Control Repository or Version Control System) – Software source code is typically stored in a source control system and accessed/managed via version control. Examples of source control repositories are Subversion, Git, and TFVC (Microsoft). Apache's definition of Subversion is: "a software versioning and revision control system" used "to maintain current and historical versions of files such as source code." *https://en.wikipedia.org/wiki/Apache_Subversion*

**REST API** (or RESTful API) – "is designed to take advantage of existing protocols. While REST can be used over nearly any protocol, it usually takes advantage of HTTP when used for web (applications). This means that developers do not need to install libraries or additional software in order to take advantage of a REST API…" "Since data is not tied to methods and resources, REST has the ability to handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia." "This…flexibility inherent in REST API design allow you to build an API that meets your needs while also meeting the needs of very diverse customers. Unlike SOAP, REST is not constrained to XML, but instead can return XML, JSON, YAML or any other format depending on what the client requests. And unlike RPC, users aren't required to know procedure names or specific parameters in a specific order." *https://www.mulesoft.com/resources/api/what-is-rest-api-design*

REST APIs are promoted by this standard as the preferred model to support data exchanges (Data Integration Framework). The use of tools such as MuleSoft and Swagger to support the creation and maintenance of REST APIs is beneficial to assure APIs adhere to the Open API standard.

**REST Web Service** - A web service that is accessible via a web address, and provides a mechanism to share information to both internal and external applications and users. The web services most commonly implemented with web applications, mobile applications, and cloud services are REST (Representational State Transfer) services. REST services commonly communicate over HTTP. (Web and Mobile Data Exchange Guideline, AASHTOWare Standards and Guidelines, *https://www.aashtoware.org/about/standards-and-guidelines/*)

**Scope** (or Scoping) – The practice of estimating the effort (otherwise referred to as level of effort) to complete software development task(s) and produce a working software product. With respect to Agile software development, multiple approaches may be used, but common models are story points and hours (or days/weeks/months).

**Scrum** – Scrum, in terms of this standard, is an agile process framework for software development. Scrum is intended for teams of ten or fewer members, who break their work into goals that can be completed within timeboxed iterations called *sprints*, and track progress in 15-minute (or less) standup meetings. Scrum requires a sprint planning meeting before the start of the sprint, a demo at the conclusion of the sprint, and a retrospective meeting following each sprint to make any needed improvements to the execution.
*https://en.wikipedia.org/wiki/Scrum_%28software_development%29*

Further definition of other Scrum terms are available at
*https://www.agilealliance.org/glossary/scrum/#q=~(infinite~false~filters~(postType~(~'page~'post~'*

*aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'scrum))~searchTerm~'~sort~false~sortDirection~'asc~page~1)*

**Scrum Artifacts** – Typical scrum artifacts are the product backlog, sprint backlog, increment, and a shared agreement (determination) of what has been done.

**Scrum Master** - The *scrum master* (also the technical lead) is the development team role responsible for ensuring the team lives agile values and principles and follows the processes and practices (specifically Scrum) that the team will use for development. The name was initially intended to indicate someone who is an expert at Scrum and can therefore coach others. The role does not generally have any actual authority. People filling this role have to lead from a position of influence (experience, technical and development prowess, software engineering, design, and/or application architecture skills) often taking a servant-leadership stance. Scrum masters benefit from specific agile training for this role.

**Scrum Roles** – Scrum roles are the product owner (which is the representative of the business, and usually represents a group of business stakeholders), the scrum master, and the developers comprising the development team.

**Sprint** – The *sprint* is a timebox of two weeks to one month during which the team produces a potentially shippable product Increment. Typical characteristics of sprints include:

- maintain a consistent duration throughout a development effort;
- a new Sprint immediately follows the conclusion of the previous Sprint: and,
- start date and end date of Sprint are fixed.

**Sprint 0** – Sprint 0 is conducted prior to development actually beginning and is used for fully describing all work (functionality/requirements) required by the project in terms of user stories and tasks. When all development work is captured in Sprint 0, the captured user stories will be the basis for the product backlog, which will also be prioritized by the product owner and scrum master. Sprint 0 will also validate the initial scoping shared with the task force and be the basis for the project development schedule. Sprint 0 will be used to initially plan releases, determine resourcing by contractors, define what is agreed upon as done, educate business participants about the Scrum process that will be followed, and conduct an initial kick off meeting.

**Sprint Backlog** – The sprint backlog is the collection of product backlog items selected for delivery in the sprint, and if the team identifies tasks, the tasks necessary to deliver those product backlog items and achieve the sprint goal.

**Sprint Planning** – A team starts a sprint with a discussion to determine which items from the product backlog they will work on during the sprint. The end result of *sprint planning* is the sprint backlog.

Sprint planning typically occurs in two parts. In the first part, the product owner and the rest of the team agree on which product backlog items will be included in the sprint. In the second part of sprint planning, the team determines how they will successfully deliver the identified product backlog items as part of the potentially shippable product increment. The team may identify specific tasks necessary to make that happen if that is one of their practices. The product backlog items identified for delivery and tasks if applicable, makes up the sprint backlog.

Once the team and product owner establish the scope of the Sprint as described by the product backlog items no more items can be added to the sprint backlog. This protects the team from scope changes within that Sprint.

**Sprint Retrospective** – At the end of the sprint following the sprint review, the team (including product owner) should reflect upon how things went during the previous sprint and identify adjustments they could make going forward. The result of the *retrospective* may be at least one action item included on the following sprint's sprint backlog.

**Sprint Review** (Sprint Demo) – At the end of the sprint, the entire team (including product owner) reviews the results of the sprint with stakeholders of the product. The purpose of this discussion is to discuss, demonstrate, and potentially give the stakeholders a chance to use the increment in order to get feedback. The sprint review is not intended to provide a status report. Feedback from the sprint review gets placed into the product backlog for future consideration. In terms of this standard a sprint review will be necessary for stakeholder acceptance of a delivered feature or requirement.

**Stand Up Meetings** – Please see the definition for daily scrum.

**Story Points** - Story points are an approach that Agile development teams use to differentiate the *relative size of user stories between each other*. A common technique is to use the Fibonacci sequence (each number in the sequence is the sum of the two preceding numbers in the sequence): 1, 2, 3, 5, 8, 13, 21, …, and so forth. The larger the story is, the more uncertainty there is around it and the less accurate the estimate will be. Story points are a way for teams to recognize this uncertainty, and to also encourage the team to decompose the user story or task into smaller pieces of work.

A common mistake is attempting to equate story points to hours, even though story points are all about time.

As an example, "development team members will consider how long each new story will take in comparison to other stories. You and I might agree this new story will take twice as long as a one-point story, and so we agree it's a two." In this way story points still reflect time, but not prescriptively define hours. *https://www.mountaingoatsoftware.com/blog/dont-equate-story-points-to-hours*

In the case where a business constraint does not support the use of story points, the product owner and scrum master may need to negotiate an alternate model. As a practical matter, after two or three sprints are completed (assuming two-week sprints), those sprint metrics can help to determine a relative mapping of hours to user stories for a given team on a specific project. This would vary team by team, project by project.

Example:

| Story Points | Equivalent Hrs / Team / Project |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 5 | 8 |
| 8 | 12 |
| 13 | 20 |

**Unit Testing** – Unit testing is a part of the Agile software development process in which the smallest testable parts of an application, called units, are individually scrutinized for proper operation. Software developers and sometimes contractor QA staff complete unit tests during the development process.

**User Acceptance Testing (UAT)** – As defined by this standard, UAT is conducted in the final sprint of the development project. "UAT is the last phase of the software testing process. During UAT, actual software users test the software to make sure it can handle required tasks in real-world scenarios, according to specifications." *https://www.techopedia.com/definition/3887/user-*

*acceptance-testing-uat-software-testing* The final sprint where UAT is conducted is also the sprint to resolve existing defects and new defects documented as part of UAT.

**User Story** – "User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template:

*As a* < type of user >, *I want* < some goal > *so that* < some reason >.

User stories are often written on sticky notes and arranged on walls or tables to facilitate planning and discussion. As such, they strongly shift the focus from writing about features to discussing them." *https://www.mountaingoatsoftware.com/agile/user-stories*

Stated another way, "a user story is an informal, natural language description of one or more features of a software system. User stories are often written from the perspective of an end user or user of a system.

**Web Service** – For the purposes of this standard, a web service will typically be a REST service that supports mobile, web, and cloud-hosted applications. Strategically, web services published consistent with the Representational State Transfer (REST) model are used to publish REST APIs, which are critical to supporting e-business and e-commerce and allow rapid evolution and efficient data exchange between web-based software platforms. REST services are stateless.

A secondary web service model, SOAP-based web services, is also useful. However, SOAP-based web services usually are heavy (larger and slower), generally don't support transactional needs as well as REST, and are restricted to XML by definition. SOAP services are typically stateful.

This page is intentionally blank.

AASHTO AASHTOWare™ SOLUTIONS

# COMMON ARTIFACTS STANDARD

## S&G Number: 1.007.01.5S

## Effective Date: April 1, 2026

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.2 | 10/13/2022 | Updated the links for the MSE and project work plan templates | 10/13/2022 Approved by T&AA |
| 1.3 | 9/30/2023 | Updated information about the data dictionary and corrected links. | 10/02/2023 Approved by SCOA |
| 1.4 | 8/07/2024 | Updated information about data dictionaries. | 12/11/2024 Approved by SCOA |
| 1.5 | 5/12/2025 | Updated information about the VPAT/ACR. | 3/12/2026 Approved by SCOA |

# Table of Contents

# 1. Purpose

This standard identifies the non-code artifacts required by the Software Development and Maintenance Standard (SDMP) and the Hybrid Agile Development and Maintenance Standard (HADM).

# 2. Task Force/Contractor Responsibilities

Task forces and contractors are responsible for the completion and approval of the artifacts contained in this standard as required by the SDMP or HADM.

# 3. Required (or Recommended) Deliverables and Artifacts

Required artifacts are identified in Section 6.

# 4. Procedures

The artifacts must be completed and approved as identified in the SDMP and HADM.

# 5. Technical Requirements

Technical requirements are identified in the SDMP and HADM.

# 6. Deliverable and Artifact Definitions

This chapter includes a description and the required content for each required deliverable and artifact defined in the Software Development and Maintenance Standard. The deliverables and artifacts listed are in the order they are prepared during the project and MSE lifecycles.

*In addition to the content listed below, each deliverable and artifact must include the appropriate document identification information, including the project/product name, contract period, version number, and date.* If needed, an introduction section that explains the purpose of the deliverable or artifact should be included.

## 6.1. Work Plan

### 6.1.1. Description

The work plan is the formal document that describes the scope and objectives of the work to be performed by the contractor during a specific contract period, requirements or specifications to be met, tasks to be performed, deliverables to be produced, schedule to be met, cost of the effort, required staffing and resources, the technical approach for accomplishing the work, and the approach for managing, monitoring, and controlling the work.

### 6.1.2. Content

*Two Microsoft Word templates are available for preparing project and maintenance, support, and enhancement (MSE) work plans. The Project Work Plan Template is used to prepare the work plan for an AASHTOWare project, and the MSE Work Plan Template is used to prepare the work plan for an annual maintenance, support and enhancement work effort of an existing AASHTOWare product. These templates include all the required information that must be included for each project or MSE work effort.*

*All sections of the selected template must be completed unless noted as optional. If a section is not applicable, note that the section is "Not Applicable" instead of removing the section.* Additional information may be included in the work plan as deemed necessary by the contractor or task force.

The URLs for downloading both work plan templates are included in the *Work Plan Templates* section of Chapter 7.

## 6.2. Review Gate Approval Request

### 6.2.1. Description

*Review Gate Approval Requests are required for work using the SDMP. They are optional for work using the HADM. Anything identified as required in this Section 6.2 applies only to work completed using the SDMP.*

*A Review Gate Approval Request is prepared by the contractor project manager and submitted to the task force chair and the appropriate AASHTO staff at each review gate. This document is also used to document and communicate the task force decision regarding the approval or denial of the review gate.*

### 6.2.2. Content

*The Review Gate Approval Request Form or an equivalent document with the same content is used for preparing all review gate approval requests.* The URL for downloading the form is included in the *Review Gate Approval Request Form* section of Chapter 7.

*The required content is defined below:*

- *Title and Header Information - Each review gate approval request includes the following title and header information: name of the review gate, project/product name, task force chairperson name, contractor project manager name, submission date, requested approval date, and a summary of the request.*
- *Deliverables – All unapproved deliverables for each review gate period are submitted with the review gate approval request. The request includes the following information for each deliverable: deliverable name, file name, version number, location of each unapproved deliverables (if not attached), and documentation on prior task force approval of deliverables and the location of the approved deliverables.*
  - *Required artifacts associated with the review gate are also submitted or the location of each artifact is provided.*
  - When available, the request form should also include recommendations for approval by stakeholders; and any other information that would assist with the approval. Any change requests that were approved since the previous review gate should be also referenced.
- *Checklist/Questions - The following checklist questions must be answered "yes" or "no":*
- *Is each major deliverable compliant with AASHTOWare standards?*
- *Have all issues regarding the deliverables or other work associated with the review gate been resolved?*
- *Does each major deliverable implement or support all user requirements in the URS?*
  - The purpose of this question is to ensure that all user requirements have been implemented in these deliverables; and to ensure that all system requirements, design elements, and test procedures support one or more of the user requirements in the URS.

- The RTM should be used to demonstrate this support in certain deliverables by including all user requirements, system requirements, design elements, and test procedures; and providing links between the user requirements and these other items.

- *Noncompliance, Issues, and Non-implemented User Requirements - If the answer to any of the above questions is "no", then a response must be included or attached that addresses the following:*

- Each area of noncompliance to standards is identified and a justification for the noncompliance provided. This should reference any prior approvals for exceptions to standards.

- Each unresolved issue is described with the plan for resolution of each issue.

- Each user requirement that is not implemented or supported in one of the major deliverables is to be listed with an explanation.

- *Contractor Acknowledgement - The contractor project manager's name and signature, and the date of signature must be provided with each review gate approval request. The signature acknowledges that the project manager approves and agrees with all information submitted.*

- *Task Force Approval - The task force chair's name and signature must also be provided, along with the date and approval decision.* The reason for not approving should be provided when applicable. Also, if needed, directions or notes to the contractor should be provided.

- *The chair's signature is provided on behalf of the entire task force and acknowledges the task force approval or denial of the review gate approval request and the submitted deliverables.*

- *AASHTO Staff Acknowledgement - The appropriate AASHTO staff member also signs and dates each review gate approval request after the task force approval decision is made. The signature acknowledges that AASHTO staff has reviewed the submission material and the approval decision.*

Note: Signatures may be in any form agreed upon by both the task force and contractor, such as written signatures, electronic images of signatures, or a note in signature block referencing an email or other method of approval.

## 6.3. Product Installation Package

### 6.3.1. Description

*The Product Installation Package is a required deliverable that contains all procedures, executables, and documentation needed to implement and operate the product at the production site.*

The product installation package may include components, if not all, that may be delivered electronically. The fact that an item has been electronically delivered should be noted on the checklist that is included in the package. If the entire package is delivered electronically, it must still include all items (electronic checklist, contents list, etc.).

### 6.3.2. Content

*There is no rigid format required for the Product Installation Package; however, the content listed below must be included.*

#### *6.3.2.1. Name of Product being shipped*

*The complete name of the product must be clearly stated on all items in the installation package.*

#### *6.3.2.2. Cover Letter*

*The cover letter must include information such as: whom the installation package is being delivered to, who is sending the package, what is included in the package, and for what reason.*

#### *6.3.2.3. Checklist*

*The Installation Package Checklist is used to assist in preparing the Product Installation Package and the completed checklist must be included with the package. This checklist is provided in Chapter 6.*

#### *6.3.2.4. Contents List*

*A contents list is included with the installation package showing what content is being shipped. The content list must clearly state what platform (computing environment) the installation package was prepared for. An example Installation Package Contents List is provided in Chapter 7.*

#### *6.3.2.5. User Documentation*

*The purpose of user documentation is to provide sufficient information to facilitate the unassisted and correct use of the software product.*

*User documentation is required for new applications.* For existing applications, the user documentation may be provided as updates to existing documentation or as a complete replacement for existing user documentation. If updates are provided, clear instructions for updating existing documentation must be included.

#### *6.3.2.6. System Documentation*

The purpose of system documentation is to provide installers and product managers with sufficient information to safely integrate the software product into their computing environment and to understand the consequences of such integration. This documentation should be distributed to all licensees.

*System documentation is required for all new releases of AASHTOWare products.* For existing products, the system documentation may be provided as an update to existing documentation or as a complete replacement for existing system documentation. If updates are provided, clear instructions for updating existing documentation must be included.

This documentation should be written to satisfy the needs of product installers, managers, and administrators. Because the system documentation may contain sensitive information such as security administration, it should be structured such that the sensitive material can be distributed only to those persons authorized to use it.

System documentation should be divided into the functions described below.

##### *6.3.2.6.1. Implementation Documentation*

*Implementation documentation is provided to assist customer support staff in the installation, setup, configuration, customization, maintenance, and de-installation of the AASHTOWare product in the customer's IT environment.*
The following are recommended components of the implementation documentation.

- Differences - Provide brief descriptions of the differences (deltas) between the current version of the product and the previous version. Release level, maintenance level, fixes applied, and testing level information should also be supplied in this section.
- Environment - Provide descriptions of environment and resource requirements. These descriptions should include documentation of interactions with systems and communications software, dependencies on interfaces with other products, resource requirements, hardware feature or device requirements, and performance characteristics.
- Warnings - Provide warning messages with clear descriptions of any potential for destroying or corrupting data as well as any irreversible actions.
- Uninstallation - Provide instructions for uninstalling or removing the product.
- Installation - Provide instructions for installation of the complete product, maintenance, and fixes.
- Problem resolution - Describe the methods and procedures that should be employed for isolating, identifying, documenting, and reporting errors.
- Interfaces to systems and other applications software - Describe data formats and methods of interaction.
- Required maintenance (system care and feeding, not changes)
- Customization features - Describe customization features such as generation or installation parameters. Explain implications of choosing different options.
- User maintainable system parameters such as initialization files, account profiles, performance parameters, or configuration definitions should be documented.
- User exits, hooks, and replaceable modules should be documented along with the processes and procedures necessary to activate them.

#### *6.3.2.6.2. Security Management*

*This component of the system documentation contains information appropriate for distribution to installation security managers.* This component should be separately packaged.

#### *6.3.2.6.3. Administration Documentation*

This component of the system documentation is prepared when the product will require management or administration by personnel separate from the installers or maintenance personnel. This component should be separately packaged. Some examples of such management are data file maintenance, performance monitoring, problem resolution, resource allocation, account management, database maintenance, work scheduling, and report distribution.

#### *6.3.2.6.4. Operator Documentation*

*Operator documentation, where separate from user documentation as in the case of shared use systems (servers), should be separately packaged. This documentation contains all operator messages.* These messages should be segregated by severity and by whether they require responses.

#### *6.3.2.6.5. Platform Specific Installation Instructions*

*Any instructions specific to the platform this installation package is to be installed on should be included.*

#### *6.3.2.6.6. Special Instructions*

*Any special instructions unique to the customer should* be included in the shipment. *All known malfunctions must also be clearly noted with the appropriate workarounds documented.*

#### *6.3.2.6.7. Summary of Changes in the Release*

*A summary of new, changed, or removed features must be included in the shipment.*

### *6.3.2.7. Software*

*All software included in the Product Installation Package must be provided electronically or shipped on extended life media that provides ease of installation and use to the recipient. A duplicate copy of the software provided electronically or on extended life media must be supplied to AASHTO for archival.*

*The following software items must be included with the installation package.*

#### *6.3.2.7.8. Product Software*

*All software that the licensees are entitled to must be included in the product installation package.*

#### *6.3.2.7.9. Command Language Procedures*

*Command language procedures needed to install or run the product must be included.*

#### *6.3.2.7.10. Database Definition Procedures*

*The necessary procedures and schema needed to setup the customer chosen (and supported) database must be included.*

#### *6.3.2.7.11. Installation Jobs*

*Installation jobs and procedures to install the product on the platform the installation package is being prepared for must be included.*

#### *6.3.2.7.12. Third Party Software*

If the AASHTOWare software requires third-party software, the following should be considered. If the third-party software is distributed with the AASHTOWare software, the latest release of the third-party software that has been tested should be included. If the third-party software is not included, it should be clearly stated in the install document what third party software is needed and what release it should be.

### *6.3.2.8. Security Key*

*If the product requires a security key to operate, the key must be included in a shipment to a first-time recipient. In the case of first-time shipments, arrangements must be made to provide this key to the recipient.* If this installation package is an update of the software and the update does not require a change in the security key, a new one need not be provided.

### *6.3.2.9. Virus-Scan has been passed*

*Before the Product Installation Package is shipped, the media containing all or parts of the package must be scanned for viruses if a commonly used virus-scanning product is available for that media. The virus-scan software must be of current release and an industry leader. The scan must show no viruses on the media.*

## 6.4. Data Dictionary

This data dictionary standard provides requirements for documenting data entities and attributes within the AASHTOWare ecosystem. *The data elements required for the implementation and use of AASHTOWare software and* ***AASHTOWare-supplied application programming interfaces*** *must be provided* while expressly excluding data elements input from or output to intermediate files that users can access. The exclusion of user-accessible temporary file interactions ensures the focus remains on the core data managed within the AASHTOWare products. *The final data dictionary must be uploaded to AASHTOWare OpenAPI before a product issues the corresponding software release anytime the data dictionary changes.*

### 6.4.1. Data Entity Definition

- ***Entity Name: Name of the data entity.***
- ***Description: Brief description of the entity's purpose and usage.***
- ***Attributes: List of attributes associated with the entity, including name, data type, description, and any additional metadata.***

### 6.4.2. Attribute Definitions

- ***Attribute Name: Name of the attribute.***
- ***Data Type: Data type of the attribute (e.g., string, integer, Boolean, etc.).***
- ***Description: Description of the attribute's meaning and usage.***
- ***Constraints: Any constraints or validation rules applicable to the attribute.***
- ***Example Values: Example values demonstrating the format and content of the attribute.***

### 6.4.3. Metadata

- ***Version: Version of the data dictionary.***
- ***Last Updated: Date of the last update to the data dictionary.***
- ***Author: Name or identifier of the individual or team responsible for maintaining the data dictionary.***
- ***Change History: Record of changes made to the data dictionary, including dates and descriptions of modifications***

### 6.4.4. Usage Guidelines

- Consistency: Ensure consistency in naming conventions, data types, and descriptions across all data entities and attributes.
- Clarity: Provide clear and concise descriptions for each data entity and attribute to facilitate understanding and usage.
- Accessibility: Make the product data dictionary easily accessible to developers, stakeholders, and other relevant parties.
- Maintenance: Regularly update the product data dictionary to reflect any changes or additions to the data model.

### 6.4.5. Example

```
brdr_bridge ⌄ {
  description:
                    Bridge: - bridge

  id*               string
  uid*              string($uuid)
  display_name*     string
  ordinal           integer($int32)
                    minimum: 0
  flags             integer($int64)
                    minimum: 0
  version           string($binary)
  admin_area        string
                    title: orgin source - ADMINAREA,

                    Administrative area or geographic stratification for the structure. This is used in sub

  altir_load        number($float)
                    title: orgin source - ALTIRLOAD,

                    Alternate inventory load rating. Optional non-NBI field to hold a load rating by some a

  altirmeth         string
                    title: orgin source - ALTIRMETH,

                    Alternate inventory rating method. Optional non-NBI field to indicate the method used i

  altor_load        number($float)
                    title: orgin source - ALTORLOAD,

                    Alternate operating load rating. Optional non-NBI field to hold a load rating by some a

  altormeth         string
                    title: orgin source - ALTORMETH,

                    Alternate operating rating method. Optional non-NBI field to indicate the method used i

  app_spans         number($double)
                    title: orgin source - APPSPANS,

                    Number of approach spans. NBI Item 46: Record the number and indicate with a 4-digit nu
                    all spans of most bridges, the major unit only of a sizable structure, or a unit of mat

  bb_brdgeid        string
                    title: orgin source - BB_BRDGEID,

                    Neighbor structure number for structures crossing a state border. Should be coded with
                    neighboring state's 15-digit NBI structure number for any structure noted in Item 98. T
                    entire 15-digit field must be accounted for including zeros and blank spaces whether th
                    item is blank.
```

This standard establishes requirements for documenting data entities and attributes within the data dictionary. Input from and output to intermediate files that users can access are excluded from these requirements. By excluding these interactions, organizations can maintain clarity, focus, and consistency in their product documentation and data management practices.

## 6.5. Application Infrastructure Component List

### 6.5.1. Description

*The Application Infrastructure Component List is a required artifact that contains the application infrastructure components, which support the development, maintenance, or operation of the application.* Refer to the *Critical Application Infrastructure Currency Standard* for additional information.

## 6.6. Accessibility Conformance Report (ACR)

### 6.6.1. Description

The Accessibility Conformance Report (ACR) is developed by completing a Voluntary Product Accessibility Template (VPAT), and details how the AASHTOWare product complies with the federal Section 508 of the Rehabilitation Act and the Web Content Accessibility Guidelines (WCAG). *An ACR must be prepared and submitted to AASHTO for each project where a new software product is developed, when a product is redeveloped, a change is made to accessibility requirements, or an update is applied to an application user interface that affects accessibility. The contractor must determine if*

*the ACR needs to be updated, make the appropriate modifications, and send the modified ACR to the appropriate AASHTO staff member for publishing.*

*All products must meet Section 508 and WCAG Level AA requirements.*

### 6.6.2. Content

*The AASHTO VPAT template is available at* https://www.aashtoware.org/about/standards-and-guidelines/. *Additional information on the VPAT and ACR is available on the Information Technology Industry Council web site at https://www.itic.org/policy/accessibility/vpat.*

*Additional information on Section 508 and WCAG guidelines is available on the following sites:*

*https://www.section508.gov/*

*https://www.w3.org/WAI/*

## 6.7. Project/MSE Archive Package

### 6.7.1. Description

*The Project/MSE Archive Package is an archive of the final product, project materials, and development artifacts. A Project Archive Package must be prepared and submitted to AASHTO at the closeout of each project, and an MSE Archive Package is prepared and submitted at the closeout of each MSE work effort.*

### 6.7.2. Content

*This project archive package includes the Product Installation Package, ACR, plus all approved and unapproved deliverables and review gate approval requests approved and rejected during the lifecycle of the project. In addition, the required artifacts for a project must be included in the project archive package, including the Application Infrastructure Component List, Technical Design Specification (TDS), Development and Maintenance Documentation, other artifacts created during the life of the project including the source code, build procedures, and any other information or documentation needed to setup, configure, change, and rebuild the final product.*

*An MSE archive package includes the same content with the exception of the RTM, Technical Design Specification (TDS), and the Development and Maintenance Documentation, which are not required for MSE work.*

# 7. Forms and Templates

## 7.1. Work Plan Templates

Two Microsoft Word templates are available for preparing project and MSE work plans. These templates include all the required information that must be included for each project and MSE work plan.

Both templates are available for download on the AASHTOWare SharePoint workspace and on the AASHTOWare web server at the URLs listed below:

Project Work Plan Template
https://www.aashtoware.org/wp-content/uploads/2022/10/AASHTOWare_Project_Work_Plan_Template_08292022.docx

MSE Work Plan Template
https://www.aashtoware.org/wp-content/uploads/2022/10/AASHTOWare_MSE_Work_Plan_Template_08292022.docx

## 7.2. Review Gate Approval Request Form

The following form or an equivalent form with the same content must be used for submitting review gate approval requests by the contractor and for documenting the approval decision by the task force.

This form is available for download on the AASHTOWare SharePoint workspace and on the AASHTOWare web server at:

*https://www.aashtoware.org/wp-content/uploads/2018/03/AASHTOWare_Review_Gate_Approval_Request_Form_05282010.docx*

**AASHTOWare**
**XXXXX Review Gate**
**Approval Request**

**To:** Task Force Chairperson

**From:** Contractor Project Manager

**Project/Product Name:** nnnnnn

**Submission Date:** mm/dd/yyyy **Requested Approval Date:** mm/dd/yyyy

Provide summary or comments regarding the review gate approval request.

### Deliverables

The following deliverables are attached and submitted for approval with the review gate approval. Reference materials regarding prior stakeholder review and approval are also noted and attached.

| Deliverable Name | Document/File Name | Version | Reference Material |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

### Checklist

Answer “yes” or “no” to each of the following questions.

| Question | Yes | No |
|---|---|---|
| Is each major deliverable compliant with AASHTOWare Standards? | | |
| Have all issues regarding the major deliverables or other work associated with the review gate been resolved? | | |
| Does each major deliverable implement or support all user requirements in the URS? | | |

### Noncompliance/Open Issues/Non Supported Requirement

If “no” is answered to any of the above questions: (1) Describe any areas of noncompliance with existing standards and include the justification, (2) Describe all open issues and the planned resolution for each, and/or (3) Explain why any user requirement(s) are not implemented or supported in a major deliverable. If needed, include an attachment.

| Noncompliance/Issue/Requirement | Planned Resolution/Justification/Explanation |
|---|---|
| | |
| | |
| | |
| | |
| | |

**Contractor Acknowledgement**

Acknowledged by: _____ on behalf of _____

Signature: _____ Date: _____

**Task Force Approval**

Approved (Yes or No): ____

Reasons for Rejection and/or Directions to the Contractor

| |
|---|
| |

Approved by: _____ on behalf of _____

Signature: _____ Date: _____

**AASHTO Project Manager Review**

Reviewed by: _____ on behalf of _____

Signature: _____ Date: _____

AASHTO Project Manager Comments

| |
|---|
| |

## 7.3. Example Status Report

The following example status report satisfies the status reporting requirements described in this standard and the Project and MSE Work Plan Templates. Any other report format may with the same content. Additional content may be added as deemed appropriate by the task force or contractor.

This document is available for download on the AASHTOWare SharePoint workspace and on the AASHTOWare web server at:

*https://www.aashtoware.org/wp-content/uploads/2018/03/AASHTOWare_Status_Report_Template_07012011.doc*

**[Product/Project Name]**
**Status Report**

**Project Manager:** **Date:**

**Reporting Period:**

**Project Stage:**

## Summary View

*Indicate the red, green, yellow status of each key status area of the project/work effort.*

| Area | This Period | Last Period |
|---|---|---|
| Schedule | Green | Yellow |
| Scope | Green | Green |
| Budget | Red | Yellow |
| Deliverables | Green | Green |
| Task Force Communication | Green | Green |
| Risk/Issue Management | Yellow | Green |
| Change Management | Red | Yellow |

## Accomplishments for This Period

*List the major accomplishments completed in this period from the project/work effort schedule.*

## Planned Activities for Next Reporting Period

*List the activities from the project/work effort schedule that are planned for the next reporting period.*

## Budget Status

*Identify the current budget status, including the initial budget from the work plan, last approved budget, current estimate, reason for new estimate, and cost expenditures to date.*

## Milestones/Deliverables

*List the major milestones and deliverables (including project closeout), the planned and actual start dates, planned and actual end dates, and the percent complete for each milestone/deliverable.*

## Changes Requests

*Identify change requests that occurred during this reporting period. Provide the status of the new change requests and all open change requests.*

## Risks

*List the current highest risk factors for the project/work effort and any actions taken to mitigate the risk.*

**Issues**

*List all open issues, the actions taken to address each issue, and the status of the actions.*

## 7.4. Installation Package Checklist

This checklist is for the vendor to use in preparing the Product Installation Package. Each item should be checked as the item is completed.

| ITEMS | X |
|---|---|
| **Documentation** | |
| • Name of product is included on all items in the installation package | |
| • Cover Letter | |
| • Contents List | |
| • Content List states the platform (computing environment) the installation package is for | |
| • Summary of package contents | |
| • User documentation or updates for existing documentation (including update instructions) | |
| • System documentation or updates for existing documentation (including update instructions) | |
| • Platform specific installation instructions | |
| • Special instructions | |
| • Summary of changes in this release | |
| • Checklist | |
| | |
| **Software** | |
| • Appropriate media used | |
| • Product software | |
| • Command Language Procedures (Scripts, JCL, EXECs, EXEs) | |
| • Database Definition Procedures | |
| • Installation jobs | |
| • Third party software at appropriate release (if applicable) | |
| • Virus scan has been passed | |
| | |
| **Hardware** | |
| • Hardware security device (if applicable) | |

This document is available for download on the AASHTOWare SharePoint workspace and on the AASHTOWare web server at:

*https://www.aashtoware.org/wp-content/uploads/2022/08/AASHTOWare_Installation_Package_Checklist_05062022.docx*

## 7.5. Installation Package Contents List

This is an example of a contents list which is shipped as part of the Product Installation Package.

| **Shipped From:** | **Shipped To:** |
|---|---|
| Shipment Date _____/_____/_____ | Release Number -- > |
| Product Name | |
| Platform / Version (computing environment) | |
| Distribution Method (Electronic or Media Type) | |
| Hardware Security Device or Software Security Key (if applicable) | |
| | |
| **Documentation** | **Documentation Type** |
| User Documentation | ( ) New Manual / ( ) Updates |
| Implementation Documentation | ( ) New Manual / ( ) Updates |
| Security Management Documentation | ( ) New Manual / ( ) Updates |
| Manager or Administration Documentation | ( ) New Manual / ( ) Updates |
| Operator Documentation | ( ) New Manual / ( ) Updates |
| | |
| | |

This document is available for download on the AASHTOWare SharePoint workspace and on the AASHTOWare web server at:

*https://www.aashtoware.org/wp-content/uploads/2022/08/AASHTOWare_Installation_Package_Contents_List_05062022.docx*

This page is intentionally blank.


# QUALITY ASSURANCE STANDARD

## S&G Number: 1.010.03.7S

## Effective Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 3.4 | 8/09/2017 | Changed SCOJD to SCOA. Made minor edits and corrections. | 8/16/2017 Approved by T&AA |
| 3.5 | 3/19/2018 | Updated to match current process. Minor edits and corrections. | 6/22/2018 Approved by SCOA |
| 3.6 | 12/17/2019 | Updated to match current process. | 7/10/2020 Approved by SCOA |
| 3.7 | 7/26/2023 | Updated the AASHTOWare logo | 10/02/2023 Approved by SCOA |

# Table of Contents

# 1. Purpose

The purpose of the Quality Assurance (QA) Standard is to define the responsibilities of the project/product task forces and contractors in ensuring that products are being developed and implemented in compliance with AASHTOWare Standards. The activities in the standard focus on evaluating if required deliverables and artifacts are created in compliance with standards; and if required processes in the standards are being followed.

The activities do not address whether a deliverable or artifact meets its intent or purpose. Review and acceptance are the responsibility of the task force and should be completed prior to submission for QA evaluation. The activities also do not require areas of noncompliance to be resolved; however, recommendations for resolution and common problems found will be used for process improvement within the applicable standards and within the internal procedures used by each task force and contractor.

*This standard applies to both AASHTOWare projects and annual MSE work efforts and the required deliverables and artifacts that are required by all AASHTOWare standards. The requirements for compliance with this standard are shown in red italicized text.* ***New requirements that were not in the prior version are shown in bold italicized text.***

An artifact is defined as a tangible by-product of software development, maintenance, or project management activity; a deliverable is an artifact that shall be delivered to the task force and approved.

Examples of deliverables include the System Requirements Specification (SRS), Requirements Traceability Matrix (RTM), and Beta Test Results Report. Examples of required artifacts are Review Gate Approval Requests, Application Infrastructure Component List, and Technical Design Specification (TDS).

# 2. Task Force/Contractor Responsibilities

The project/product task force and contractor responsibilities in regards to the AASHTOWare Quality Assurance (QA) Standard are summarized below. Additional details on these responsibilities are provided in the *Procedures* section of this document.

- *Plan the QA activities defined in this standard in all project and MSE work plans, including the annual QA meeting.*
- *Provide a list of completed deliverables and required artifacts once a year, as requested by the QA analyst.*
- *Work with the AASHTOWare QA Analyst before the annual QA meeting to review the meeting agenda and discuss deliverables and artifacts to be reviewed and any concerns with the Standards and Guidelines Notebook.*
- *Submit requested deliverables, artifacts, and supporting information to the AASHTOWare QA Analyst.*
- *Meet with the QA Analyst once a year. Lead a discussion on the deliverables and artifacts produced and any concerns encountered during the year. The contractor work site is the preferred location for the QA meetings.*
- *Provide the QA Analyst with access to deliverables and artifacts as requested.*
- *Review evaluation reports and provide comments.*

# 3. Required Deliverables and Artifacts

The following summarizes the required deliverables and artifacts that shall be created and/or delivered to comply with the Quality Assurance Standard.

- *Email a list of deliverables and artifacts completed during the fiscal year to the QA analyst.*

# 4. Procedures

The following provides detailed descriptions of quality assurance procedures that involve the project/product task force and/or contractor.

All correspondence sent regarding the QA procedures should be sent or copied to the task force chair, AASHTO Project Manager (PM), SCOA liaison, T&AA liaison, and QA analyst.

## 4.1 Plan QA Activities

*Each task force shall plan the QA activities by the task force and contractor in the appropriate project or MSE work plan. This includes preparing the list of completed deliverables and artifacts, submitting selected items to the QA analyst, meeting once a year with the QA analyst, and reviewing and responding to evaluation reports.*

## 4.2 Schedule QA Meeting

*The task force chair or AASHTO project manager (PM) shall schedule a QA meeting once a year at the contractor work site or an alternate site selected by the chair or AASHTO PM.* This visit will normally be scheduled after the end of the fiscal year between August 1 and November 30.

The meeting may be scheduled in conjunction with a planned task force meeting or as a separate meeting. *The minimum attendees shall be the contractor project manager or designee, task force chair or designee, AASHTO project manager, and the QA analyst.* It is also recommended that the T&AA liaison, SCOA liaison, and other contractor staff attend.

The QA analyst will send a reminder message to the task force chairs and AASHTO PMs regarding the scheduling of the QA meeting by the end of the current fiscal year.

## 4.3 Prepare the QA Meeting Agenda

The QA analyst will provide the AASHTO project manager, task force chair, T&AA liaison, and contractor project manager the draft QA meeting agenda approximately one month before the QA meeting. The agenda may be finalized by web meeting, conference call, email, or other methods of communication requested by the agenda review participants.

The purpose of finalizing the agenda is to identify the discussion topics to be included in the QA meeting, identify the version of the product for which artifacts will be reviewed, and to identify any artifact exclusions.

## 4.4 Conduct the QA Meeting

The QA analyst will meet with the contractor project manager, task force chair, T&AA liaison, AASHTO PM, and other interested parties between August 1 and November 30. The purpose of the meeting will be to:

- Discuss deliverables, artifacts, emails, and other items that will be included in the review;
- Solicit feedback from the contractor staff and task force representatives on the current QA process and standards;
- Collect suggestions for improving the QA process and standards;
- Discuss and obtain feedback on new standards currently under development and existing standards currently being revised; and
- Discuss standards, processes, or new technologies used in or planned for the product, and other related items.

### 4.5 Review Deliverables and Artifacts

After the meeting, the QA analyst will begin evaluating each selected item for compliance against the applicable standard(s). The results of each evaluation are documented in a preliminary QA evaluation report. The report will document where the items are not in compliance with the applicable standards and will reference any exceptions that have been granted. The report also includes recommended actions to address the areas of noncompliance. When completed, the preliminary evaluation report is sent to the task force chair and AASHTO PM with a request for comments and feedback.

### 4.6 Review Evaluation Reports and Provide Comments

After receiving the preliminary evaluation report, the task force chair should distribute the report to the task force members and contractor. The QA analyst will follow up with the task force chair, AASHTO PM and contractor representatives, as needed, to review the evaluation results and noncompliance issues and answer questions.

### 4.7 Resolve Issues and Provide Comments

*The task force shall review the preliminary evaluation report and decide if any corrective actions will be taken to resolve the noncompliance issues. The task force chair shall then prepare a response to the evaluation report and send the response to QA analyst and copy the task force members, contractor, AASHTO PM, and T&AA and SCOA liaisons.* If the task force or contractor has any suggestions to improve the QA process and/or to minimize noncompliance, these suggestions should also be included in the response.

The decision to resolve or not resolve noncompliance issues should be included in the response. If a major deliverable or artifact will be updated, a target date should be provided for the revised deliverable. Revised deliverables and artifacts should be submitted through the task force chairperson to the QA analyst for re-evaluation.

### 4.8 Prepare and Distribute Final Evaluation Reports

After receiving the task force response, the QA analyst will prepare a final QA evaluation report, which includes the task force response. If an item was resubmitted and re-evaluated, these results/actions are included in the final report.

A cover letter or email to the task force chairperson will be prepared and sent with the final QA evaluation report. A copy is provided to the T&AA liaison and AASHTO PM. The task force chairperson distributes the reports to the task force members and contractor representatives. Any additional distribution should be handled by the recipients.

### 4.9 Prepare and Review Annual QA Summary Report

Following the completion of all QA evaluations, the QA analyst will produce a report that summarizes the results, findings, and recommendations of all evaluations performed during the fiscal year. If any trends are observed, the report will also document these along with any recommended actions to address the trends.

The QA analyst will provide the report to the T&AA task force for review and comment and make appropriate updates. After the T&AA review, the T&AA chair will send the report to SCOA and AASHTO staff.

### 4.10 Recommend Improvements

Based on comments and recommendations received from the review of the QA Summary Reports, trends found, and findings from the QA meetings, SCOA and T&AA will determine if changes are needed to the QA standard, changes are needed to other existing standards, or new standards are needed.

If changes or new standards are needed, SCOA will provide direction to T&AA regarding the time frames for planning and implementation of new and revised standards.

# 5. Technical Requirements

There are no technical requirements for this standard.

# 6. Deliverable and Artifact Definitions

## 6.1 QA Evaluation Report

### 6.1.1 Description

This report is not prepared by the task force or contractor; however, the task force and contractor should review the report and provide a response to the findings.

### 6.1.2 Content

The results of the QA evaluation will be provided in this report. The report will include the following content. Other content may also be added.

- The date the report was prepared.
- Name of the project or product reviewed.
- Meeting overview, date, location, and attendees.
- Deliverables, artifacts reviewed
- Approval documentation reviewed
- Findings and recommendations from the QA review, including any area of noncompliance.
- Recommended actions to address noncompliance.
- Summary of other agenda items from the QA review meeting (reviewed of existing standards, planned standards, suggestions, etc.)

## 6.2 QA Summary Report

### 6.2.1 Description

This report is prepared by the QA analyst and summarizes QA results for the fiscal year.

### 6.2.2 Content

The report is prepared for SCOA and includes the date, location, and attendees for all QA meetings held during the fiscal year. The report also summarizes all QA evaluation results and findings and may include recommended changes to the standards and guidelines.


# 2 – Technical Standards and Guidelines

This page is intentionally blank.


# SECURITY STANDARD

## S&G Number: 2.020.01.11S

## Effective Date: April 1, 2026

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.7 | 3/20/2019 | Removed bold from new requirements introduced in 2018. | 7/30/2019 Approved by SCOA |
| 1.8 | 10/18/2021 | Updated broken links to web pages. | 10/22/2021 Approved by T&AA |
| 1.9 | 9/30/2023 | Updated the AASHTOWare logo and corrected links. | 10/02/2023 Approved by SCOA |
| 1.10 | 8/07/2024 | Added multifactor authentication and vulnerability scanning. | 12/11/2024 Approved by SCOA |
| 1.11 | 9/11/2025 | Removed bold from new requirements introduced in the last update. | 3/12/2026 Approved by SCOA |

# Table of Contents

# 1. Purpose

AASHTOWare recognizes its responsibility for providing secure applications. Further, AASHTOWare endorses and demands that applications delivered meet user needs and maintain the highest level of application, data, and infrastructure security as practical. This standard defines the security requirements and responsibilities that must be met when developing AASHTOWare products.

*This standard applies to all new development and major security related enhancement projects.* The standard does not normally apply to small enhancements and software maintenance efforts; however, it should be reviewed when these efforts involve security. In addition, the standard primarily addresses multi-user applications except where noted otherwise.

*The requirements for compliance with this standard are shown in red italicized text.* ***New requirements that were not in the previous version of the Standards and Guidelines are shown in red bold italicized text.***

# 2. Task Force/Contractor Responsibilities

The product task force and contractor responsibilities for the Security Standard are summarized below:

- *Ensure that business specific security requirements are defined and implemented.*
- *Ensure that the security technical requirements defined in this standard are implemented in the product when applicable.*
- *Ensure that industry best security practices and emerging security trends are considered and implemented appropriately.*

# 3. Required Deliverables and Artifacts

*The following summarizes the required deliverables and artifacts that must be created and/or delivered in order to comply with the Security Standard. Definitions and content requirements are provided in the Deliverable and Artifact Definitions section of this document.*

- *Security Requirements – must be included in the System Requirements Specification (SRS).*
- *System Roles – must be included in the SRS.*

# 4. Procedures

## 4.1 Establish Security Requirements

For each new development or major enhancement effort, the task force and/or contractor should:

- Analyze the business needs, expectations, and constraints that impact the data, application, and system security,
- *Define the applicable security requirements and system roles for the effort and include in the System Requirements Specification (SRS).*

## 4.2 Include AASHTOWare Security Technical Requirements

*Where applicable, the task force and/or contractor must ensure that the technical requirements listed below are included in the SRS.*

## 4.3 Review Impact to Existing Security

For each enhancement or modification to an existing application, the task force and/or contractor should ensure that there is no impact to the existing security introduced by the implementation of the enhancement or modification.

### 4.4 Test and Implement the Security Requirements

The task force and contractor should ensure that all security requirements in the approved System Requirements Specification are tested and implemented.

# 5. Technical Requirements

Research performed by T&AA reveals that there is a wide variety of tools, products, and computer environments in use at member agencies. Such variety exists that identifying detailed security requirements is not practical. Therefore, the following high-level security requirements are identified.

*In addition to the standards listed below, product contractors and task forces are responsible for ensuring that industry best security practices and emerging security trends are considered and implemented appropriately.*

## 5.1 Authentication

*User authentication routines must support one or more of the following.*

- *Lightweight Directory Access Protocol (LDAP)*
- *Integrated Windows Authentication (IWA)*
- *WS-Federation (e.g. Active Directory Federated Services)*
- *OAuth 2.0*
- *OpenID 2.0*
- *OpenID Connect*
- *SAML 2.0*

Internal application user authentication is allowed, but should not be promoted as the primary authentication method for AASHTOWare products.

References:

| | |
|---|---|
| LDAP | https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol |
| Integrated Windows Authentication | https://en.wikipedia.org/wiki/Integrated_Windows_Authentication |
| WS-Federation | https://en.wikipedia.org/wiki/WS-Federation |
| OAuth 2 | https://oauth.net/2/ |
| OpenID | https://openid.net/developers/specs/ |
| SAML 2 | https://en.wikipedia.org/wiki/SAML_2.0 |

### 5.1.1 Multifactor Authentication

Multifactor authentication (MFA) should be supported for all authenticated access. Requiring MFA should be configurable at the organization level to allow customers to opt in or out.

## 5.2 Encryption of Sensitive Data

*User accounts, passwords, and any other data identified as being sensitive must be encrypted while in transit or at rest using methods and techniques accepted by the industry as being reliable and secure.* This includes, but is not limited to, data transmitted on internal, external, public, or private networks and data stored in a database management system such as Oracle, Microsoft SQL Server, etc.

References:

| | |
|---|---|
| Data encryption standards | https://en.wikipedia.org/wiki/Data_Encryption_Standard |
| Microsoft SQL encryption | https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/sql-server-encryption?view=sql-server-ver16 |
| | https://www.databasejournal.com/ms-sql/sql-server-2005-security-part-3-encryption/ |
| Encryption and SQL injection | https://owasp.org/www-community/attacks/SQL_Injection |
| Oracle Transparent Data Encryption | https://docs.oracle.com/en/database/oracle/oracle-database/26/dbtde/index.html |
| Configuring data encryption and integrity | https://docs.oracle.com/cd/E11882_01/network.112/e40393/asoconfg.htm#ASOAG020 |
| Payment Card Industry standards | https://www.pcisecuritystandards.org/ |

### 5.2.1 Transport Layer Security (TLS)

The Transport Layer Security (TLS) protocol provides secure communications on the Internet for such things as e-mail, Internet faxing, and other data transfers. The primary benefit of TLS is the protection of web application data from unauthorized disclosure and modification when it is transmitted between clients (web browsers) and the web application server, and between the web application server and back end and other non-browser based enterprise components.

*Any AASHTOWare-related web site for customer access, including web applications and support sites, must use TLS 1.2 or higher.*

### 5.2.2 Secure Hash Algorithm 2 (SHA-2)

SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the United States National Security Agency (NSA). Security certificates issued by a trusted certificate authority should be using the SHA-2 cryptography.

*Any AASHTOWare-related web site for customer access, including web applications and support sites, must use SHA-2 based security certificates.*

## 5.3 Role Based Security

Applications must use role-based security. Roles must be defined within the application, and controlled by either the application or groups in a common directory service (e.g. Active Directory Security Groups). Applications should not require users to have accounts that access databases directly. Instead, applications should use a proxy account to perform database create, read, update, and delete actions.

## 5.4 Industry Standard Passwords

*Passwords must follow industry recognized standards for minimum length, makeup (i.e., characters, numbers, or symbols), and change frequency*.

References:

| | |
|---|---|
| Digital Identity Guidelines, Appendix A | https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63B-4.pdf#page=98 |

| | |
|---|---|
| NIST Digital Identity Guidelines | https://pages.nist.gov/800-63-4/ |

### 5.5 Appropriate Levels of Hardening

*Hardware and software provided to AASHTOWare customers that is exposed to external network users, including Internet users, must be hardened to levels accepted by the industry as appropriate and effective for the hardware and software being used.*

References:

| | |
|---|---|
| World Wide Web Consortium Security | https://www.w3.org/mission/security/ |
| SANS Institute | https://www.sans.org/ |
| | https://isc.sans.org/ |
| Windows hardening guidelines | https://www.cisecurity.org/benchmark |
| | https://public.cyber.mil/stigs/downloads/ |
| Open Web Application Security Project (OWASP) | https://owasp.org/ |

### 5.6 Security Patches

AASHTOWare contractors should assist in identifying and monitoring security patches for third-party components used in AASHTOWare products. In addition, contractors should notify the licensees of the location where the patches may be obtained and provide any specific instructions needed to incorporate the patches into AASHTOWare products within a reasonable timeframe from when the manufacturer of the third-party component makes patches available.

### 5.7 Security Assessments

#### 5.7.1 Source Code

Application source code should be analyzed regularly using an appropriate tool to identify potential security vulnerabilities. Said vulnerabilities should be resolved prior to release to production environments or delivery to customers. *Source code must be analyzed at least once annually by a trusted third-party assessor to ensure that current industry best practices are being implemented.*

#### 5.7.2 Hosted Environment

*Hosted environments must have a security assessment performed at least once annually by a trusted third-party assessor to ensure the environment follows current industry best practices.*

*A Systems and Organization Controls 2 (SOC2), Type II report is required. For more details, see the Hosting Services Standard.*

## 6. Deliverable and Artifact Definitions

### 6.1 Security Requirements

#### 6.1.1 Description

*The security requirements of the proposed application, system, database, or enhancement must be included in the System Requirements Specification (SRS). In*

*addition, the security requirements must be included in the appropriate test procedures for alpha and best testing.*

### 6.1.2 Content

*The SRS must include a section where all security requirements are documented. Other methods that allow all security requirements to be easily identified may be used in lieu of this method.*

The security requirements should define:

- Privacy concerns associated with the application or data;
- The types of users that have access to the applications, systems, databases, and data (see system roles below);
- What each type of user has access to and the type of access allowed;
- AASHTOWare and member organization technical and organizational security requirements and constraints; and
- Security Requirements

## 6.2 System Roles

### 6.2.1 Description

*The SRS must define the roles of the various stakeholders that use and support the system.*

### 6.2.2 Content

The roles may be provided in any format that identifies the groups of users and stakeholders along with their roles and responsibilities regarding the proposed system. Example roles include users, managers, executives, system administrators, security administrators, database administrators, and application support personnel.

This page is intentionally blank.

AASHTO AASHTOWare™ SOLUTIONS

# CRITICAL APPLICATION INFRASTRUCTURE CURRENCY STANDARD

## Version: 2.030.03.5S

## Effective Date: January 1, 2025

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 3.1 | 7/01/2014 | Clarified the 24 and 12 month implementation requirement and defined “general availability” status. | 7/01/2014 Approved by T&AA Chair |
| 3.2 | 3/20/2019 | Made all columns on the application infrastructure component list required. | 7/30/2019 Approved by SCOA |
| 3.3 | 5/20/2020 | Clarified application infrastructure component list requirements and removed bold font for requirements added last year. | 7/10/2020 Approved by SCOA |
| 3.4 | 7/26/2023 | Added a column to the application infrastructure component list to identify the items to be updated and why items beyond N-1 are not being updated. | 10/02/2023 Approved by SCOA |
| 3.5 | 8/07/2024 | Removed the bold font for new requirements from last year. | 12/11/2024 Approved by SCOA |

# Table of Contents

# 1. Purpose

This document describes the requirements needed to ensure AASHTOWare products maintain compatibility with updated technology and drop support for outdated technology. Task forces and contractors shall ensure AASHTOWare products are tested with new versions of development tools, operating systems, utilities, databases, and other related infrastructure components. Changes to AASHTOWare products needed due to updated versions of critical infrastructure components shall also be planned and implemented in a timely manner. In addition, each critical infrastructure component used in AASHTOWare products shall be supported by the component's vendor.

The actions included in this standard will ensure AASHTOWare products are compatible with updated critical infrastructure components without forcing customers to upgrade their environments immediately.

*This standard applies to both AASHTOWare projects and annual MSE work efforts. The requirements for compliance with this standard are shown in red italicized text.* ***New requirements that were not in the previous version of the Standards and Guidelines are shown in red bold italicized text.***

# 2. Task Force/Contractor Responsibilities

The product task force and contractor responsibilities for this standard are summarized below:

- *Each task force and/or their contractor shall maintain a list of application infrastructure component products for each AASHTOWare product. These are those component products required to support the development, maintenance, or operation of each AASHTOWare product, such as browsers, database management systems, operating systems, and development tools.*
- *Task forces and contractors shall ensure that each product supports and is compatible with at least the most recent release of application infrastructure component products and the release immediately prior to the most recent release, often respectively referred to as N and N-1. AASHTOWare products that have been replaced and are still in use are exceptions to this requirement.*
- *At least once a year, the task force or the contractor shall review the list of application infrastructure components supported in each AASHTOWare product and identify the components that have been upgraded and those that will lose vendor support. The task force or contractor shall also indicate which components will be upgraded or removed in the work plan for the next fiscal year. If a component is beyond N-1 and not selected to be upgraded or mitigated, the reason why must be documented in the application infrastructure component list.*
- *Plans shall be created and executed to test each AASHTOWare product with updated application infrastructure component versions.*
- *Plans shall be created and executed to support new versions of the application infrastructure components in each AASHTOWare product in the time frames discussed below in the Determine Which Components Need to be Updated section.*
- *When a vendor announces the discontinuation of support for a specific version of an application infrastructure component, a plan shall be created and executed to migrate the product away from that version, as discussed below in the Determine Which Components Need to be Dropped section.*
- *The current version of the application infrastructure component list shall be included or referenced in each product or product work plan for an existing AASHTOWare product.*

- *All work activities associated with this standard shall be planned in both project and MSE work plans.*

# 3. Required Deliverables and Artifacts

The following summarizes the required deliverables and artifacts that shall be created and/or delivered in order to comply with the Critical Application Infrastructure Currency Standard.

- *Application Infrastructure Component List: This is list of all application infrastructure components, or those required to support the development, maintenance, or operation of their product(s). Application infrastructure component examples include integrated development environments, development languages, run time environments, configuration management and version control software, browser, desktop and server operating systems, testing software, and database engines. This list includes the component name; version level the product uses, supports, or depends on; and the owner/vendor of the component. If the component is open source, the license information shall also be included.*

  If available, the list should also include the next version, availability date of next version, and discontinuation of support date of the current version. Refer to *Deliverable and Artifact Definitions* section for the required content of the lists.

# 4. Procedures

## 4.1 Prepare the Application Infrastructure Component List

*The initial activity is for the contractor to prepare the application infrastructure component list. The contractor shall include all application infrastructure components in the list that are used to develop, support, or execute each AASHTOWare product.* The components should be categorized to clearly identify components that are supported by the client organizations (e.g., database, server operating system, and workstation operating system), third-party components the application relies on to operate (e.g., controls, communication protocols, and XML schemas), and components needed to develop, test, and support the application (e.g., integrated development environments, testing tools, and other items not needed in the application users' organizations).

## 4.2 Maintain the Application Infrastructure Component List

*The contractor shall maintain the application infrastructure component list by updating the list when the components needed to develop, support, or execute the application change.* New components will be added and new versions for existing components will be updated as needed. Other components or component versions will be removed from the list when the AASHTOWare product drops support for a component or a specific version of a component.

## 4.3 Review Application Infrastructure Component Available Releases

*At least once a year, the contractor shall review versions of the most recent generally available production or stable releases (referred to as the N versions) of application infrastructure components used by or relied on by the application and their associated release history.* The anticipated release dates of new versions of application infrastructure components should also be reviewed so the task force and contractor are aware of planned updates and their scheduled release dates. The contractor should also make note of any version of an application infrastructure component that the component vendor has announced a planned date when support will be discontinued and add the date to the list.

## 4.4 Determine Which Components Need to be Updated

The results of the previous step should be compared to the application infrastructure component list. *If a newer version of an application infrastructure component is available, the contractor shall develop a plan to complete the development and testing to support the*

*new version of the application infrastructure component in each AASHTOWare product within 24 months after each new component version achieves general availability status. The production support for the new version of an application infrastructure component shall be included in the next planned release of the AASHTOWare product after the 24 month date.*

*Browsers are an exception to this requirement. New versions of browsers shall be tested and implemented within 12 months after the date of general availability and supported in the next planned release after the 12-month date.*

General availability is a term used by Microsoft and other vendors that is defined as that stage of the product life cycle when the product is stable, having successfully passed through all earlier release stages (such as beta and candidate releases) and is believed to be reliable, free of serious bugs, and suitable for use in production systems. The general availability date is announced by the vendor of each component product and is typically posted on the vendor's web site.

*The plan to implement the new versions shall be included in the appropriate upcoming work plan as discussed in the Prepare the Work Plan section.*

The following includes examples of the 24- and 12-month requirements.

- If a new version of the SQL Server database management system reaches general availability status on March 15, 2015, this new version of SQL Server shall be supported in each AASHTOWare product in the next planned release after March 15, 2017.
- If a new version of the Internet Explorer (IE) browser reaches general availability status on May 10, 2015, this new version of IE shall be supported in each AASHTOWare product in the next planned release after May 10, 2016.

This timeline may require AASHTOWare products to support more than the N and N-1 versions of application infrastructure components, depending on the upgrade schedule of the components' manufacturers. In these situations, task forces may accelerate product updates so that products only support N and N-1 versions of application infrastructure components.

When the availability date of the next version of an application infrastructure component is known, these should be added to the application infrastructure component list.

## 4.5 Determine Which Components Need to be Dropped

*When a vendor announces the discontinuation of support for a specific version of an application infrastructure component, the contractor shall develop a plan to migrate each AASHTOWare product away from that version by the time the vendor drops support for the component. The plan shall be included in the appropriate upcoming work plan as discussed in the Prepare the Work Plan section.*

At this point, the contractor should also add the planned date for discontinuation of support to the application infrastructure component list

## 4.6 Prepare the Work Plan

*When developing the next project or MSE work plan for an existing AASHTOWare product, the contractor shall include or reference the current version of the application infrastructure component list in the work plan, and include the planned activities in the work plan to test new versions of components; review the current list and make necessary updates; prepare the list if none exists; modify the product as required to implement new versions of components and to discontinue outdated versions of components. This does not mean that application infrastructure component lists may only be reviewed and updated while work plans are being prepared. Application infrastructure component lists may be reviewed any time during the year. Also, application infrastructure component lists for all products being*

*supported as specified in Task Force/Contractor Responsibilities section above must be reviewed and updated at least annually, regardless of whether the application is specifically mentioned in a work plan.*

*Activities shall be included in the work plan to ensure that:*

- *New versions of critical application infrastructure components are:*
  - *Tested and ready to implement in each AASHTOWare product within 24 months after the component reaches general availability status.*
  - *Implemented and fully supported in the next planned release of the AASHTOWare product after the 24 month date.*

- *New versions of browsers are:*
  - *Tested and ready to implement in each AASHTOWare product within 12 months after the version reaches general availability status.*
  - *Implemented and fully supported in the next planned release of the AASHTOWare product after the 12 month date*
- Versions of critical application infrastructure components that will be no longer supported by the vendor will be discontinued in each AASHTOWare product by the date that the vendor drops support for the component.
- The current version of the application infrastructure component list is included or referenced in all work plans for an existing AASHTOWare product.
- The current version of the list is reviewed and kept up to date during the execution of the work plan. A list is prepared if none exists.
- The new or most recent version of the application infrastructure component list is submitted at the Closeout Review Gate. (Refer to *the Software Development and Maintenance Process Standard* for additional information).

*These activities are included in the Application Infrastructure Upgrade Services section of an MSE work plan. The current application infrastructure component list is included or referenced in this section or the Tools and Technologies section.*

*If a project includes these type of upgrade services, these activities are included in the Work Activities section of the project work plan. The current application infrastructure component list is included or referenced in this section or the Tools and Technologies section.*

*For projects developing a new product or redeveloping an existing product, a new application infrastructure component list shall be created and submitted at the Closeout Review Gate.*

# 5. Technical Requirements

There are no technical requirements for this standard.

# 6. Deliverable and Artifact Definitions

## 6.1 Application Infrastructure Component List

### 6.1.1 Description

*The application infrastructure component list is a required artifact that contains the application infrastructure components, which support the development, maintenance, or*

*operation of the application.* It is used to document the components associated with an application and to help determine when components need to be updated or dropped.

### 6.1.2 Content

*The application infrastructure component list shall contain the following items:*

- *Component Name*
- *Component Category*
  - *External - Components that are supported by client organizations (e.g., database, server operating system, and workstation operating system),*
  - *Internal - Components the application relies on to operate (e.g., controls, communication protocols, and XML schemas)*
  - *Support - Components needed to develop, test, and support the application (e.g., integrated development environments, testing tools, and other items not needed in the application users' organizations)*
- *Component Version*
- *Owner/Vendor Name of Component*
- *License Information (for open source components)*
- *Next Available Version (when known) (This is the next available version from the most recent generally available version, not necessarily the next version newer than the version in use.)*
- *Date When Next Version is Available (when known)*
- *Date When Support is Discontinued for Current Version (when known)*
- *An indicator of whether the component is included in the next fiscal year work plan.*
- *The reason the component was not selected for remediation if the component is not included in the next fiscal year work plan and the component is N-1 or older.*

This page is intentionally blank.

# DATABASE SELECTION AND USE STANDARD

## S&G Number: 2.040.04.5S

## Effective Date: January 1, 2025

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 4.2 | 3/15/2018 | Changed SCOJD to SCOA. | 6/22/2018 Approved by SCOA |
| 4.3 | 2/06/2020 | Added PostgreSQL as an option for cloud-hosted solutions | 7/10/2020 Approved by SCOA |
| 4.4 | 9/30/2023 | Updated the AASHTOWare logo and corrected links. | 10/02/2023 Approved by SCOA |
| 4.5 | 8/07/2024 | Added Amazon Relational Database Service and Microsoft Azure SQL as options for cloud-hosted solutions. | 12/11/2025 Approved by SCOA |

# Table of Contents

# 1. Purpose

Relational databases are the preferred method of data storage for application programs. This is especially true for multi-user applications, where data update coordination between many users is essential. Databases provide built-in functions that lend themselves to performance, security, and multi-user access.

It is the purpose of this standard to define requirements and best practices for the use of databases in AASHTOWare product development. In addition, the standard provides information and recommendations which promote the preservation, sharing, and exchange of data supported by AASHTOWare products. *This standard applies to new development projects and projects and MSE work that make major changes to an existing product's data storage repository.*

*All requirements for compliance with this standard are shown in red italicized text.* ***The new requirements that were not in the previous version of the Standards and Guidelines are shown in red bold italicized text.***

# 2. Task Force/Contractor Responsibilities

The project/product task force and contractor responsibilities regarding this guideline are summarized below:

- *Ensure the AASHTOWare standard database platforms are supported in all development of new products and development efforts that include the establishment or replacement of an existing application's data storage repository.* Refer to the *Technical Requirements* section in this standard for a description of the standard database platforms.
- Consider supporting the standard database platforms when developing enhancements that make major changes to an existing product's data storage repository.
- Routinely survey the current and potential user base to determine what databases are supported, planned, being eliminated, and regarded as the preferred databases.
- Participate in research and testing new database platforms.
- Recommend new database platforms to be supported in specific products.
- Notify the T&AA Task Force when new database platforms are planned.
- *Include supported database platforms and versions in the Product Catalog and on applicable product web sites.*
- *Since database management systems are considered as application infrastructure components, the contractor and task force must also comply with the requirements of the Critical Application Infrastructure Currency Standard. These requirements include maintaining supported database platforms and versions in the product Application Infrastructure Component List, testing new releases of database platforms, dropping support for outdated versions, and planning the implementation of new versions.*
- Ensure compliance with all license requirements and report potential issues to AASHTO.

# 3. Required Deliverables and Artifacts

*The standard does not require any deliverables or artifacts to be created and/or delivered; however, the following must be maintained with the database platforms and versions of those platforms supported by each product.*

- *Application Infrastructure Component List*
- *Product Catalog*
- *Product web site (if applicable)*

# 4. Procedures

## 4.1 New Database Notification

When a project/product task force is making plans to add support for a new database platform, the task force chairperson should advise the T&AA liaison or T&AA task force chairperson. This is strictly a courtesy notification and may be communicated verbally, by phone, or email. This will allow T&AA to communicate any concerns to the project/product task force and contractor early in the product development life cycle.

## 4.2 Maintain Database Platform Currency

*As noted above, database platforms are considered application infrastructure components and must be included in the activities and artifacts required to comply with the Critical Application Infrastructure Currency Standard. These include, but are not limited to, the following.*

- *Include each database platform and the specific version supported in Application Infrastructure Component List for each product.*
- *Each product must support the most recent release of each supported database platform and the release immediately prior to that release (N and N-1).*
- *New versions of database platforms must be supported in each product within 24 months after the new version reaches general availability status.*
- *Products must not support versions of database platforms that are no longer supported by the vendor.*
- *Plan and execute the work activities in the appropriate work plan to test, evaluate, and implement new versions of database platforms and to eliminate non-supported versions.*

## 4.3 Database Selection

When new database software is selected for use in AASHTOWare products, the following selection criteria and additional consideration should be used.

### 4.3.1 Database Selection Criteria

The following selection criteria are used as a basis for evaluation of database products and their recommended use in the development of AASHTOWare products.

#### 4.3.1.1 Standards Conformance

The products recommended are chosen on the basis of their conformance with industry standards such as SQL and connectivity.

#### 4.3.1.2 Platform Support

The products recommended are chosen because of their support of a broad range of development and operational platforms (operating software/hardware). Special attention is given to those platforms which are currently employed by AASHTOWare products. Consideration is also given to those products which are current industry leaders.

#### 4.3.1.3 Scalability

The products recommended are highly scalable within their product family.

#### 4.3.1.4 Security

The product recommended should have adequate security features for database administration.

#### 4.3.1.5 Development Tools

The products recommended are accessible and usable by a broad range of development tools that are suitable for the development of AASHTOWare products.

#### 4.3.1.6 Middleware and Gateway

The recommended database product families provide middleware and gateways which permit access to and from other manufacturers' database products over a variety of networking types (differing network protocols).

#### 4.3.1.7 Replication

The products chosen support replicating data across a network and to different server environments.

#### 4.3.1.8 Product Viability

All products recommended are well established in the market place and/or the user community.

### 4.3.2 Additional Considerations

New AASHTOWare product development should also consider the items listed below when determining which new database(s) to support. It is also suggested that existing products utilize the items to determine if the list of currently supported databases can be reduced.

#### 4.3.2.1 Use of the Latest ODBC and JDBC Client Drivers

Software database drivers are available for most database platforms so that *application software* can use a common *application programming interface* (API) to retrieve the information stored in a database. AASHTOWare product development should ensure that the latest stable ODBC and JDBC client drivers are used when developing and maintaining AASHTOWare products.

#### 4.3.2.2 Surveying User base

In order to stay abreast of database platforms being used in the current and potential user base, AASHTOWare management should routinely survey the member departments to determine what databases are: preferred, currently supported, not used, planned for future use, and planned for retirement.

- The project/product task force should routinely solicit this information when surveying the current organizations licensing their products, as well as potential new users.

- The SCOA and the T&AA Task Force should routinely include questions regarding database platforms in the AASHTO Annual IT survey, which is sent to the chief information officer in each of the AASHTO member departments.

#### 4.3.2.3 Maintain the Minimum Number of Databases

AASHTOWare should select and maintain support for the minimum number of database platforms required to meet the user and organizational requirements for new and existing product development.

### 4.4 Update Product Catalog and Public Web Site

*In addition, to maintaining the Application Infrastructure Component List, the supported database platforms must be updated in the next release of the AASHTOWare Product Catalog. If a public web site exists for the product, the supported database platforms with version numbers must also be kept up to date with all other published application infrastructure components for the product.*

## 5. Technical Requirements

### 5.1 Enterprise (Multi-User) User Databases

*The following enterprise databases must be supported for all new development of AASHTOWare products and development efforts that include the establishment or replacement of an existing application's data storage repository for applications hosted at sites directly managed by customers.*

- *Oracle*
- *Microsoft SQL Server*

For software-as-a-service applications or other hosted applications that are not directly supported by customers, PostgreSQL, Amazon Relational Database Service, or Microsoft Azure SQL may be used instead of Oracle or Microsoft SQL Server.

### 5.2 Standalone (Single User) Databases

When using standalone databases, the following recommendations should be considered:

- Use a single standalone database engine within the application.
- Licenses should be included and distributed with the AASHTOWare product.
- Functionality to transfer data to and from the enterprise database should be included in the application.

## 6. Deliverable and Artifact Definitions

*The content of the following deliverables and artifacts must be maintained with the database platforms and versions of those platforms supported by each product.*

- *Application Infrastructure Component List – Refer to the Critical Application Infrastructure Currency Standard for the required content of this artifact.*
- *Product Catalog*
- *Product website (if applicable)*


# SPATIAL STANDARD

## Version: 2.050.01.7S

## Effective Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.5 | 2/12/2020 | Removed bold font from requirements added last year. | 7/10/2020 Approved by SCOA |
| 1.6 | 10/18/2021 | Updated broken links to web pages. | 10/22/2021 Approved by T&AA |
| 1.7 | 10/13/2023 | Updated the AASHTOWare logo and broken links. | 10/02/2023 Apprved by SCOA |

# Table of Contents

# 1. Purpose

A wise man once said, “To solve real world problems, you need to know where you are in the real world and understand a ‘Thing’s’ place in the real world.”

The purpose of this standard is to help (1) protect and preserve the DOTs’ information investment, (2) assure that individual assets are distinguished from one another, (3) improve data quality within AASHTOWare software, and (4) to increase each product’s value to its customers. To support these goals, this standard provides specifications to uniquely identify and locate assets to ensure the asset and its information remain correctly associated and a recommended workflow to improve spatial data quality within AASHTOWare software.

This standard supports improving the ability of AASHTOWare products to integrate with one another and with DOTs’ programs and data and delivering the right information at the point of need. Additionally, this standard advocates leveraging the significant GIS investments DOTs have made over the last 30 years to better integrate public services.

Discussion

Spatially-enabled data provides the foundation for effective data exchanges between agencies. Emergency events ignore borders, boundaries, and what agency has decision authority over people and assets. Spatially-enabled data and tools allow agencies to easily coordinate efforts and work from a common operating picture to solve problems regardless if they are adjacent state DOTs or DOTs working with emergency operation centers or other government organizations.

Further, customer expectations have changed. People are evolving to a mobile, “anytime/anywhere” culture and expect to be connected and have information at their fingertips. Consider how people use their phones to find a new restaurant, hail a rideshare, evaluate entertainment options, etc.

All this convenience is driven by location-aware databases focusing on answering a person’s *current* needs. AASHTOWare software should drive similar answers to customers’ questions. In the field, relevant (local) bridge or project information should be offered first. An inspector evaluating a newly built asset should see that asset’s location and specs to then verify it was built as designed. Location-based services are fundamental to delivering the right information focused on a customer’s current needs.

Mobile solutions leverage location to focus information to what is relevant to the user. Projects, assets, and other elements of interest are influenced by the things around them. Location-based evaluation of information is a natural method for finding and identifying important patterns to make business decisions.

Location acts as a “super key” to unlock data sharing for traditional data systems that don’t easily connect. Consider the following example. New regulations are announced about pedestrian safety, and agency leadership needs to know what highway projects fall within a newly defined 10,000-foot zone around all public schools. If the data is spatially enabled, this can take seconds to perform in a GIS but could take weeks to develop the project list if performed manually.

This power is only available if location is captured correctly. If poor methods are used, the cliché is realized: “garbage in/garbage out” (GIGO). Every member in the chain of AASHTOWare data creation, management, and use, from the software developer to the data consumer, has an important role to play in developing the best information that is necessary to make decisions that can save lives, stimulate the economy, improve efficiency, and help the traveling public.

This Spatial Standard helps protect DOT information investments from loss. Real-world coordinates “anchor” the information to the Earth (where the asset is located), so that it can consistently be searched for, found, retrieved, and used correctly in analysis regardless of what other reference keys get lost or change.

Traditional location descriptions in DOT systems (linear reference systems [LRS] such as route-milepoint or station-offset) and road name-cross street references often develop errors over time and require increased maintenance to keep current. Beginning points for station-offset and other station markers get destroyed. The locations described by them often cannot be relocated years later. Road network changes force linear referencing shifts over time. Road names and route names change.

For example, a school pedestrian crossing is located at “US-123, milepoint 7.258” in a DOT’s database. No location maintenance is performed. Five years later, after a new bypass is built, the crossing “jumps” many miles away when new network infrastructure is brought into the LRS. See Figure 1.

How did this happen? A bypass is built to improve traffic on the US highway. The “old” section of the US highway is re-designated to a state road with a new route number (SR-6789), and milepoint (2.123). While the asset (school crossing) hasn’t changed, its LRS designation has.

What this means for AASHTOWare is if a bridge, pavement, project, or other data asset is similarly referenced with an LRS, each DOT must invest staff time in maintenance work to recalibrate the data to correct errors. If LRS changes are ignored, analysis and reporting will deliver incorrect information to the customers, who depend on AASHTOWare to be the record of truth.

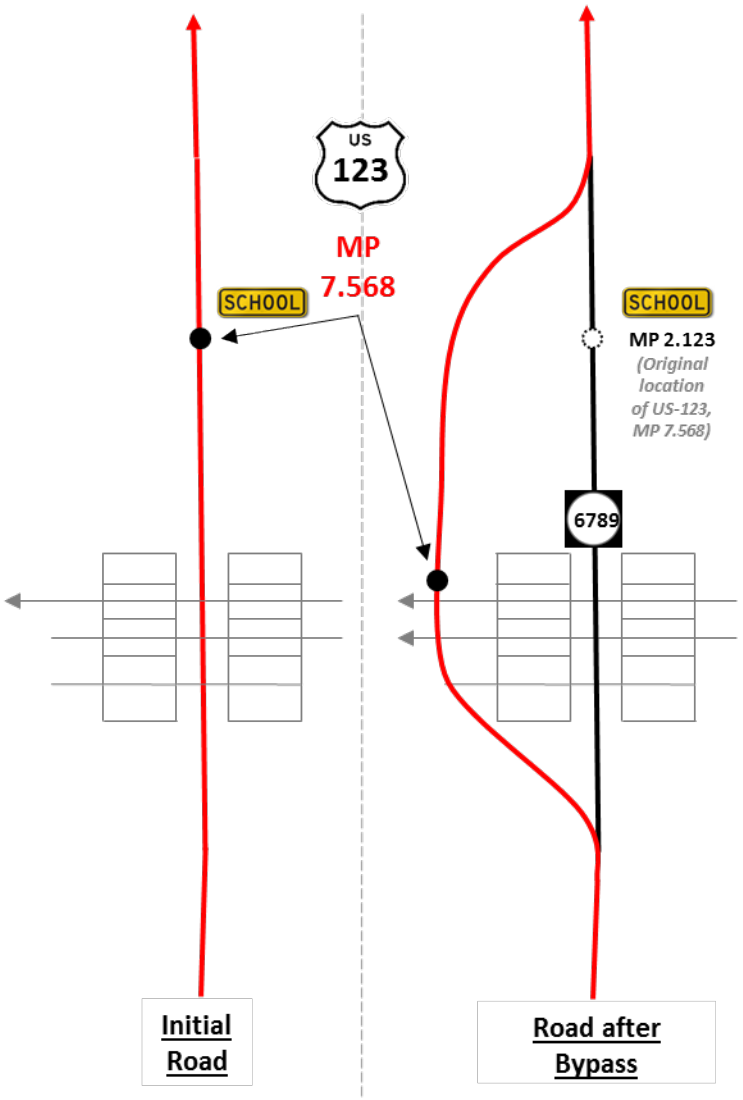



**Figure 1**: LRS Milepoint Jumps

In another example from Kentucky, surveyors, engineers, and other customers would come to the DOT office because they needed to find out information about a location, and the historical project as-builts held that valuable information. In the past, they would come to the DOT and search by the modern name of the road or project number through more than 200,000 project microfilm cards trying to find the data. These searches often took days or even weeks. Often the road names changed. A US highway in 1931 becomes a local road off the improved highway built in 1962. The project number’s meaning is lost to time as are the station and offsets, but the

reference map on the plan title page still connects the project details (parcel boundaries, owner names, etc.) to the Earth and has value to people today. By (1) building the spatial index of all these projects, (2) connecting it to scans of the microfilms, and (3) publishing an interactive map of them to the web, customers find what they are looking for in seconds or minutes rather than hours or days. This information is one of the most popular map pages for the Kentucky Transportation Cabinet. Accurate location is continuing to deliver ROI on projects built 75 or more years ago.

Lack of location accuracy also increases the risk of decision errors. This standard provides specifications to assure that individual assets are uniquely identified and located to assure an asset's information remains correctly associated. Figure 2 (below) shows two real-world examples where simple rounding errors of the latitude and longitude would result in the erroneous association of bridge data. The green labels show how information gathered about a state road bridge would flip to a US highway. The yellow labels provide an example of a county bridge being assigned to a US highway.


**Figure 2**: Rounding Error

One might argue that having the road name would prevent such events from occurring. However, road names, route designations, and milepoints change over time, as the road example (Figure 1) above highlighted. Real-world coordinates with the proper accuracy and precision assure long term that the information gathered for an asset stays associated with that asset.

Ignoring these problems is reminiscent of the following quote, "… right now, someone in your organization is about to make a poor decision based on data that you have paid enormous amounts to gather and assemble." (from CEB article[CIO3072112SYN-CEB]: *From Big Data to Better Decisions*, p.4). DOTs have invested millions to gather and analyze AASHTOWare-based information, and location is a key element of that data.

This standard also provides a recommended workflow to improve spatial data quality within AASHTOWare software. Figure 3 highlights a real-world example of Kentucky Transportation Cabinet project location data stored in AASHTOWare Project prior to manual cleanup. At the time the sample was gathered, there was not any validation on the latitude and longitude columns. Since users were not presented with the results of what they entered in a meaningful way to evaluate, they accidentally stored gross errors. PROJECT data shows Kentucky working in Georgia, the Arctic Ocean north of Greenland, and in western Chad.

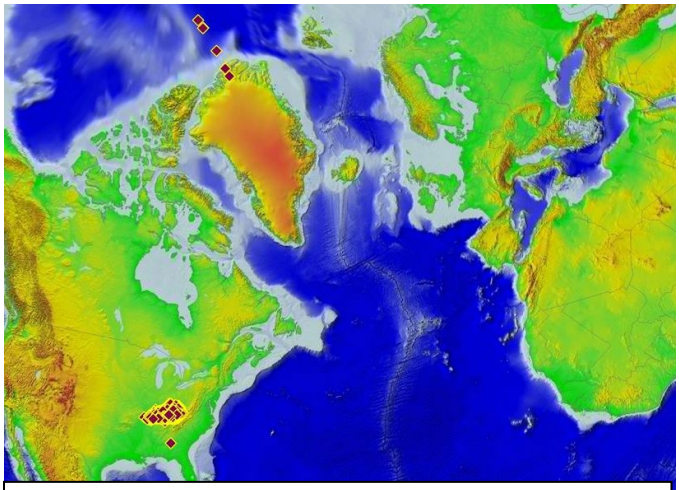


**Figure 3**: QA Testing via map visualization.

Section 4.1.1 of this standard covers the “search, place, and confirm” workflow to minimize these types of GIGO problems, so data quality is naturally improved, and customers can have confidence in their data-driven decision making.

Spatially enabling and opening an AASHTOWare product’s data for integration into the larger DOT data ecosystem increases each product’s value to its DOT customers. By making AASHTOWare data easily accessible to GIS software, the data can easily be merged with data from completely different systems that share no traditional database connections. Using location information, a DOT staff member can quickly evaluate a broad range of potential threats to the bridge program, all active road projects, etc.

In Example 4 (below), a scenario was posed: “FHWA has new guidelines that all bridges 100 feet or longer must address runoff impact to wetlands designated as wooded within 500 feet of the bridge.” The DOT’s bridge management team can leverage BrM data fused with environmental wetland data to define the answer quickly.

**Figure 4a: STEP 1-- Pull All Kentucky Bridges into GIS**
(Count = 18,422)

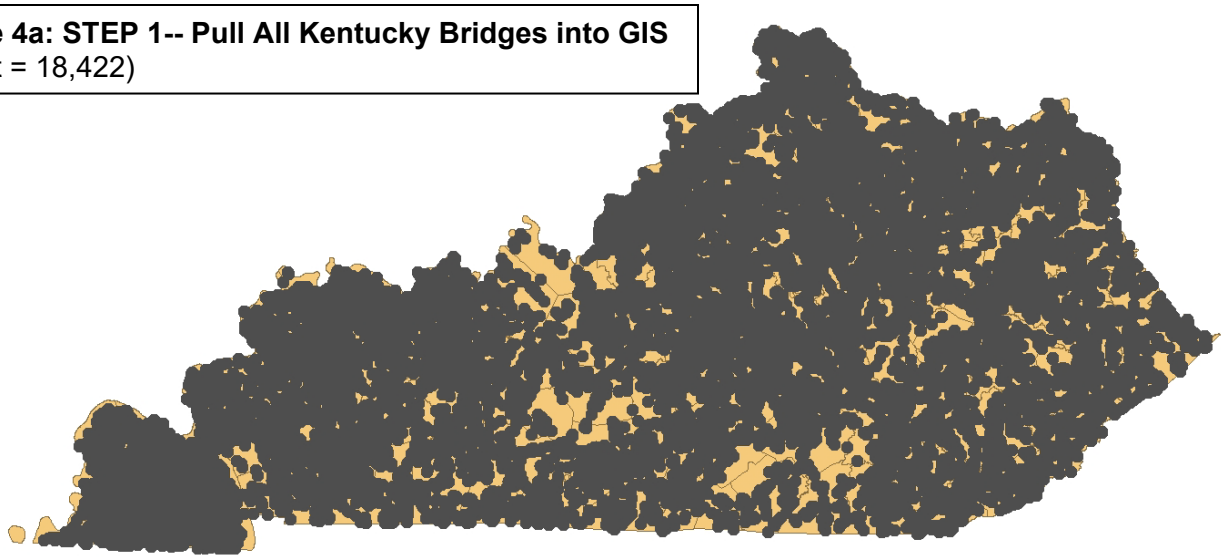


**Figure 4b: STEP 2— Filter Kentucky Bridges**
To >= 100 Ft.
(Count = 8,129)

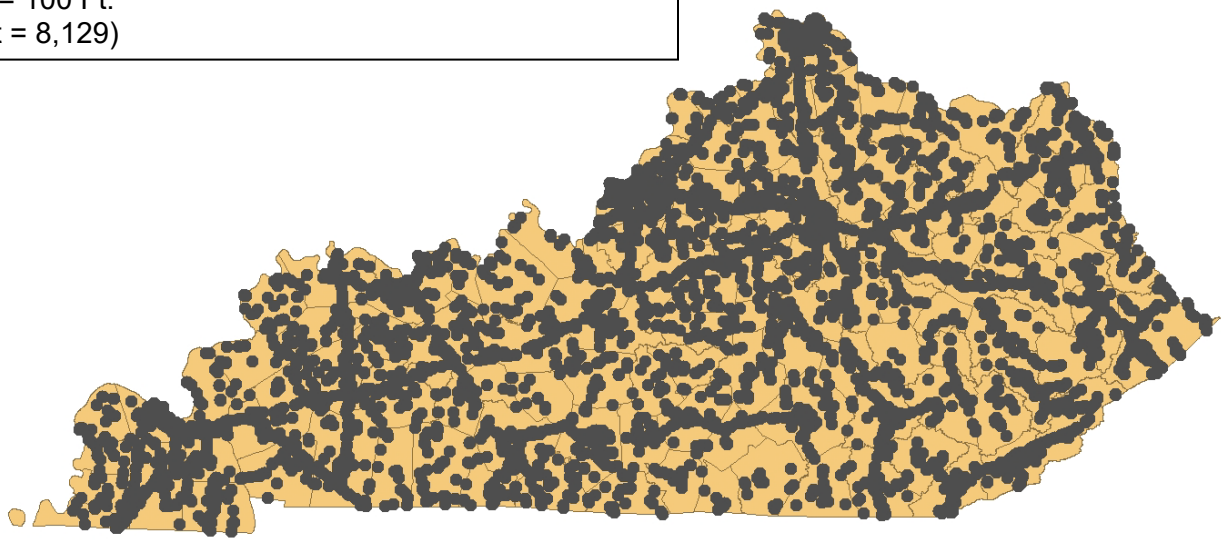
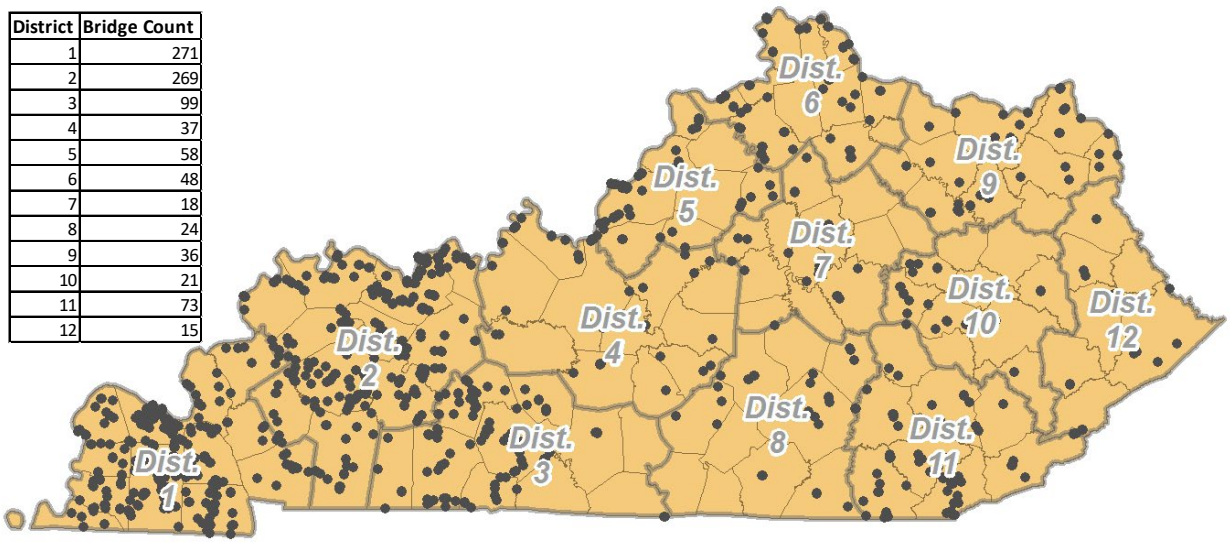


**Figure 4c: STEP 3— Spatially Select Kentucky Bridges**
To > 100 Ft. & within 500 Ft of Forested Wetlands
(Count = 969)

| District | Bridge Count |
|---|---|
| 1 | 271 |
| 2 | 269 |
| 3 | 99 |
| 4 | 37 |
| 5 | 58 |
| 6 | 48 |
| 7 | 18 |
| 8 | 24 |
| 9 | 36 |
| 10 | 21 |
| 11 | 73 |
| 12 | 15 |




The analysis can be performed across the *entire* database of bridges rather than researching each individual bridge. This saves weeks of staff time that would be consumed determining whether there was a correlation using traditional paper map methods. The step by step example above took 15 minutes since the data was already easily accessible. The spatial search results can be folded into traditional reporting the way customers are accustomed to reading. In this example, staff now know the scale of the problem in their respective districts by referencing the table and map together. It could easily be broken down to county summaries and bridge lists provided to allow action.

A recommended AASHTOWare software enhancement would be to allow ingesting results like this and flag records for special instructions to prompt the next inspector or whoever gets assigned to work that asset, guidance on actions to take using the system, rather than having independent processes outside the system. Again, the goal is to facilitate AASHTOWare being integrated into the dynamic flow of work (and data) within DOTs to assure that AASHTOWare is irreplaceable as it serves our customers' needs.

Part of “locating” includes describing the quality of the location description (metadata) to let users know how accurate the location description is. For example, in developing a billboard application, a general location accurate to 50 feet would be acceptable for many purposes, and depending on the particular DOT, this level of accuracy might be sufficient since visual sighting of a billboard should be easily achieved within 50 feet. The database table might store latitude/longitude. But for underground gas lines, the same DOT might require a sub-foot level of accuracy to avoid impacts since the risks and consequences if impacted are much higher. The Spatial Standard accommodates that flexibility while communicating the differences.

The technology environment has changed radically over the last decade. Systems today are built on a mobile and web-friendly, service-based architecture. Consuming and sharing data from and to external systems are now normal processes. Additionally, location-based analysis and reporting are normal elements within applications DOTs use regularly, as well as individuals in their private lives. The Spatial Standard helps task forces prepare applications and data for the needs embedded within the DOTs today as well as future needs yet to be identified. Those needs often are not the direct user of the AASHTOWare application screens, but DOTs are compelled to blend data from many different sources to answer today’s pressing problems. By opening the applications’ information up to a larger customer base, DOTs realize a much higher value for their investment in AASHTOWare.

**What this standard does not do**.

*This standard does not force DOTs to be spatially enabled or aware*. Customers are able not to use spatial functionality if they choose. But just as some members of DOTs thought computers were a fad, spatial analysis is too powerful and the need too great to be ignored long term. Over the last 20 plus years, DOTs have invested significantly in GIS to solve real-world problems, and all 50 DOTs have GIS embedded in their agency processes. AASHTOWare must position itself for the norm, not the anomaly.

*This standard does not require all task forces to uniquely spatially locate every element, nut, bolt, wire, etc. stored within AASHTOWare applications*. Two key objectives of this standard are to accurately locate assets stored within the applications so that the information can be consistently accessed long term regardless of the name or LRS changes that occur over time and increase the value of AASHTOWare by simplifying the integration of high-value data into DOT analyses to solve real-world problems.

During the Data Integration Summit held in January 2019, DOT members consistently and clearly communicated the need to locate assets using both LRS and real-world location information. Of the 28 respondents to a survey at the end of the summit, 26 said both LRS and “GIS” locations were needed. The remaining two said GIS only. This is a significant shift from traditional approaches within the software and will take time to implement, but the customers are clear on their needs.

*This standard applies to new development projects and work efforts (projects and MSE work) that make major changes to an existing product’s data storage repository. All requirements for compliance with this standard are shown in red italicized text.* ***The new requirements that were not in the previous version of the Standards and Guidelines are shown in red bold italicized text.***

## 2. Task Force/Contractor Responsibilities

The product task force and contractor responsibilities for this are summarized below:

- *The contractor shall ensure that the AASHTOWare spatial standard is supported in all development of new products and development efforts that include the establishment of new data structures or mobile solutions. Refer to the Technical Requirements section in this standard*
- The contractor and task force should consider supporting the spatial standard when developing enhancements that make major changes to an existing product's data structure or the addition of a mobile solution.
- *The contractor and task force shall evaluate their existing data structures and assure that those spatial elements (like latitude/longitude [lat/long]) currently stored are being collected and stored using methods that assure the data quality that provides the intended value to the user.* For example, storing lat/long without performing simple validation checks allows the user to falsely assume spatial accuracy when data entry errors could be recording coordinates on the wrong side of the Earth. Remember GIGO.
- *If location data (like lat/long) are being stored without proper validation checks embedded in the software, the developer and task force shall develop alternative guidance workflows to help users test, validate, and correct their spatial data.*
- The task force should routinely survey the current and potential user base to determine what interest there is for mobile and location-based solutions.
- The contractor and task force should participate in research and testing of new location-based data structures.
- The contractor and task force should recommend/provide feedback on location-based services that support specific products.
- The task force should collaborate with the T&AA Task Force liaison whenever new mobile applications or location-based reporting are discussed.
- *If spatial data is being embedded in the database and application, the contractor shall store the data in one of the supported DBMS (Oracle or SQL Server). For point data, simple x and y fields can be stored in traditional tables. For lines and polygons, a spatial repository with spatial data types will need to be developed. Both Oracle and SQL Server have implementations that support the Open Geospatial Consortium's Simple Feature Access structure. Data for mapping displays shall be delivered via platform-independent Open GIS Web Service formats (WMS or WFS). Other semi-standard spatial formats (GeoJson, REST, etc.), and the third party vendor formats listed in Appendix A may be offered as secondary alternates should the customer prefer it.*
- *The contractor shall research and document any applicable licensing terms and conditions, then provide recommendations to avoid potential issues to the task force and AASHTO Project Manager to ensure compliance with all license requirements.*
- *The contractor shall provide metadata at the table and record level as appropriate; see details in Section 5.3.*

# 3. Required Deliverables and Work Products

The following summarizes the required deliverables and work products that shall be created and/or delivered in order to comply with this standard.

- Table of Spatial Feature Classes*: As part of the product documentation, a table shall be provided describing the feature classes (e.g., assets, bridges, projects, etc.) that resides within an application's database that has been spatially enabled. Each record in the table shall include the feature class name, its spatial representation (point, line, polygon, multi-part, and raster), description of what the feature class is, and projection (See example below). This is intended as a high-level view of objects. Additional metadata details about these feature classes would be described in their attached metadata.*

| Feature Class (Table) Name | Spatial Representation | Description | Projection |
|---|---|---|---|
| State Bridges | Point | Point location of all bridges within the state. | Decimal Degrees (WGS84: EPSG 4326) |
| Project Footprints | Polygon | Project Area for all state projects active in the next two years. | UTM, Zone 16, NAD83, Meters. (EPSG:26916) |

- *Metadata properly attributed shall be attached to data tables/records (see Section 5.3).*

# 4. Procedures

## 4.1 Modification of Spatial Data Entry

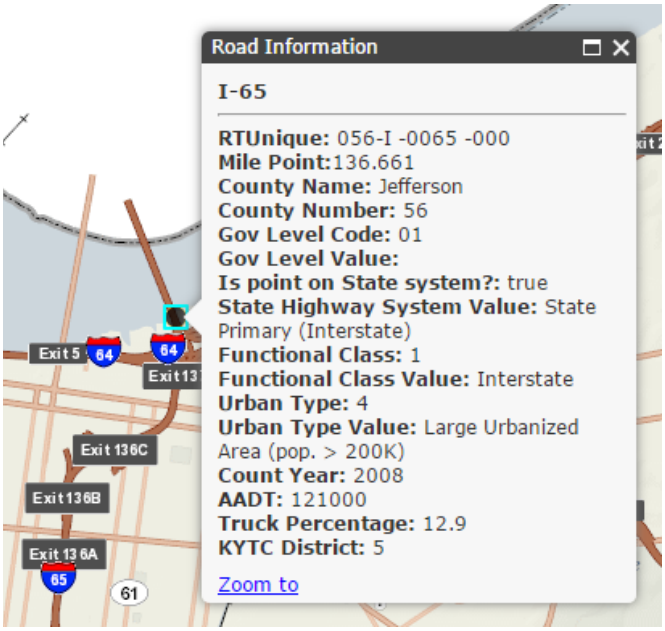
Correctly locating information is a critical foundation for integrating that information with other systems. Often applications allow users to enter tabular reference location information (what city/county, milepoint, etc.) even as the user is also allowed to put a "pin on the map." This creates opportunities to have erroneous and conflicting information. When these errors are discovered later, it causes integrity concerns since there is ambiguity about which location information is correct. This data integrity issue can have a snowball effect reducing confidence in other data within the database and increase the users' maintenance efforts.

### 4.1.1 Recommended Solution: Refine the User Workflow

- *The task force shall define, and the developer implement, an application workflow that guides the user through a "search, place, and confirm" process. The workflow shall not allow the direct input of location reference information (e.g., inside city limit, linear reference measure [county-route-milepoint], etc.) into the data table.* The steps are:
    - Provide search tools for the user to find their location of interest similar to Google or Bing.
    - Display the location (map and LRS/lat/long, etc.) and prompt for user confirmation or placement selection.
    - Store locations in real-world coordinates into appropriate location fields in the correct table(s). Storing locations by linear referencing or other systems that can change over time creates on-going maintenance concerns.

- Submit the location to web-based services that request the correct information from authoritative GIS data sources (sourcing will vary from agency to agency) to return the spatial reference information needed. For example, using the stored lat/long, an application can leverage web services to return, milepoint, county, city, state, AADT, functional class, etc. See Figure 5 below.

**Figure 5: Example of a user's map-click capturing real-world location and that location via web services returning supplementary information.**




- Only store spatial reference information (non-real-world coordinates) in data tables when there are application performance issues.
- *Developers shall use web services to return spatial reference information as needed and display reference information as the customer desires it.*
- When the user desires to change the location, use the same search/confirm method described above to update location information.

- When the user only wants to record non-specific location (e.g., city, county, etc.), *developers shall use the same search by, then verify on map workflow as above to capture and write to the most precise field and populate other required spatial reference information.* For example, if a user chooses "Nashville," the county would automatically populate and not allow the user to edit it, and would display the county, "Davidson," and the state, "Tennessee."

# 5. Technical Requirements

## 5.1 Define Coordinate System for all Spatial Feature Classes

*All spatially enabled data shall have a coordinate system defined*. Coordinate systems are the "fabric" that define how a location on the Earth's surface is described. This allows features and feature classes to be displayed in relation to one another. Modern software allows feature classes to be displayed together even if they are described with different coordinate projections, but they must be properly defined. Tabular (non-spatial) data stored

with linear reference attributes may not have a direct projection defined, but when it is processed as an "event" in GIS, it is draped into the base network's projection.

*The Spatial Reference System Identifier (SRID) shall be used to correctly describe the projection of the data.* SRIDs can be found here: *https://spatialreference.org/*. The International Association of Oil & Gas Producers *(IOGP)* inherited the maintenance of these from the European Petroleum Survey Group (EPSG). The SRIDs are commonly referenced to assure standard projection descriptions. Modern GIS software and relational database management systems (RDBMs) have tools embedded to define and manage spatial data including projections.

### 5.1.1 Preferred Coordinate System: Latitude/Longitude Decimal Degrees (WGS84 [EPSG 4326])

GPS units commonly provide coordinates in the 1984 World Geodetic System (WGS84) as a default. Modern mapping packages can re-project and display most projections on the fly. Therefore, for general mapping (non-survey grade) purposes, the key is to correctly define the coordinate system the data is captured and stored in rather than focus on what it is stored in, but having a standard default tied to the base technology default reduces the possibility of error.

A detailed definition of the projection can be found *here.*

```
GEODCRS["WGS 84",
  DATUM["World Geodetic System 1984",
    ELLIPSOID["WGS 84",6378137,298.257223563,LENGTHUNIT["metre",1.0]]],
  CS[ellipsoidal,2],
    AXIS["latitude",north,ORDER[1]],
    AXIS["longitude",east,ORDER[2]],
    ANGLEUNIT["degree",0.01745329252],
  ID["EPSG",4326]]
```

*A consistent single projection definition shall be used for each table/feature class.* While there have been many refinements with new epochs (datum adjustments) published since its initial definition and any of these may be used, for non-survey purposes, the original definition is the preferred.

### 5.1.2 Alternate Coordinate System: Latitude/Longitude Decimal Degrees (NAD83-1986 [EPSG 4269])

Many GIS applications use NAD83- 1986 coordinates as their default of decimal degree data. Again, modern mapping packages can re-project and display most projections on the fly. This projection provides a reasonable alternative that proficient GIS practitioners can easily integrate with other data. The difference between the two coordinate systems is minor for mapping grade applications (about four feet), but data coordinate systems should not be "mixed and matched" within a single feature class.

A detailed definition of the projection can be found *here.*

The well-known text version of the definition provided by IOGP's EPSG is:

```
GEODCRS["NAD83",
  DATUM["North American Datum 1983",
    ELLIPSOID["GRS
1980",6378137,298.257222101,LENGTHUNIT["metre",1.0]]],
  CS[ellipsoidal,2],
```

AXIS["latitude",north,ORDER[1]],
AXIS["longitude",east,ORDER[2]],
ANGLEUNIT["degree",0.01745329252],
ID["EPSG",4269]]

*A consistent single projection definition shall be used for each table/feature class.* While there have been many refinements with new epochs (datum adjustments) published since its initial definition and any of these may be used, for non-survey purposes, the original definition is the preferred.

### 5.1.3 Alternate Coordinate System: Web Mercator

This is not a preferred alternate for data capture or more complex analysis. Additional steps must be taken if distance or area measurements are used in an application that uses this projection (see *ESRI's Blog post*). It is provided here as a reference since Google Maps, Bing Maps, and ESRI maps, among others, use it for web mapping.

A detailed definition of the projection can be found *here.*

The well-known text version of the definition provided by IOGP's EPSG is:

- WGS_1984_Web_Mercator_Auxiliary_Sphere
  - WKID: 3857 Authority: EPSG
  - SRID/EPSG: 3857
  - ESRI WKT
    - PROJCS["WGS_1984_Web_Mercator_Auxiliary_Sphere",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Mercator_Auxiliary_Sphere"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",0.0],PARAMETER["Standard_Parallel_1",0.0],PARAMETER["Auxiliary_Sphere_Type",0.0],UNIT["Meter",1.0]]

### 5.1.4 Alternate Projection: Local Projection

State transportation agencies and many state GIS entities use a state plane or other local projection as their default coordinate system for storing location coordinates. While this may require more processing to integrate with other external data in a larger geographic integration, it should be accommodated if that is a local business requirement.

AASHTOWare contractors should assure that the local projection is properly defined as described, including the SRID by the EPSG.

### 5.1.5 Linear Referenced Tabular Information

It is very common for transportation agencies to store location information using linear referencing (route and milepoint[s]). Prior to the widespread adoption of GPS and GIS technology, this was probably the most common means of tracking and describing the location of assets within DOTs. With advances in focused GIS-based transportation

solutions, the ability to maintain accurate location descriptions with linear referencing has improved dramatically.

However, it has also exposed gross errors between the different information systems storing LRS based information. The limitation is that *any* information not integrated into these large scale LRS maintenance systems is prone to synchronization errors or "location dissonance." Every DOT's road network and the assets that make them up are changing daily. Bypasses are added, and existing route names/numbers are reassigned to it. New route names/numbers are assigned to old, existing pavements. Curves are straightened to improve safety. Bridges are replaced. If one does not adjust the linear reference measures of the network assets as in the bypass example above (Figure 1), then you get assets assigned via LRS to new road sections that can be miles from the actual real-world asset location. For the curve straightening example, "bunching errors" occur where assets shift closer together or stretch farther apart if a road is extended depending on the modification of the network. If one does adjust the network but does not make synchronization adjustments to the other LR data within the DOT, the data, when plotted, will "jump" to incorrect locations. These errors create false correlations between different sources' data and hide meaningful data clusters. The result is poor decisions made on bad information.

*To protect data integrity and the decisions made with that information, disconnected LRS-referenced data must store not only the LRS information but also the LRS metadata such as the capture time stamp and the source.* This information, if consistently sourced, could be stored at the table level or, if dynamic, can be stored at the record level. See the table below.

| Field Name | Type | Length-Precision-Scale | Description |
|---|---|---|---|
| **LRSType** | **Text (VARCHAR2)** | **255** | Commonly one of three types: (1) Route & Distance measure along centerline, (with optional Offset), (2) Intersection & Offset along route, (3) Station and Offset along centerline. |
| **LRSSource** | **Text (VARCHAR2)** | **255** | The source system name of the LRS or web service used to pull the LRS values. |
| **LRSCreatedDate** | **Date/Time** | **8** | Captures date/time of when the LRS source was pulled. This should be populated by the DB or by a service leveraging those. Note that the timestamp of the true LRS creation may be significantly different than the record creation date stamp. Some DOTs only publish LRS data annually or semi-annually. This date stamp is the best strategy for documenting which published LRS was used in case of a need to reconcile discrepancies. |
| **LRSLastUpdateDate** | **Date/Time** | **8** | Captures the last date/time when a record's LRS was modified. |
| **LRSComments*** | **Memo (VARCHAR2)** | **1024** | This filed captures any other information regarding LRS that would not properly fit within an existing field. |

Ultimately, it is up to the individual DOTs to implement the maintenance processes to maintain synchronization of LRS information, but AASHTOWare must be capable of facilitating those maintenance processes. Capturing the LRS metadata greatly improves the quality of integration and analysis efforts. LRS-based location analyses should adjust all the different LRS/time combinations to a common LRS-time. This is often ignored, but the shifting LRS causes false correlations, and missing patterns will be delivered to the customer to make bad decisions. Location and temporal integrity are important for truly leveraging the value of the data.

Linear referenced information will never go away, but there are inherent challenges keeping information in sync with a constantly changing network external to the master LR repository. AASHTOWare is not attempting to be the master repository for linear referenced data. There are several existing, firmly established vendors. The recommended method is to store real-world coordinates and then use spatial tools to "snap" to the network to pull and report the linear description. This assures that the objects being tracked stay where they actually are, provides an LR description that

DOTs are used to and comfortable working with, and assures a more simplified and accurate integration with other DOT data.

#### 5.1.6 Vertical Projections

*For all North American data, any elevations (z) recorded shall use the original NAVD 88 or one of its epochs. NAVD 88 is the official vertical datum in the National Spatial Reference System (NSRS). It is "the official civilian vertical datum for surveying and mapping activities in the United States for the United States performed or financed by the Federal Government." See the Federal Register and NOAA's page on datums.*

*A single vertical datum or epoch shall be used per table or feature class. They shall not be mixed within a table or feature class.*

#### 5.1.7 Units of Measure

Both US and metric units of measure are acceptable for area and length measurements. Consistency and conformity to local requirements is the key with appropriate metadata documentation.

### 5.2 Define the spatial accuracy either at the feature class or record level.

The evolution of mapping and data sourcing integration has resulted in a wide variety of data sets being "mashed together" in most modern GIS environments. Today, an agency's data is an "ecosystem" with inputs flowing in, through, and out of discrete systems and into other systems rather than remaining in a single isolated application/database.

This means that spatially-aware field data and other methods and workflows not imagined 20 years ago have emerged. Customers want to blend this dynamic and variable information with data captured and held in traditional workflows as well. This evolution of data opportunity and customer expectations over the past decade inserts much more variability within the data held within AASHTOWare applications.

*To correctly serve as the authoritative repository for transportation information, AASHTO software shall properly capture, store, and report how data presented in its software is gathered, and its accuracy in order to allow customers to make informed decisions. Spatial accuracy is a critical part of that metadata.*

#### 5.2.1 Survey Grade Accuracy (under 2 inches)

*All capture of location information that is required for survey work or for use in engineering work shall be under the guidance of professional engineers/surveyors and follow the local and state requirements to meet those standards and specifications for accuracy. If information stored in AASHTOWare is going to be reused with survey-grade accuracy, then the storage and display of information shall preserve and present location data at the same original quality. Metadata confirming the accuracy shall also be attached and provided to the data consumer.*

#### 5.2.2 Mapping/Resource Grade Accuracy (>2 inches [0.0508 Meters])

"Mapping grade" accuracy provides a general-purpose location of a feature, feature class, or boundary that is useful for the creation of general maps for conveying information about objects and their general location and their relationships to one another. It is not designed to be used for establishing property rights, or act as the

authoritative location description in a legal proceeding. It can be used for illustrative purposes.

Field work is a critical part of transportation work processes. As applications have moved from the office to the field, location-based workflows and the underlying accuracy of available GPS-based technology must be understood.

A GPS-enabled smartphone is typically accurate to within three to five meters, or up to about 16 feet. In 2017, a GPS chip became available to enable mobile devices to improve accuracy to within 30 centimeters, or under one foot. The key is to be aware that accuracy will continue to increase with common hand-held devices.

#### 5.2.2.1 Spatial Accuracy Guidance

Application developers must keep in mind that weather, sun activity, blocked satellites due to line of sight limitations, multi-path, etc. decrease the locational accuracy delivered by these devices. While these units will continue to provide a "precise" location (the same number of decimal places as before), the accuracy of the location can vary greatly. This can have a significant impact on the quality of data, including placing features on the wrong side of a road or outside of a project boundary.

Awareness and intelligent UI design are needed. Designers must be very clear on how and what they allow users to capture. *Designers shall properly capture and communicate location quality within their applications to help users make informed decisions as they use location in their analysis.*

#### 5.2.2.2 Feature class accuracy statement

*Designers shall guide users to follow accuracy reporting as described by the Federal Geographic Data Committee's Accuracy Standard (FGDC, Pg 3-5&6, 3.2.3).* Of particular note, the recommendation to use the "compiled to meet..." statement is very relevant to modern mobile development when multiple devices, with varying user experiences, are sourcing the location.

The goal is to give the data consumer an understanding of the quality of the information (location) that they are using to make a decision. A careful description of the methodology and tested accuracy will greatly improve the value of the information. If no documentation is provided, the consumer will fill in the blanks, often to their own detriment.

#### 5.2.2.3 Record level accuracy capture

With highly variable means of location capture within a single feature class or table, documenting accuracy at the record level becomes important to help data consumers evaluate the different data elements they are using. It also protects against degrading all records to the least common denominator. See below.

### 5.3 Properly Document and Attach Metadata to Each Feature Class

Modern GIS tools, as well as relational databases, provide the means to capture "data about the data." "Metadata," the descriptive information about "who, what, when, how, and why" of the data is important for the correct use of the data. This supporting information becomes even more important as the data gets consumed by people/groups other than the original creators. In modern technology where "big data" and mashups are the norm, it is critical that

as data travels, descriptive information about its origin, quality, and utility travel with it to help the data consumer to properly understand and use the information.

### 5.3.1 General Feature Class (Table level) Metadata

*Designers shall provide interfaces to allow users to complete metadata attached/linked to the appropriate feature class/database table*. This should include the following:

- A full description of the table's contents (Abstract) and a brief description (Summary) are required. These should capture the what, why (Purpose), when, where, and who.
- Use complete sentences in sentence case.
- Write out the full phrase that an acronym represents and then follow it with the acronym in parentheses, for example, "the Office of Information Technology" (OIT).
- Specify purpose, source, and time range validity of data in the Abstract.
- Provide a list of key words that will assist search functions to quickly surface data during searches.

The Federal Geographic Data Committee (FGDC) provides additional guidance *https://www.fgdc.gov/metadata*.

### 5.3.2 Specific Feature Class Record Level Metadata

As noted above, record-level metadata provides the data consumer critical information to allow them to make better decisions. Below are the fields needed to capture record level location metadata.

| Field Name | Type | Length-Precision-Scale | Description |
|---|---|---|---|
| LOCMethod | Text (VARCHAR2) | 2 | • F1 – Field Survey Grade GPS<br>• F2 – Field Mapping Grade GPS<br>• F3 – Field Map notation – using a map in the field and selecting a feature location based on other referenced information.<br>• O1 – Office Heads up digitizing from Maps/Imagery in a GIS environment.<br>• O2 – Office location recording from web base mapping source (e.g., Google/Bing Maps)<br>OO – Office Other – Location recorded from some other source/method. |
| LOCQuality | Text (VARCHAR2) | 2 | • A – 3D Survey Grade<br>• B – 2D Survey Grade |

| Field Name | Type | Length-Precision-Scale | Description |
|---|---|---|---|
| | | | • C – Sub Meter 2D Grade<br>• D – 1-5 Meter 2D Grade<br>• E – 6-10 Meter 2D Grade<br>• F – 10-20 Meter 2D Grade<br>• G – 20+ Meter 2D Grade<br>• U -- Unknown Quality |
| LOCIssues* | Memo (VARCHAR2) | 1024 | Memo field to capture an issue about the location that needs to be addressed and cannot be by the person performing the data entry. This can then be addressed at a later date but allows for quickly filtering data to focus and correct problems. |
| LOCCreatedDate | Date/Time | 8 | Captures date/time at the point of record location creation. This should be populated by the OS or DB or by a service leveraging those. |
| LOCCreatedBy | Text (VARCHAR2) | 70 | Person that created the record location. This should be populated by the OS or DB authenticated username or by a service leveraging those. |
| LOCLastUpdateDate | Date/Time | 8 | Captures date/time at last point of record location modification. |
| LOCLastUpdateBy | Text (VARCHAR2) | 70 | Person that last updated the record. This should be populated by the OS or DB authenticated username. |
| EndActiveDate* | Date/Time | 8 | Captures date/time when the record was retired. This allows history to be stored and temporal analysis. |
| Comments* | Memo (VARCHAR2) | 1024 | This is to capture any other non-location based ideas that would not properly fit within an existing field. |

Note: Fields that can be populated optionally are marked with an asterisk (*). LOCQuality – there are many sources for determining the spatial accuracy of the tools, maps, and methods employed. The task force shall develop specific guidance to assist users in properly capturing and recording location information in their respective application.

# 6. Deliverable and Work Product Definitions

## 6.1 Application Adjusted to Leverage Location

### 6.1.1 Description

The first deliverable is an application designed to properly capture, store, and use the spatial data within the application.

### 6.1.2 Content

*The application shall leverage spatial information as well as tabular data and associated metadata. It shall allow users to interact with spatial and tabular data to make more informed business decisions.*

## 6.2 Spatially Enabled Database

### 6.2.1 Description

The second deliverable is a database designed to capture, store, and deliver spatial and tabular data as needed to applications and web services.

### 6.2.2 Content

*The database and its tables and/or feature classes shall maintain spatial data at the appropriate accuracy for the business purposes required. The database and tables shall also capture and maintain metadata at the table and record level as appropriate.*

## 6.3 Spatially Enabled REST Web Services

### 6.3.1 Description

The application should provide secured RESTful web services to allow the proper connection of the application to other external data sources and applications to allow for larger integration to the enterprise. See *Web Application Development Guideline and Architecture Goals, Section 14* for a more detailed discussion of REST services.

### 6.3.2 Content

Depending on customer needs, the web services may provide a read-only access to the data, thereby allowing external integration with other systems, including the enterprise's GIS.

It may also be structured to allow 3rd party data submission that should have validation checks prior to integration into the final data tables to assure proper adherence to business & data rules.

Any spatial web services should be offered in an Open Geospatial Consortium compliant format. A proprietary format compatible with a major GIS software vendor can be available as an optional service.

Mobile applications should have a configuration to define device accuracy to record the LOCMethod and LOCQuality information.

# 7. Appendix A

Appendix A provides a series of links to external reference information relevant to the spatial standards discussed above.

## 7.1 Standards

### 7.1.1 Federal Geographic Data Committee Standards

See FGDC *https://www.fgdc.gov/*

Federal Geographic Data Committee's Accuracy Standard *(FGDC, Pg 3-5&6, 3.2.3).*

Metadata Standard https://www.fgdc.gov/metadata

### 7.1.2 Open Geographic Data Committee Standards

7.1.2.1 WFS Standard: *https://www.ogc.org/standard/wfs/*

7.1.2.2 WMS Specification: *https://www.ogc.org/standard/wms/*

## 7.2 Terms

ESRI's GIS Glossary: *http://wiki.gis.com/wiki/index.php/GIS_Glossary*

ESRI's GIS Dictionary: *https://support.esri.com/en-us/gis-dictionary*

GISGeography.Com's Dictionary: *https://gisgeography.com/gis-dictionary-definition-glossary/*

## 7.3 GIS Fundamentals

ESRI's "What is GIS?" *https://www.esri.com/en-us/what-is-gis/overview*

GIS Lounge—GIS Essentials *https://www.gislounge.com/gis-essentials/*

## 7.4 Major GIS Vendors

AutoDesk -- *https://www.autodesk.com/products/autocad/overview*

Bentley -- *http://www.bentley.com/*

ESRI – *https://www.esri.com/en-us/home*

Trimble – *https://www.trimble.com/*

Hexagon (formerly Intergraph) – *https://hexagon.com/*

## 7.5 Open Source Web GIS

MapServer -- *https://www.mapserver.org/*

GeoServer -- *https://geoserver.org/*

This page is intentionally blank.

AASHTO AASHTOWare™ SOLUTIONS

# PRODUCT NAMING CONVENTIONS STANDARD

## S&G Number: 2.060.05.4S

## Effective Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 4.0 | 3/13/2011 | Changed name from Glossary of Product Terminology. Updated Turbo Relocation & DARWIN-ME information. | 6/06/2011 Approved by SCOA |
| 4.1 | 6/08/2011 | Corrected “Version” in component parts of Release Number to “Release”. Corrected spelling. | 6/30/2011 Approved by T&AA |
| 5.0 | 4/01/2012 | Added information on AASHTOWare Rebranding and updated logos and icons. | 7/01/2012 Approved by T&AA Chair for T&AA |
| 5.1 | 6/27/2013 | Changed S&G number from 3.060.05.0S to 2.060.05.1S. Made minor changes and corrections. Updated splash screen and about screen examples. | 7/01/2013 Approved by T&AA Chair for T&AA |
| 5.2 | 6/04/2014 | Changed information regarding AASHTOWare Branding and made minor corrections. | 7/01/2014 Approved by SCOA |
| 5.3 | 5/02/2018 | Updated the AASHTOWare Product Identification section and made minor grammatical edits. | 6/22/2017 Approved by SCOA |
| 5.4 | 8/22/2023 | Updated the AASHTOWare and product logos. | 10/02/2023 Approved by SCOA |

# Table of Contents

# 1. Introduction

The Product Naming Conventions Standard was established to assist AASHTOWare contractors and users in proper use of the AASHTOWare terminology for product nomenclature and identification. AASHTO reserves the right to change this standard at any time at its discretion. *The AASHTOWare contractors shall comply with this standard as amended from time to time.*

The Product Naming Conventions Standard provides a source for consistent and correct usage for terms and graphics that are specific to the AASHTOWare products. *This standard is applicable to all AASHTOWare documentation and packaging describing the AASHTOWare products and services. The requirements for compliance with this standard are shown in red italicized text.*

To comply with the AASHTOWare Product Naming Conventions standard it is important to understand and differentiate the usage of the term AASHTO and AASHTOWare.

## 1.1. AASHTO

The term AASHTO is the acronym for American Association of State Highway and Transportation Officials and is a registered trademark of the American Association of State Highway and Transportation Officials, Inc.

## 1.2. AASHTOWare

The term AASHTOWare is a registered trademark and service mark of AASHTO. It collectively represents all intellectual property including computer software products resulting from the AASHTO Cooperative Software Development Program.

# 2. AASHTOWare Rebranding

During FY 2023, AASHTO initiated a rebranding project to establish consistency among AASHTO programs, services, and products. This effort affected the AASHTOWare program and AASHTOWare products.

The updated logos are shown below in the “AASHTOWare Product Identification” section. Rules governing how to display the logos are available here.

# 3. AASHTOWare Product Nomenclature

The AASHTOWare product nomenclature provides definitions of terms specific to the AASHTOWare environment for uniform naming of the AASHTOWare products. AASHTOWare product names based on this nomenclature are generally submitted to the United States Patent and Trademark Office to obtain official trademark registration.

## 3.1. Owner Name

This term represents the name of the legal owner of the AASHTOWare products. An AASHTOWare product may include intellectual property or components legally licensed by AASHTO for distribution. AASHTO is the designated Owner Name for all AASHTOWare products. AASHTOWare is the designated Master Brand of the software products.

## 3.2. Family Name

This term designates a group of AASHTOWare products designed for a specific transportation-engineering domain. The use of Family Name for AASHTOWare product naming is optional. Project and Bridge are examples of the existing AASHTOWare Family Names. The Family Name corresponds to the AASHTOWare Software Brand.

## 3.3. Product Name

This term designates an AASHTOWare product that provides information and functionality for an identifiable or definable segment within a transportation-engineering domain. ME

Design, Analyst, and Preconstruction are examples of some of the existing AASHTOWare Product Names

### 3.4. Module Name (Optional)

The term Module Name designates a portion of an AASHTOWare product that can operate independently but is usually data compatible with the other portions or modules of the product. The use of Module Name for AASHTOWare product naming is optional. Superstructure and Substructure are examples of two modules for the AASHTOWare Bridge Design product.

### 3.5. Version Name

This term Version Name designates the technical architecture of an AASHTOWare family name, product name, or module name. Web-based Project Preconstruction is an example of a version name for the Project Preconstruction web-based product. Client Server SiteManager and Web SiteManager, are not currently used, but represent possible version names for an AASHTOWare product.

### 3.6. Release Number

The Release Number represents a specific designation for each compiled component of an AASHTOWare Product. The Release Number is composed of four distinct numerical terms separated by decimal points (MAJOR.MINOR.MAINT.BUILD) specifying the chronological order of the software productions. The AASHTOWare contractor should maintain a written record of Release Number with description of software changes associated with each release. *A complete Release Number shall appear on the About Dialog Box for product identity in each AASHTOWare product. The Release Number can be truncated to the first two terms for display in the AASHTOWare product splash screen and other documentation.*

Each component of the four part Release Number (MAJOR.MINOR.MAINT.BUILD) is described below.

#### 3.6.1. Major Release Number

The first term designates the major revisions to the AASHTOWare product, which usually include major functional additions and enhancements. AASHTOWare Task Force approval is required to update this term.

#### 3.6.2. Minor Release Number

The second term designates minor changes to the AASHTOWare product such as minor functional additions, improved performance, and improved user interface. AASHTOWare Task Force approval is required to update this term.

#### 3.6.3. Maintenance Release Number

The third term designates maintenance updates to the AASHTOWare product resulting from software malfunction correction. The AASHTOWare contractor can update this term with every software maintenance release.

#### 3.6.4. Build Release Number

The fourth term designates incremental software build indicator. The AASHTOWare contractor should update this term with every build of the AASHTOWare product.

### 3.7. Platform Name (Optional)

The term Platform Name is an optional naming convention that designates a technology platform for the AASHTOWare product. The technology platform includes the operating system and any other operating environment software necessary for designed technical use of the AASHTOWare product. The AASHTOWare product naming convention requires the use of the word "for" before the Platform Name.

The syntax for using Platform is Name Owner Name [Family Name] Product Name [Module Name] Release Number for Platform Name, with the brackets representing optional

components. Examples of possible Platform Names are listed below (note these are not currently used):

- AASHTOWare Project Preconstruction 3.00 for Microsoft Windows 8.1
- AASHTOWare Bridge Design 6.6 for Microsoft Windows 8.1 and DB2

## 4. AASHTOWare Product Identification

*AASHTOWare product identification using appropriate graphic elements is required.* This section provides information of different types of graphic elements recognized for AASHTOWare product identification.

*Products identified in the AASHTOWare catalog as having a registered trademark must use logos that include the ® symbol. Products identified in the AASHTOWare catalog as having a trademark must use logos that include the ™ symbol.*

*Regarding the AASHTOWare product logos, alteration of color and aspect ratio is not allowed.* Refer to the AASHTO Logo Guidelines Update for additional details on the correct usage of the logos and icons.

### 4.1. AASHTOWare Main Brand Logo

The AASHTOWare main brand logo is a trademark and service mark of AASHTO. This logo should be used to identify a product as an AASHTOWare product.


*AASHTOWare Main Brand Logo*

### 4.2. AASHTOWare Family Brand Logo

The AASHTOWare family brand logos should be used to identify a product as a member of an AASHTOWare family of products.



*Sample AASHTOWare Family Brand Logos*

### 4.3. AASHTOWare Product Brand Logo

AASHTOWare product brand logos are the most visible form of product identification elements. The product brand logos within an AASHTOWare product family have common graphical elements to allow visual association of individual products within a product family.


*Sample AASHTOWare Product Brand Logos*

### 4.4. AASHTOWare Product Icon

AASHTOWare product icons are the most recognizable graphical element for the product user. Consistency should be maintained in updating product icons between major releases of the AASHTOWare products.

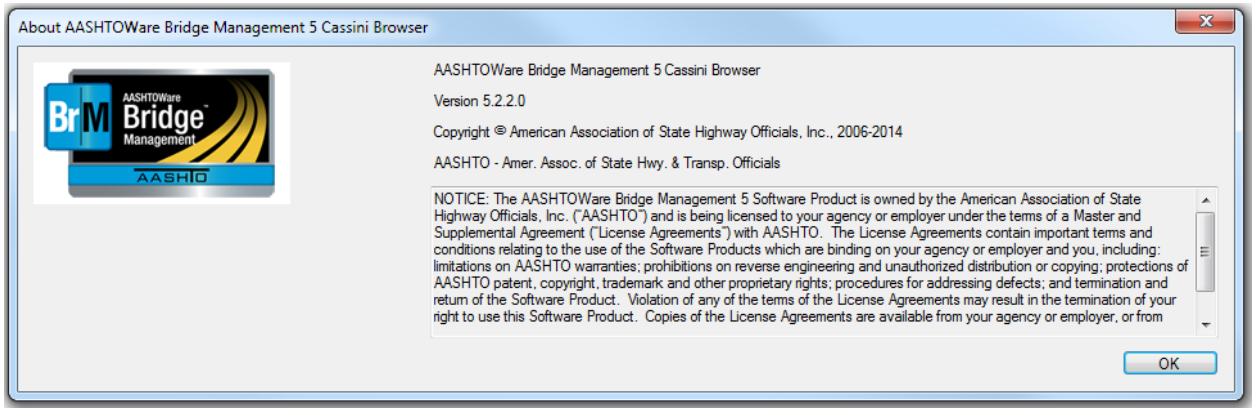

*Sample AASHTOWare Product Icons*

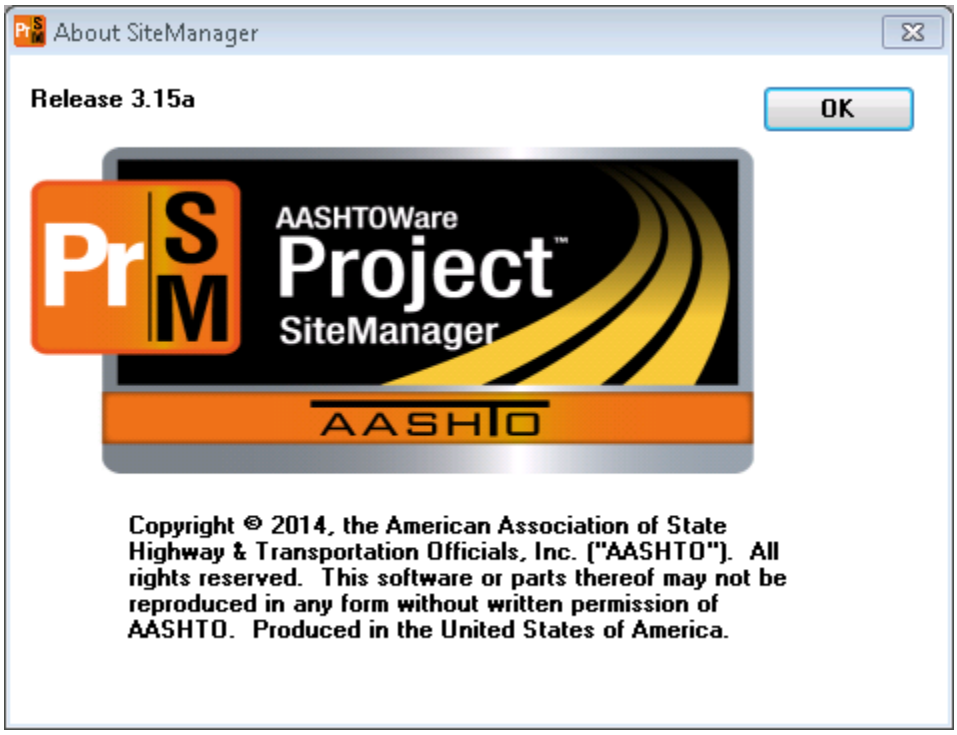
### 4.5. AASHTOWare Product Splash Screen

The AASHTOWare product splash screen should be used to illustrate product quality and consistency. A splash screen can serve as a strong identification mark for a family of AASHTOWare products. The splash screen should contain the product name and logo or other graphics representing the product. The AASHTOWare product brand logo is suitable for a splash screen.

### 4.6. AASHTOWare Product About Dialog Box

The AASHTOWare product About Dialog Box is the most significant product identification component. *The About Dialog Box shall contain complete product nomenclature, copyright notices, product icon, and information for product registration and support.*




*Sample AASHTOWare Product About Dialog Boxes.*

This page is intentionally blank.


# BACKUP AND DISASTER RECOVERY STANDARD

## S&G Number: 2.070.04.5S

## Effective Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 4.2 | 3/15/2018 | Changed new requirements in previous version from bold red italics font to red italics font and changed SCOJD to SCOA. | 6/22/2018 Approved by SCOA |
| 4.3 | 6/04/2019 | Added requirements for cloud services backups and updated backup media information and included using AASHTO's online system to submit backups. | 7/30/2019 Approved by SCOA |
| 4.4 | 2/12/2020 | Removed bold font from requirements added last year. | 7/10/2020 Approved by SCOA |
| 4.5 | 7/26/2023 | Updated the AASHTOWare logo. | 10/02/2023 Approved by SCOA |

# Table of Contents

# 1. Purpose

When a contractor works on an AASHTOWare project or MSE work effort, AASHTO is investing both time and money into this effort. Until a release point is reached, AASHTO has nothing in hand to show for this investment. If a disaster were to happen at the contractor's site, the work the contractor had done on the project might be lost. This would be a loss to AASHTO in both time and money invested to the point of the disaster.

The purpose of this standard is to make certain that AASHTOWare's assets (source code, test scripts, tools, etc.) in all active development and maintenance environments are protected and/or could be recovered in the case of an emergency. The standard defines the actions that AASHTOWare contractors shall take to safeguard AASHTO's development investment in a project or product should a natural or man-made disaster occur.

This standard applies to all AASHTOWare projects and product Maintenance, Support, and Enhancement (MSE) work efforts and, specifically, applies to the software development and maintenance environments, where:

- Development environment describes the operational software development environment for an AASHTOWare project including the software, tools, code, data, deliverables, and documentation; and
- Maintenance environment describes the operational maintenance, development, and support environment for a product MSE work effort.

*All requirements for compliance with this standard are shown in red italicized text.* ***New requirements that were not in the previous version of the Standards and Guidelines are shown in red bold italicized text.*** *If any of the requirements defined in this standard cannot be met, an exception to this standard will need to be documented and approved by SCOA.*

# 2. Task Force/Contractor Responsibilities

The responsibilities of the task force and contractor, regarding this standard, are summarized below. Each responsibility is further described in the *Procedures* section.

- *The contractor organization shall have a Disaster Recovery (DR) Plan that is maintained and annually exercised. The DR plan shall protect the AASHTOWare assets required by this standard, shall be referenced in each MSE and project work plan, and made available on request,*
- *The contractor shall prepare and maintain a Backup Plan, as required by the standard, and include or reference this plan in each MSE and project work plan.*
- *The contractor shall execute the Backup Plan during each project or MSE work effort.*
- *The contractor shall send backup data and checklists to AASHTO headquarters, as defined by this standard.*
- *If infrastructure services are procured to host or support a product's development, maintenance, and/or production environment, the AASHTOWare contractor must ensure that the service level agreement meets all disaster recovery and backup requirements of this standard.*

# 3. Required Deliverables and Artifacts

The following lists the required deliverables and artifacts that shall be created and delivered to comply with this standard. Each item is described in the section of this standard shown following the item.

- *Disaster Recovery (DR) Plan - Prepare and Maintain Disaster Recovery Plan*
- *Backup Plan - Prepare and Maintain Backup Plan*
- *Acknowledgement Letter - Restore Exercise Acknowledgement Letter*

- *December and June Backup Data - Send Checklist and Backup Data to AASHTO*
- *Contractor Backup Checklist - Complete Contractor Backup Checklist*

# 4. Procedures

The section defines the procedures and activities for planning and executing backups and restores.

## 4.1. Prepare and Maintain Disaster Recovery Plan

A Disaster Recovery (DR) Plan includes a comprehensive statement of consistent actions to be taken before, during, and after a natural or human-induced disaster. *A DR plan is required by this standard to support the objective of protecting all AASHTOWare assets associated with each active development and maintenance environment. The contractor organization shall have a documented DR plan that is in use by the organization by the beginning of contract work on any project or MSE work plan. The intent of this requirement is not for the contractor to create a DR plan specifically for AASHTOWare. The expectation is that the contractor organization already has a corporate DR plan that currently protects organizational IT assets, and this same DR plan will be capable of protecting AASHTOWare assets.*

### 4.1.1. Corporate DR Plan Requirements

*The corporate DR plan must document the actions that will be taken to prepare for a natural or man-made disaster, the actions that will be taken during a disaster, and the specific steps that will be taken after a disaster occurs. The specific actions in the DR plan are generally left up to the contractor; however, the DR plan shall include the following AASHTOWare requirements:*

- *Execute a backup plan that meets the requirements of this standard, including the frequency of backup, media, retention, and off-site storage requirements.*
- *Perform an annual exercise that ensures the ability to fully restore all systems, applications, data, and services from backup data and/or media, including all applicable AASHTOWare environments. The requirements for this exercise are defined below.*
  - *The exercise shall include the restoration of AASHTOWare backup data from the most active development environment of the current project or MSE effort.*
  - *If the exercise does not include a complete restore of the environment, then the subset of backup data selected for the exercise shall be different from that used in the previous year.*
  - *The exercise shall include activities to verify that the backup data was successfully restored.*
  - *A description of the backup data included in the exercise and the verification activities performed shall be documented and submitted with an acknowledgement letter as described in the Perform and Acknowledge Annual Restore Exercise section.*
- *Define the recovery time for restoring all systems, applications, data, and services after a disaster or disruption, including all applicable AASHTOWare development, maintenance, and support environments. If the recovery time in the corporate DR plan is not acceptable to the task force or SCOA, an agreed on recovery time, specific to the AASHTOWare environments shall be included in the work plan and/or contract.*

*A summary of the AASHTOWare requirements for the Disaster Recovery Plan is included in the Deliverable and Artifact Definitions section.*

*The current version of the corporate Disaster Recovery Plan shall be referenced in all MSE and project work plans; and shall be available on request by the task force chair or AASHTO project manager.*

*If the contractor organization does not have a corporate DR plan for reference in the work plan, the work plan must include the plan to develop an AASHTOWare specific DR plan (see below) as a deliverable during the execution of the work plan. The DR plan must be developed and approved by the task force prior to beginning any software analysis, design, or development tasks for the work plan. AASHTOWare will provide no funding toward the development of a corporate DR plan.*

### 4.1.2. AASHTOWare Specific DR Plan Requirements

*If an AASHTOWare specific DR plan is developed, this plan shall only address the AASHTOWare environments; and shall include the same basic requirements described above in the Corporate DR Plan Requirements section. These are summarized below.*

- *Include actions that will be taken to prepare for a disaster, during a disaster, and after a disaster occurs.*
- *Execute a backup plan that meets the requirements of this standard.*
- *Perform an annual restore exercise as described above.*
- *Define the recovery time (acceptable to the task force and SCOA) for restoring all applicable AASHTOWare environments after a disaster or disruption.*

*Funding toward the development of an interim plan and funding to perform the annual exercise in an interim plan must be approved by SCOA.*

## 4.2. Prepare and Maintain Backup Plan

The Backup Plan includes what will be backed up, the frequency of backups, type and retention of each backup, off-site storage procedures, type of media and software used for backup and recovery, roles and responsibilities, backup procedures, procedures to recover individual files or the complete development environment; and any specific needs of the project or product.

*The contractor shall prepare and maintain one or more Backup Plans that cover each active development and maintenance environment that are under contract for services*. Typically, one Backup Plan will cover all on-going projects and MSE efforts; however, the contractor may also choose to have multiple plans.

*The current version of the applicable Backup Plan shall be included or referenced in all MSE and project work plans, and the current plan shall be posted in the project repository. If a Backup Plan does not exist when the work plan is prepared, then the tasks to prepare and approve the Backup Plan shall be included in the work plan. In this case, the Backup Plan shall be prepared, submitted, and then approved by the task force prior to any software analysis, design, or development tasks.*

The Backup Plan should be reviewed on an annual basis, modified as required, and the revised version included in the next work plan prepared.

*When preparing the Backup Plan, the contractor shall consider and address all the topics in this section; and, if applicable, any specific needs of the project or product.* The *Compliancy Backup Checklist*, which is included in the Appendices, is provided to assist in comparing the Backup Plan with the requirements of this standard.

### 4.2.1. What Will be Backed Up

The primary reason for backing up data is to safeguard against the loss of data and to minimize the re-entry of data and/or the redo of tasks when a loss occurs. *To fully protect the development or maintenance environment, all development and maintenance related files shall be backed up along with all tools, data, documentation, and other*

*resources that would be required to restore the complete environment and continue operations at an alternate site.*

*The data that will be included in each backup shall be identified in the Backup Plan. This shall include, but is not limited to, the following:*

- *Source code*
- *All deliverables, artifacts, reports, and documentation*
- *Test scripts and test databases*
- *Data repositories used for project management and software engineering processes and tools. (Such as a problem/issue database, requirements database, or database of design artifacts.)*
- *Copy of tools and development software used*
- *Copy of backup and restore software*

Most of the data to be backed up is identified by defining specific computers and servers and the folders and files to be included or excluded in the backups.

#### *4.2.1.1. Hosted Development Tools*

Hosted development tools, such as Azure DevOps or Jira Software Cloud, cannot be backed up by the contractor in traditional ways. There is an expectation for all hosted artifacts to be extractable and importable into the hosted tool from which those artifacts came. For example, Git repositories in Azure DevOps can be fully exported and imported into the Azure DevOps platform. Similarly, build and release tasks and backlog items can be exported into machine-readable formats.

*Artifacts produced in a hosted development tool must be backed up in machine-readable formats.*

### 4.2.2. Types of Backups

*The Backup Plan shall define the type of backups to be used during the backup cycle defined in the next section.* The basic types of backups are defined below:

- Full backup – A full backup is the starting point for all other types of backup and contains all the data in the folders and files that are selected to be backed up. Since full backups store all files and folders, frequent full backups result in faster and simpler restore operations; however, the restore process may take longer.
- Differential backup – A differential backup contains all files that have changed since the last full backup. The advantage of a differential backup is that it shortens restore time compared to a full backup or an incremental backup; however, if a differential backup is performed too many times, the size of the differential backup might grow to be larger than the baseline full backup.
- Incremental backup – An incremental backup stores all files that have changed since the last full, differential, or incremental backup. The advantage of an incremental backup is that it takes the least time to complete; however, during a restore operation, each incremental backup shall be processed, which could result in a lengthy restore job.
- Hosted services backup – A backup done by a hosted service is semi-transparent to the contractor. *If the contractor utilizes hosted services, they must describe how the hosted service is meeting the minimum required backup cycle. If the hosted service provider cannot meet the minimum required backup cycle, the contractor must supplement with their own data extractions.*

### 4.2.3. Frequency and Retention of Backups

*The Backup Plan shall define how often backups will be performed (frequency) and how long each backup shall be kept (retention). The minimum required backup cycle is listed below with the type and the retention of each backup.*

| Frequency | Type | Retention |
|---|---|---|
| *Daily* | *Incremental or Full* | *7 days* |
| *Weekly* | *Full* | *8 weeks* |
| *Monthly* | *Full* | *12 months* |
| *Yearly* | *Full* | *12 months or life of the project/MSE if the length of project/MSE extends past two years.* |

The Yearly backup is the same as the 12th monthly backup for each year of the project or MSE effort.

### 4.2.4. Off-site Storage

One of the reasons for doing backups is to safeguard the work and investment that is being done against loss in the event of a disaster. Disasters come in many forms, such as hardware failures, fires, floods, or human-induced disasters. Storing copies of the backup media off-site is essential to recovering your systems and data in the case of a natural disaster.

*The Backup Plan shall define what backups will be stored off-site, the location of the backup site, the distance from the main site, the environment of the off-site location, and the frequency that the backup media is moved to the off-site storage location. The requirements for off-site storage are listed below:*

- *Copies of all weekly, monthly, and yearly backup data shall be stored at an off-site location.*
- *Backup data shall be moved to the off-site storage location weekly.*
- *Copies of all software and tools (including the backup software) needed to re-establish all systems and data shall also be stored at the off-site location.*

Some of the things that should be considered when selecting an off-site location are listed below.

- The off-site storage location should be at a great enough distance from the main site so that any disaster at the main site will not impact the off-site location.
- The environment at the off-site storage location should appropriate for storage of the backup data/media.
- The best type of place to store off-site backup media/data is at a site set up specifically for this function. Another contractor site could be used if the location and environment were adequate.

### 4.2.5. Backup Media and Backup Software

*The Backup Plan shall include the type of media used for backups and the product name, version, and manufacturer of the software that will be used for backups, off-site data transfer, and restores.*

Magnetic tapes are no longer the most common media used for backups. External hard drives, Blu-ray or DVD discs (M-DISC), and USB thumb drives are media types to be used for backups.

The backup media and the equipment used to read and write the media should be common products that can easily be obtained from a market leader that supplies this type of hardware or media. The software that is used to create, write, access, transfer, and restore the backup data is also important. As with the media, it is best to use backup and restore software that is common to many sites and can easily be obtained from a market leader that supplies this type of software.

One of the best ways to make sure your media is compatible is to perform a test restore at the backup location or a location with the same hardware and software as the backup location. *The contractor's DR plan (corporate or AASHTOWare specific) shall include an*

*annual restore exercise as described in the Prepare and Maintain Disaster Recovery Plan section.*

### 4.2.6. Care of Backup Media

*Once a backup is performed, the media shall be properly cared for. The manufacturer's guidelines for the care of the media being used shall be followed to ensure the best chance of having readable media when needed.* Some of the things these guidelines will typically cover are environment, handling, age, and usage.

### 4.2.7. Backup Documentation

Documentation is an important part of the Backup Plan. Should a disaster occur at the main site and the development or maintenance environment needs to be restored at another site, it is important to know what is on all backup media. A log that documents each backup media used, what is on each backup media, any excluded files, and the date of the backup is essential to performing an accurate recovery and minimizing the time to perform the recovery.

*A log documenting all backups shall be prepared and maintained throughout the project/MSE lifecycle. A copy of the log shall be stored at the off-site location.*

### 4.2.8. Additional Backup Plan Elements

In addition to the above, the Backup Plan should typically define the procedures that will be executed for backups and restores, roles and responsibilities, and any specific needs for the project or product.

## 4.3. Execute the Backup Plan

*After the Backup Plan is prepared and approved, the contractor shall execute the plan as documented; including the following:*

- *Perform daily, weekly, monthly, and yearly backups*
- *Log all backups*
- *Retain backup data/media*
- *Move data/media, copies of tools and software, and logs to off-site storage*
- *Use high-quality backup media and backup and restore software*
- *Care for the media and replace as required*
- *Restore files, systems, data, etc., as required*

## 4.4. Additional Backup and Restore Requirements

*In addition to the above activities, the following backup and restore activities shall be performed:*

### 4.4.1. Perform and Acknowledge Annual Restore Exercise

*As described in the Prepare and Maintain Disaster Recovery Plan section, the contractor's Disaster Recovery Plan (corporate or AASHTOWare specific) shall include an annual exercise that ensures the ability to fully restore all systems, applications, data and services from backup data and/or media, including all applicable AASHTOWare environments.*

*After this annual exercise is performed, the contractor shall send a letter to the task force chair and AASHTO project manager that acknowledges the successful completion of the annual exercise (Restore Exercise Acknowledgement Letter). The documentation describing the backup data included in the exercise and the verification activities performed is submitted with the acknowledgement letter.*

### 4.4.2. Complete Contractor Backup Checklist

*The contractor shall complete the Contractor Backup Checklist twice a year after completing the December and June monthly backups.* The checklist is used to

demonstrate compliance with this standard and the contractors Backup Plan. The *Contractor Backup Checklist* is shown in the Appendices of this standard and may be downloaded from the AASHTO web site or SharePoint workspace.

The Appendices also includes the *Compliancy Backup Checklist* which provides the minimum compliance level for the Backup and Disaster Recovery Standard

#### 4.4.3. Send Checklist and Backup Data to AASHTO

*The contractor shall send the Contractor Backup Checklist to AASHTO headquarters with a copy of the December and June monthly backup data by the 15th day of the following month.* The backup data may be sent to AASHTO on an external hard drive or another type of removable media or submitted via AASHTO's online system.

The AASHTO Project Manager will review each submitted checklist for consistencies with the documented Backup Plan and report any inconsistencies to the task force chair and contractor. The Backup Plan and Contractor Backup Checklists will also be reviewed during the annual QA review.

The completion of the checklists and the submissions to AASHTO should be planned activities and typically should be included in the Backup Plan.

# 5. Technical Requirements

There are no technical requirements for this standard.

# 6. Deliverable and Artifact Definitions

## 6.1. Disaster Recovery Plan

### 6.1.1. Description

*Disaster Recovery Plan includes a comprehensive statement of consistent actions to be taken before, during, and after a natural or human-induced disaster*

### 6.1.2. Content

*The Disaster Recovery Plan shall include, but is not limited to the following content:*

- *Actions that will be taken to prepare for a natural or man-made disaster, actions that will be taken during a disaster, and specific steps that will be taken after a disaster occurs. The actions to prepare for a disaster shall include:*
  - *Executing a backup plan that meets the requirements of this standard; and*
  - *Performing, verifying, and documenting an annual restore exercise as described in the Prepare and Maintain Disaster Recovery Plan section.*
- *Actions for fully restoring each applicable environment at an alternate site and resuming normal operations within a specified number of days following a disaster event. The specified number of days will be agreed upon by both AASHTO and the contractor organization.*

The format of the plan and all other content is left up to the contractor and any product or project-specific requirements.

## 6.2. Backup Plan

### 6.2.1. Description

*The Backup Plan defines the plan for backing up the AASHTOWare development or maintenance environments.* Typically, one Backup Plan will cover all on-going projects and MSE efforts; however, the contractor may also choose to have multiple plans.

### 6.2.2. Content

*The Backup Plan shall contain the following items and shall conform to the requirements in Prepare and Maintain Backup Plan section of this standard.*

- *What will be backed up*
- *Types of backups*
- *Frequency and retention of backups*
- *Off-site storage*
- *Backup media and backup software*
- *Care of backup media*
- *Backup documentation*
- Procedures for backups and restores (as required to use the above)
- Responsibilities (as required for the above)
- Any product- or project-specific requirements.

The format of the plan is left up to the contractor.

## 6.3. Contractor Backup Checklist

### 6.3.1. Description

*The Contractor Backup Checklist shall be completed by the contractor twice a year after completing the December and June monthly backups. The checklist is sent to AASHTO headquarters with a copy of these backups.*

### 6.3.2. Content

*The Contractor Backup Checklist form in the Appendices, or an equivalent form with the same content, shall be used by the contractor. The form is available for download on the AASHTOWare web site or SharePoint workspace.*

## 6.4. Restore Exercise Acknowledgement Letter

### 6.4.1. Description

*This letter is sent to the task force chair and AASHTO project manager acknowledging the successful completion of the contractor's annual restore exercise.*

### 6.4.2. Content

*No specific content is required other than a statement that acknowledges the success of the annual restore exercise, as documented in the Disaster Recovery Plan. In addition, documentation describing the backup data included in the exercise and the verification activities performed shall be submitted with the letter.*

# 7. Appendices

## 7.1. Contractor Backup Checklist

**Contractor Backup Checklist**

| **General** | |
|---|---|
| Is there a written Disaster Recovery Plan? (yes/no), Does it include a minimum recovery time? (yes/no), Does it include an annual restore exercise? (yes/no) | |
| Is there a written Backup Plan? (yes/no) | |
| What is being backed up? | |
| Source code (yes/no) | |
| Product/product deliverables, artifacts, reports, & documentation (yes/no) | |
| Test scripts and test databases | |
| Data used for project management & software eng. processes & tools (yes/no) | |
| Other | |
| Has an annual restore exercise been performed? (yes/no), When?<br>Is the restore exercise planned? (yes/no), When?<br>Was the restore verified and documented? (yes/no)<br>Was an acknowledgement letter with exercise documentation sent to the TF Force Chair and AASHTO PM after the exercise was completed? (yes/no), When? | |
| **Infrastructure (On-Premise or as a Service)** | |
| Are daily backups being done? (yes/no), What type of backup? (full/incremental?), What is the retention period? | |
| Are full weekly backups being done? (yes/no), What is the retention period? | |
| Are full monthly backups being done? (yes/no), What is the retention period? | |
| Are full yearly backups being done? (yes/no), What is the retention period? | |
| Is the media and software being used common, a market leader? (yes/no) | |
| List type and brand of media:<br>_____ | |
| Name, version, and manufacturer of backup software:<br>_____ | |
| Is the media being tracked for age and use? (yes/no) | |
| Off-site Storage | |
| Which backup media is being stored off-site? (daily, weekly, monthly, yearly) | |
| Distance main site is from off-site storage location | |
| Does off-site storage have a controlled environment for media storage? (yes/no) | |
| Location of off-site storage:<br>_____ | |
| How often is media transmitted to off-site location? | |
| Is the log of what is being stored on each backup media being stored off-site with the media? (yes/no) | |
| Are copies of the software and tools required to restore the files from the backup media and to re-establish the operational environment being stored at the off-site storage? (yes/no) | |
| **Hosted Services (e.g. Azure DevOps, AWS, Jira Cloud)** | |
| Is the configuration and data for the hosted service actively replicated to multiple datacenters or regions? | |
| Does the hosted service provide AASHTOWare sufficient recovery options for data and configuration? | |

| | |
|---|---|
| If No, are supplemental backups of data and configuration being done by the contractor? | |

Download at: *https://www.aashtoware.org/wp-content/uploads/2019/07/AASHTOWare_Contractor_Backup_CheckList_07302019.docx*.

## 7.2. Compliancy Backup Checklist

The Compliancy Backup Checklist, shown below, provides the minimum compliance level for this standard in *red italic*.

**Compliancy Backup Checklist**

**If not performed during the first 6-month period, shall be performed during second 6-month period. In this case, the planned date is provided and other dates are N/A.*

| **General** | |
|---|---|
| Is there a written Disaster Recovery Plan? (yes/no), Does it include a minimum recovery time? (yes/no), Does it include an annual restore exercise? (yes/no) | *Yes, Yes, Yes* |
| Is there a written Backup Plan? (yes/no) | *Yes* |
| What is being backed up? | |
| Source code (yes/no) | *Yes* |
| Product/product deliverables, artifacts, reports, & documentation (yes/no) | *Yes* |
| Test scripts and test databases | *Yes* |
| Data used for project management & software eng. processes & tools (yes/no) | *Yes* |
| Other | |
| Has an annual restore exercise been performed? (yes/no), When?<br>Is the restore exercise planned? (yes/no), When?<br>Was the restore verified and documented? (yes/no)<br>Was an acknowledgement letter with exercise documentation sent to the TF Force Chair and AASHTO PM after the exercise was completed? (yes/no), When? | *Yes*, Date*<br>*Yes*, Date**<br>*Yes**<br>*Yes*, Date* |
| **Infrastructure (On-Premise or as a Service)** | |
| Are daily backups being done? (yes/no), What type of backup? (full/incremental?), What is the retention period? | *Yes, Incremental or Full, 7 days* |
| Are full weekly backups being done? (yes/no), What is the retention period? | *Yes, 8 weeks* |
| Are full monthly backups being done? (yes/no), What is the retention period? | *Yes, 1 year* |
| Are full yearly backups being done? (yes/no), What is the retention period? | *Yes,1 year or Life of project/MSE if longer than 2 yrs.* |
| Is the media and software being used common, a market leader? (yes/no) | *Yes* |
| List type and brand of media:<br>_____ | |
| Name, version, and manufacturer of backup software:<br>_____ | |
| Is the media being tracked for age and use? (yes/no) | *Yes, or redundant Cloud Storage* |
| Off-site Storage | |
| Which backup media is being stored off-site? (daily, weekly, monthly, yearly) | *Weekly, Monthly, Yearly* |
| Distance main site is from off-site storage location | *Miles* |
| Does off-site storage have a controlled environment for media storage? (yes/no) | *Yes* |
| Location of off-site storage:<br>_____ | |
| How often is media **transmitted** to off-site location? | *Weekly* |
| Is the log of what is being stored on each backup media being stored off-site with the media? (yes/no) | *Yes* |
| Are copies of the software and tools required to restore the files from the backup media and to re-establish the operational environment being stored at the off-site storage? (yes/no) | *Yes* |

| **Hosted Services (e.g. Azure DevOps, AWS, Jira Cloud)** | |
|---|---|
| Is the configuration and data for the hosted service actively replicated to multiple datacenters or regions? | *Yes* |
| Does the hosted service provide AASHTOWare sufficient recovery options for data and configuration?<br>If No, are supplemental backups of data and configuration being done by the contractor? | *Yes, No – but supplemental backups are being performed.* |

Download at: *https://www.aashtoware.org/wp-content/uploads/2020/08/AASHTOWare_Compliancy_Backup_CheckList_02122020.docx*.


# MOBILE APPLICATION DEVELOPMENT GUIDELINE

## S&G Number: 2.080.01.6G

## Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.4 | 2/3/2014 | Initial Version. | 2/03/2014 Approved by T&AA Chair |
| 1.5 | 3/15/2018 | Changed SCOJD to SCOA. | 5/22/2018 Approved by T&AA |
| 1.6 | 9/30/2023 | Updated the AASHTOWare logo and broken links. | 10/02/2023 Approved by SCOA |

# Table of Contents

# 1. Purpose

The guideline defines an initial set of practices and technologies that should be used when developing AASHTOWare mobile applications. The expectation is that updates will be made to this guideline as AASHTOWare contractors, task forces, and other joint development stakeholders gain more experience and knowledge in the development of mobile applications. It is also expected that a standard will be published at a later date with mandatory requirements.

The demand for mobile applications in business is increasing exponentially as users discover the benefits during personal use of smart phones and tablets. The ability to leverage the features of a mobile device (phone, camera, video, voice or GPS) makes them ideal for use in the field. The challenge for AASHTOWare and other organizations is the development and maintenance of applications for the different mobile operating system platforms and various types of mobile devices.

Currently there are two leading mobile platforms, Apple iOS and Android; however, there no clear winner to become the industry standard in the short-term. This poses a challenge for organizations that develop and market mobile applications. Unless the customer base standardizes on one platform, most organizations may choose to support both of these platforms and possibly support the third or fourth place platforms, Windows and BlackBerry. Recent AASHTO IT Surveys indicate that supporting iOS and Android appears to be the appropriate approach for AASHTOWare. Since most AASHTOWare is written to operate on existing Windows desktop operating systems, supporting a Windows mobile platform is also recommended.

The practices and technologies described in this guideline provide an initial approach for developing mobile applications that can execute on multiple operating systems platforms and on multiple types and sizes of mobile devices, while attempting to meet the following objectives:

- Minimize the cost and level of effort required for the initial development of mobile applications, as well as for the long term maintenance and enhancements; and
- Avoid long-term vendor dependence or lock-in to specific operating systems, tools and technologies.

The optimal solution to meeting these objectives is to develop cross-functional mobile applications that use a single code base to support all required platforms and devices through the use of open standards and non-proprietary technologies.

# 2. Types of Mobile Applications and Web Sites

When business needs require an organization to support multiple mobile platforms and devices, the next step is to determine the appropriate development approach for supporting each platform. In many cases, the development approach is initially driven by the decision to create mobile applications (or apps) for users to download and run on their mobile device or to create mobile websites that are accessed from a web server at specific URLs. The look and feel of mobile apps and web sites may be similar at first-glance; however, the difference in features, functionality, and operation can be significant, as can the cost to develop and maintain each type.

Creating a mobile web site is normally the least costly approach for developing a mobile web application and, in most cases, is the best starting point for organizations beginning a mobile development effort. Mobile web sites also tend to be more limited than a mobile app. Access to

most of the native features of a mobile platform or device; such as the camera, geolocation, local storage, and gestures normally require a mobile app.

Understanding the differences between mobile apps and web sites is the key to deciding the best mobile development approach. The types of mobile apps and web sites that can be created are equally important. The following subsections describe four types of mobile development, as well as the key differences. These include two categories of mobile apps, native and hybrid, and two categories of web sites, mobile web sites and responsive design web sites.

The recommended mobile development approaches for AASHTOWare are responsive web design and/or creating hybrid apps with Apache Cordova, an open source toolkit. Although not recommended, there may also be cases where creating a mobile web site or a native mobile app may better fit the business requirements for a new mobile web site or app.

Although this is a guideline, SCOA has requested to be kept aware of mobile development that does not follow the recommendations of this guideline. The task force chair should contact SCOA early in the project lifecycle when the development approach is not responsive web design or a hybrid app.

## 2.1. Mobile Web Site

A mobile website is similar to any other website in that it consists web pages created using the HyperText Markup Language (HTML), uses Cascading Style Sheet (CSS) instructions to control the layout and presentation format of the web pages, and is accessed over the Internet though a browser. Like other web sites, mobile web sites also typically use JavaScript to provide interaction with users and to add behaviour to the web pages.

The main characteristic that distinguishes a mobile website from a standard website is the fact that it is designed and optimized for the small screen and touch interface of a mobile device. The layout, presentation and navigation of the web pages are normally specifically optimized to provide the best viewing and user experience on a smartphone.

Although a mobile site may share many of the back end resources a primary web site (full site), a mobile web site is separate and must be developed and maintained in parallel. Since, the mobile web site is different from the full site, it must also use a different domain. Many companies choose to differentiate their mobile site from the primary web site with an “m.” prefix, as in “m.company.com”. Some sites also redirect a user to the mobile site from the full site based on the type of device being used to view the site.

In most cases, a tablet will work appropriately with either the mobile web site or full site depending on the screen size and resolution the tablet. Additional changes may also be required to ensure that a mobile web site displays and navigates pages appropriately with both smartphones and tablets and the wide array of screen sizes and resolutions. Without these changes, the user may be required to switch back and forth between the mobile site and full site for certain pages.

As noted above, mobile web sites run inside a browser and do not require local installation on a mobile device. Since a mobile website does not reside on the mobile device, it is much more dynamic than a mobile app in terms of flexibility to update content. Once a mobile web site is updated on the server, the changes are immediately available to all users.

Most mobile web sites are created with the latest versions of HTML and CSS (HTLM5 and CSS3), as well as the latest version of JavaScript (ECMAScript 5.1). Mobile web sites also

typically use pre-prepared open source libraries and frameworks that are compliant with the HTML5, CSS3 and JavaScript standards.

The overall development approach for mobile web sites generally allows the code to be written once and run on any mobile device there is a browser that is compliant with the HTML5, CSS3 and JavaScript standards.

The downside is that a mobile specific site must be written and maintained in addition to the primary web site, and the mobile web site is normally optimized for smartphones only. Additional development may be needed to support a wide range of tablet and smartphone displays. Mobile web sites also have limited access to native platform and device features, such as the camera, geolocation, and local storage. Broader access to native features requires a native or hybrid mobile app.

Although, mobile web sites are normally cross-functional on multiple platforms and devices, this guideline recommends responsive web design sites in lieu if web sites targeted for specific mobile devices. Responsive web design sites, which are discussed next, are designed to operate on a wide variety of devices.

## 2.2. Responsive Design Web Site

Wikipedia describes Responsive Web Design (RWD) as a web design approach aimed at crafting sites to provide an optimal viewing experience, with easy reading and navigation and minimum of resizing, panning, and scrolling, across a wide range of devices from mobile phones to desktop computer monitors.

A web site designed with RWD adapts the layout to the viewing environment by using fluid, proportion-based grids, flexible images, and *CSS3* *media queries*. Media queries allow the web page to use different CSS style rules to adapt the display and rendering of a web site based on characteristics of the device the web site is being displayed on, including the device display width, height, resolution and aspect ratio. Flexible images and fluid grids size the web content correctly to fit the screen, resolution and orientation.

The advantage of responsive web design is that the content of a web site remains the same from device to device. A website built with a RWD coding techniques is not built for any specific type of device. There is no need for a mobile-only version since all devices use the same site and a single domain name.

A RWD website is completely fluid, scaling on the fly as the site is accessed by different size devices, as well as when a browser window is adjusted to a smaller or larger size. The goal is to provide the users with an optimal viewing experience across a wide range of devices from the largest desktop monitor to the smallest smartphone display. Although it may display differently, the basic content of the web site will be the same for each user accessing the site regardless if they use a smartphone, tablet or desktop device.

This guideline does not go into the details required to develop a RWD web site. As with other web sites, RWD web sites are developed with HTML5, CSS3, JavaScript, and predefined libraries and frameworks. The specific tools, libraries and methods used for RWD techniques are left up to the contractor; however, in order to meet the objectives of this guideline, proprietary tools and libraries should be avoided.

Creating responsive design web sites for access with mobile devices is one of two preferred mobile development approaches recommended by the guideline. In order to meet the objectives of this guideline, the contractor should develop the web site using a single set of code for all platforms supported. In addition, the web site should be developed using open

web technologies standards, including the latest versions of HTML, CSS (currently HTML5, CSS3), and JavaScript. Add on libraries and frameworks should be open source; proprietary tools and dependencies should be avoided.

Using this approach, an RWD web site should adapt, display and operate correctly on any device and platform that uses a browser compliant with the HTML5, CSS3, and JavaScript standards.

As with mobile specific web sites, responsive design web sites have limited access to native platform and device features, such as the camera, geolocation, and local storage. Access to native features requires a native or hybrid app. When native features are needed, it is recommended that a hybrid mobile app using Apache Cordova be developed in lieu of a native app.

## 2.3. Native Mobile Applications

Where a mobile web site or a responsive design web site can be run from any device with a browser, a native mobile application (app) can only be run on a single platform. Native mobile applications are typically written in the native language of the device/operating system. For example, applications are written in Java or Dalvik for Android; Objective C for iOS; and C# or C++ for Windows Phone. Each mobile platform provides a different toolset for their native app developers including the tools and materials needed for app store submission.

Native mobile apps are downloaded to each device from an app store, such as the Apple App Store or Google Store, activated through icons on the device's home screen, and executed locally of the device's operating system. Since these applications are developed specifically for one platform, native apps and can take full advantage of all the device features, such as the camera, geolocation, accelerometer, and list of contacts, and gestures. The ability to use all native features and functions, typically allows a native to provide to a richer device specific user experience and improved performance from that of a web site. Also, since native apps run locally, they have the ability to run offline.

The downside is that a native app is developed for one platform will not run on another mobile platform. Separate native apps are required for each platform supported. Since different versions of an application are required for each platform; developing, maintaining, and supporting applications for each platform introduces a significant increase in cost and effort than that of a single cross-functional application.

Due to the parallel development and maintenance work, native apps are not currently recommended and should only be considered for specific business processes or user interactions when neither a responsive web design site nor a hybrid mobile app (discussed next) is feasible.

## 2.4. Hybrid Mobile Applications

Hybrid mobile applications (apps) can be thought of as mobile web sites that have access to certain native features. Hybrid apps are also packaged to download and execute on a mobile device in the same manner as native apps. Hybrid mobile apps are developed using a Mobile Enterprise Application Platform (MEAP). A MEAP solution is a suite of products and services that enable the development of mobile applications that can execute on multiple mobile platforms. MEAP solutions allow hybrid apps to be created using HTML5, CSS3, and JavaScript in lieu of the typical languages use for native apps. In addition, MEAP solutions provide a set of device APIs that allow a mobile app developer to access

native device functions from JavaScript. The MEAP solution packages the HTML5, CSS3, and JavaScript code; access to the native functions; and an embedded copy of the browser as a native application that can be distributed through an app store.

As with mobile and RWD web sites, hybrid applications can also use pre-prepared libraries and frameworks in addition the MEAP specific libraries that are used to access native device features.

There are many different MEAP solutions available to build hybrid mobile apps; however, many of these use proprietary tools and/or libraries. At this point, Apache Cordova, an open source MEAP solution which was previously referred as PhoneGap, is the only MEAP solution recommended when developing for in AASHTOWare mobile development. Cordova has no cost and Apache's licensing agreement allows the mobile applications and Cordova to be distributed without any additional costs.

Since hybrid apps are written with HTLM, CSS3, and JavaScript and run within an embedded browser, an assumption could be made that a hybrid app could use responsive web design techniques. The compatibility of RWD techniques and hybrid apps is unknown and not addressed or recommended in this guideline.

# 3. Features

When developing web sites or mobile apps, the following features and functionality should be considered when developing mobile applications:

## 3.1. Usability

Interface and operational processes need to focus on usability. Focus on delivering relevant information and experience; eliminate every possible click, or tap, from the design. Ask for the minimum amount of information. Consider the need for voice recording, text input, pictures, and positional location as a minimum.

## 3.2. Speed/Performance

Priority should be given to the speed and execution of the application.

## 3.3. Security

Include authentication and data security as necessary; consider encryption for transmissions and storage as warranted. Consider privacy needs.

## 3.4. Offline Capabilities

Build in content, functions and features as needed that do not rely on a connection. Provide offline storage and synchronization with the primary data store when a connection is available. Provide a non-obtrusive indicator showing network connectivity if not otherwise provided for as part of the device.

## 3.5. Customization

Provide for the adjustment of settings for the app: colors, font sizes, etc.

## 3.6. Feedback Mechanism

Provide a mechanism for the user to submit feedback. Users feel more involved when they have the capability to offer suggestions, report bugs, and criticisms. The mechanism (email, form submittal, or link) is less important than the capability – and the response to the feedback.

## 3.7. Keep It Simple

Focus on the basic requirements of the application. Resist the urge to add unneeded features into the mobile app even if easily done. Concentrate on getting the job/task done and ease of use. Remember it is a small form factor mobile device.

## 3.8. Include Analytics

Incorporate analytics into your mobile application such as location, page visits, download counts and connection activity. The data gathered may be helpful in updating the application.


# WEB APPLICATION DEVELOPMENT GUIDELINE AND ARCHITECTURE GOALS

## SG Number: 2.085.01.5G

## Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.3 | 1/11/2016 | Made changes and corrections after review by contractors and task force members. | 2/22/2016 Approved by T&AA & SCOJD |
| 1.4 | 4/06/2016 | Made minor spelling/grammar/naming corrections. | 4/08/2016 Approved by T&AA Chair |
| 1.5 | 7/26/2023 | Updated the AASHTOWare logo | 10/02/2023 Approved by SCOA |

# Table of Contents

# Table of Figures

# 1. Introduction

As a guideline this document is intended to promote approaches and practices for developing web-based applications (i.e. - web application) for AASHTOWare. This guideline does not rigidly dictate an application architecture model, although current best practices have defined preferred models. Similarly, this guideline does not dictate a single technology stack, technology platform, or technical architecture since AASHTOWare customers rely on a variety of technology vendors, technical architectures, and have unique and different IT environments.

This guideline is intended to be generally non-technical in the context of Information Technology (IT) and software development. The guideline's contents should be readily understood by both contractor technical staff, as well as task force members, committee members, project managers, and other AASHTOWare stakeholders.

# 2. Purpose

This guideline does intend to establish a consistent high-level approach for contractors such that existing AASHTOWare software products evolve in a consistent and recognizable fashion which will help to align product architecture over time.

New products under consideration or just beginning development should adopt the strategies and recommendations of this guideline, the intent being to build and deliver a product with an application architecture that is sustainable, extensible, scalable, adaptable, and capable of future evolution.

# 3. Web Application Characteristics

Web Applications (Web Apps) have a number of characteristics that are worth identifying for this guideline document:

- Web Apps rely on a commonly available client that is available on mobile devices and workstations, namely, a Web Browser;
- A Web Browser will commonly run on a variety of operating systems and devices, which makes a Web App both device and operating system independent.
- Because devices are of all shapes and sizes, Web Apps must responsively adapt to the device form factor so that the applications are useable no matter which form factor is in front of a user.
- Web Apps rely on a hosting platform for the client device and browser to interact with; a Web App may interact with multiple hosting platforms simultaneously via web services.
- Web Apps typically display data, and also collect and store data, which may require a database platform (e.g. – Database Management System, or DBMS, may be used interchangeable with database platform.)
- Web Apps may interact with multiple database platforms simultaneously.
- Web Apps can be used anywhere and frequently are used anywhere.
- Web Apps may need to be spatially aware and capable of displaying maps and spatially related information.
- Web Apps may interact with client device features, or devices attached to the client device.

- Web Apps rely on web services to implement interfaces to other systems, and to support transactions with platforms both inside and external to their hosting environment; REST (Representational State) services are the preferred web service model used to support web applications. SOAP services can provide a stateful interface between systems, but are not typically used for web-based transactions.

# 4. Open Standards

This guideline promotes the adoption and use of open standards. Open Standards should be supported to the greatest extent practicable, and adopted widely for AASHTOWare Web Applications, as well as other AASHTOWare products. Open standards are generally recognized as a key factor in a product's ability to integrate efficiently, and interact easily, with other software products, software libraries (component libraries), and other specialized systems. Web Applications are realistically flawed as-built if they don't support open standards, which limits their useful life, increases their cost for support, and makes them less likely to work on any device, operating system (OS), or browser.

## 4.1 Data, Interfaces, Services

Open standards help to support a product's ability to communicate with other data consumers and data providers, and to support efficient development of interfaces with systems. Open standards provide the foundation for web services, and web service models such as SOAP and REST.

Service Oriented Architecture (SOA) relies upon a foundation established by open standards. Open standards provide the underlying mechanisms that allow dissimilar, heterogeneous IT environments to communicate with each other, effectively allowing transparent sharing of information and services.

Information Models such as NIEM (National Information Exchange Model) promote efficient exchange of data that reduces the need for data to be transformed or translated between business systems. Information models are based upon open standards technologies, and they further extend and promote open standards related to data, interfaces, and services.

## 4.2 OS and Device Independence

Web Apps that adopt open standards (such as HTML5) accommodate an application's need to run on a variety of Web Browsers, and on any device. Open standards support a Web App's ability to be substantially operating system and device independent.

## 4.3 Open Spatial Standards

Open standards such as the Open Geospatial Consortium (OGC) Standards support the definition and use of spatial web service models that allow a software product to readily interact with a spatial data service or GIS platform.

## 4.4 Open Communication and Networking Standards

Open standards support the functional specialization of network and system platforms to support networking, security, application hosting, and virtually all functional elements that are the basis of web applications.

# 5. Web Browser Agnostic

Perhaps the greatest value of Web Applications is their ability to use a common client, a web browser, which resides on virtually every modern day device used for business productivity or personal use. This includes modern smart phones, tablets, laptops, desktop computers, and even televisions, stereos, vehicles…the list goes on and on.

For a web application to be of greatest use it should be able to function on any of the web browsers commonly used by AASHTOWare customers. In other words, the application should be browser agnostic, and provide all of its capabilities to the user whether that user prefers Internet Explorer, Firefox, Chrome, Safari, or any future Open Standards Based web browser.

Essentially, if a web application is web browser agnostic, it has also made itself device and operating system agnostic. A web application with this characteristic will work on a Windows desktop, a Linux Server, an Apple Laptop, an Android or Apple phone, and so on.

# 6. Database Agnostic

Web Application products should be database agnostic so that they may be readily consumed by AASHTOWare customers.

## 6.1 Object Relational Mapping

The proposed web application architecture promotes the adoption of common object relational mapping (ORM) technologies and approaches, which are functionally implemented within the persistence layer ("persistence" as defined in *Figure 1*). This strategy allows developers to access and request data from the database, update the database, and generally interact with the data structures supporting the application's transactions, without using database languages (SQL) within the other layers of the application architecture.

The persistence layer also supports the applications ability to interact with any database provider, including multiple flavors of databases at the same time, without significant custom coding for each deployment of the application.

Prior to common adoption of ORM tools for Web Applications, an application may have required custom coding that limited their ability to easily interact with a variety of database technology providers. Implementing a persistence layer and ORM tools provides better security, speeds development, simplifies maintenance, and generally makes the application more adaptable, and much less brittle.

## 6.2 Spatial Databases

ORM technologies and tools also support Spatial Databases and Spatial Objects. While many GIS platforms will easily support and prefer service level interactions, AASHTOWare customers may have Linear Referencing Systems based upon a spatial database. ORM libraries do interact with Spatial Databases and spatial objects, such that spatial interactions and manipulations can occur via mechanisms established and maintained in the persistence layer similar to any non-spatial database.

# 7. Separation of Concerns

Web Applications require a high level of application architecture abstraction to support commonly used design patterns such as "Model View Controller". The software engineering

design principal of SoC (Separation of Concerns) is generally accepted as the guiding principal for the Web Application Architecture described below. Both Technical and Application architectures must be adaptable, provide for scalability, availability (fault tolerant), and implement and support open API's for publishing and consuming services.

## 7.1 Web Application Architecture

The following example architecture distinguishes between a number of different functions that occur within a web application, and separates those functions into unique domains which are represented by separate layers within the diagram, and typically within the application build itself. The unique domains may also rely on specialized component libraries provided by technology providers.

Example of typical SoC abstraction construct (refer to diagram in *Figure 1*):

- Presentation (User Interface)
- Business
- Services
- Testing
- Persistence

Testing is represented as a unique layer, and commonly is maintained in the code repository. However when applications are packaged for deployment unit testing and other testing artifacts may be omitted as an option to consume fewer application server resources.

Security is also designed, implemented, and managed as a functionally abstracted element in modern web application architecture. Security impacts each functional element, or layer, of the architecture representation, and has been referred to by a number of technology providers as a cross-cutting architecture element.

## 7.2 Elements of Application Architecture Functionality Implemented as Unique Platforms and/or Products

Additional application architecture abstraction layers may be defined to support other elements of application functionality, such as reporting, business rules, business intelligence, or spatial/GIS capabilities. In many cases these functions are separated completely from the application and maintained within their own unique technical architecture (separate platforms). When specific application functionality is maintained as its own specialized platform, the capabilities of those platforms may be implemented by adoption of component libraries accessed within the different layers of the application, or via web services.

### 7.3 Generalized Web Application Architecture Model

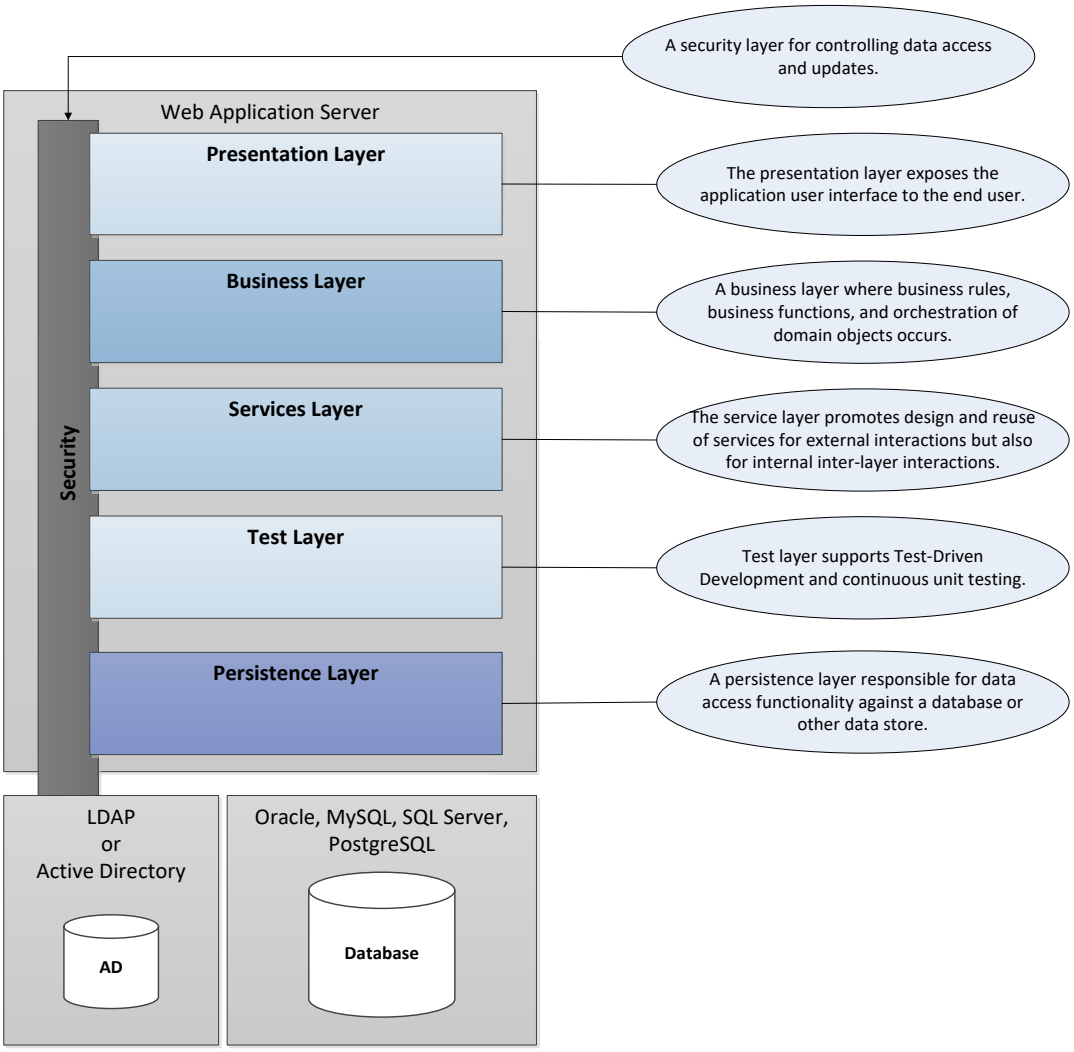



**Figure 1 - Generalized Representation of a Web Application Architecture**

# 8. Design Patterns

In the context of software engineering a design pattern is a generalized solution or strategy to a commonly occurring problem or design goal. A design pattern is not communicated or distributed as a finished set of software classes, objects, lines of source code or component libraries, but rather a template or pattern developers can use to solve a recognized problem or design goal. Design patterns are generally recognized as a best practice, and should be adopted as a standard design goal for AASHTOWare Web Applications.

One of the most common design patterns in use with modern web applications is the Model, View, Controller pattern, or MVC. There are many other patterns commonly used, but probably none is of greater significance than the MVC pattern when designing and deploying applications for the web. The MVC design pattern separates the internal representations of information from how that information is displayed to a user. Specifically:

- The Model represents the data as stored within the application database. The Model is how data is maintained and managed.

- The View is how that Model data is presented to a user via the presentation layer to interact with the application.
- The Controller handles the user interaction and input, and supports the communication between the View and the Model. Expressed alternatively, Controllers handle events that affect the Model or the View.

The MVC pattern allows developers to develop, test, and manage the user interface separate from the data model (database), as well as separate from the business logic. The MVC pattern makes it easier to test the application continuously, and also simplifies group development and also allows multiple development methodologies.

### 8.1 Software Design Pattern Retrospective

Design Patterns are a well understood concept by engineers and architects. Patterns for Software Engineering came into play with the advent of Object Oriented Programming. Once reusable software became a reality, software engineers embraced the common patterns that revealed themselves to accelerate development and improve the maintainability of code.

A number of books promoted design patterns for software development, but the most well recognized was published in 1994 and titled “Design Patterns: Elements of Reusable Object-Oriented software”. A Java (J2EE) design pattern book was published in 2003 which extended the previous discussion of patterns and best practices and made them relevant for web applications.

Microsoft has promoted similar architecture practices, and in their second guide, “Microsoft Application Architecture Guide 2nd edition, October 2009”, presents a web application architecture model consistent with this AASHTOWare guide. Chapter 10 of Microsoft’s guide lists the common patterns found in the software design pattern book listed above (1994).

### 8.2 MVC Pattern and Web Request Response Cycle

The MVC Pattern and web request-response cycle is represented in terms of Web Application Architecture Layers in *Figure 2* on the follow page.

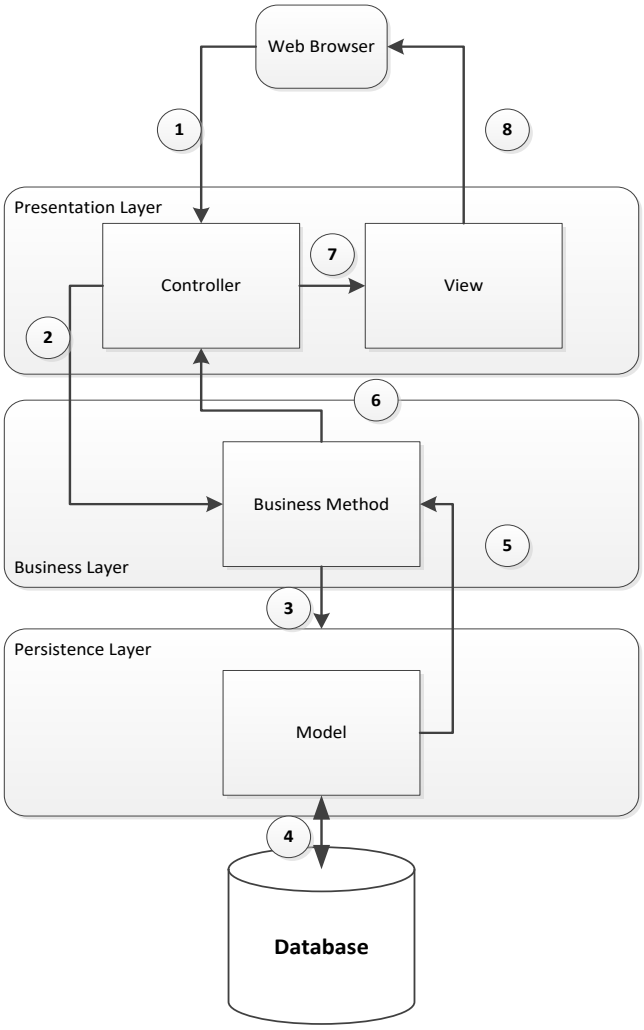



**Figure 2 - Web Request and Response Cycle**

# 9. Responsive Design

Web applications must support multiple device form factors. This is accomplished by making the application responsive. The presentation layer of the application architecture is responsible for the interaction of the application with the customer on a web browser. All elements of the user interface (UI) live within the presentation layer of the application. Responsive Design is important due to its ability to seamlessly support client devices of any form factor, such as smart phones, tablets, laptops, and desktop workstations. More to the point, Responsive Design is a technique for supporting multiple screen/web page layouts for an application simultaneously.

## 9.1 Responsive Technologies

Responsiveness depends on three technologies for current web applications. Those technologies are:

- HTML5
- CSS3
- JavaScript

### 9.2 Libraries and Templates

Responsive applications commonly adopt a set of tools and component libraries to simplify and standardize development and which are useful in accelerating development, and delivering a common look and feel to software products. For responsive user interfaces, technologists usually adopt a stylesheet template and JavaScript libraries that will provide the best user experience for their finished product.

- Example CSS3 templates: Bootstrap;
- Example JavaScript libraries: jQuery, AngularJS.

In order to promote alignment of AASHTOWare products adoption of a CSS3 template such as Bootstrap is encouraged. Adoption of a style sheet library promotes a common look and feel across the AASHTOWare product line, and promotes the branding efforts of the organization.

### 9.3 Open standards

The open standards browser markup language promoted by W3C (World Wide Web Consortium) and technology companies alike is HTML5. HTML5 is foundational to responsive user interfaces, and a key element in making web applications function on any web browser and on any device or operating system.

## 10. Business Rules

Many predecessor client-server applications, (or rich client applications), embedded business processing on the database. This model can inject an unwanted database dependency into the application. Additionally, in terms of the web application model, database execution of business rules will add latency and impact performance.

For web applications the optimal location for execution of business logic is in the business layer of the application on the web application server. This model also supports use of business logic by other layers of the architecture, such as the service layer.

## 11. Security

Web applications must have the capacity to sustain their customer's security policies, the customer's technical environment, and the customer's data. Web applications that accommodate authentication as well as authorization functionality have the potential of supporting the majority of customer security needs. Adopting a security architecture that sustains transaction security across all levels of the application architecture is fundamental, and supportable for all AASHTOWare products.

## 12. Spatial Integration

AASHTOWare designs delivers software products for transportation entities. As a matter necessity most if not all web application software from AASHTOWare should have the capability to integrate with spatial systems in use by customers such as:

- Linear Referencing Systems,
- GIS platforms, and

- Spatial databases.

Additionally, AASHTOWare web applications must have the ability to implement a variety of Mapping API's such as Esri or Google JavaScript Mapping API's. AASHTOWare Web applications must be able to perform the above independent of the technologies or platforms in use at a specific DOT's location.

### 12.1 Spatial Integrations

The web application architecture promoted by this guideline has the intrinsic capability to integrate and interact with spatial technologies, systems, and platforms.

- Service-level integration with a platform or GIS provider is accommodated by consuming or publishing services to interact with that given technology provider's products. Services would allow data and information sharing, displaying a map within the application with data from a service, or completing a transaction as examples.
- Including a JavaScript library from a Mapping API provider within the presentation layer of the application architecture supports a web application's ability to present data and information via a mapping presentation.
- The persistence layer and ORM technology promoted by this guideline allows a web application to map to a spatial database and interact with the spatial objects within the application.

### 12.2 Vendor Agnostic Spatial Capabilities

Spatial and mapping capabilities must be vendor agnostic within the web product, such that products from Google, Esri, Hexagon Geospatial (formerly Intergraph), and other technology vendors may be used that are unique to each AASHTO customer environment.

The application architecture model promoted by this guideline supports technology agnosticism, and continues to promote the use of OGC standards and standards-based approaches to delivering spatial capabilities.

## 13. Web-Oriented Architecture

Web-Oriented Architecture (WOA) is a substyle of service-oriented architecture (SOA) that leverages Web architecture. WOA focuses on Web services models, primarily RESTful modes of interoperability. These services are frequently exposed through an open API model.

WOA has been heavily used by major Web services providers such as Amazon and Google. WOA is actively being used by architects and developers for delivery of enterprise applications, although adopters don't readily identify the following interface elements as specific to just WOA. WOA currently emphasizes the following generic interface elements:

### 13.1 Universal Resource Identification (URI)

A common form of URI is the uniform resource locator or URL. URL's are referred to informally as an *internet (web) address.*

### 13.2 Hyper Text Transfer Protocol (HTTP).

HTTP is the common interaction mechanism or language between a web browser and a given web URL. Common HTTP commands are GET and POST.

### 13.3 Multipurpose Internet Messaging Extensions [MIME] types.

MIME is an Internet standard that extends the format of email to support:

- Text in character sets other than ASCII;
- Non-text attachments such as audio, video, images, etc. …,
- Message bodies with multiple parts, and
- Header information in non-ASCII character sets.

Virtually all human-written Internet email and a fairly large proportion of automated email is transmitted via SMTP in MIME format

### 13.4 Hypermedia as the engine of application state (Hyperlinks).

Hypermedia, used as an extension of the term hypertext, is a medium of information which includes graphics, audio, video, plain text and hyperlinks.

### 13.5 Application Neutrality

Application neutrality, which refers to an application being web browser and device independent, requires the implementation and use of open standards-based technologies and architectures. In simple terms, a web application should not require a specific web browser or device manufacturer to be useable.

# 14. Web Services

Web Services by their definition insulate a web application server and its hosted web applications from another system's technical dependencies. Services are the preferred mechanism for system to system communication, and provide maximum flexibility to both data providers and data consumers.

### 14.1 REST

Web applications are typically best served by light-weight REST-based services which do not interfere with a web application's functioning and are less likely to diminish performance. Representational State Transfer (REST) services are simple to implement for interaction with other platforms and systems that do not require a transaction state be maintained.

**REST** services were built upon HTTP (and use HTTP constructs) from inception, with the intent that they would be used for the Web. As an example, a REST client sends an HTTP request such as GET, PUT, or POST to a URI to perform a simple action. Web Browsers provide native support for HTTP, and are the ubiquitous client supporting REST-based transactions. Web apps and mobile apps relying on web browsers also use light weight JSON (JavaScript Object Notation), which is native to Java Script, making JSON the preferred "object" relied upon to consume RESTful services through a browser.

**REST/JSON**

- Smaller message size for improved mobile/web per performance
- Easier to consume in web and mobile applications

- Easier to consume for simple web services (simple Java Script)
- Stateless

### 14.2 SOAP

Back-end processes, some of which will require that transaction state be maintained, are best served by a SOAP-based service. SOAP refers to “Simple Object Access Protocol”. Web Applications and Web Application Servers can also support SOAP services, *but not for the purposes of supporting transactions with customers using the application on the web.*

**SOAP** Services expose method calls available to remote servers and respond with a SOAP response. SOAP responses have a (usually) larger payload, which is XML. By their nature SOAP-based services are more suited to handling internal server to server transactions of greater complexity. SOAP-based transactions have been designed to leverage additional infrastructure elements such as service busses, which can maintain service state and provide service orchestration.

**SOAP/XML**

- Larger message size
- Easier to consume for complex web services
- Service Orchestration (Service Bus)
- Stateful (via Service Bus)

## 15. Summary

This guideline presents an application architecture that enhances a software product’s sustainability and adaptability. The design philosophies also promote a web application’s ability to be useable on any web browser, and on multiple device form factors. In short, the guideline focuses on the strategy for AASHTOWare customers to use AASHTOWare web applications on whatever hardware and software that has been adopted in their respective organizations with the least technical issues.

# 16. Glossary

Application Architecture – Application architecture is the organizational design of a software application.

ASCII – American Standard Code for Information Interchange. A character encoding scheme.

Database – A technology used to store, maintain, and manage data to support application transactions. Database Platforms use a common language, SQL, to support the actions of *Create, Read, Update*, and *Delete*.

Geographical Information System (GIS) – A type of application that specializes in spatial data management and spatial analysis. GIS products also generate and display maps via web browsers, and allow the interaction with GIS servers and databases via mapping interfaces.

GUI – Graphical User Interface.

Hosting Platform – Another name for a web application server.

HTML – Hypertext Markup Language. HTML5 is the most recent W3C HTML standard, and is the language used to render a web page for display by a web browser. HTML5 extend HTML significantly and supports many new capabilities and features.

HTTP – Hypertext Transfer Protocol. Common HTTP commands are GET and POST.

Hyperlink - A hyperlink is a reference to data that the reader can directly follow either by clicking or by hovering. A hyperlink points to a whole document or to a specific element within a document.

Hypertext - Hypertext is text with hyperlinks.

Information Model – An XML data model used to define and support data exchange between applications.

Model-View-Controller – A design pattern that separates the user interface from other functional elements of the application. See Figure 2. As an example: The controller translates the user's interactions with the view into actions that the model will perform.

Persistence Layer – Data Access Layer (See Figure 1).

Responsive Design – The combined technologies of HTML5, CSS3, and JavaScript are used to implement/create a responsive design. Responsive Design (RD) loosely means any web site or web application that adapts itself to the device form factor in which it appears. RD relies upon display rules established within style sheets, used to create display templates for web pages. The current version of Cascading Style Sheets is version 3, and is referred to as CSS3, which is a World Wide Web Consortium (W3C) standard. Cascading style sheets (templates) are applied to HTML and when animated and enhanced with JavaScript methods become the responsive user interfaces seen in modern web applications.

Service Oriented Architecture - An architecture model that emphasizes and promotes interactions between applications and hosting platforms by web services.

SMTP – Simple Mail Transfer Protocol.

SQL – Structured Query Language. SQL consists of a data definition language, data manipulation language, and a data control language. SQL is an International Standards Organization (ISO) and an American National Standards Institute (ANSI) standard.

TCP/IP – Transmission Control Protocol/Internet Protocol. The basic network communication protocol in use on the internet.

Transaction State – Common transaction states are: Active, Partially Committed, Failed, Aborted, or Committed. A transaction is a unit of program execution that accesses and/or updates various data items. To maintain the integrity of an application's data a transaction's state must be maintained in certain execution circumstances.

Web Application – An application accessed and used via a web browser, and hosted on a Web Application Server.

Web Application Server – Web Application Servers host web applications. Examples of web application servers: Internet Information Server (IIS), Tomcat, JBoss, WebLogic, WebSphere.

Web Browser – A GUI web software client that is used to access internet-based resources (applications and web sites) using common protocols such as HTTP and TCP/IP. Examples of web browser are: Internet Explorer, Chrome, Firefox, Safari, Opera, and Mozilla.

Web Service – A web service is accessible via a web address, and provides a mechanism to share information to both internal and external applications and users. The web services most commonly implemented with web applications are REST (Representational State Transfer) services. REST services commonly communicate over HTTP.

AASHTO AASHTOWare™ SOLUTIONS

# WEB & MOBILE DATA EXCHANGE GUIDELINE

## S&G Number: 2.090.01.5G

## Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.2 | 6/12/2017 | Updates based on task force/contractor comments. | 8/01/2017 Approved by T&AA |
| 1.3 | 7/23/2018 | Minor grammar updates and updated URL. | 8/08/2018 Approved by T&AA |
| 1.4 | 10/18/2021 | Updated a broken link to a web page. | 10/22/2021 Approved by T&AA |
| 1.5 | 9/30/2023 | Added the Public API section, updated the AASHTOWare logo, and corrected broken links. | 10/02/2023 Approved by SCOA |

# Table of Contents

## Table of Figures

# 1. Introduction

The user of a web or mobile application expects (and requires) fast interactions between their apps/devices and all hosting platforms supporting them. Additionally, since contemporary mobile devices are architected to interact and rely on cloud-based server resources, a common model has emerged to support interaction with those platforms. A mobile user doesn't expect to manually create or initiate a data exchange; it occurs naturally without complex actions to manage the transaction.

In support of the above-stated user expectations, this document is intended to promote approaches and practices for developing web and mobile device data exchanges for AASHTOWare. Similar themes are promoted within this guideline consistent with the *Web Application Development Guideline and Architecture Goals.* From this point forward, this guideline will refer to the aforementioned guideline as the WADG document.

This document is intended to be general in nature with light coverage of technical content in the context of information technology (IT) and software development. The guideline's contents should be readable by task force members, committee members, project managers, other AASHTOWare stakeholders, and contractor staff.

# 2. Purpose

This guideline is intended to establish a high-level approach for contractors such that existing AASHTOWare software products evolve in a consistent and recognizable fashion. This will help promote sustainable data exchange models and methods for web and mobile applications within the product portfolio and also with non-AASHTOWare web or mobile products.

New products under consideration or just beginning development should adopt the strategies and recommendations of this guideline; the intent is to build and deliver new web and mobile products that can efficiently and effectively interact and also embrace open standards.

# 3. Web Application Characteristics Redux

In the prior *WADG*, the following characteristics were presented. Given the focus of this guideline, the same listing is also worth presenting since they provide context for data exchange discussions:

- Web apps rely on a commonly available client that is available on both mobile devices and workstations, namely, a web browser.
- A web browser will commonly run on a variety of operating systems and devices, which makes a web app both device and operating system independent.
- Because devices are of all shapes and sizes, web apps must responsively adapt to the device form factor so that the applications are useable no matter which form factor is in front of a user.
- Web apps rely on a hosting platform for the client devices and browsers to interact with, and web apps typically interact with multiple hosting platforms and client devices simultaneously via web services.
- Web apps typically display data and collect and store data, which may require a database platform (database management system, or DBMS, may be used interchangeably with database platform).
- Web apps may interact with multiple database platforms simultaneously.

- Web apps are commonly used anywhere and anytime.
- Web apps may need to be spatially aware and capable of displaying maps and spatially related information.
- Web apps may interact with client device features or devices attached to the client device.
- Web apps rely on web services to implement interfaces to other systems and to support transactions with platforms both inside and external to their hosting environment; REST (Representational State Transfer) services are the preferred web service model used to support web applications. SOAP (Simple Object Access Protocol) services can provide a stateful interface between systems but are not typically used for web-based transactions.

# 4. Data Exchanges in a Web and Mobile World

Typical business data exchanges between a web application user and the web hosting environment, a mobile web user and a hosting environment, or a mobile user on a mobile device and a hosting environment, occur via common protocols and rely entirely on connectivity. Interactions by necessity need to be lightweight and as fast as possible. The common language for web and mobile data exchanges is HTTP. The preferred lightweight data exchange mechanisms are REST web services and JSON (JavaScript Object Notation).

For the purposes of this guideline, peer-to-peer, or mobile device-to-mobile device data exchanges that occur without the benefit of a TCP/IP connection will be omitted. Examples of this type of exchange would be peer-to-peer exchanges via Bluetooth wireless connections. This type of connection might occur between a laptop or mobile computing device and survey, sample collection, or testing equipment.

## 4.1 Connectivity Expectations and Data Exchange Needs for Web and Mobile Products

The mobile workforce supporting transportation-related work depends on connectivity when in the field. If connectivity is unavailable, the expectation is that the device and application will allow work to proceed and simply connect when possible and complete transactions (data exchanges) without any extra effort or loss of productivity. Whether connectivity is available or not, adopting REST web services and using JSON provides a lighter and faster data exchange mechanism than SOAP or bulk file load models.

## 4.2 SOAP Web Services & XML-Based Data Exchanges

T&AA promotes that both SOAP and REST web service models have a place in AASHTOWare's current web ecosystem: (1) REST services and JSON for user/application-based transactions and mobile app/device interactions and (2) SOAP services and XML for "heavy lift" and back-end processes between systems. Information models such as NIEM (National Information Exchange Model) promote a common dialect and data structure. Furthermore, SOAP-based data exchanges using NIEM's XML definitions promote the opportunity for disparate systems to interact and share data.

# 5. Public APIs

Web APIs enable a broader data ecosystem and facilitate efficient data exchanges. They empower both internal and external development and help maintain the relevance of the originating application. Public APIs support diverse operation modes and integrations, particularly in open system architectures and digital transformation strategies. These integrations include web application interfaces, mobile apps, business intelligence, external

integrations, and automation tooling, all leveraging the functionality the public API provides. Developing public APIs involves traditional software development lifecycle practices and requires a combination of development and operations (DevOps) skills. It also incorporates utility services concepts formalized through data and operation contracts.

The Swagger project started as a tool to automate API documentation generation and code artifact generation for client software development kits. It was initially developed for a non-profit dictionary website project using JSON and REST principles. Additional features were added, such as schema validation and rule-based constraints, to enhance its functionality. As more APIs adopted RESTful architectures, Swagger became valuable in bridging the gap between structured and proprietary platforms and REST paradigms. The project gained popularity and attracted a community of users and contributors. Eventually, Swagger became the preferred RESTful API platform, surpassing competing projects. It was renamed as the OpenAPI Specification and placed under the stewardship of the OpenAPI Initiative. The founders of Swagger went on to establish SmartBear Software, which continued to develop tooling suites for the OpenAPI Specification.

Swagger was chosen for its efficient API documentation generation, code artifact generation for client software development kits, support for JSON and REST principles, schema validation capabilities, and the broad adoption and large community supporting it.

This section addresses achieving progressive API designs, extensions, integration methodologies, deployment, enhancement strategies, and API-Ops concerns (the extension of DevOps methodologies to API delivery). APIs have a lifecycle that involves managing the improvements and maintenance over time. The steps and methods in that progression are outlined below.

### 5.1 API-First Approach, API-Design Focus

The API-First and API-Design Focus results from the desired architecture benefits in a distributed API solution. An API-centric architecture allows flexibility, scalability, control, and adaptability within a solution while simultaneously achieving (API consumer) platform and channel indifference. The API solution supports a wide range of use scenarios that support multiple independent use cases and users rather than writing individual, custom APIs for each use case/requestor. The first architectural input is the formulation of the API goals that equate to end-result targets. The steps to achieve these goals are decomposed into specific objectives. Goals and objectives are design artifacts collaboratively produced, formally quantified, and rigorously documented. These artifacts later initiate the specifications used to define the API. This definition is central to all future architectural activities.

### 5.2 Source of Truth

In an API-First approach, the API serves as the primary source of truth for information. It provides a direct and reliable channel to access the accepted truth store. While other data resources may exist, the API stands out as a trustworthy representation of the truth store. Architectural considerations include data warehousing, transactional databases, data lakes, message busing, and external repositories. Data flows directly from the underlying provider to ensure the reliability of the source of truth. Relying on secondary system relays does not meet this design requirement. ***To sustain the source of truth, APIs must enforce product business logic and security rules.***

### 5.3 Simplicity, Uniformity, Predictability

Once goals, objectives, and sources of truth are defined, the API is organized to an optimal structure. The structure architecture accounts for the end-developer experience (actual use of the API). This architecture should be easy to understand, consistent with explicable conventions, predictable in operations, and compliant with commonly accepted tooling/automation resources. The architectural lifecycle artifacts at this phase are API guideline principles, concepts, rationale, abstractions, and foundations, which can later become definitive and enforceable rules.

### 5.4 RESTful Style Continuity

During the architectural lifecycle, API definition conventions are established as architectural style guidelines and enforced through validation rules. These guidelines provide verifiable policies and serve as an architectural artifact for validation. The OpenAPI Standard utilizes domains to structure and encapsulate foundational components in the API definition, enabling reuse. In the case of the AASHTOWare API definition, the API conventions are documented in the “AASHTOWare Components” domain and referenced within the definition itself.

### 5.5 Credentialing

The API design provides the definition of supported and expected methods of authentication. Incorporating credentialing or a representation of previously validated credentials in the data payload of every API request allows the credentials and subsequent authentication support to serve as a stamp to determine if the message requesting a resource from the API should be fulfilled. The credential assertion can be extremely simple, or provide higher degrees of functionality, depending on the mechanism used. This ranges from anonymous, in which no credentials are necessary, to a more formalized capability in which subsequent on-behalf-of relays are possible. As part of the open but not public mantra, the requestor still must provide some information to ensure sanctioned use, even with an anonymous (no credentials) request. That minimum proof is done with the inclusion of a subscription key. The combination of a subscription key and credentials suffice to perform prerequisite checks.

### 5.6 Discoverability

At initial presentation, discoverability may seem to be more of a service attribute than an architectural phase. However, discoverability includes developer experience concerns, target operating model facilitators, and implementation automation. The architectural task at this phase is to distribute, curate, contextualize, and organize API metadata for access by developers, tooling frameworks, and in some cases, robotic process automation configuration. Oversight of discoverability is a critical activity that directly impacts ease of adoption.

### 5.7 Decoupling of Concerns

The decomposition exercise is initiated by API definition, objective mapping, and reconciliation to truth stores. The decomposition exercise establishes dependency boundaries with the intent of *functional modularization*. The architectural lifecycle action here is to clearly separate concerns while facilitating functional, implementation, and operational interests. Establishing modular, single-responsibility, and practical architectural elements is a non-trivial exercise. Furthermore, the current operating model may introduce

challenges to achieving the ideal separation. This effort is nuanced, requiring deliberation and extensive review by technical domain experts. This architecture lifecycle phase produces a modularization strategy, quantifies options, describes tradeoffs, and concludes with a recommendation.

### 5.8 De-Monolithing

Once a modularization plan is in place, the next architectural task is to compose an optimal separation strategy for existing assets. When working with greenfield resources, the composition is straightforward. However, in the more common scenario where assets already exist, the composition effort becomes more complex due to the monolithic nature of those assets.

In this case, the architecture tasks involve identifying dependencies on proposed separations and locating isolated areas within the existing monolith. The objective is to break down the monolith into sustainable modules that can be maintained and updated independently. The architectural deliverable at this phase is an implementation overview organized by module, with the associated deconstructed abstractions from the existing monolith(s).

### 5.9 Microservices using PaaS (Platform-as-a-Service) Architecture

Once module implementations are outlined, the architectural task is to complete the implementation with a microservice approach. At this point, most of the system dependencies are clear and suitable for a microservice pattern. Each microservice will be independently deployable, testable, and scalable. Each microservice isolates its configuration, execution, operation, and instrumentation.

Based on the module's functional requirements, aspects are added to the microservice design. This will vary by microservice; however, all aspects of that microservice are intended to be self-contained. Coordination between shared resources (data stores, offloaded cache, etc.) will fall to the microservice containers or orchestration environment.

Microservices reduce a great deal of code complexity by deferring orchestration to an infrastructure layer. At this architectural phase, infrastructure orchestration is responsible for stewarding the executing microservices to achieve solution characteristics, such as fault tolerance, scalability, capacity, and instrumentation. How orchestration is achieved at the infrastructure layer is based on preference and governance policy. Microservices can execute through common containers like Kubernetes or platform-as-a-service (PaaS) strategies that provision elastic/auto-scaling resources.

At this architecture lifecycle phase, determinations of infrastructure orchestration techniques are made. On this architecture topic, there is limited risk to evolving over time (hence the benefit of the microservice approach).

The architecture deliverable at this point is the formal microservice description (modeling language diagram, configuration markup, and resource templates) per module.

### 5.10 Consistency, Availability, Partitioning

The consistency, availability, partitioning (CAP) theorem asserts that only two of the three listed properties can be optimally maintained within a distributed system. An API distributed system is no different. However, in the outlined API distributed system architecture, these properties take on different qualities against the CAP basis. Partitioning is reflected in the separation of concerns and microservice distribution. Consistency is even more nuanced

with strict isolation regarding truth stores. While the meaning of availability is the same in this case, an API system, specifically a RESTful API, reclassifies availability against web conventions.

The lifecycle task here is to prioritize which two of the three CAP properties are most consequential to the architecture and then engineer accordingly. Microservice scale-out approaches will benefit both availability and partitioning. The introduction of a distributed cache can dramatically improve availability but at the cost of consistency. Given the asynchronous and stateless mechanisms at play, most scenarios will prioritize availability and partitioning simply because the consistency compromise is unavoidable. The RESTful underpinnings again increase the options to increase availability using protocol-level directives.

# 6. Web Services

Web services by their definition insulate a web application server and its hosted web applications from another system's technical dependencies. Services are the preferred mechanism for system-to-system communication and provide maximum flexibility to both data providers and data consumers.

## 6.1 REST

Web applications are typically best served by lightweight REST-based services, which do not interfere with a web application's functioning and are less likely to diminish performance. REST services are simple to implement for interaction with other platforms and systems that do not require that a transaction state be maintained.

**REST** services were built on HTTP (and use HTTP constructs) from inception, with the intent that they would be used for the Web. As an example, a REST client sends an HTTP request such as GET, PUT, or POST to a URI to perform a simple action. Web browsers provide native support for HTTP, and are the ubiquitous client supporting REST-based transactions. Web apps and mobile apps relying on web browsers also use lightweight JSON, which is native to JavaScript, making JSON the preferred object relied on to consume RESTful services through a browser.

**REST/JSON:** Smaller message size for improved mobile/web performance; easier to consume in web and mobile applications, and stateless

REST Service Summary:

- Contemporary web services rely primarily on RESTful modes of interoperability.
- REST services are frequently exposed through an open API model.
- REST services use HTTP methods explicitly, are stateless, expose URIs, and can transfer JSON, XHTML, XML, or other common MIME types.

A REST service is:

- Platform-independent,
- Language-independent, and
- Standards-based (relies on HTTP).

### 6.2 SOAP

Back-end processes, some of which will require that transaction state be maintained, are best served by a SOAP-based service. Web applications and Web application servers can also support SOAP services *but not for the purposes of supporting transactions with customers using the application on the web.*

**SOAP** services expose method calls available to remote servers and respond with a SOAP response. SOAP responses have a (usually) larger payload, which is XML. By their nature, SOAP-based services are more suited to handling internal server to server transactions of greater complexity. SOAP-based transactions have been designed to leverage additional infrastructure elements such as service busses, which can maintain service state and provide service orchestration.

**SOAP/XML:** Larger message size, may be a more robust alternative for complex web services supporting complex interfaces; service orchestration and statefullness via service bus.

A SOAP/XML service is also:

- Platform-independent,
- Language-independent, and
- Standards-based (relies on SOAP/XML).

## 7. Web Service Description

Web services have proven to be one of the most effective ways to support a web-based transaction between a web browser user and a web application. This guideline is promoting the REST services model. The following is a short summary of how a REST service is described/characterised versus how a SOAP service is described.

### 7.1 REST Web Service Description

REST services usually support one business function. Also, REST services rely on HTTP calls such as GET and POST. Therefore, REST services do not need a complex web service description language like SOAP services. Instead REST services have adopted a lighter model referred to Web Access Description Language, or WADL.
*https://www.w3.org/submissions/wadl/*

WADL is a machine-readable *XML* description of *HTTP*-based *web services*. WADL models the resources provided by a service and the relationships between them. WADL is intended to simplify the reuse of web services that are based on the existing HTTP architecture of the Web. WADL is platform and language independent and aims to promote reuse of applications beyond the basic use in a web browser.
*https://en.wikipedia.org/wiki/Web_Application_Description_Language*

### 7.2 SOAP Web Service Description

SOAP services can and do support multiple business functions, or actions. By extension, SOAP services typically provide access to multiple methods and method calls per service. The Web Service Description Language, or WSDL, provides a specification to describe SOAP services. A WSDL is somewhat similar to a “method signature” in a programming

language. *SOAP services do not rely on web browsers or HTTP. SOAP services do require a web application server.*

WSDLs are an *XML*-based *interface definition language* that is used for describing the functionality offered by a SOAP-based *web service*. The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.
*https://en.wikipedia.org/wiki/Web_Services_Description_Language*

### 7.3 REST API vs Universal Description and Discovery (UDDI)

REST APIs provide a model to publish, via the Web, how a web application (which has developed and published REST web services), or web client, can perform data exchanges using HTTP. APIs (application programming interfaces) have evolved with the growth of web applications. REST APIs may be published in a traditional API model or published directly to the Web to promote easy communication between web applications and between web applications and web clients. Mobile computing and the explosion of social media, cloud-based applications, and services, have elevated REST APIs to a higher significance than traditional API models or UDDI.

# 8. REST APIs

HTTP is not just for serving up HTML pages. It is also a powerful platform for building web APIs, using a handful of verbs (GET, POST, and so forth) plus a few simple concepts such as URIs and headers (*https://learn.microsoft.com/en-us/aspnet/web-api/overview/older-versions/build-restful-apis-with-aspnet-web-api*).

REST API Examples:

Paypal - *https://developer.paypal.com/api/rest/*

Gmail - *https://developers.google.com/gmail/api/reference/rest*

Google Maps - *https://developers.google.com/maps/documentation/*

ArcGIS (Server) - *https://developers.arcgis.com/rest/*

Amazon (AWS) Lambda - *https://docs.aws.amazon.com/lambda/latest/dg/API_Operations.html*

AASHTOWare contractors responsible for developing, enhancing, and maintaining AASHTOWare products will enhance the value of AASHTOWare web-based products to AASHTOWare and its consumers by implementing and publishing REST APIs. AASHTOWare web software could then more easily be supported within customer environments, integrated with third-party products, and integrated with other AASHTOWare (web) products deployed by any customer.

**REST API examples:**

### 8.1 Rest API Example (Google Maps Directions API)

Google Maps Directions API through an HTTP interface (*https://developers.google.com/maps/documentation/directions/start*):

The following example requests the driving directions from Disneyland to Universal Studios Hollywood, in JSON format:

*https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studios+Hollywood4&key=YOUR_API_KEY*

*A sample response, in JSON:*


```
{
   "geocoded_waypoints" : [
      {
         "geocoder_status" : "OK",
         "place_id" : "ChIJRVY_etDX3IARGYLVpoq7f68",
         "types" : [
            "bus_station",
            "transit_station",
            "point_of_interest",
            "establishment"
         ]
      },
      {
         "geocoder_status" : "OK",
         "partial_match" : true,
         "place_id" : "ChIJp2Mn4E2-woARQS2FILlxUzk",
         "types" : [ "route" ]
      }
   ],
   "routes" : [
      {
         "bounds" : {
            "northeast" : {
               "lat" : 34.1330949,
               "lng" : -117.9143879
            },
            "southwest" : {
               "lat" : 33.8068768,
               "lng" : -118.3527671
            }
         },
         "copyrights" : "Map data ©2016 Google",
         "legs" : [
            {
               "distance" : {
                  "text" : "35.9 mi",
                  "value" : 57824
               },
               "duration" : {
                  "text" : "51 mins",
```

```
                  "value" : 3062
               },
               "end_address" : "Universal Studios Blvd, Los Angeles, CA
90068, USA",
               "end_location" : {
                  "lat" : 34.1330949,
                  "lng" : -118.3524442
               },
               "start_address" : "Disneyland (Harbor Blvd.), S Harbor
Blvd, Anaheim, CA 92802, USA",
               "start_location" : {
                  "lat" : 33.8098177,
                  "lng" : -117.9154353
               },
  ... Additional results truncated in this example[] ...
…
         },
         "summary" : "I-5 N and US-101 N",
         "warnings" : [],
         "waypoint_order" : []
      }
   ],
   "status" : "OK"
}
```


### 8.2 Example REST API Specification Example Template

Example REST API specifications (*https://www.docuwriter.ai/posts/api-documentation-template-example*).

*Doc Template - *https://docs.google.com/document/d/1HSQ3Fe77hnthw8hizqvXJU-qGEPHavMkctvCCadkVbY/edit?pli=1#* .

**API Specification Doc** (simplified excerpt from above Specification Link*).

| Version | Date | Author | Description |
|---|---|---|---|
| | | | |

**Index**



**Methods**

### #. get updates

Get the new updates

**Request**

| Method | URL |
|---|---|
| POST | api/updates/ |

| Type | Params | Values |
|---|---|---|
| HEAD<br>POST | auth_key<br>version | String<br>number |

**auth_key**

The auth_key that was given in response to /api/login

**version**

The current version of an internal recipe database. Each time when updates are pulled from the server through the web service, the internal database version is incremented.

**Response**

| Status | Response |
|---|---|
| 200 | Response will be an object containing a list (array) as well as the updated database. Each item in the array has the following structure. (Example from a 'Recipe' datastore.)<br><br>{<br>"recipe_id": 10,<br>"title": "Green Chilly Salad",<br>"category": 1,<br>"ingredients": {<br>"Green Chilly": "1 kg",<br>"Salt": "0.5 tbsp"<br>},<br>"steps": [<br>"First clean and cut the chillies",<br>"Now you can eat."<br>], |

| | |
|---|---|
| | <pre>"remarks": "serves 2 people"<br>}<br><br>An example response is:-<br>{<br>"recipes": [<br>{<br>"recipe_id": 10,<br>"title": "Green Leaf Curry",<br>"category": 1,<br>"ingredients": {<br>"Green leaf": "1 kg",<br>"Salt": "0.5 tbsp"<br>},<br>"steps": [<br>"First clean and cut the leaves",<br>"Now you can eat."<br>],<br>"remarks": "serves 2 people"<br>}<br>],<br>"version": "4"<br>}</pre> |
| 400 | <pre>{"error":"Please specify database version."}</pre> |
| 400 | <pre>{"error":"Invalid database version."}</pre> |
| 401 | <pre>{"error":"Invalid API key."}</pre> |
| 500 | <pre>{"error":"Something went wrong.  Please try again later."}</pre> |

# 9. Securing REST Services

OAuth has been the defacto standard for authentication between web applications and web sites. Many large technology companies (e.g. Google, FaceBook, Twitter, etc.) rely on OAuth to authenticate against their APIs (*https://oauth.net/*). The authentication this guideline is presenting is OAuth version 2. “The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service. “

## 9.1 OAuth2

OAuth2 sounds like an evolution of OAuth1, but it is a completely different take on authentication that attempts to reduce complexity. OAuth2’s current specification removes signatures, so you no longer need cryptographic algorithms to create, generate, and validate signatures. All the encryption is now handled by TLS, which is required.

OAuth2 does have some detractors that promote a fallback to OAuth1 or alternative approaches. However, even detractors direct OAuth implementers to Facebook (*https://developers.facebook.com/docs/facebook-login/security*), Google, Twitter, and Amazon to see how OAuth2 should be implemented.

### 9.2 Examples from Google OAuth2 Processes

*https://developers.google.com/identity/protocols/oauth2*

Google's simple representation of the OAuth 2 sequence to access a Google REST API endpoint is below. For details, see "Using OAuth 2.0 for Web Server Applications" (*https://developers.google.com/identity/protocols/oauth2/web-server*).

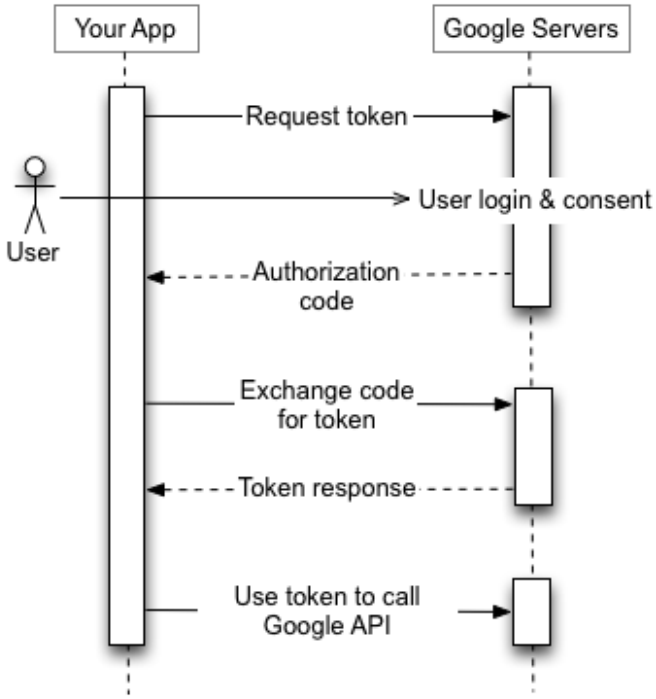



**Figure 1 - Using OAuth 2.0 for Web Server Applications**

### 9.3 Client-side (JavaScript) applications

"The Google OAuth 2.0 endpoint supports JavaScript applications that run in a browser. The authorization sequence begins when your application redirects a browser to a Google URL; the URL includes query parameters that indicate the type of access being requested. Google handles the user authentication, session selection, and user consent. The result is an access token, which the client should validate before including it in a Google API request. When the token expires, the application repeats the process." For details, see "Using OAuth 2.0 for Client-side Applications" (*https://developers.google.com/identity/protocols/oauth2/javascript-implicit-flow*).

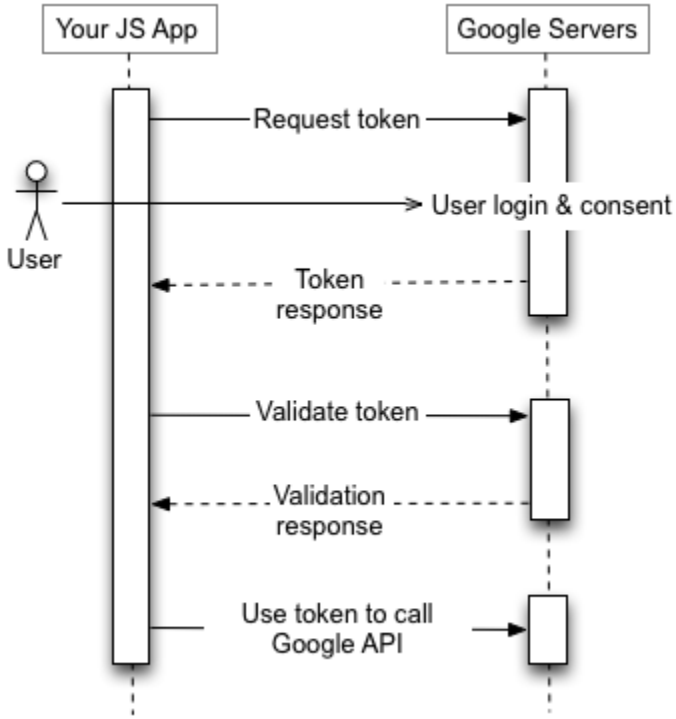



**Figure 2 - Using OAuth 2.0 for Client-side Applications.**

### 9.3.1 Service accounts

Some APIs can act on behalf of an application without accessing information. In these situations, the application needs to prove its own identity to the API, but no user consent is necessary. Similarly, in enterprise scenarios, your application can request delegated access to some resources. For these types of server-to-server interactions a **service account** is required, which is an account that belongs to your application instead of to an individual end-user. Your application calls server APIs on behalf of the service account, and user consent is not required. For details, see the service-account documentation at *https://developers.google.com/identity/protocols/oauth2/service-account*.

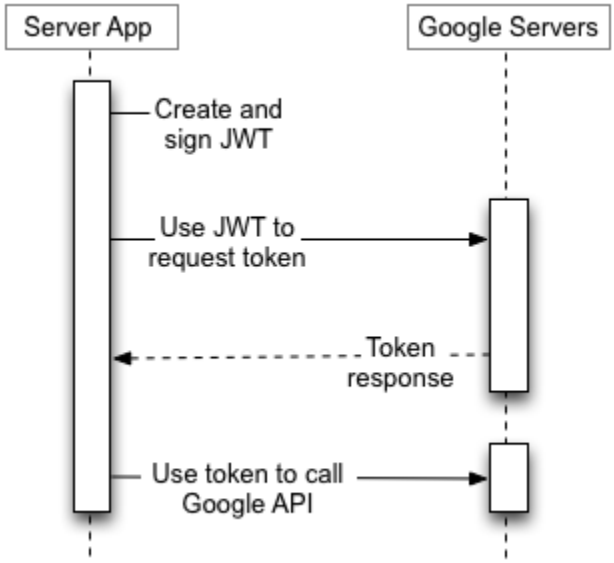



**Figure 3 - Using OAuth 2 for service-accounts.**

# 10. An Introduction to JavaScript Object Notation (JSON)

From json.org (*http://json.org/* ): "**JSON** (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the *JavaScript Programming Language*, *Standard ECMA-262 3rd Edition - December 1999*. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language."

JSON objects can be a collection of name/value pairs, or an ordered list of values.

- Collections of name/value pairs are realized as objects, records, hash tables, keyed lists, or associative arrays.
- Ordered lists of values in most are realized as arrays, vectors, lists, or sequences.

Most modern programming languages support the above data structures in one form or another. JSON is simply a JavaScript manifestation of these universal data structures used to support data exchanges.

Responsive web applications that rely on JavaScript are better able to manipulate JSON objects for data exchanges than an XML payload within a SOAP envelope. Data formatted according to the JSON standard is easily and very quickly parsed, which is a requirement for mobile and web application users. JSON is platform independent which makes it an ideal data exchange medium for web and mobile applications.

Microsoft has been a proponent of JSON for web application data exchanges for the last decade. MSDN provided a primer on JSON in 2007 (*https://learn.microsoft.com/en-us/previous-versions/dotnet/articles/bb299886(v=msdn.10)* ), and also a comparison between JSON and XML.

In summary, and paraphrasing from W3C (*https://www.w3schools.com/js/js_json_intro.asp*) JSON is:

- JSON is a lightweight data exchange format that is fast to execute.
- JSON is language independent (JSON uses JavaScript syntax, but the JSON format is text only; text can be read and used as a data format by any programming language.)
- JSON is "self-describing" and easy to understand.
- JSON is syntactically identical to the code for creating JavaScript objects. Therefore, a web application using JavaScript can convert JSON into native JavaScript objects.
- JSON is easier to use than XML for web-based (data exchanges) transactions.

From JSON.ORG - JSON objects take on these forms:

### 10.1 JSON Set of Name/Value Pairs

An *object* is an unordered set of name/value pairs. An object begins with `{`(left brace) and ends with`}` (right brace). Each name is followed by`:` (colon) and the name/value pairs are separated by`,` (comma).

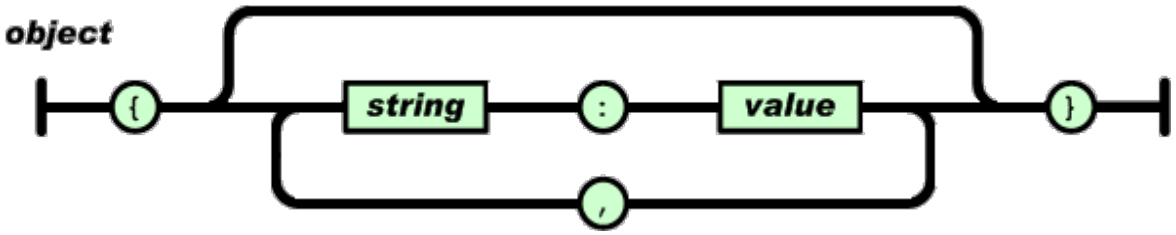



**Figure 4 - JSON Set of Name/Value Pairs**

### 10.2 JSON Array

An *array* is an ordered collection of values. An array begins with `[`(left bracket) and ends with] (right bracket). Values are separated by, (comma).

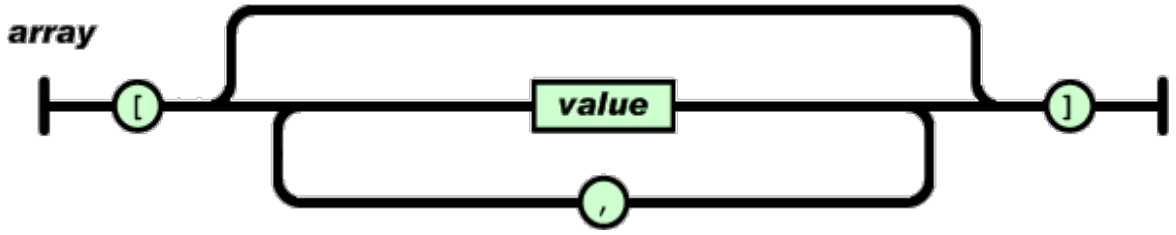



**Figure 5 - JSON Array**

### 10.3 JSON Value

A *value* can be a *string* in double quotes, or a *number*, or `true` or `false` or `null`, or an *object* or an *array*. These structures can be nested.

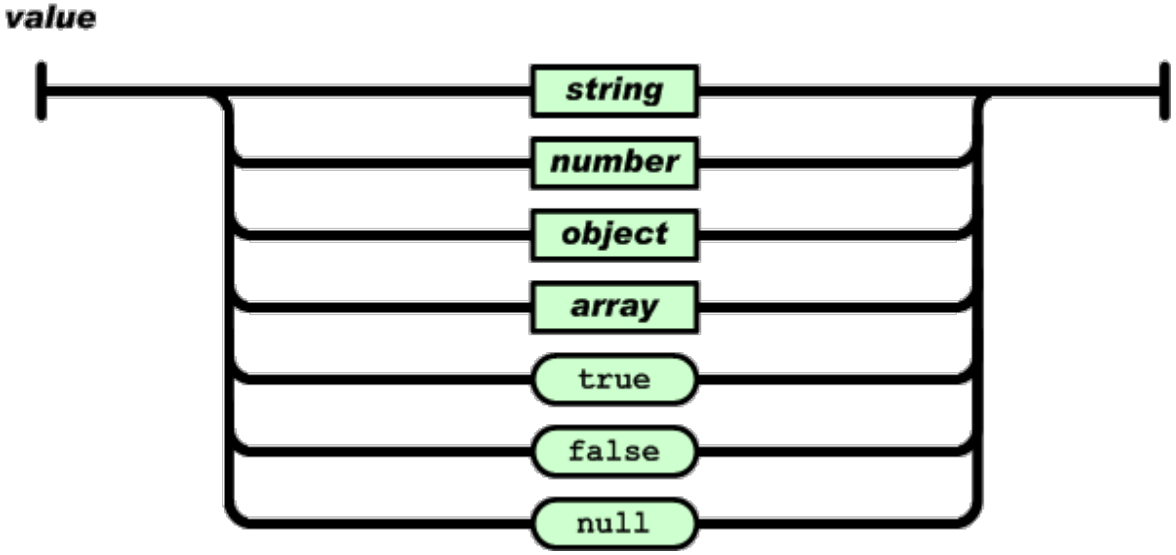



**Figure 6 - JSON Value**

### 10.4 JSON String

A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.

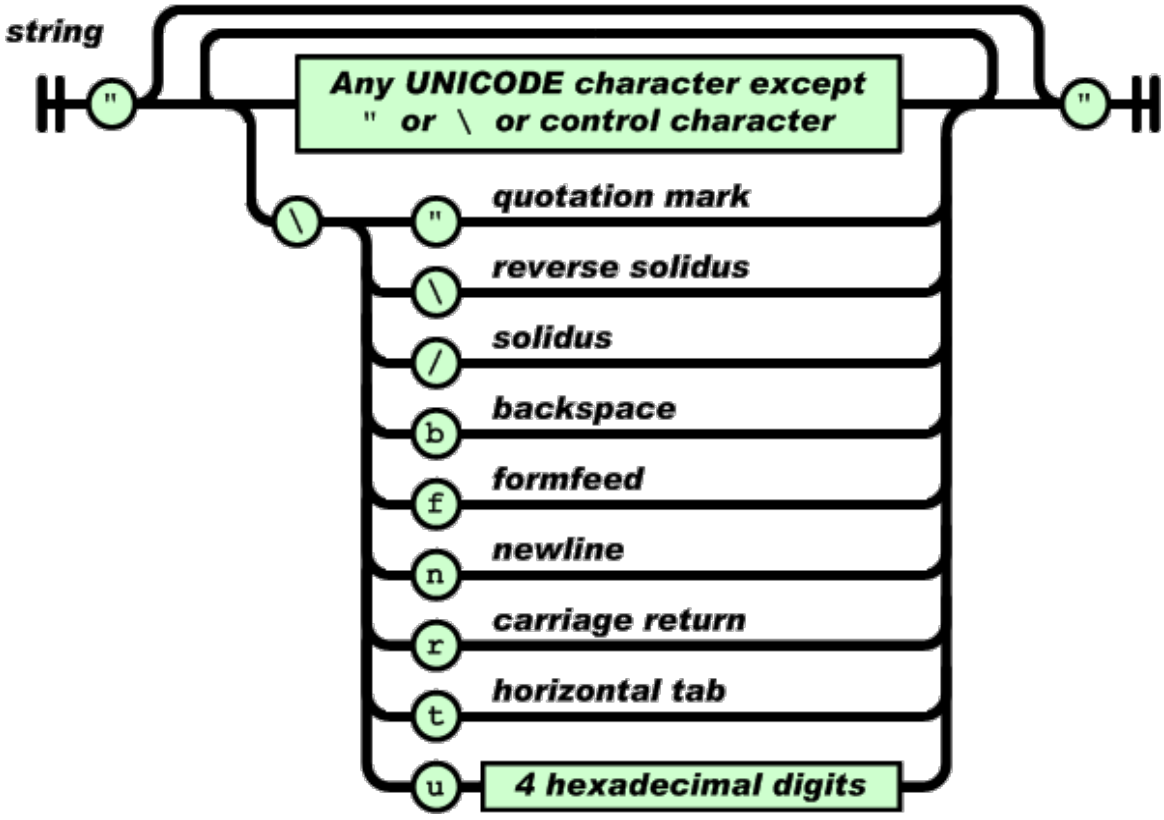


**Figure 7 - JSON String**

### 10.5 JSON Number

A number is very much like a C or Java number, except that the octal and hexadecimal formats are not used.

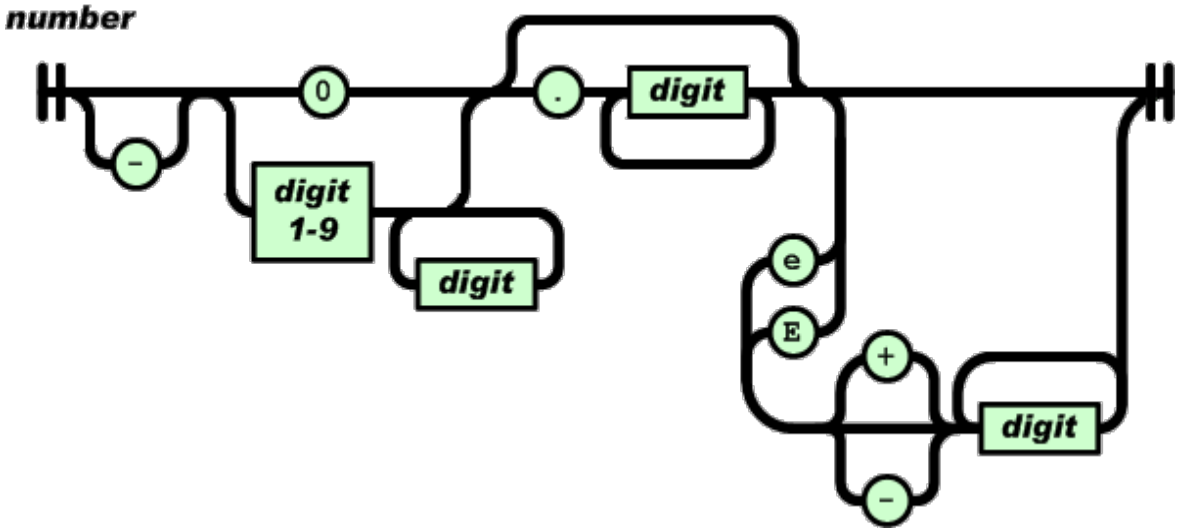



**Figure 8 - JSON Number**

# 11. Open Standards & Web/Mobile Interactions

This guideline promotes the adoption and use of open standards. Open standards should be supported to the greatest extent practicable and adopted widely for AASHTOWare web applications and other AASHTOWare products. Open standards are generally recognized as a key factor in a product's ability to integrate efficiently, and interact easily, with other software products, software libraries (component libraries), and other specialized systems. Web applications are inherently flawed as built if they don't support open standards, which limits their useful life, increases their cost for support, and makes them less likely to work on any device, operating system (OS), or browser.

### 11.1 Data, Interfaces, Services

Open standards help to support a product's ability to communicate with other data consumers and data providers, and to support efficient development of interfaces with systems. Open standards provide the foundation for web services, and web service models such as SOAP and REST.

Service Oriented Architecture (SOA) relies on a foundation established by open standards. Open standards provide the underlying mechanisms that allow dissimilar, heterogeneous IT environments to communicate with each other, effectively allowing transparent sharing of information and services.

Information models such as NIEM promote efficient exchange of data that reduces the need for data to be transformed or translated between business systems. Information models are based on open standards technologies, and they further extend and promote open standards related to data, interfaces, and services.

### 11.2 OS and Device Independence

Web apps that adopt open standards (such as HTML5) accommodate an application's need to run on a variety of web browsers and on any device. Open standards support a web app's ability to be substantially operating system and device independent.

### 11.3 Open Spatial Standards

Open standards such as the Open Geospatial Consortium (OGC) Standards support the definition and use of spatial web service models that allow a software product to readily interact with a spatial data service or GIS platform. It also supports application independence from a single vendor and cross-vendor communication. This allows customers to use one vendor's solution while consuming data hosted by a different vendor.

### 11.4 Open Communication and Networking Standards

Open standards support the functional specialization of network and system platforms to support networking, security, application hosting, and virtually all functional elements that are the basis of web applications.

# 12. Web Browser Agnostic

Perhaps the greatest value of web applications is their ability to use a common client, a web browser, which resides on virtually every modern-day device used for business productivity or personal use. This includes modern smart phones, tablets, laptops, desktop computers, and even televisions, stereos, vehicles; the list goes on and on.

For a web application to be of greatest use it should be able to function on any of the web browsers commonly used by AASHTOWare customers. In other words, the application should be browser agnostic, and provide all its capabilities to the user whether that user prefers Internet Explorer, Firefox, Chrome, Safari, or any future open standards based web browser.

Essentially, if a web application is web browser agnostic, it has also made itself device and operating system agnostic. A web application with this characteristic will work on a Windows desktop, a Linux server, an Apple laptop, an Android, or Apple phone, and so on.

# 13. Spatial Integration

AASHTOWare designs delivers software products for transportation entities. As a matter necessity, most if not all web application software from AASHTOWare should have the capability to integrate with spatial systems in use by customers such as:

- Linear Referencing Systems,
- GIS platforms, and
- Spatial databases.

Additionally, AASHTOWare web applications must have the ability to implement a variety of mapping APIs such as Esri or Google JavaScript mapping APIs. AASHTOWare web applications must be able to perform the above independent of the technologies or platforms in use at a specific DOT.

### 13.1 Spatial Integrations

The web application architecture promoted by this guideline has the intrinsic capability to integrate and interact with spatial technologies, systems, and platforms.

- Service-level integration with a platform or GIS provider is accommodated by consuming or publishing services to interact with that given technology provider's products. Services would allow data and information sharing, displaying a map within the application with data from a service, or completing a transaction as examples.
- Including a JavaScript library from a mapping API provider within the presentation layer of the application architecture supports a web application's ability to present data and information via a mapping presentation.
- The persistence layer and object-relational mapping technology promoted by this guideline allows a web application to map to a spatial database (or multiple databases) and interact with the spatial objects within the application.

### 13.2 Vendor Agnostic Spatial Capabilities

Spatial and mapping capabilities must be flexible enough to be vendor agnostic within the web product, such that products from Google, Esri, Hexagon, and other technology vendors may be used that are unique to each AASHTO customer environment.

The application architecture model promoted by this guideline supports technology agnosticism and continues to promote the use of OGC standards and standards-based approaches to delivering spatial capabilities.

# 14. Web-Oriented Architecture

Web-oriented architecture (WOA) is a substyle of service-oriented architecture (SOA) that leverages web architecture. WOA focuses on web services models, primarily RESTful modes of interoperability. These services are frequently exposed through an open API model.

WOA has been heavily used by major web services providers such as Amazon and Google. WOA is actively being used by architects and developers for delivery of enterprise applications, although adopters do not readily identify the following interface elements as specific to just WOA. WOA currently emphasizes the following generic interface elements:

### 14.1 Universal Resource Identification (URI)

A common form of URI is the uniform resource locator or URL. URLs are referred to informally as an *internet (web) address*.

### 14.2 Hyper Text Transfer Protocol (HTTP).

HTTP is the common interaction mechanism or language between a web browser and a given web URL. Common HTTP commands are GET and POST.

### 14.3 Multipurpose Internet Messaging Extensions [MIME] types.

MIME is an Internet standard that extends the format of email to support:

- Text in character sets other than ASCII;
- Non-text attachments such as audio, video, images, etc;
- Message bodies with multiple parts; and
- Header information in non-ASCII character sets.

Virtually all human-written Internet email and a fairly large proportion of automated email are transmitted via SMTP in MIME format

### 14.4 Hypermedia as the engine of application state (Hyperlinks).

Hypermedia, used as an extension of the term hypertext, is a medium of information which includes graphics, audio, video, plain text, and hyperlinks.

### 14.5 Application Neutrality

Application neutrality, which refers to an application being web browser and device independent, requires the implementation and use of open standards-based technologies and architectures. In simple terms, a web application should not require a specific web browser or device manufacturer to be useable.

# 15. Summary

As presented previously, contemporary web services for modern web applications rely primarily on RESTful modes of interoperability. REST services are frequently exposed through an open API model.

A REST service is:

- Platform-independent,
- Language-independent, and
- Standards-based (relies on HTTP).

AASHTOWare web products should implement REST services, publish REST APIs, and prefer JSON over XML for web-based client transactions with web application servers and web-based platforms.

# 16. Glossary

Database – A technology used to store, maintain, and manage data to support application transactions. Database Platforms use a common language, SQL, to support the actions of *Create, Read, Update*, and *Delete*.

GIS - Geographic Information System. A type of application that specializes in spatial data management and spatial analysis. GIS products also generate and display maps via web browsers and allow the interaction with GIS servers and databases via mapping interfaces.

GUI – Graphical User Interface. The GUI is the visualization that the user sees and interacts with on a computing device. It often contains windows, tools, buttons, menus, etc. that the user engages to navigate and focus on information the user wants to find and understand.

HMAC - Hash Message Authentication Code. HMAC is a specific type of *message authentication code* (MAC) involving a *cryptographic hash function* and a secret *cryptographic key.*

Hosting Platform – Another name for a web application server.

HTML – HyperText Markup Language. HTML5 is the most recent W3C HTML standard, and is the language used to render a web page for display by a web browser. HTML5 extends HTML significantly and supports many new capabilities and features, such as location awareness.

HTTP – Hyper Text Transfer Protocol. Common HTTP commands are GET and POST.

Hyperlink - A hyperlink is a reference to data that the reader can directly follow either by clicking or by hovering. A hyperlink points to a whole document or to a specific element within a document.

Hypertext - Hypertext is text with hyperlinks.

Information Model – An XML data model used to define and support data exchange between applications.

JSON – JavaScript Object Notation (*https://www.ecma-international.org/publications-and-standards/standards/ecma-404/* ). The fat-free alternative to XML (*http://www.json.org/fatfree.html*).

JWT – JSON Web Token. JSON Web Token is an open standard (*RFC 7519*) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (HMAC algorithm) or a public/private key pair using RSA.

MIME - MIME stands for Multipurpose Internet Mail Extensions. MIME type examples - JSON, XHTML, XML.

OAuth – OAuth is an *open standard* for *authorization*. OAuth provides to clients a "secure delegated access" to server resources on behalf of a resource owner. (*https://en.wikipedia.org/wiki/OAuth*)

Responsive Design – The combined technologies of HTML5, CSS3, and Javascript are used to implement/create a responsive design. Responsive Design (RD) loosely means any web site or web application that adapts itself to the device form factor in which it appears. RD relies on display rules established within style sheets, used to create display templates for web pages. The current version of Cascading Style Sheets is version 3, and is referred to as CSS3, which is a World Wide Web Consortium (W3C) standard. Cascading style sheets (templates) are applied to HTML and when animated and enhanced with JavaScript methods become the responsive user interfaces seen in modern web applications.

REST – Rest stands for representational state transfer and is a way to allow web applications and computer systems to interoperate over the Internet.

RSA – A cryptographic system developed by Rivest, Shamir, and Adleman. RSA is widely used for secure data transmission. In such a *cryptosystem*, the *encryption key* is public and differs from the *decryption key* which is kept secret. In RSA, this asymmetry is based on the practical difficulty of *factoring* the product of two large *prime numbers*, the *factoring problem*. (*https://en.wikipedia.org/wiki/RSA_(cryptosystem)* )

SOA – Service Oriented Architecture. An architecture model that emphasizes and promotes interactions between applications and hosting platforms by web services.

SOAP – Simple Object Access Protocol. SOAP is a protocol that supports exchanging information between web applications and web services.

SMTP – Simple Mail Transfer Protocol.

SQL – Structured Query Language. SQL consists of a data definition language, data manipulation language, and a data control language. SQL is an International Standards Organization (ISO) and an American National Standards Institute (ANSI) standard.

TCP/IP – Transmission Control Protocol/Internet Protocol. The basic network communication protocol in use on the internet.

TLS - Transport Layer Security and its predecessor, Secure Sockets Layer (SSL), both frequently referred to as SSL, are *cryptographic protocols* that provide *communications security* over a *computer network*, and/or over the internet. (*https://en.wikipedia.org/wiki/Transport_Layer_Security*)

Stateful – A web service is defined as stateful if during execution its transaction state is maintained. REST services are typically not stateful. Stated another way, REST services are typically stateless.

Transaction State – Common transaction states are: Active, Partially Committed, Failed, Aborted, or Committed. A transaction is a unit of program execution that accesses and/or updates various data items. To maintain the integrity of an application's data, a transaction's state must be maintained in certain execution circumstances.

UDDI - Universal Description, Discovery, and Integration: An XML-based registry business and government entities publish and expose on the web. Its ultimate goal is to streamline online transactions by enabling companies to find one another on the Web, support interoperability, and interface development.

URI – Universal Resource Identifier. A common type of URI is a universal resource locator (URL). URLs are referred to informally as internet (web) addresses.

Web Application – An application accessed and used via a web browser and hosted on a Web Application Server.

Web Application Server – Web Application Servers host web applications. Examples of web application servers are: Internet Information Server (IIS), Tomcat, JBoss, WebLogic, WebSphere.

Web Browser – A GUI web software client that is used to access internet-based resources (applications and web sites) using common protocols such as HTTP and TCP/IP. Examples of web browser are: Internet Explorer, Chrome, Firefox, Safari, Opera, and Mozilla.

Web Service – A web service is accessible via a web address, and provides a mechanism to share information to both internal and external applications and users. The web services most commonly implemented with web applications are REST (Representational State Transfer) services. REST services commonly communicate over HTTP.

WSDL - Web Services Description Language: An *XML*-based *interface definition language* that is used for describing the functionality offered by a *web service*. The acronym is also used for any specific WSDL description of a web service (also referred to as a *WSDL file*), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. Therefore, its purpose is roughly similar to that of a *method signature* in a programming language. (Ref: *https://en.wikipedia.org/wiki/Web_Services_Description_Language* )

This page is intentionally blank.

# HOSTED SERVICES STANDARD

## S&G Number: 2.095.01.5S

## Effective Date: April 1, 2026

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.3 | 7/26/2023 | Updated the AASHTOWare logo. | 10/02/2023 Updated by SCOA |
| 1.4 | 9/28/2024 | Added sections for infrastructure location and security compliance reporting and made other updates. | 12/11/2024 Updated by SCOA |
| 1.5 | 2/18/2026 | Added requirements for minimum necessary resources and a minimum of two environments. Removed bold from new requirements introduced in the last update. | 3/12/2026 Approved by SCOA |

# Table of Contents

# 1. Purpose

This document provides details for Service Providers to deliver applications as a hosted service to Customers. This standard applies to any new and existing development which has a contractor-provided hosting option.

For this standard's purposes, the following capitalized terms have the following definitions.

- Service Provider – The AASHTOWare contractor responsible for maintaining the AASHTOWare product(s) and ensuring services are available. This includes all subcontractors.
- Customer – The user of the application or services. Typically, this would be a state department of transportation employee.

*All requirements for compliance with this standard are shown in red italicized text.* ***New requirements that were not in the previous version of the Standards and Guidelines are shown in red bold italicized text.***

# 2. Task Force/Contractor Responsibilities

The product task force and contractor (Service Provider) responsibilities for the Hosted Services standard are summarized below:

- *Ensure business-specific hosting requirements are defined and implemented.*
- *Ensure technical hosting requirements defined in this standard are implemented in the product when applicable.*
- *Ensure industry best hosting practices and emerging hosting trends are considered and implemented appropriately.*
- *The Service Provider must specify whether the hosted environment is a single or multi-tenant (SaaS) environment.*
- Where feasible, multi-tenant environments are more cost effective and should be encouraged.

# 3. Required (or Recommended) Deliverables and Artifacts

The following summarizes the required deliverables and artifacts that must be created and/or delivered in order to comply with the standard.

*Technology Architecture: See Section 2.4.5.3 in the Software Development and Maintenance Process Standard.*

*Service Level Agreement: The Service Level Agreement (SLA) defines required service categories and related performance requirements between Service Provider and Customer.*

# 4. Procedures

## 4.1 Establish Hosting Requirements

For each new development or major enhancement effort, the task force and/or Service Provider should:

- Analyze the business needs, expectations, and constraints that impact the data, application, and hosting architecture,

## 4.2 Include AASHTOWare Hosting Technical Requirements

*Where applicable, the task force and/or Service Provider must ensure that the technical requirements listed below are included in the SRS.*

### 4.3 Review Impact to Existing Hosting Architecture

For each enhancement or modification to an existing application, the task force and/or Service Provider should ensure that there is no impact to the existing hosting or integration introduced by the implementation of the enhancement or modification.

### 4.4 Test and Implement the Hosting Requirements

The task force and Service Provider should ensure that all hosting requirements are thoroughly tested and implemented. ***The Service Provider will collaborate with the task force or relevant agencies to assess and determine the necessary resources allocated to meet the application's requirements for availability and performance. These include processor, memory, and other infrastructure components. The Service Provider will focus on scalability and resource optimization and, when necessary, will adjust resources. The Service Provider will collaborate with the task force or relevant agencies to accommodate future adjustments based on demand or usage.***

## 5. Technical Requirements (or Technical Recommendations)

This section is not applicable.

## 6. Deliverable and Artifact Definitions

### 6.1 Technical Architecture

#### 6.1.1 Description

*See Section 2.4.5.3 in the Software Development and Maintenance Process Standard.*

#### 6.1.2 Content

*See Section 2.4.5.3 in the Software Development and Maintenance Process Standard.*

#### 6.1.3 Infrastructure Location

*The infrastructure that hosts any part of an AASHTOWare application must reside within the United States unless an alternative location is specified in the Service Level Agreement. Any third-party application used must also reside within the United States, or an alternative location must be specified in the Service Level Agreement.*

#### 6.1.4 Hosting Environments

Service Provider should provide a minimum number of hosting environments to allow for the controlled development, testing, and deployment of applications. ***Unless otherwise defined in the SLA, a minimum of two (2) environments must be provided: Test and Production.*** However, it is recommended that a third environment, Development (DEV), also be provided to support full application lifecycle activities.

- Development (Recommended): An environment used as a sandbox for development and experimentation without impacting the production environment or end users.
- ***Test/Stage (Required): An environment used for validating functionality, performance, and stability before deployment to production.***
- ***Production (Required): An environment used for live application usage and access by end users.***

### 6.2 Service Level Agreement

#### 6.2.1 Description

*The Service Level Agreement (SLA) defines required service categories and related performance requirements between Service Provider and Customer. An SLA must be*

*executed for each hosted service or group of related services. At a minimum, the SLA must define the basic computing environment, whether it is single or multi-tenant, service availability, scheduled maintenance and upgrade times, backup and retention services, support and monitoring services and times, escalation paths, mean time to respond and mean time to repair, and minimum-security standards.*

### 6.2.2 Content

*The SLA must, at a minimum, address the following service categories.*

#### 6.2.2.1 Customer Points of Contact (POC)

*The SLA must include named individuals for each of the roles below. Each POC must specify their first name, last name, email address, and a phone number where they can be reached in case of an emergency requiring contact outside of normal business hours.*

*Business POC – Must be the AASHTOWare End User Designee (EUD).*

*IT POC – Must be the IT technical resource responsible for coordinating all changes with other systems. Depending on the state and how IT is managed, this may be a state central IT resource.*

*Security POC – Must be the individual responsible for handling data breaches and other security issues which can be the chief information security officer (CISO), their designee, or a similar role.*

The Customer is responsible for updating points of contact as required. *The Service Provider must verify the points of contact at least annually with the EUD.*

#### 6.2.2.2 Data Ownership

*The Customer must own all rights, title, and interest in its data that is related to the services provided. AASHTO retains all rights, title, and interest in the application system related to the services provided. The Service Provider must not access or make use of any Customer account or data except when expressly required to provide services or at the Customer's written request.*

#### 6.2.2.3 Data Protection

*Protection of personal privacy and sensitive data must be an integral part of the business activities of the Service Provider to ensure that there is no inappropriate or unauthorized use of information at any time. Multi-factor authentication (MFA) must be part of the security tools used to protect the customer's data unless an exception is included in the Service Level Agreement.*

#### 6.2.2.4 Data Encryption

*Any data stored by a Service Provider must be encrypted at rest and in transport.*

#### 6.2.2.5 Data Sovereignty

*All services of a system provided by a Service Provider must remain in the Customer's country of origin unless the Service Provider and Customer otherwise explicitly agreed upon other terms. This includes backup data and disaster recovery locations. The Service Provider will not permit its personnel and contractors to access any system or data except as required to provide support services.*

#### 6.2.2.6 Integration Availability

*The Customer must be provided with a way to get live transactional data through an API interface or direct RDBMS connectivity. If a request is made, the Customer must be provided with data extracts, at regular intervals, defined as at least once a day.*

#### 6.2.2.7 Availability and Performance

*Applications and their dependencies in the hosted service model are expected to be hosted in a fault-tolerant, highly available environment as agreed to by the Customer and Service Provider. Service Provider must provide AASHTOWare and the Customer with an on-demand, automated way of accessing availability and performance metrics.*

#### 6.2.2.8 Disaster Recovery

*The SLA must include Recovery Time Objective (RTO), Recovery Point Objective (RPO), and Recovery Service Level (RSL) definitions.*

*In the event of a data center outage or loss, the Service Provider must bring the hosted solution online in another physical location within the Customer's SLA agreed infrastructure country with minimal data loss. This must be completed within the recovery time frame defined in the SLA. Minimal data loss must be defined by the Customer. Service Provider should attempt to mitigate any data loss.*

*An annual disaster recovery exercise must be completed. The scope of the exercise and the month to perform the exercise must be specified in the SLA. The results of the exercise must be reported to the Customer and AASHTO staff. The report must include the RTO, the time taken from when the system went down until the system was fully restored.*

#### 6.2.2.9 Backup and Recovery

*Any data hosted by the Service Provider must follow the following backup policies:*

1. *Daily backups must be kept for a minimum of six months unless otherwise defined by the Customer.*
2. *Backups must be stored in a data center geographically dispersed from the data center hosting operations. For example, a product hosted in a data center on the East Coast must store backup data in a data center on the West Coast unless otherwise specified by the Customer.*
3. *Point in time recovery (down to the second) must be kept for a minimum of 24 hours unless otherwise defined by the Customer and agreed to in the SLA.*
4. *Service Provider must facilitate a way to bring a hosted system back to a specific date or point in time, as defined in Backup and Recovery items one and two.*

#### 6.2.2.10 Service Provider Compliance

*Service Provider must meet any applicable laws or rules established by the Customer.*

#### 6.2.2.11 Audit and Logging

*Any system hosted by Service Provider must record and retain audit-logging information sufficient to answer the following questions:*

- *What activity was performed?*
- *Who performed the activity?*
- *Where was the activity performed?*
- *When was the activity performed?*
- *What was the status, outcome, or result of the activity?*

*Logs must be made available to the Customer on-demand through a vendor and the Customer agreed upon method. Logs must be kept for a minimum of six months.*

#### 6.2.2.12 Security Scanning

A security scan should be performed against the hosted service and its infrastructure with a security scanning service which scans against the following databases:

1. Mitre's CVE (Common Vulnerabilities and Exposures) list
2. The U.S. National Vulnerability Database

Remediation

Vulnerabilities found by the security scanning service should be scored utilizing the *Common Vulnerability Scoring System Version 4* (CVSSv4.0).

1. Vulnerabilities with a Qualitative Severity Rating of 9.0-10.0 should be remediated within two weeks.
2. Vulnerabilities scoring a Qualitative Severity Rating of 7.0-8.9 should be remediated within one month.
3. Vulnerabilities scoring a Qualitative Severity Rating of 4.0-6.9 should be remediated within three months.
4. Vulnerabilities scoring a Qualitative Severity Rating of 1.0-3.9 should be remediated within a year.
5. Vulnerabilities scoring a Qualitative Severity Rating of 0 do not have a defined remediation timeline.

#### 6.2.2.13 Security Compliance Reporting

*The Service Provider must provide a Systems and Organization Controls 2 (SOC2), Type II report no older than one year when requested by a Customer or AASHTO staff.*

*It must address the five Trust Services Criteria defined by the American Institute of Certified Public Accountants (AICPA):*

1. *Security – How the hosted site and applications are protected from attack? This will include addressing the "SOC2 Common Criteria List." See this reference for more details.*
2. *Availability – Verifies that the data and applications are available for use as defined in the SLA. It should also affirm that the controls and monitoring are in place to assure smooth operations and maintenance of the system.*
3. *Confidentiality – Verifies that the host has the proper elements in place to assure sensitive information is protected.*
4. *Processing Integrity – Verifies that the processing delivers complete, valid, accurate, timely, and authorized access to information to meet the SLA for the customer.*
5. *Privacy – Verifies that any personally identifiable information (PII) that is collected, used, retained, shared, and disposed of agrees with any jurisdictionally appropriate laws, regulations, or policies and is compliant with any additional SLA requirements.*

*This report must be provided to the Customer and AASHTO staff within two weeks of completion.*

#### 6.2.2.14 Security Incident Response

*Service Providers hosting systems for Customers are required to:*

1. *In the event of a breach, comply with the appropriate local, state, and federal reporting laws for the jurisdictions where the hardware resides and the Customer's principal office. In case of conflict, the Customer's office takes priority.*
2. *Report to the AASHTOWare product staff and the Customer's Security POC within one hour of its identification of any breach of security, including but not limited to unlawful access, suspected intrusions, theft, or other actions that compromise the security of information technology resources owned or hosted by the Service Provider.*
3. *Cooperate with the Customer and AASHTO staff during investigations of suspected computer security incidents by providing all requested information to the Customer and AASHTO staff.*
4. *Establish any additional security controls that are deemed necessary by the Customer and AASHTO staff.*

#### 6.2.2.15 Change Control

*The Service Provider must establish a change control process, which will, at a minimum, notify the IT POC of any changes that occur to the infrastructure or application code of a hosted service. The SLA must define the lead time required for non-emergency notifications prior to a change being implemented. At a minimum, the change control notification must include:*

1. *Description of the change.*
2. *Reason for the change.*
3. *Expected impacts.*
4. *Change date.*
    a. *If this is an emergency (less than 24-hour notice) change, the Service Provider must clearly indicate this change as an emergency change.*
5. *Date by which the Customer must notify the Service Provider of any concerns. (The amount of time allowed must be appropriate for the expected impact of the change.)*

*The Service Provider must maintain a change log and notify all Customer IT POCs when the change has been completed.*

#### 6.2.2.16 Termination and Suspension of Service

*In the event of termination of the contract, the Service Provider must implement an orderly return of all data in a mutually agreeable format. The Service Provider must guarantee the subsequent secure disposal of all data upon confirmation by the Customer.*

1. *Suspension of services: During any period of suspension or contract negotiation or disputes, the Service Provider must maintain all Customer data.*
2. *Termination of any services or agreement in entirety: In the event of termination of any services or agreement in entirety, the Service Provider must retain all Customer data for a period of 90 days after the effective date of the termination.*
    a. *Within this 90-day timeframe, the Service Provider will continue to secure and back up Customer data covered under the contract.*
    b. *After such 90-day period, the Service Provider must have no obligation to maintain or provide any Customer data and must thereafter, unless*

*legally prohibited, dispose of all Customer data in its systems or otherwise in its possession or under its control as specified in section (4) below.*

3. *Post-Termination Assistance: Within ninety (90) days from the effective date of the agreement termination, the Customer must be entitled to any post-termination assistance generally made available with respect to the services unless a unique data retrieval arrangement has been established as part of the Service Level Agreement. This includes the Service Provider delivering a complete copy or copies of the data to the customer in a digital format acceptable to the Customer.*
4. *Secure Data Disposal: Ninety (90) days after the effective date of the agreement termination, when requested by the Customer, the Service Provider must destroy all requested data in all forms, for example disk, CD/DVD, backup tape, and paper. Data must be permanently deleted and must not be recoverable according to National Institute of Standards and Technology approved methods, and certificates of destruction must be provided to the Customer.*

This page is intentionally blank.

AASHTO AASHTOWare™ SOLUTIONS

# AASHTOWARE OPENAPI STANDARD

## S&G Number: 2.100.02.0S

## Effective Date: April 1, 2026

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 1.0 | 9/28/2024 | Initial release | 12/11/2024 Approved by SCOA |
| 2.0 | 12/04/2025 | Changed the standard name and added technical API requirements | 3/12/2026 Approved by SCOA |

# Table of Contents

# 1. Purpose

This document establishes AASHTOWare OpenAPI standards and guidelines for the development and management of REST-based APIs within the AASHTOWare ecosystem. It provides a structured framework to ensure consistency, interoperability, and efficiency in API design, implementation, and maintenance. The document outlines methodologies for API versioning, security, resource management, data interchange, and compliance with industry standards. Additionally, it addresses key focus areas, including public-facing APIs, partner-limited integrations, and business-driven use cases, with the goal of enhancing system reliability, scalability, and user experience.

This document uses the term “OpenAPI Specification” or “OAS” to refer to the industry OpenAPI specifications and standards.

# 2. Responsibilities

The product task force and contractor responsibilities are summarized below:

**API Platform Team (API PT) Responsibilities**

- Defines the scope and objectives of the AASHTOWare OpenAPI standards.
- Ensures alignment with industry best practices and regulatory requirements.
- Coordinates with stakeholders to gather requirements and feedback.
- Reviews and approves changes to the document.
- Maintains oversight of implementation and adoption across relevant teams.

**Task Force and Contractor Responsibilities:**

- Ensures the development and implementation of API components align with established standards.
- Enforces compliance with security, versioning, and documentation requirements.
- Verifies the API implementation to ensure proper testing and validation for reliability.
- Ensures ongoing support and maintenance for API-related issues.
- Facilitates the documentation of findings and recommendations to drive continuous improvement.

# 3. Required Deliverables and Artifacts

The following summarizes the required deliverables and artifacts that must be created. ***These artifacts must be created when an API is created and updated when necessary.***

- ***API Definition Documentation – A comprehensive document outlining the APIs structure, including the following at a minimum:***
  - ***Endpoints***
  - ***Data models***
  - ***Security, access control, and authentication methods***
  - ***Expected responses***
  - ***Detailed schema definitions for API responses, including supported data types, pagination methods, and request body structures***
  - ***Security and Access Control.***
- ***Testing and Validation Framework – A set of test cases and automated validation mechanisms to ensure interoperability and compliance with AASHTOWare OpenAPI standards.***

- ***API Usage Guidelines – Instructions for API consumers on proper usage, including examples, error handling, and optimization techniques.***
- ***Operational and Monitoring Policies – Standards for product API deployment, tracking, instrumentation, and performance monitoring.***

# 4. Procedures

The Procedures section outlines the structured steps and best practices required to ensure compliance with the AASHTOWare OpenAPI Standards. These procedures provide a clear framework for API design, implementation, testing, deployment, and maintenance, ensuring consistency, security, and interoperability across all AASHTOWare APIs.

By following these procedures, API developers and stakeholders can align with industry standards, streamline integration efforts, and enhance the reliability and usability of APIs. This section covers essential processes such as API registration, security enforcement, version control, performance monitoring, and governance practices.

Each procedure is designed to facilitate seamless API development and operations, providing guidance on how to properly implement, test, and maintain APIs throughout their lifecycle.

General Procedures

- ***Define and document all API functionalities.***
- ***Implement version control and registration processes for APIs.***
- ***Ensure security schemes are properly defined and enforced.***
- ***Maintain compliance with required API headers, resource paths, and data models.***

Implementation and Design Procedures

- ***Adopt RESTful principles in API development, adhering to the OpenAPI Specification.***
- ***Use standard naming conventions for API paths, resources, and operations.***
- ***Ensure correct implementation of HTTP methods.***
- Follow pagination, filtering, and content negotiation guidelines.

Testing and Validation Procedures

- ***Perform rigorous API testing, including functional, integration, and security testing.***
- ***Validate API compliance against OpenAPI Specifications and AASHTOWare OpenAPI standards.***
- ***Ensure backward compatibility with previous API versions where applicable.***

Operational and Monitoring Procedures

- ***Establish logging, diagnostics, and instrumentation to monitor API performance.***
- ***Define and enforce API rate limiting, caching, and compression policies.***
- ***Regularly review API metrics and usage analytics to ensure optimal performance.***

Governance and Maintenance Procedures

- ***Maintain proper documentation and update API definitions as needed.***

# 5. Technical Requirements

The sections cover various practical concerns for implementing RESTful services. Some sections may not be relevant for basic web API development. For more complex cases requiring additional considerations, these sections offer best practices informed by industry examples, formal standards, accepted formats, and consensus conventions.

API Info

An API-first approach is a development strategy that prioritizes the design and implementation of APIs before other software components. By placing APIs at the core of the development process from the outset, this methodology ensures well-defined, consistent, and reusable APIs that serve as the foundation for the entire system. The API-first approach enhances interoperability, accelerates development, and ensures that APIs effectively meet the needs of all stakeholders before additional system components are built.

The API definition establishes a standardized framework that enhances the developer experience by setting clear expectations for data exchange. At its core, the API defines a structured data language tailored to its specific use, ensuring consistency and eliminating ambiguity regarding available data, exchange mechanisms, and system interactions. This formalized approach streamlines integration, prevents misinterpretation, and facilitates seamless interoperability between systems.

Additionally, the API definition includes comprehensive documentation, providing detailed explanations and sample structures to guide developers. As the API evolves, its data language dynamically adapts to incorporate schema additions, refinements, and extensions. These updates are accompanied by thorough documentation, enabling data consumers to effectively leverage new features and maintain compatibility with evolving standards.

## 5.1 Key Standards, Specifications, Syntax, and Forms

- Internet Engineering Task Force (IETF) Requests for Comment (RFCs)—the foundation for Internet standards—are prioritized standards for RESTful aspects and web protocols.
- ***The OpenAPI Specification (OAS), 3.1, must be used when developing web/HTTP-delivered RESTful APIs.***
- ***The JSON Schema specification must be used as the primary data description and serialization convention for primitive object types.*** JSON serialization should be used as the primary serialization method for non-primitive object types.
- Extended Backus-Naur Form (EBNF) should be used as the meta syntax convention for complex semantic uses such as filtering clauses and rules' expression. EBNF meta syntax is used by several well-known standards and platforms allowing for a degree of simplified translation. Alternatives to EBNF are less suitable for (query) parameter usage.
- ***Augmented Backus-Naur Form (ABNF) must be used for runtime expressions as part of the OpenAPI Specification in defining Link and Callback objects.***
- ***CommonMark (Markdown) syntax must be used in API definition documentation.***

## 5.2 API Versioning, Deprecation, and Management

This standard establishes initial requirements for API versioning and deprecation, specifically for externally consumed APIs, which include Partner APIs and Public APIs (defined below). While Internal APIs are also defined in this section, these requirements do not apply to them.

### 5.2.1 API Classifications

#### *5.2.1.1 Internal APIs*

Internal APIs are product-specific APIs developed and maintained by AASHTOWare contractors to provide direct access to an AASHTOWare product's services and data. These APIs serve as the foundation for both internal development and external partnerships, ensuring data integrity, security, and consistency. ***Internal APIs must establish consistent, design-by-contract, reusable definitions.***

**Best Practice:** To maintain control and ensure secure access, only the AASHTOWare product itself should directly interact with its Internal APIs. Any external access to an Internal API should be facilitated through Partner APIs or Public APIs, following structured access control measures.

#### *5.2.1.2 Partner APIs*

Partner APIs enable deep integrations between AASHTOWare products and external agencies, partners, or other AASHTOWare products. These APIs grant access to AASHTOWare product data and services, allowing for customized interactions and mission-critical collaborations.

Examples of Partner API Integrations:

- Agency partners accessing AASHTOWare product data via secured API channels.
- Secure data exchanges and interoperability between AASHTOWare Project, AASHTOWare Bridge Management, AASHTOWare Bridge Design and Rating, and AASHTOWare Pavement ME Design.

Partner APIs:

- Are designed to facilitate seamless, secure, and controlled interactions, ensuring reliability while maintaining strict access governance.
- ***Must commit to diligent stakeholder and technical reviews,***
- ***Must establish consistent, design-by-contract, reusable definitions, and***
- ***Must ensure lifecycle and (API) versioning coordination between all participating trusted parties.***

#### *5.2.1.3 Public (Authenticated) APIs*

Public APIs are developed and maintained by AASHTOWare contractors for use by external developers and the broader public. These APIs expand an AASHTOWare product's capabilities, enabling integrations with third-party applications and allowing external entities to access valuable AASHTOWare data and services.

Examples of Public API Integrations:

- Posted bridges and bridge clearance data for public reference.
- Active construction location data for real-time traffic and planning applications.

Public API access is strictly managed and controlled through AASHTOWare OpenAPI, ensuring that usage remains secure and aligned with AASHTOWare's access policies. These APIs are provisioned in a more restrictive manner than Partner APIs, limiting exposure while maintaining accessibility for authorized public users.

***Public APIs must defensively account for "bad practice" usage scenarios through platform-level aspects (usage limiting, application developer identification, activity monitoring, etc.) and avoid breaking changes, version-imposed disruption to reliability or stability, and deviation from standardization controls. Public APIs must distinguish between public, partner, and internal access modes.***

### 5.2.2 Versioning

The API definition will evolve with numerous enhancements, extensions, and revisions that provide significant new capabilities to the API consumer. However, if these improvements do not break any previous functionality (e.g., definition backwards compatibility), neither the major nor the minor version will change. This may be counterintuitive to traditional software application updates in which at least a minor version would change. However, for the API definition, an ideal enhancement roadmap would never go beyond v1.0 if the updates are always backwards compatible. Major and minor versions are indicators to the API consumer and the API developer writing code of changes that would potentially require retesting, redeployment, and potential redevelopment of core functionality.

Minor version updates would indicate a modification that invalidates the previous definition in some aspect. These may be one or multiple breaking changes in operations, parameters, or schema elements that cannot be reconciled via translation or deprecation.

A major version change would indicate a significant product, architectural, design convention, or platform change in addition to invalidating the previous definition. In almost all cases, the implementation artifacts for a major version would correspond to a significant code change, or even a fork in extreme cases.

As compared to product iterations, major and minor version changes to the API definition are not inevitable. Minor version increments occur when a breaking aspect is unavoidable. Major version increments occur when the “philosophical” core has fundamentally changed. This change is drastic enough to consider the new major version as a new embodiment unto itself.

As stated above, API versioning differs from application and standard product versioning, is a lifecycle aspect, and has resource implications. RESTful conventions express resources. ***(API) Versioning must be global and resource specific.***

***API versioning must not be used in resource (URL) paths.*** API version context may be included in headers, query parameters, or policy/context metadata.

Version context assignment should be limited to these structures:

- Header assignment
  - api-version
  - x-aw-apiversion
  - accept, v={version};version={version} (media content formatting)
- Query parameter assignment
  - api-version
  - v
  - version
- Caller metadata

Published API definitions should limit version designations to major and minor.

#### *5.2.2.1 Production Release Numbers*

*The release or version number for production APIs must adhere to the following format. Periods will be used between the following release numbers.*

#### *5.2.2.2 Major Release Number*

*The major release number is shared by the API definition and platform supporting deployments (i.e., AASHTOWare OpenAPI platform Infrastructure Services).*

#### *5.2.2.3 Minor Release Number*

*The minor release number is shared by the API definition and platform assets.*

#### *5.2.2.4 Maintenance Release Number*

The maintenance release number is specific to the supporting deployment asset and not the API definition.

#### *5.2.2.5 Pre-Production Release Numbers*

*Pre-production APIs will follow the production release number format followed by a dash and an alpha-numeric string.* The content of the string is left to the developer. Possible contents are:

- “-RC1” : release candidate 1

- “-test1” : test identifier
- “-20230808” : date

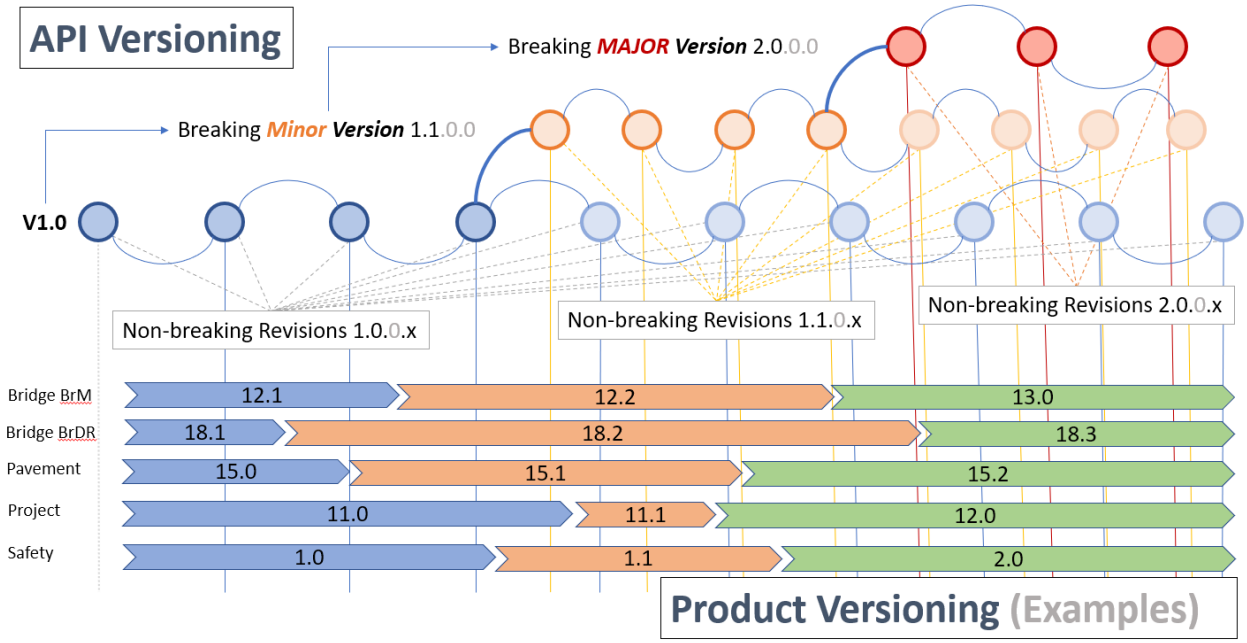



*Figure 1. API Lifecyle Phase Diagram – Tandem, API vs Product Versioning*

### 5.2.3 Deprecation and Decommissioning

The AASHTOWare OpenAPI Standard includes the ability to classify operations, parameters, and schema elements as deprecated. The deprecation designation does not remove functionality; it simply recommends that it **should** be transitioned out of use.

Deprecation is included in the API definition, requires a revision, and requires changelog information, but remains version adherent. The later removal of a previously deprecated element **is** a major or minor version change, likely a breaking change, and is treated accordingly. Deprecation, in theory, should be rare. Decommissioning (“sunset” declaration per IETF *RFC8594*) should be rare. In short, the RFC guidance to remove an operation includes forewarning relayed in the operation “sunset” header assignment. The “sunset” header includes policy metadata to indicate alternative support and final expirations. The following decommissioning is the full removal, error generating, functional elimination of an element of the definition that was previously in place. Decommissioning an aspect is the forceful assurance that reliance or use of that component will produce an error.

If an element does require removal from the definition, the lifecycle approach is to first: avoid at all costs, second: deprecate, third: decommission via the sunset technique. As part of the avoidance strategy, although undesirable, existing elements can be translated into new elements. This is a purely semantic mapping but serves to avoid the deprecation path and aids in functional backwards compatibility. Translation elements are an implementation responsibility but are associated with definition revisions. As such, they are listed in the changelog under the translation moniker. Translation, deprecation, and sunset are changelog entry categories for this lifecycle area.

*An AASHTOWare OpenAPI API must be maintained for at least two years after providing notice that it is to be deprecated or decommissioned.*

## 5.3 API Definitions

***RESTful web APIs must use the OpenAPI Specification (OAS). OAS API definitions must be written in YAML or JSON.*** YAML is recommended for “free hand” authoring for improved readability within the context of design or communications’ reference.

Elements of an OAS API definition may be written as isolated components for external reuse. Formally, these referenced elements are expressed through the 'External Document Object' in the API's definition. Behaving much like code snippets, includes, or using/linking directives, the External Document Object contents extend the definition. When creating finalized published API definitions, those definitions should include all external documents as resolved resources. Publishing with resolved external references places all necessary information in a single asset.

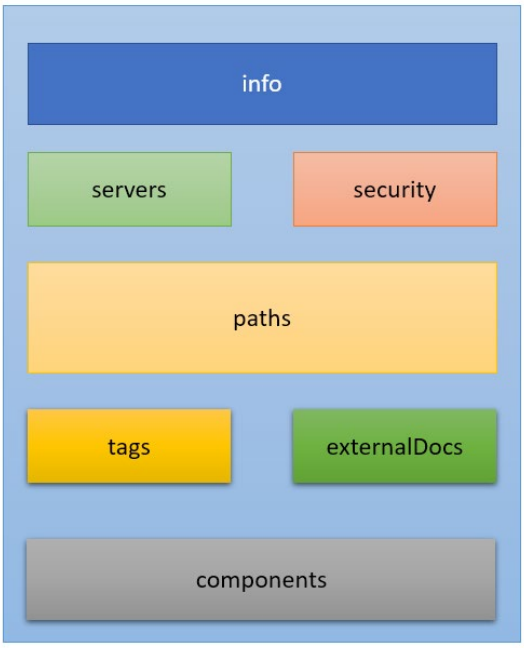



*Figure 2 OpenAPI Specification 3.0 Root Elements*

***API definitions must include meaningful descriptive (text) content explaining key information.*** API definitions should contain thoughtful and simple (code and text) content populated on the 'example' object members. ***API definitions must be written for human and machine consumption use cases.*** Web-styled publishing formats for end-developer users is an example of human consumption use case. Use of existing coding standards for nomenclature is a consideration for source generation and an example of a machine consumption use case. Developer end-user technical references, beyond but including the API definition, should be provided as a more comprehensive knowledge base.

***API definitions must be written in English using the American dialect: U.S. English, referenced by IETF RFC 4646 as "en-US". Case sensitive values must employ a standardized naming convention within the API definition.*** Case sensitive values should not use kebab casing. ***API definitions must provide a value for 'operationId' on every operation*** (OAS mandates uniqueness, case sensitivity, but not assignment requirements). API operation 'operationId' values should be lower snake case (e.g., "awproject_get_project"). Automation-suitable rules are recommended for assigning 'operationId' values such that context, function, schema (association), and resource can be uniquely and easily inferred programmatically. Specification extension properties (prefixed with "x-") may be used to provide additional object properties. ***Extension properties that are global to AASHTOWare must be prefixed with "x-aw-". Text content in examples, titles, summaries, descriptions, and terms of service must be written such that the information is simple, clear, easily translatable to non-"en-US" languages/dialects, and well formatted to allow for various indexing mechanisms.***

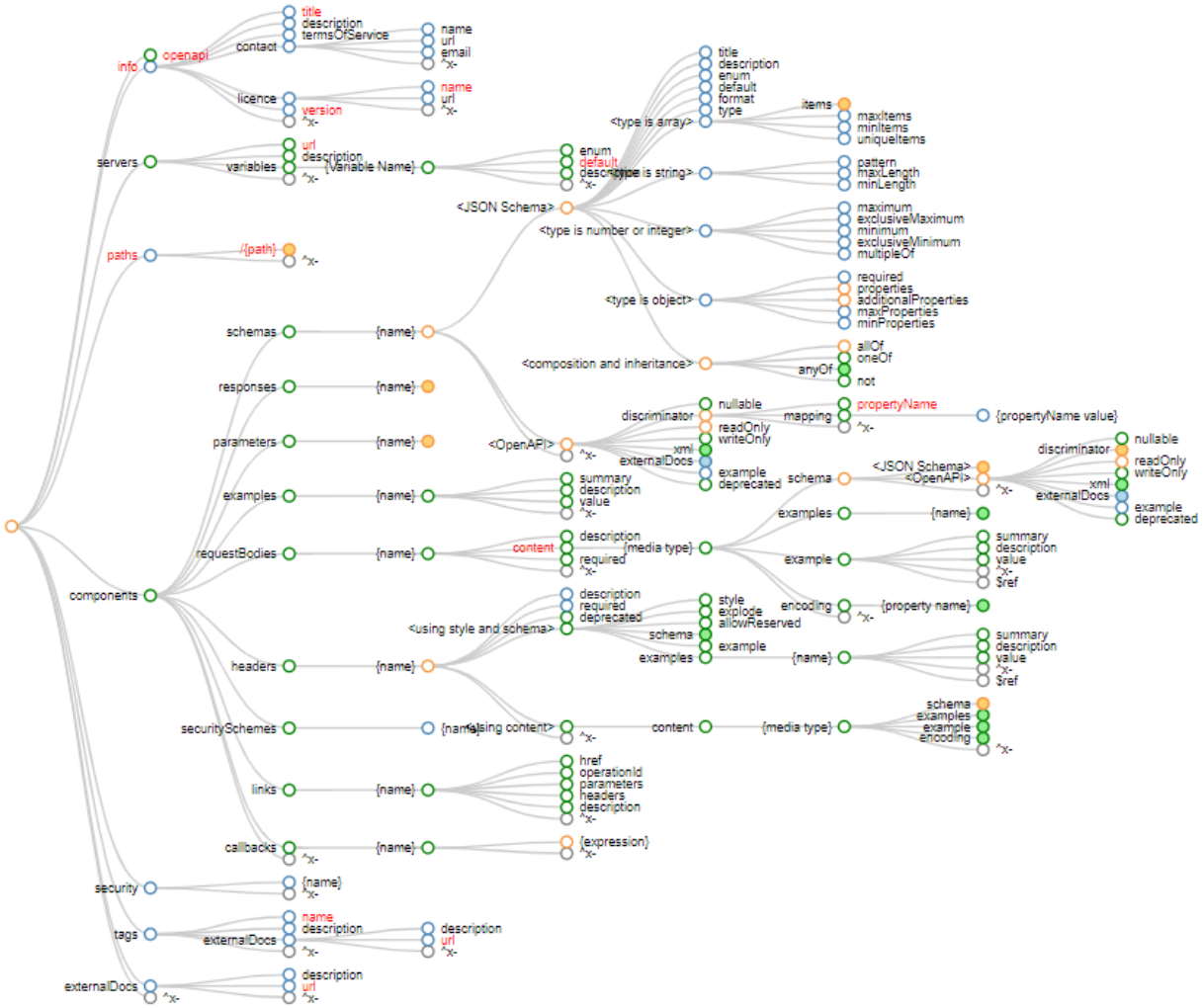



*Figure 3 Expansion of OpenAPI Specification Structure*

***API definitions must be semantically correct. RESTful semantics express resources that are resolved in a uniform way, subject to a set of consistent actions. Resources must be defined as (English) nouns while the actions on resources must be framed as (English) verbs.***

***API definitions must include descriptive metadata.*** API definitions should include extended properties when additional specific classifying information is required for governance, automation, and reference. Extended properties should adhere to a naming convention to establish a classification system, similar to namespaces in other programming languages. ***Extended properties must begin with the prefix "x-".***

***API definitions must have a title, description, and contact.***

***API definitions must have a version number that adheres to Semantic Version 2.0 except for rules 9 and 10.*** API versioning may incorporate semantic versioning rules for pre-release designation (rule 9). ***API versioning must not incorporate appended build metadata (rule 10).***

***API definitions must include the extended property 'x-api-terms' to declare terms of use.***

API tags should be implemented and applied to operation definition to organize, illustrate, and categorize their functional context. It is recommended that API definition operations only use one tag. The use of a singular tag facilitates and simplifies certain client and provider code generation scenarios.

## 5.4 API Registration

***API internal names must not conflict with one another.*** API internal names should enforce a namespace-style classification. API internal names and aliases should be included in the API definition. ***When included in API definitions, API internal names and API aliases must use the following extended properties:***

- x-aw-api-internal-name
- x-aw-api-alias

***Property values must be universally unique.*** The API internal name property should be kebab cased, clear, concise, condensed, and meaningful. The API alias should be lowercased, short, and simple but discernible by abbreviation or convention.

Access modes define the API request caller context or mechanism. The API classification (internal, partner, and public) in conjunction with the access mode will impact some use cases.

## 5.5 Security Schemes

Security schemes enumerate allowed or supported credentialing, authentication, and a subset of authorization permissions. ***APIs must comply with the Security Standard and enforce application business rules.*** Authorization scopes should be defined using role-based and claim-based access policies.

***Every API definition must explicitly state its applicable security scheme. If no security scheme is applied, the related definition information and metadata must also qualify any relevant restrictions in the terms of use extended property 'x-api-terms'.***

Security schemes are defined by the OpenAPI Specification. APIs may use multiple security schemes. OAuth2 and OpenID security schemes are recommended as preferred practice.

## 5.6 Headers

Primary standard and well-adopted headers may be used. Header usage should be explicitly explained with meaningful descriptions in the API definition. ***Use of standard headers must be correct. Use of content headers must be formally backed by request, response, and (gateway) policy behaviors when supported.***

Header naming should use an upper kebab case format.

Content-Location

***"Content-Location" must be used for resource identification while "Location" must be used for redirection.***

Authorization

***The "Authorization" header must be used for credentialing structures of "Bearer" and "Basic".***

***"X-API-Key" is reserved for API Key-based security schemes.***

ETag

The "ETag" header should be used for resource versioning, caching extensions, concurrency support. The "ETag" value may be computed or an explicitly assigned value. Data repository "row versions", and equivalents are recommended candidates for ETag usage. Computational hashes may be used for ETag values. As a resource version identifier, the ETag value does not require universal uniqueness nor cryptographically strong hash qualities. The ETag value must express a unique value to a unique (version) state of a resource.

Accept-Encoding

Compression should be used. The standard client request header "Accept-Encoding" will dictate compression formats. ***The standard response header of "Content-Encoding"***

***will indicate the compression method used for the returned payload.*** Gzip, brotli (br), and deflate are appropriate HTTP compression formats.

Non-Standard

***Non-Standard headers must be prefixed with 'X-'.***

***Rate and usage limiting header directives must use the existing IETF draft guidance:***

- X-RateLimit
- X-RateLimit-Limit
- X-RateLimit-Remaining
- X-RateLimit-Reset
- X-RateLimit-Policy
  - ***Valid***
    - ***X-RateLimit-Policy: 10;w=1, 50;w=60, 1000;w=3600, 5000;w=86400***
    - ***X-RateLimit-Policy: 10;w=1; burst=1000, 1000;w=3600***
  - ***Invalid***
    - ***RateLimit-Policy: 10;w=1, 10;w=60***

Long-running asynchronous operations should include an originating request or resource identifier. Use X-Request-Id for request identification, allowing caller-assigned values, preferably a randomly generated UUID.

***For complex resources, use X-Entity-Uid, a strongly typed, UUID-formatted, 128-bit header. If a UUID/GUID mechanism is unavailable, apply RFC 4122-compliant hashing (UUID v3 MD5 or v5 SHA-1) to ensure uniqueness without cryptographic strength requirements.***

The following header conventions should be used for pagination metadata if required:

- X-Page-First, indicates location of first segment in collection set
- X-Page-Next, indicates location of next segment in collection set
- X-Page-Previous, indicates location of previous segment in collection set
- X-Page-Last, indicates location of last segment in collection set
- X-Page-Self, indicates location of current segment in collection set
- X-Page-Count, indicates total count of items in collection set
- X-Page-Size, indicates total maximum items per segment
- X-Page-Index, indicates one-based index reference if present on current request
- X-Page-Offset, indicates absolute (not segment page count) offset if present
- X-Page-Sortby, sorting expression used by request if present
- X-Filter-Expression, filtering expression used by request if present
- X-Filter-Target, filtering expression target used by request if present

***Note: Cursor based paging for locations (first, last, next, previous, self) must include the "page_cursor" parameter. Offset based paging for locations must include all parameters necessary to calculate a collection segment "seek" within the set.***

## 5.7 Access Modes

Access modes define the API request caller context or mechanism. The API classification (internal, partner, or private) in conjunction with the access mode will impact some use cases.

Access mode information may be included in the API definition. ***When access mode information is included, the following extended property naming conventions must be used:***

- ***x-client-type, expected values: public or confidential***
  - ***Public: "uncontrolled" API client/application, zero trust***
  - ***Confidential: controlled or partially trusted API client/caller***
- ***x-provider-type, expected values include product, partner, aashtoware, global***
  - ***product implies product specific functionality***
  - ***partner implies external system/service specific functionality***
  - ***aashtoware implies aggregation/generalized/platform functionality***
  - ***global implies universal utility functionality***
- ***x-usage, expected values include application, shared, partner, public***
  - ***application implies API client/caller is controlled (does not determine trust which is defined by client type)***
  - ***shared implies multiple API client/caller context or applications that are controlled by coordinating (internal) teams***
  - ***partner implies API client/caller is controlled by a partner entity adhering to specification or convention***
  - ***public defines API consumption in a public-use role***

Combinations of client type, provider type, and usage are unique. The API definition may include multiple values for each. ***The API definition must accurately describe sanctioned access modes through these listed extended properties.***

## 5.8 Data Model, Schema, Types and Formats

***ISO-defined and IETF-defined standardization must be used. Data types and formats must reflect the OpenAPI Specification. The following options establish the primitive types:***

- String
- Number
- Integer
- Boolean
- Array
- Object

The OpenAPI Specification 3.0 data types and formats are based on the JSON Schema Validation (vocabulary). Suitable data types and formats will be limited to those defined in the OpenAPI Specification. Type definitions are singular. Mixed types are defined by variant qualifiers "oneOf" and "anyOf". An unassigned type will allow any type. Primitive types allow nullability through a "nullable" attribute. Objects define complex types. Arrays define collections.

Data type formats extend primitive types. These formats are recommended to define language-generalized equivalency:

- Integer Type Formats
  - int32 (4 byte)
  - int64 (8 byte)

    - bigint (unbounded, signed)
- Number Type Formats
    - float (single precision decimal)
    - double (double precision decimal)
    - decimal (unbounded, decimal)
- String Type Formats
    - byte (bin64 encoding, RFC 7493)
    - binary (bin64 encoding, byte array, RFC 7493)
    - date (date-only, ISO 8601, RFC 3339)
    - date-time (full date, with time, ISO 8601, RFC 8601 section 5.6)
    - uuid (textual representation 128bit globally/universally unique identifier)

Formats outside of the OpenAPI Specification-defined options are allowed. ***Types outside of the OpenAPI Specification are not allowed.***

Even with ISO/RFC standardized treatments, serialization, deserialization, interpretation, and over/underflow conditions may be nuanced between platforms.

The string type can define enum options to declare a specific set of possible values. Enum-support string types (still) allow nullability.

## 5.9 Resource Paths

Resources are accessed from endpoint targets expressed as paths. Examples of resource paths are:

- /orders
- /customers
- /trends/playlists

Discussed in later sections, methods to access, query, manipulate, or interrogate (context) against resources define API operations. Paths and methods are used to create operations. Forward slashes "/" delineate path segments. Path segments should reflect logical hierarchical relationships, dependencies, or associations.

Path templating defines URI parameters and structures to compose a resource location. Path parameters are enclosed by curly braces {}. Examples of templated paths containing parameters are:

- /orders/{id}
- /vendors/{ein}
- /books/{isbn}/authors

Paths, templated parameters, and resource hierarchy organization can have non-trivial functional impacts. ***Path parameters must not allow null, empty values, or trailing slashes. Path parameters must be encoding- and URL-friendly.*** Paths should not use "/api" as a root or base convention. ***Paths must not use version directives "/v1", "v2", "version2.1". Resource collection names must be pluralized. Path segments must be lower snake case "/orders/{id}/order_items".***

Linguistically, path-defined resources are nouns. ***Paths and resources must not use (linguistic) verbs: /catalog_entries/search_results is correct /search_catalog is incorrect.***

From domain driven design, an aggregate root is an entity from which logical and dependent associations form an outward hierarchy. API paths should reflect aggregate resources and

define path segments as to match any strict dependencies. For example, a strict dependency exists between orders and order items. Resource paths should encompass important aggregate roots. ***Aggregate roots must use domain-based meaningful names.*** An association between products and orders may exist without the same strict dependency seen with orders and order items. For example, "/orders/{id}/order_items" is better than "/order_items?order_id={id}".

Each path resource may express distinct (functional) responsibilities. Separation of responsibilities is applicable to path and segment structure. Access control may be applied to resources on a specific path.

Course Registration API Example

Aggregate root and dependent entities include courses, students, and registrations. Consider the paths:

- /courses/{id}/students
- /students/{id}/courses
- /registrations?student_id={student_code}&course_id={course_code}
- /registrations/{registration_confirmation_code}

If the course data model includes registered students, the first path is unnecessary. If students registered in a course requires access control, the first path has advantages. If student data, course data, and registration information are delivered from disparate systems, separate microservices, or distinct repositories, the second path elegantly hides that complexity. The third path illustrates an antipattern – registrations are not an aggregate root, and the resource resolution is not based on the resource (entity). The fourth path illustrates an acceptable, non-nested, non-aggregate root resource. Non-nested resource paths may be used to directly access subordinate association entities. Resources, paths, and path segments should be well-composed to the domain.

***Complex (entity) graphs must allow expansion, navigation, traversal, denormalization, and unique (key) distinction.*** Resource associations should be flattened to achieve denormalization through the addition of scalar/primitive properties. Object graph complexity requires deliberation and evaluation against resource path complexity. Caching, concurrency, and scalability can be influenced by resource path convention strategies. Graph complexity can reduce data request frequency.

Resources and path composition should model complete business capabilities, achieve API goals, and reconcile to specific functional objectives. ***Dependent resources must be defined through hierarchical path segments (as seen in the previous examples), where each segment is a valid resource.***

Path resource resolution may use contextual discriminators:

- /employee_profiles/{id} ➔ /employee_profiles/me
- /workspaces/{assignment_id} ➔ /workspaces/home
- /timecards/{period_close_id} ➔ /timecards/current

Path resource depth should not exceed five nesting levels. Path resource length, including parameters, is limited by URL character limits.

Paths should not end with trailing forward slashes.

## 5.10 Common Parameters, Formats, Content

Parameter definitions in this section describe URL path segment parameters and URL query strings. ***Parameters must use lowered snake case formats and not camel case.*** Path parameters should use simple and non-exploded styles:

Simple, Non-Exploded

| | | |
|---|---|---|
| /product_types/{code} | /product_types/7 | product type with code 7 |
| /product_types/{code} | /product_types/7,8,9 | product types with codes 7,8, or 9 |

Query parameters may use form and exploded or form and non-exploded styles:

Form, Exploded

/product_types?flag={flag} /product_types?flag=trending&flag=new

Form, Non-Exploded

/product_types?flag={flag} /product_types?flag=trending,new

Complex parameter values should rely on simple and standards-based formats and styles. For object serialization or (functional) expressions, JSON Schema, and extended Backus-Naur form (EBNF) should be used.

EBNF Expression for Filtering Example

ProductName ne 'Aniseed Soup' and LastRestockDate ge 2023-05-01T00:00:00Z

Path parameters should be simple, clear, and readable. URL encoding requirements impact these considerations. Query parameters should adhere to similar requirements. Query parameter constraints are less stringent than path parameters. Path parameters identify resources and sub resources through their (path) segments' composition. Parameter naming should be qualified to avoid ambiguity. The parameter name "created" is subject to interpretation as a Boolean flag or a date. Further qualifying the name as "created_at" or "created_flag" reduces confusion.

The following common-use parameters should use the conventions listed in the next sections.

### 5.10.1 Selection

{id}: Unique identifier with variant data typing. Distinguishes item(s) within a set or collection.

| | | |
|---|---|---|
| /items/{id} | /items/11 | /items/xyz |

{uid}: Specialized universally unique identifier, 128bit data type, includes globally unique identifier formats. Formally, should be limited to versions/classes of 3 and 5 (namespace, name hashes) and 4 (random).

| | |
|---|---|
| /items/{uid} | /items/6bed7b8e-db77-4f3a-8d2f-d3252ece8088 |

{*_id}, {*_uid}, {id}, {uid}: Multiple, composite, or compound key ids.

/items/{item_id}/sub_items/{sub_item_id}

/items/11,19,75/sub_items/21,78

/items/[…]/sub_items/[…]

### 5.10.2 Pagination

Pagination parameters are prefixed with "page_" to assist in disambiguation and potential naming conflicts. The contextual definition occurs after the suffix and does not compound the definition (as later seen in "page_offset").

{page_index}: one-based index, page reference location used in conjunction with the page size parameter and the page_cursor if a cursor-based techniques are employed.

{page_size}: number of items to include in one page (set count).

{page_offset}: Absolute count offset "skip" value. Adds to the quantity determined from the page_size and page_index values. Is not an offset of pages.

{page_sortby}: Sort by clause, comma delimited, implied ascending, explicit ascending or descending by '+' or '-' prefix.

/locations?&page_index=1&page_size=1000&page_sortby=timezone, region,name

{page_cursor}: Against a predetermined sequential or sortable and unique identifier, the page cursor is a value for that identifier used in the collection (set). This may be an ID, UID, composite key, set-spanning version (akin to a relational database row version or timestamp), or any equivalent reference. Consider a collection of 1,000 locations, uniquely identifiable by an incremented numeric ID:

"First"

/locations?page_size=100

/locations?page_size=100&page_cursor=

"Next"

/locations?page_size=100&page_cursor=MTAx

"Previous"

/locations?page_size=100&page_cursor=MQ==

"First"

/locations?page_size=100&page_cursor=MQ==

"Last"

/locations?page_size=100&page_cursor=OTAx

### 5.10.3 Content Negotiation

Content negotiation parameters provide flexibility in data retrieval without modifying the resource structure. Content negotiation techniques should be used to improve performance, extend utility via embedded resources, and ensure efficient/optimal data transfer based on the requesting caller's needs.

{filter_expression}: Filter clause to apply to a collection or set. The filter expression may use any format, style, syntax, or standard. The value should use extended Backus-Naur form or a simple expression style (form, non-exploded). Analogous to the OData $filter keyword.

{filter_target}: Filter clause supplement to provide formatting and style switches, EBNF as default. The filter_target may reference fields, formats, or styles that are parsed for interpretation in conjunction with the filter_expression parameter.

{transform}: Result transform parameter that must be interpreted for processing. The transform value does not have a convention or strict enum definition. The transform is a supporting concept to allow dynamic switching of the result.

{fields}: (Advanced) Limits model data to members listed in parameter value. Analogous to fields in GraphQL or the OData $select keyword.

{expand}: (Advanced) Expands model data to related members. Provides resource expansion or associated data required for entity navigation based on the parameter value(s). Expansion is defined explicitly.

{expand_depth}: (Advanced) Expands model data to related members based on a traversal depth, regardless of entity structure or model definition.

### 5.10.4 Temporal/Versioning

Temporal parameters are used in time-based expressions that include versioned, historical, or changed-tracked entity models. Temporal data operations complement

data repository features such as point-in-time queries or change tracking. Most modern data storage systems (including enterprise relational databases) support temporal query capabilities as of the SQL:2011 standard. ***Temporal parameters are prefixed with "temporal_" to avoid parameters with the same name***, but more general date/time context. Temporal parameters correspond to standard temporal operations.

{temporal_start}: Inclusive starting point for temporal range, often used in conjunction "temporal_end" parameter.

{temporal_end}: Inclusive ending point for temporal range, often used in conjunction with "temporal_start" to achieve a "between" spanning range.

{temporal_asof}: Singular date point used for a point-in-time expression. "temporal_asof" is suitable for use for individual entity selections and collections.

{temporal_version}: Version marker, allows variant typing, corresponds to single-entity version regardless of change tracking or historical retention. Temporal versions may (re)use "ETag" header conventions. Temporal versioning values may use simple numeric incrementing values. Temporal version values must be unique for a given individual entity.

Collection resource extensions may be added to support structured temporal results. Filtered partial results are necessary to limit total data returns.

Single item resource extensions may be added to support structured temporal results.

/items/point_in_time/{temporal_asof}

/items/{id}/item_history

/items/{id}/item_history/{temporal_version}

Canonical extension options and conventions for temporal treatment are listed in Canonical Extensions section.

### 5.10.5 Tracking

Tracking parameters are used to facilitate tracing, collation, and audit by action, operation, event, request, or response. Fundamentally, use of these parameters coincides with a stateless transactional flow that requires the ability to "reconstruct" processing steps.

{request_id}: Recommended UUID/GUID used to identify a specific request, may allow type variation. In long-running asynchronous processes, a non-failing acknowledgement status may be provided while the actual result is prepared. Later, resolving the result to the originating request would be done through the "request_id" value.

{correlation_id}: UUID/GUID used to identify and correlate a sequence of requests or events as part of a larger operational transaction.

{event_id}: UUID/GUID used to identify an instance of a raised event. The generation of an event instance may occur in an operation's defined callback or in a distinct webhook. The event_id is distinctly separate from any request_id value but may be associated through a correlation_id. The values of request_id(s), event_id(s), and correlation_id(s) should not be reused or reassigned across those parameters.

{event}: String used to qualify and uniquely identify an originating source of an event. For a webhook, this value would be the webhook name. For a callback, this would be a callback name. The event value should serve as a specialized partitioning key against associated event_id(s).

## 5.11 Data Interchange

Standard media types should be used for standard operations. ***The JSON media type must be supported with conventional data interchange operations where an incoming or outgoing data payload is present.*** The XML media type support should be made

available as a complement to the JSON media type support when needed. This does not apply to non-standard operations that return images, videos, files, or application-specific formats.

### 5.11.1 Input Payload

Data modification object payloads should reflect their corresponding data retrieval payloads. A POST operation payload to create an item would approximate the data returned in the item resource GET operation. A partial update PATCH operation should be expressed as a subset of the data returned in the item get operation. Data modification operations include POST, PATCH, PUT, and DELETE methods against resources.

Data modification payloads that provide batch operation should reflect their equivalent operations with extension to array object collections.

Data modification payloads requiring additional context, metadata, properties, or information “enrichment” may envelope contents in an object “root” wrapper to support more complex structures.

The following root objects, subordinate collection, and enrichment field naming options are recommended when a convention or practice does not exist:

- Root/Wrapper Object Names
  - input (preferred default)
  - parameters
  - data
  - settings
- Subordinate Principal Collection Names
  - items (preferred default)
  - values
  - results
  - elements
  - children
  - details
- Common Enrichment Fields
  - code
  - status_code
  - status
  - info
  - message
  - correlation
  - date
  - source

Standard data modification operations that do not include file content should not use HTTP multipart requests. Instead, complex inputs as described above should be used to achieve input parameterization segmentation. Standard data modification operations that do not include file content should not require multiple input types (JSON, XML, form, etc.). Operation input should be type-consistent or type-appropriate.

***Input payloads that do not execute data modification, the rare GET body case, must define root-expressed data object. Input payloads that do not execute data modification operations must adhere to the same guidance defined in this section.***

#### *5.11.1.1 Specialized Formats*

Existing purpose-fit standards and formats should be employed whenever possible in lieu of custom or proprietary definitions. Formal standards are wide-ranging, sometimes obscure, but preferable to non-standards-based alternatives. IETF-published references on JSON aspects are essential resources for composing potential payload structures.

### 5.11.2 Output Payload

***Standard data operations that do not return file or file streamed content must firstly support standard media types.*** Standard data operations should reflect resource contexts and exclude mixed-function data enrichment. Data operations may support more than one return type object or model. Examples of multiple return type options, expressed using the OpenAPI Specification keywords oneOf | allOf | anyOf, include the model implied by the resource, error, or information(al) objects.

Standard output object collection payloads should not wrap returned data in an enclosing object. Complex object payloads that extend resource data may provide dedicated (wrapping) model objects to best structure data elements. These complex objects should follow the previous guidance and conventions for root objects, subordinate collection, and enrichment field naming options. Output adjustments include:

- Root/Wrapper Object Names
  - result (preferred default)
  - data
  - output

#### *5.11.2.1 Error Object Base*

The following error object base model is recommended for operations returning 4XX status:

YAML definition

```
error:
      type: object
      required:
        - status_code
        - message
      properties:
        status_code:
          summary: Error code to reflect 4XX, 207, or other status result
          type: integer
          format: int32
        message:
          summary: meaningful error message that is suitable for API dev, and not for
end user
          type: string
        correlation_id:
          summary: correlation id for tracking the error, generated by the API operation
          type: string
          format: uuid
```

The error object structure may be extended to include adornments listed in the previous *5.11.1 Input Payload* and *5.11.2 Output Payload* sections. A sub-collection of errors may be added to the error object as a “details” property.

An error object should also be used in batch or bulk operations that include any raised exceptions appropriate to a status 207, multi-status.

#### *5.11.2.2 Information Object Base*

The following information base object model is recommended for operations that do not require a defined model result (DELETE operations, non-failing 202 Accepted):

YAML definition

```
info:
      type: object
      required:
        - completed
        - status_code
        - message
      properties:
        completed:
          type: boolean
        status_code:
          summary: Status code to reflect 2XX result
          type: integer
          format: int32
        message:
          summary: meaningful informational message that is suitable for API dev, and
not for end user
          type: string
        correlation_id:
          summary: correlation id for tracking the operation, generated by the API
operation
          type: string
          format: uuid
        result:
          summary: result of the operation, if applicable
          type: object
          nullable: true
        request_id:
          summary: client-originating initiating request id, generated by the API
request caller
          type: string
          format: uuid
        resource:
          summary: (url) resource result, resource that was created, updated, deleted,
or necessary for subsequent operation
          type: string
        percent_complete:
          summary: percent complete of the operation
          type: integer
          minimum: 0
          maximum: 100
```

Similar to the error object base, the information object structure may be extended to include adornments listed in the previous *5.11.1 Input Payload* and *5.11.2 Output Payload* sections. A sub-collection of “info” objects may be added to the object as a “details” property which provides utility in a status 207, multi-status scenario.

### 5.11.3 Enumeration

The OpenAPI Specification supports nullable and reusable enum definitions. Enumerations are defined as strings. Constrained option sets should be created as enum types. Enum options value declarations must use upper snake case.

### 5.11.4 Entity Naming

Entity naming will reflect model schema concerns and resource (path) treatments. For convention and consistency, entity naming will have implications on serialization formats and styles.

***Entity names as a model representation and a resource representation must be correct in syntax (rules, grammar) and semantics (meaning, context).***

***Entity naming must prioritize resource (URL) format and styles over serialization formats and styles.*** Resources expressed with entity-based segments may be adjusted to ensure the greatest possible compatibility and adherence to convention. Resources should be lowercased and translated easily, even if not perfectly, across

(coding) languages. Lower kebab case is suitable for resource path segments, but it is not prevalent in most coding standards.

Entity names should use lower snake casing to facilitate resource path composition.

***Entity naming conventions must allow for serialization/deserialization flexibility across platforms, conventions, and formats.***

An entity naming example …

### 5.11.5 Entity Member Field Naming

***Entity members must be named with a consistently enforced convention. Entity member names must allow for translation or mapping from one format to another format. For example, an entity containing two distinct member fields of "StatusCode" and "status_code" would not allow translation between formats due to collision/repetition of a field name.***

Entity member naming should accommodate (URL) parametric formats and styles while prioritizing serialization formats and styles. Query parameters expressed with entity member-based may be adjusted to ensure the greatest possible compatibility and adherence to convention. A member name of "DisplayName", "display_name", or "displayname" may be reference as a related, but not matching, parameter such as "name". Consistency should be prioritized conciseness.

There is no unified industry consensus on entity member naming formats.

Syntactically and semantically, various formats have "machine" and "human" pros and cons. Which will vary by platform, ecosystem, stack, and stylistic disposition (machines and humans).

Broadly, underscores, hyphens, and case variation (Pascal and camel) all yield compound word naming structures. Machine/system treatment may vary with case sensitivity and hyphenation versus underscoring (underscoring is indistinct while hyphenation establishes breaks). For (human) readability, underscore formatting can be obscured with font embellishments/adornments in web styling (e.g., hyperlinks) or in printed media (e.g., underlining). In most code languages, hyphenation is not used as a naming aspect (CSS and XML as definite exceptions). In weighing all these factors and practical implications on clarity and efficiency, the following order represents the better practices in member naming formats:

1. Lower snake case
2. Camel case
3. Pascal case

Entity member naming should use the above formats.

***Entity member names must be correct in syntax and semantics.***

***Entity member names must express a meaningful representation of the entity model.***

### 5.11.6 Graph Flattening

Related entities in a model or schema definition, including navigation traversal properties, may be flattened (denormalized) to simplify data payloads. Navigation key(s), property dependent value(s), and a human-readable label/descriptor/display name should be included as additional members as a de-normalizing pattern.

Graph depth traversal de-normalization should be reflected in member names to express hierarchical associations, namespace aspects, or subordinate discriminators. For example, an "order item" that includes a product and a category for that product would be expressed to include the following members:

- product_id
- product_displayname

- product_category_id
- product_category_displayname

***Graph flattening for de-normalized serialization optimization must not violate resource behavior principles. For example, the "order item" cache eviction would occur if the quantity of the product has changed, but not if the product category has changed. As such, if an ETag or cache key is based on a deterministic hash (versus a simple version or modification date), that computation must exclude members that are used for enrichment of subordinate values.*** From above, changing the product name should also not invalidate the cache.

### 5.11.7 Graph Expansion

Graph expansion provides a serialization scaffolding for a normalized, traversable, associated ***resource*** expansion. Graph expansion is not the graph model structure of a resource model entity. Graph expansion is a HATEOAS and resource discovery aspect to achieve the embedding of subordinate resources.

Furthermore, graph expansion differs from an extended model (serialization) graph as the resource is distinct. That resource distinction allows for clear(er) separation of function. A resource can apply access, action, and various other behaviors that are less concise when implemented by extending an entity model and its serialization.

Consider order items' retrieval through a resource path: /orders/{id}/order_items. Modification of the order's items may require special behaviors which can be achieved against an action [GET], [POST], … If there is only a consolidated entity without a resource extension, that will require a more complex treatment of the resource /orders/{id}. However, for graph expansion, the goal is to allow embedding supplements to the resource without incurring the responsibility of behaviors of the embedded subordinate resources. Graph expansion is a content negotiation aspect discussed in *5.10 Common Parameters, Formats, Content*.

A graph expansion for the /orders/{id} resource could take the form orders/{id}?expand=(order_items) as a [GET] action. A collection property of "order_items" would be populated with elements mirroring the resource /orders/{id}/order_items.

Expansion strategies are non-trivial. Expansion allows for improved backwards compatibility preservation. Expansion may be leveraged to improve aggregate performance and scalability. Expansion does not supersede resource composition guidelines.

## 5.12 Requests

### 5.12.1 Operation Identification

***All resource operations must be uniquely identified.*** A resource may support multiple actions. Each action on a resource is an operation. ***Each operation must be given an operation ID reflecting the API information or abridged alias, a primary (included) OpenAPI Specification tag value, the action context, the resource, and cardinality (singular item or a collection of items pluralized). For example, adding an order, in an eCommerce app API: the operation ID of "acmecomm_ordering_add_order" would represent a [POST] action on the resource /orders, which is part of the "ordering" aspect (OpenAPI tag "ordering") for the Acme eCommerce public API. Retrieving orders would have an operation ID of "acmecomm_ordering_get_orders" representing /orders [GET].***

Id collisions are avoided through the API aliasing, one distinct OpenAPI tag, action context, resource target, and implied cardinality via pluralization/singularization semantics.

The following action designation should be used in the absence of an existing defined convention:

- [GET]: read, get, get_all, query
- [POST]: add, create, post, insert, add_batch, create_batch, post_batch, insert_batch
  - query – when a [GET] action is appropriate, but the parameter complexity requires a body payload a [POST] action may be used. The resource may employ a "result" variant if necessary: /orders [GET] and /order_results [POST] in this case. This is a canonical deviation.
- [PATCH]: update, modify, patch (note, the patch action usually applies to a single, non-batched, item)
- [DELETE]: delete, remove, delete_batch, remove_batch
- [PUT]: update, upsert, modify, put, update_batch, upsert_batch, modify_batch, put_batch
- [HEAD]: append suffix "_preamble" to the corresponding [GET], read_preamble, get_preamble
- [OPTIONS]: append suffix "_options", read_options

"_batch" designations imply multiple discrete processing elements in the operation. ***These types of operations must conform to guidance listed in 5.13 Responses, specifically status 207 considerations.***

### 5.12.2 HTTP Verb Usage

***Requests must correctly use HTTP verbs in resource actions.***

***Action behavior must correspond to its verb. Action behaviors include consideration of resource modification, implications of "replay" activity, and suitability for caching.***

***Each verb behavior is qualified by:***

- ***Safety (effectively read only resource retrieval)***
- ***Idempotency (action replay determinism)***
- ***Ability to cache (produces a response that remains valid for a duration)***

#### *5.12.2.1 [GET]*

***[GET] actions must be fundamentally read-only as it applies to the target resource.*** [GET] resource actions are used to retrieve single items or collections. An unresolved single item [GET] should return a 404, not found, status code. A zero-length collection result should return an empty array along with a 200, success, status code. A missing collection should return a 404, not found, status code.

[GET] actions should implement filtering and pagination patterns.

[GET] canonical exceptions should be extremely rare and ***must be strict when used***. Canonical extensions are provided in 5.13 Responses. This includes the use of body content in the request, although not explicitly forbidden, it is canonically outside of the [GET] method directive.

***[GET] actions must be safe and idempotent.*** [GET] actions should be cacheable.

#### *5.12.2.2 [POST]*

[POST] actions are used to create new resources. Item creation should target a pluralized-named resource target. [POST] actions may create multiple items in a batch scenario.

***Successful [POST] actions must return a success status code, 2XX class, result.***

If a single item is created, the operation should return a status 201, created, result. If successful or partially successful, a [POST] batch creation action should return a

status 207, multi-status successful, result as listed in *5.14 OpenAPI RESTful Design Aspects*. A long-running asynchronous, non-failing, [POST] should return a status 202, accepted, result. When the [POST] action does not return a resource payload, a status 204, no content, should be returned.

[POST] actions should create distinguished resources which generate identifying values not provided by the client request (i.e., database record ID, repository entry key, etc.).

The [POST] action request body should approximate the resource that will be created (e.g., an order item [POST] body is well aligned to the single order item [GET] result structure).

[POST] actions are not safe nor idempotent. [POST] actions may implement idempotency schemes. [POST] actions should not be cacheable.

[POST] actions are used to create new resources. Item creation should target a pluralized-named resource target. [POST] actions may create multiple items in a batch scenario.

Successful [POST] actions must return a success status code, 2XX class, result.

If a single item is created, the operation should return a status 201, created, result. If successful or partially successful, a [POST] batch creation action should return a status 207, multi-status successful, result as listed in *5.14 OpenAPI RESTful Design Aspects*. A long-running asynchronous, non-failing, [POST] should return a status 202, accepted, result. When the [POST] action does not return a resource payload, a status 204, no content, should be returned.

[POST] actions should create distinguished resources which generate identifying values not provided by the client request (i.e., database record ID, repository entry key, etc.).

The [POST] action request body should approximate the resource that will be created (e.g., an order item [POST] body is well aligned to the single order item [GET] result structure).

[POST] actions are not safe nor idempotent. [POST] actions may implement idempotency schemes. [POST] actions should not be cacheable.

#### *5.12.2.3 [PATCH]*

[PATCH] actions are used to modify part of a resource, or by extension, apply a partial modification update to a resource. A [PATCH] action body payload content does not have to approximate the resource target. ***The [PATCH] action body content must express the instructions necessary to modify the resource.*** The [PATCH] action body content may use any standard, format, structure, or style. The [PATCH] action body content should align to the resource's structure or a well-known standard such as JSON Merge Patch (IETF RFC 7396) or more ideally JSON Patch (IETF RFC 6902).

***Successful [PATCH] actions must return a success status code, 2XX class, result.***

A simple, single-resource-affecting [PATCH] action should return a status 200, OK, result. A successful batch [PATCH] action should return a status 207, multi-status successful, result. A successful [PATCH] action that does not return response content should provide a status 204, no content, value. Non-failing [PATCH] actions that are asynchronous and long-running should return status 202, accepted, result.

[PATCH] actions are not safe nor idempotent. [PATCH] actions may implement idempotency schemes. [PATCH] actions should not be cacheable.

#### *5.12.2.4 [DELETE]*

[DELETE] actions assert the request to remove a resource representation. This implies the deletion of a single item or a collection of items as expressed by the action. Mechanics of removal are separate from the resource representation. For example, archival, cascade/propagation, “soft deletion”, and eventually consistent patterns will be outside of the [DELETE] action response status. If supported by the resource, [DELETE] actions should scope to a single item. Constrained batch [DELETE] actions may be used to remove multiple items. ***[DELETE] actions must not allow unconstrained removal of a resource collection. For example, this constrained [DELETE] action is allowed: [DELETE] /orders/{id}/order_items?quantity=0. This unconstrained [DELETE] action is not allowed: [DELETE] /orders, as it would semantically delete all orders.***

***Successful [DELETE] actions must return a success status code, 2XX class, result.***

A status 204, no content, success is appropriate for a successfully completed [DELETE] action. In an asynchronous long-running, non-failing, scenario, the [DELETE] action should return a status 202, accepted, result. A subsequent [GET] or a repeated [DELETE] in the long-running asynchronous scenario should return a status 410, gone, result. A subsequent [GET] or repeated [DELETE] after the action processing completes should return a (simple) status 404, not found, result.

[DELETE] actions do not explicitly forbid request body content.

[DELETE] actions are not safe nor cacheable. [DELETE] actions should be inherently idempotent.

#### *5.12.2.5 [PUT]*

[PUT] actions express a resource submission. If the resource exists, the [PUT] action performs a complete replacement. If the resource does not exist, the resource is then (newly) created. [PUT] action request body content should approximate the target resource structure(s). By extension, [PUT] action request body content should resemble [POST] action request body content. [PUT] actions may apply batch operations. A successful or partially successful batch operation [PUT] action should return a status 207, multi-status successful result. A long-running asynchronous, non-failing, [PUT] should return a status 202, accepted result. If the [PUT] action does not require a data return, for the empty result, a status 204, no content should be returned.

***[PUT] actions must be idempotent.*** [PUT] actions are not safe nor cacheable.

A collection-based [PUT] action may be used to effectively “wipe and replace” when the collection is constrained. [PUT] /orders/{id}/order_items would allow for a complete add, update, delete combine on all order items, constrained to the order referenced by the parameter “id”.

#### *5.12.2.6 [HEAD]*

[HEAD] actions are semantically similar to [GET] actions. [HEAD] actions do not return body content, instead, only headers are provided.

[HEAD] actions should align to a [GET] operation. [HEAD] actions usage includes determining a resource’s state or version leveraging the “ETag” header and related information or in interrogating metadata of a large resource instead of accessing it.

Successful [HEAD] actions should return status 204, no content. [HEAD] actions should not return a response body content payload.

[HEAD] actions are safe, idempotent, and cacheable.

#### *5.12.2.7 [OPTIONS]*

[OPTIONS] actions are used to interrogate available methods and options for a given resource. The [OPTIONS] action should be used to provide access control hints (by excluding base methods that are not allowed), cross-origin resource sharing conditions, and encoding support.

[OPTIONS] actions should not return response body content. [OPTIONS] actions should return status 204, no content, when successful.

[**here**]

### 5.12.3 Links

Link objects were introduced to the OpenAPI Specification in version 3.0. Link objects help provide discoverability and a HATEOAS aspect. Link objects are design-time declarations that allow runtime expressions to define an association, relationship, or navigation traversal (resource) path based on the current operation. A Link object may incorporate any component of the operation, using the expression syntax, to define its target (resource). The target may be defined by an existing operation ID using the “operationId” property (which must be uniquely identified within the scope of its OpenAPI Specification definition document), a relative path against its definition using the “operationRef” property, or an absolute path which may include external references using that same “operationRef” property. For example, the customer on an order is resolved from a link using a separate “crm_customers_get_customer” operation, using the returned order in the body content by its “customer_id” property value:

```yaml
paths:
  /orders/{id}:
    operationId: ecomm_orders_get_order
    parameters:
    - name: id
      in: path
      required: true
      description: the order identifier, as order id
      schema:
        type: string
    get:
      responses:
        '200':
          description: the order being returned
          content:
            application/json:
              schema:
                type: object
                properties:
                  customer_id: # the unique customer id
                    type: string
                    format: uuid
          links:
            customer:
              # the target link operationId
              operationId: crm_customers_get_customer
              parameters:
                # get the `customer_id` field from the request path parameter named `id`
                userId: $response.body#/customer_id
```

The following expressions are valid for use in the link object:

- $method – HTTP Method
- $request.header.accept – Media Type
- $request.path.{parameter} – (Defined) Parameter from operation
- $request.body#/{path} – Request body payload element reference
- $url – Request URL
- $response.body#/{path} – Response body payload element reference

Operation links are part of the API definition but do not apply any special or additional behavior to their targets (the link target can be restricted to the caller – existence of a link does not guarantee access).

Link objects are especially applicable to long-running asynchronous operations. In this case, the Link object would include the target that provides the result after processing. This is an extension of the canonical-required use of the “Location” header parameter assignment. Including a Link object here allows a more formal and explicit design-time definition.

### 5.12.4 Canonical Extensions

Canonical extensions are meant to supplement core semantics where there are clear gaps in definitive guidance. These extensions occur when certain scenarios are recognized in common or practical situations, while at the same time their ideal resolution is not explicitly forbidden. Additionally, these extensions may be used when a pending guiding standard is not yet formalized.

Canonical extensions are rare, limited in impact, focused on application, and should ideally solve a specific set of problems.

#### *5.12.4.1 HTTP [GET] Method with Body Payload*

[GET] actions do not explicitly forbid body content. Some complex [GET] actions may require parameter complexity that is not (elegantly) suitable for URL query parameter translation, either due to formatting or total potential size. URL Query string length may be limited and enforced by policy or server platform. Additionally, the combination of [GET] and a body payload may also be restricted by rule or platform. With limitations, when the scenario is better fit with a body payload, complex [GET] actions may use body payload when no alternative exists.

#### *5.12.4.2 HTTP [DELETE] Method with Payload*

As illustrated with the previous GET with body payload example, a complex or conditional [DELETE] action (especially in a batch operation) may also benefit from a content body payload input. The same considerations, tradeoffs, and constraints apply: URL parameter length restrictions VS server/platform support for [DELETE] with content body.

#### *5.12.4.3 HTTP [HEAD] Concurrency Check Preamble*

A resource’s version is delivered through the “ETag” header. A change in a resource’s “ETag” value indicates a change in the resource. For resource updates, if the initial “ETag” header value has changed prior to the commit of the resource modification, there is a potential concurrency conflict where the modification will inadvertently overwrite resource data.

In the context of preemptive concurrency state validation, using the [HEAD] action enables the retrieval of the “ETag” value to ensure the modification is against the intended resource version.

When a resource concurrency scheme or versioning strategy is required, it should implement metadata retrieval using the “ETag” header value for version, through the [HEAD] action.

#### *5.12.4.4 HTTP [HEAD] Passive Pagination Metadata*

The ability to traverse a large data set through pagination is essential for scalability and performance. However, computing the associated paging metadata can add additional overhead to the retrieval operations and may not be required in all scenarios. Providing collection-based resource metadata separate from data retrieval reduces the resource burden.

Pagination metadata data may be provided from a [HEAD] action against a collection-based resource. Pagination headers should derive from those listed in "Headers" section.

***Paginated data retrieval must support serial and parallel fetch strategies. When supported, passive pagination metadata must include threshold and bounding values, explicit or derived (e.g., first, last, count).***

### 5.12.5 Common Request Bodies

The body content of most requests will adhere to the resource model structures for [POST] and [PUT] actions. [PATCH] actions may also align to the resource model or a formalized standard such as the JSON Merge Patch or JSON Patch list inside of the *5.13 Responses* section.

Batch operation collections will extend this structure as simple arrays of the underlying resource model.

Primitive parameter collections should be defined using a simple name, value pairs, where the value is either a single non-object type or an array of the same non-object typed elements. This parameters' structure may take the form:

```
parameters:
      type: array
      items:
        type: object
        properties:
          name:
            type: string
          value:
            oneOf:
             - type: string
             - type: number
             - type: integer
             - type: boolean
             - type: array
```

Complex parameter collections extend the previous example with type variation and the inclusion of the non-primitive object type. The complex parameters' structure may take the semantic form:

```
complex_parameters:
      type: array
      items:
        type: object
        properties:
          name:
            type: string
          value:
            anyOf:
             - type: string
             - type: number
             - type: integer
             - type: boolean
             - type: array
             - type: object
```

When distinguishing request body content elements, a set-based identifying approach should be used based on a combination server origination, client origination, or sequence indexing. The following model member names are recommended for identifying usage:

- id - server generated, variable type support, expresses uniqueness in a collection-based resource (often associated with a repository key).
- uid – allows client generation/origination, a formal UUID value that is either random (version 4) or convention based with hashing (version 3 and 5), ensures repository-independent uniqueness, and facilitates idempotent implementation strategies.
- ordinal - in a collection set or subset, expresses an index locator that is unique, sequence based, and is an integer data type. The ordinal value may support explicit sorting directives that are leveraged as a processing support.

When possible, common request body formats should be normalized through the use of base definitions in conjunction with extension through the “allOf” discriminator. This polymorphic technique is illustrated below:

```yaml
components:
  schemas:
    vehicle:
      type: object
      required:
        - make
        - model
        - year
      properties:
        make:
          type: string
        model:
          type: string
        year:
          type: integer
    car:
      allOf:
        - $ref: '#/components/schemas/vehicle'
        - type: object
          required:
            - body_style
          properties:
            body_style:
              type: string
    truck:
      allOf:
        - $ref: '#/components/schemas/vehicle'
        - type: object
          required:
            - bed_length
          properties:
            bed_length:
              type: number
```

## 5.13 Responses

Operation responses can be classified into a limited number of generalized categories. These include singular object results, object array/collection results, no-content results (differs from empty array results), informational digest results, and error message results.

Safe and cacheable operations should implement “ETag” header value assignments in their response.

Header value assignments and status code values must be correct and consistent to the operation response.

### 5.13.1 Common Response Bodies

#### *5.13.1.1 Info*

An informational result body should be derived from a standardized base type object such as the example listed in *5.11 Data Interchange*. An informational result should be provided instead of a “no content” result in the absence of any special exceptions.

***An informational result must not be used for error results.*** An informational object may be used in conjunction with an error result (object) as part of a multi-status return collection.

Extending informational result object types to include basic, non-sensitive, telemetry is recommended (e.g., time references, processing, localized/regionalized context messages).

#### *5.13.1.2 Error*

An error result body should be derived from a standardized base type object (also illustrated in *5.13 Data Interchange*) to ensure a consistent behavior for exception conditions.

An error result body should be used for error results. An error result should supplement a multi-status result when an error condition is encountered as part of the operation.

Extending error result object types to specific functions and conditions is recommended. ***Error result objects must not include sensitive information in production environments.*** Error result objects may include extended debug, trace, logging, and instrumentation data outside of production environments. Error results should include functional metadata that can be used by the API developer in a debugging capacity. Error results may include non-sensitive metadata that can be presented to an end user.

#### *5.13.1.3 Resource Links*

To achieve greater discoverability and advanced (higher maturity) API capabilities, resource links may be included as part of a response body data return. This approach differs from the OpenAPI Specification Link object as it is a runtime aspect delivered in the response data as compared to a design-time definition based on the operation resource and resource expressions. Although the use of response data-delivered resource links is similar to operation Link objects, the runtime composition will allow for greater flexibility that may not be achieved in defining the Link object. The definition below represents an example base object for a resource link data structure:

```
resource_link:
  description: A base type of representing links to resources.
  type: object
  required:
   - href
  properties:
    href:
      description: A protocol and path to the resource, e.g.,
https://example.com/api/objects/123
      type: string
      format: uri
    relation_type:
      description: The relation type, as defined by rfc 8288 options
      type: string
```

When used as a collection, the collection (array) property should be named “links”. When used as a property, the name “link” should be used.

Additional information properties may be added to objects derived from the “resource_link” which can be used to better match the OpenAPI Link object.

#### *5.13.1.4 Page(d) Result*

A paged data model represents a result subset of items within a collection-based resource. Simple object array results should be used over a paged data model when possible. When a paged data model is required, it must include pagination supports for its target methodology (offset or cursor) and allow for potential high-volume/transaction iteration over the full collection.

The model below provides a representative base “paged_result” structure (offset and cursor properties both included:

```
paged_result:
      type: object
      required:
            - items
      properties:
        description: A paged result of collection items
        items:
          description: items in the result subset
          type: array
          items:
            type: object
        first:
          description: first page of the result set
          type: string
```

```
    format: uri
last:
  description: last page of the result set
  type: string
  format: uri
next:
  description: next page of the result set
  type: string
  format: uri
previous:
  description: previous page of the result set
  type: string
  format: uri
self:
  description: current page of the result set
  type: string
  format: uri
page_index:
  description: current page index
  type: integer
page_cursor:
  description: current page cursor
  type: string
page_size:
  description: current page size
  type: integer
page_offset:
  description: absolute offset used in addition to the page index
  type: integer
page_sortby:
  description: current page sort by expression
  type: string
filter_expression:
  description: current page filter expression
  type: string
filter_target:
  description: current page filter target
  type: string
count:
  description: count of items in the current page result subset
  type: integer
total_count:
  description: total count of items in the full collection result set
  type: integer
```

Cursor-based traversal will require serial traversal and does not easily allow for parallelization strategies. Offset pagination does allow for simple parallel execution strategies.

Paged results should explicitly define and enforce page size limits.

Paged results are suitable for caching support. Paged results should implement caching, however, cache invalidation and ETag value determination may be non-trivial depending on the underlying object collection and model structure.

The paged result data model return can require significantly more processing overhead in comparison to an equivalent simple object array result. The page result must return all necessary metadata required perform the paging operations but should not return more than minimally required. Additional paging strategy supports are discussed in *5.14.5 Discoverability*. Alternative approaches are also listed in *5.12.4.4 HTTP [HEAD] Passive Pagination Metadata*.

### 5.13.2 Status codes

***API operations must use standard HTTP status code classes (1XX, 2XX, 3XX, 4XX, 5XX).*** API operation results should limit status codes to standard well-adopted conventions with expected corresponding behaviors. Implementations may use sub-indexed HTTP status codes for internal instrumentation and logging but should not use internal sub indexing status code structures as operation results.

These well-adopted conventionally used status codes (listed by class) should be used appropriately:

#### *5.13.2.1 Status 1XX - Informational*

100 Continue: Rare usage, discussed further in *5.14.9 Purpose*, may be implemented as a preamble check or validation when the client request includes the 'Expect: 100-continue' header. Returning status 100 effectively affirms the request is suitable for processing based on the provided headers, not a series 200 status code [SUCCESS] return, but a confirmed non-failing precondition. ***If header values do not fulfill a precondition check, a failing result with status HTTP status code 417 is returned.***

#### *5.13.2.2 Status 2XX – Success*

200 OK: Operation succeeded. Successful operations should return specific status codes based on their HTTP verb and function. Status code 200 may be used for a generalized success condition. ***Successful GET operations will return status 200.***

201 Created: Successful creation of a resource from a POST or PUT operation. ***201 must not be used when a resource is not created. GET, HEAD, DELETE, and PATCH operations must not return status 201.***

202 Accepted: Non-failing completed request indicating accepted (for processing) but incomplete final processing with indeterminant action. Status 202 only indicates operation validity and simply 'Accepted'. Status 202 alludes to a 'pending' action. Long-running asynchronous operations should use status 202 to support process orchestration.

204 No Content: Successful operation in which no (body) content is returned. Status 204 is appropriate for PUT operations that modify existing resources. Status 204 should not be used in GET operations that return collections with no results (empty array structures are appropriate).

206 Partial Content: Successful return of a (binary) data subset from the operation. Appropriate to GET streaming file data scenarios. ***Streaming-enabled file GET operations to support client requests that include 'Range' HTTP headers must return status 206 in conjunction with response headers of 'Content-Range' and 'Content-Length'.***

207 Multi-Status: Completed operation in which multiple resources are entrained in a response, subject to individual status codes, and not explicitly homogeneous. The WebDAV (web distributed authoring and versioning) HTTP extensions include usage of status 207 but this status more broadly, and simply, connotates operations that may include differing status results. ***Operations that are candidates for caching must not allow status 207.*** Multi-status 207 should be reserved for POST, PUT, DELETE, and PATCH operations. Collection-based POST, PUT, DELETE, and PATCH should support status 207 results. 207 status-resulting operations should return a content body with a singular root object, optionally include root-level object properties, and ***must include a collection of result objects with the corresponding status, resource,*** and may include a status message plus a location (URL).

#### *5.13.2.3 Status 3XX – Redirection*

300 Multiple Choices: Rare usage, useful as a deprecation/sunset preamble directive where HATEOAS, multi-linking options exists. Multi-Choice 300 should be reserved for deprecated/sunset operations that have been subsequently "split" where the new resources are resolved separately. 300 status-resulting operations should return a content body with a singular root object, optionally include root-level object properties, and must include a set of choice objects, each indicating the HTTP verb and destination, and which may also supply an operation ID, a human-readable label, and a descriptive note. The 'Location' header should not be assigned unless a definitive 'default' resource (replacement) exists.

301 Moved Permanently: Redirection directive for GET operations that are no longer supported. Allows deprecation/sunset backwards compatibility. Non-GET operations should not use 301 status operations for redirection. ***The 'Location' header must be assigned for a 301 status-returned action.***

308 Permanent Redirect: Redirection directive for operations containing a request payload that are no longer supported. Allows deprecation/sunset backwards compatibility. (Simple) GET operations should not use 308 status operations for redirection. ***The 'Location' header must be assigned for a 308 status-returned action.***

#### *5.13.2.4 Status 4XX – Client Error*

400 Bad Request: Failing operations should return specific status codes based on their HTTP verb and function. Status code 400 may be used for a generalized failure condition(s). Status code 400 reflects failure resulting from the request (payload, format, syntax, or precondition).

401 Unauthorized: Credentialing or authentication is absent. ***Operations that require authentication must return status 401 when credentialing or authentication does not exist.*** If authentication, identity, or credentialing is present but privileges are insufficient for the operation, status 401 should not be used. Status code 403 Forbidden is the appropriate response when required permissions are not present.

403 Forbidden: Current identity does not have adequate permissions for the resource or operation. ***Operations that require authorization (as compared to authentication) must return 403 when the acting identity does not have sufficient privileges.*** Status 403 should not be used when no identity is present nor when credentialing authentication has not occurred. Status code 401 Unauthorized is appropriate when required authentication has not occurred.

404 (Resource) Not Found: Referenced resource does not exist. Status 404 should be returned in cases where the operation is valid, but the resource is not present. As an example, /items/1 would return status 404 if the templated value /items/{item id} does not contain an item with item id = 1 in the items collection.

405 Method Not Allowed: (HTTP verb) Method is not allowed for resource. Status 405 should be returned when a resource method is not allowed, not implemented, or not supported. Status 404 reflects the absence of a resource, status 405 represents resource existence without (specific) method support.

409 Conflict: Resource state conflicts with attempted (modification) operation. When optimistic concurrency is applied, status 409 may be returned as an appropriate error response. Status code 409 should not be returned when explicit pessimistic locking strategies are employed. Pessimistic locking conflicts should return status 423.

423 Locked: Resource is locked from modification due to an explicitly assigned state. Status 423 should be used in a pessimistic concurrency scheme when preventing resource modification. Status 423 should not be used in optimistic concurrency strategies. Status 423 is employed with WebDAV protocol extensions, but it is not exclusive to WebDAV. Operations should include response (body) information about the resource, lock condition, and available action(s).

#### *5.13.2.5 Status 5XX – Server Error*

500 Internal: Unexpected/unhandled server error condition. This generalized (server) error should not be raised within the context of a resource operation. This generalized error may be used as a default error exception response.

501 Not Implemented: Requested resource or resource method is not supported. Status 501 may be returned by a "not implemented" exception for a resource or resource method and should be returned when the resource or resource method is expected but not processed.

503 Service Unavailable: Service is temporarily unavailable. Status 503 should be used to indicate a temporary disruption. The 'Retry-After' header may be assigned to indicate the service resumption.

## 5.14 OpenAPI RESTful Design Aspects

API designs should be non-fragile as well as robust. API designs should be conservative. API designs should be deliberate and cognizant of future extension requirements from the perspective of backward compatibility. API designs should be deliberate and restrictive in definition (formats, lengths, ranges, validation, composite constraints). API designs should rely on known patterns and well-accepted conventions as a foundation. API design quality is measured by its resulting developer experience. Mature APIs provide better developer experiences than less mature APIs. ***API designs must accommodate multi-modal usage scenarios (e.g., headless, daemon processing, On-Behalf-Of delegation, user interaction driven, automation, high volume transactional).***

### 5.14.1 REST API Maturity Levels

The Richardson REST Maturity Model (RMM) has been used to measure how well a web-based API follows the principles of REST (Representational State Transfer) as an architectural style for designing web services. The RMM was originally suggested by Leonard Richardson in 2008. The model has four levels, from 0 to 3, each adding more features and constraints to the API design. Level 0 uses only one URI and one HTTP method (usually POST) for all operations. Level 1 introduces multiple URIs for different resources but still uses only one HTTP method. Level 2 uses different HTTP methods (such as GET, POST, PUT, DELETE) to indicate different operations on the resources. Level 3 adds hypermedia controls (such as links and forms) to the responses, allowing the client to discover and navigate the API dynamically.

RMM will be applied to the OpenAPI Specification (OAS) usage as a standardizing format for web API design elements. Design guidance rooted in RMM against the OAS will include these key attributes:

- URI templates and path parameters to define multiple resources
- HTTP methods and status codes to define different operations and responses
- Schema definitions and examples to validate and document the request and response bodies
- Links and callbacks to describe hypermedia controls and asynchronous operations

The four maturity levels and their target thresholds are detailed in the following sections.

***API designs must achieve Level 2 or higher aspects.***

#### *5.14.1.1 Level 0 - "Swamp of PO*"*

Level 0 is the lowest level of maturity and the least conforming to REST principles. A web service at this level exposes only one URI for the entire application and uses only one HTTP method, typically POST, for all operations. The request and response bodies are typically in an object-serialized format and contain information about the operation and the resource. PO* represents "plain old xml", "plain old json", or any equivalent substitute.

This level has several drawbacks, such as:

- It does not leverage the full potential of HTTP, such as using verb differentiation for specific actions, using status codes for errors, and using headers for metadata.
- It does not support resource identification and addressing, which means that each resource cannot be uniquely accessed by its own URI.
- It does not support hypermedia, which means that the client cannot discover and navigate through related resources by following links or forms.

- It cannot follow the OpenAPI Specification as a standardization framework for RESTful APIs.

A web service at this level is not RESTful and should be considered an antipattern reference.

#### *5.14.1.2 Level 1 - Resources*

Level 1 of the RMM begins to reflect rudimentary concepts of RESTful design, where each resource in the system is assigned a unique URI. The resource is a formal abstraction that can be accessed or manipulated by client requests. Consider an employee, a product, or a booking as resources in a proposed API.

In Level 1, the API uses different URIs to identify the different resources, but still relies on a single HTTP method (typically POST) to perform all operations on them. This means that the API does not adhere to correct standard HTTP methods (i.e., GET, PUT, PATCH, DELETE) that correspond to the common CRUD (create, read, update, delete) actions on resources. A Level 1 API might use the following URIs and HTTP methods:

- /employees: POST to create a new employee or get a list of all employees
- /employees/{id}: POST to get, update, or delete an employee by ID
- /products: POST to create a new product or get a list of all products
- /products/{id}: POST to get, update, or delete a product by ID

In the OpenAPI Specification, a Level 1 API can be defined using the paths and operations objects. Each path represents a resource URI, and each operation represents an HTTP method and its parameters and responses. For example, the /employees path can have one operation with the POST method:

```yaml
paths:
  /employees:
    post:
      summary: Create a new employee or get a list of all employees
      parameters:
        - in: query
          name: action
          schema:
            type: string
            enum: [create, list]
          description: The action to perform on the employees resource
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Employee'
        description: The employee data to create (required for create action)
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Employee'
        '400':
          description: Bad request
        '500':
          description: Internal server error
```

Level 1 is an improvement over Level 0, where the API uses only one URI and one HTTP method for handling all requests. However, Level 1 still does not fully leverage the semantics of HTTP. A RESTful design requires achieving Level 2 with correctness in HTTP method use for resource operations.

#### *5.14.1.3 Level 2 - HTTP Verbs*

At Level 2 of the RMM, the API design uses different HTTP methods (such as GET, POST, PUT, and DELETE) to perform operations on different resources, which are each identified by unique URIs. This structure is more consistent and expressive than Level 1, which only uses one HTTP method for all operations, and aligns to correct semantic use. Level 2 fits the OpenAPI Specification in its standard format for describing RESTful API characteristics, including all URIs, HTTP methods, parameters, responses, and schemas. By using the OpenAPI Specification, a service can easily document and communicate its Level 2 maturity characteristics to its API consumer and developer audience. For example, here is a snippet of an OpenAPI definition that shows how to create, read, update, and delete a book resource using different HTTP methods:

```yaml
paths:
  /books:
    get:
      summary: Returns a list of books
      responses:
        '200':
          description: A JSON array of books
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Book'
    post:
      summary: Creates a new book
      requestBody:
        description: A book object
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Book'
      responses:
        '201':
          description: Book created successfully
        '400':
          description: Invalid book object
  /books/{id}:
    parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
    get:
      summary: Returns a book by ID
      responses:
        '200':
          description: A book object
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Book'
        '404':
          description: Book not found
    put:
      summary: Updates a book by ID
      requestBody:
        description: A book object
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Book'
      responses:
        '200':
          description: Book updated successfully
        '400':
          description: Invalid book object
        '404':
```

```
          description: Book not found
    delete:
      summary: Deletes a book by ID
      responses:
        '204':
          description: Book deleted successfully
        '404':
          description: Book not found
components:
  schemas:
    Book:
      type: object
      properties:
        id:
          type: integer
          readOnly: true
        title:
          type: string
        author:
          type: string
```

#### *5.14.1.4 Level 3 - Hypermedia Controls*

Level 3 is the highest level of maturity in the RMM, and it requires the use of hypermedia as the engine of application state (HATEOAS). API responses should include links to other resources or actions that are relevant to the current state of the application. For example, a response to a GET request for a product resource could include links to add the product to a shopping cart, write a review, or view similar products. Level 3 with HATEOAS provides API (capability) discovery which is a formal design goal.

Using hypermedia links at Level 3 has several benefits for API design:

- It decouples the client and the server, as the client does not need to know the URLs or the logic of the server in advance. The client can simply follow the links provided by the server based on the application state.
- It improves discoverability and usability of the API, as the client can easily navigate through the resources and actions available without requiring additional documentation or out-of-band information.
- It enhances evolvability and scalability of the API, as the server can change its URLs or logic without breaking existing clients. The server can also add new links or resources without affecting existing functionality.

However, using hypermedia links at Level 3 also poses some non-trivial challenges and pitfalls in the design execution:

- It increases the complexity and verbosity of the API, as the responses need to include additional information about the links and their semantics. The client also needs to parse and interpret the links correctly.
- It reduces performance and efficiency of the API, as the responses may contain redundant or unnecessary information that is not relevant to the client's needs. The client may also need to make multiple requests to follow the links and achieve its goals.
- It requires disciplined standardization and consistency of the API, as the links need to follow a common format and convention that is understood by both the server and the client. The API also needs to adhere to the OpenAPI Specification principles and goals, such as being clear, complete, correct, consistent, and current.

Designing an API at Level 3 of the RMM, consider these trade-offs and better practices:

- Use hypermedia links only when they add value and meaning to the API responses. Avoid adding links that are obvious, irrelevant, or redundant.

- Use descriptive and meaningful link relations that convey the semantics and purpose of the links. Prefer standard link relations defined by RFC 8288 or registered with IANA over custom ones.
- Use consistent and concise link formats that are easy to parse and follow. Prefer URI templates over absolute or relative URLs for parameterized links.
- Use appropriate media types that support hypermedia links and provide sufficient information about their structure and semantics. Prefer standard media types such as HAL, JSON:API, or Siren over custom ones.
- Use OpenAPI Specification (OAS) to document and validate your API design at Level 3. OAS supports hypermedia links through its Link Object and Callback Object components.

Two well-known industry APIs, provided by GitHub and Stripe, illustrate Level 3 qualities and are ideal references.

The GitHub API uses the HAL media type to provide hypermedia links in its responses. For instance, a response to a GET request for a repository resource includes links to its owner, collaborators, issues, pull requests, commits, branches, and others to establish a full associative graph. The client can follow these links to discover and interact with the API without relying on brittle references, URL locators, or traversal discovery/query logic.

The Stripe API personifies the RMM Level 3 aspects in conjunction with the OpenAPI Specification rigor by providing rich and consistent hypermedia controls throughout its resources. For example, the Stripe API uses the x-expandableFields vendor extension to indicate which fields can be expanded by passing an expand parameter in the request. This allows the client to retrieve more information about related resources without making additional requests. By using these hypermedia controls, the Stripe API achieves a high level of maturity and rigor in accordance with the OpenAPI Specification and the RMM Level 3 aspects. The Stripe API enables the client to discover and interact with its resources in a flexible and consistent way, without requiring prior knowledge of the model or resource structure.

### 5.14.2 Developer Experience

Developer experience (DX) is the term used to describe how easy and accessible it is for developers to use a certain product or service, such as an API. DX is influenced by many factors, such as the design, documentation, performance, demonstrative examples, reliability, and security of the API. A good DX can increase the adoption, retention, and satisfaction of the developers who use the API.

Use of the OpenAPI Specification, in part, is meant to improve the DX of an API. Use of a standard and language-agnostic way to describe the structure and behavior of an API is fundamental to DX. The OpenAPI Specification can help developers understand what the API does, how to interact with it, and what to expect from it. The OpenAPI Specification can also be used to generate documentation, code samples, mock servers, testing tools, and other resources that can simplify and streamline the development process. By fully using the OpenAPI Specification as a technical authoring convention, API designs can offer a more consistent and clearer DX to their developers.

#### *5.14.2.1 Documentation*

Technical documentation is an essential component of an API's developer experience, as it provides the necessary information and guidance for developers to use the API effectively and efficiently. Technical documentation can include various elements, such as:

- An overview of the API's purpose, features, and benefits
- A description of the API's architecture, design principles, and standards
- A reference of the API's endpoints, parameters, responses, and error codes

- A tutorial or a getting started guide with step-by-step instructions and code samples
- A list of common use cases and scenarios with examples and best practices
- A FAQ section with answers to frequently asked questions
- A changelog with updates on the API's version history and future plans

As mentioned, the OpenAPI Specification provides a comprehensive language-agnostic format for describing RESTful APIs. The OpenAPI Specification allows developers to define the API's structure, behavior, data models, and security schemes in a machine-readable and human-readable way. Using the OpenAPI Specification also enables the generation of interactive documentation tools, such as Swagger UI or Postman Explorer, which allow developers to explore, test, and debug the API directly from the browser.

Documentation quality is a key attribute of a mature API, as it reflects the API's reliability, usability, and maintainability. Documentation quality can be measured by various criteria, such as:

- Accuracy: The documentation should be consistent and up to date with the API's functionality and behavior and should not contain any errors or outdated information.
- Completeness: The documentation should cover all aspects of the API's usage and integration and should not omit any essential details or features.
- Clarity: The documentation should be written in a clear, concise, and coherent way, using simple and precise language and terminology.
- Organization: The documentation should be structured in a logical and intuitive way, using headings, subheadings, tables, lists, diagrams, etc. to facilitate navigation and comprehension.
- Accessibility: The documentation should be easy to access and consume, using responsive design, cross-browser compatibility, search functionality, etc. to enhance user experience as compared to a singular monolithic reference volume.

Quality technical documentation is vital for an API's developer experience, as it helps developers understand, learn, and use the API effectively and efficiently. By following the OpenAPI Specification and applying best practices for documentation quality, developers can create robust and user-friendly APIs that meet the needs and expectations of their consumers.

#### *5.14.2.2 Examples*

The OpenAPI Specification allows in-line examples and detailed descriptions for all the capabilities of an API design. This can help document APIs more clearly and consistently, as well as enable various tools and frameworks to interact with the API more easily. To best use and fully leverage OpenAPI support for in-line examples and detailed descriptions, consider these guidelines:

- For each operation, you can use the summary and description fields to give a brief overview and a detailed explanation of what the operation does, what parameters it accepts, what responses it returns, and what errors it may encounter. You can also use the tags field to group your operations by categories or functionalities.
- For each parameter, you can use the name, in, description, required, schema, and example fields to specify the parameter's location (query, path, header, or cookie), description, data type, validation rules, and an example value. You can also use the deprecated field to indicate if the parameter is no longer supported.

- For each response, you can use the description and content fields to describe the response's status code, media type, and schema. You can also use the examples field to provide one or more example values for the response body. You can also use the headers field to specify any response headers that are returned along with the response body.
- For each schema data model, you can use the type, properties, required, and example fields to define the data type, attributes, validation rules, and an example value for the schema. You can also use the description field to give a detailed explanation of what the schema represents and how it is used in your API.
- For all the core aspects of a well-defined API definition document, such as info, servers, paths, components, security, and externalDocs, you can use the description field to provide additional information and context for each element. You can also use the x- prefix to add any custom extensions that are not part of the standard OpenAPI Specification.

By using these fields consistently and accurately throughout your API definition document, you can create comprehensive and informative documentation assets. You can also use in-line examples in addition to detailed descriptions as illustration on common API consumption scenarios.

#### *5.14.2.3 Guardrails*

One of the features of the OpenAPI Specification is that it can describe validation, use limits, and restricted use of an API as a way to help inform API developers of existing guardrails.

Validation in this context refers to the process of checking if the requests and responses (or elements of them) conform to the schema defined in the OpenAPI document. For example, a content validation policy can enforce the size or payload of a request or response body against the API definition. Validation can help prevent errors, improve security, and ensure consistency in the API with meaningful and structured message patterns to guide the API developer.

Use limits refer to the restrictions on how many times or how frequently a client can call an API. Use limits can help protect the API from abuse, overload, or denial-of-service attacks. Use limits can be specified in the OpenAPI document using extensions, such as x-ratelimit-limit and x-ratelimit-remaining which support a variety of throttling strategies. Use limits can also be enforced by external tools or services, such as API gateways or proxies which may leverage these same conventions.

Restricted use refers to the conditions or requirements that a client must meet in order to access the API. Restricted use can help control who can use the API and for what purposes. Restricted use can be specified in the OpenAPI document using security schemes, such as apiKey, http, oauth2, or openIdConnect. Restricted use can also be implemented by external tools or services, such as authentication or authorization servers that extend the underlying API definition.

By describing validation, use limits, and restricted use of an API in the OpenAPI Specification, API developers benefit from a clear and consistent representation of the API's capabilities and constraints. API developers may also leverage various tools and services that support the OpenAPI Specification to generate code, test cases, documentation, or policies based on the definition.

### 5.14.3 Compatibility

Expect API consumers to have independent release lifecycles which require stability. API designs should be progressive based on extension and avoid (breaking) modifications. Compatibility is best preserved with an extension strategy and diligent planning. When compatibility cannot be preserved new versions should be introduced

while supporting older versions. Eventual decommission should be achieved through the use of deprecation techniques, finalized by official sunset (RFC 8594) definitions.

Compatibility is more easily maintained using the following techniques and design parameters to evolve an API definition:

- ***Fields additions must be optional.***
- ***Semantic definitions must be definitive, including data model member names: do not change field names.***
- ***Do not add restrictions.*** Complex constraints or basic validation may be loosened ***but not tightened:*** success validation should not later fail after an API revision.
- ***Enumeration reduction must coincide with deprecation support of old values.***
- Resource and operation substitution should be supported with redirection (status 301, moved permanently).

Schema extensibility is provided by the JSON schema attribute “additionalProperties” used with the OpenAPI Specification. ***Parameter and model definitions must not set the “additionalProperties” value to false.***

If version changes become unavoidable, the recommended approach is to add a resource variant rather than apply breaking modifications to an existing resource.

### 5.14.4 Performance

Performance aspects focus on response speed, capacity, and operation compute time with some overlapping dependencies between the three.

#### *5.14.4.1 Compression*

Response speed can be improved by reduction in response payload size. This may be achieved through compression, but at the cost of compute time and potentially processing capacity. Operation payloads should be compressed with multiple algorithm options, ideally tuned for net reduction of size and optimized speed (gzip, deflate, brotli).

#### *5.14.4.2 Caching Support*

Response speed, capacity, and compute times benefit from caching. Suitable resource operations, which can support caching, should implement caching.

Caching (key) determination may include, header contexts (including authorization identity), path parameters, query parameters, and any other discriminators to ensure that the cached data fulfills any partitioning requirements.

Caching mechanisms may include:

- HTTP response caching where the client caller infrastructure is responsible for the data cache, eliminating (API) server overhead completely.
- A caching intermediary such as a platform gateway where a secondary store, such as Redis, in conjunction with a policy engine, where the caching logic is defined outside of the API operation implementation.
- In-memory or distributed memory caching that is employed in the API operation implementation.

Caching strategies should include measurable and quantifiable metrics that are later used for tuning refinements.

#### *5.14.4.3 Trimming*

Data trimming reduces data payload size and, in some cases, graph traversal compute processing. Data trimming can be achieved by dynamic field selection, which is non-trivial (See *5.10.3 Content Negotiation*). Dynamic field selection may

have negligible benefit in single-item resource operations but becomes more impactful against collection-based resource operations. Collection pagination and filtering directives, with server implementation, also achieve trimming functions.

#### *5.14.4.4 Expansion (reduce round trips)*

In contrast to data trimming, expansion can be used to include related data results for a given resource model. The expansion may apply to a model (entity) graph or tangential items that are not part of a strict association.

In both cases, the performance benefit of data expansion is to reduce caller round trips with a server consequence of reduced compute time and net reduction on capacity pressure.

#### *5.14.4.5 Aggregate Stat Exclusion*

In complex data returns against collection-based resources, aggregate calculations should only occur when required and useful. Pagination implementations should allow for subset-only returns. Aggregate stats may be provided in a single operation as compared to an adornment to every paged-result data response.

Aggregate values that can be derived on the caller from the returned data should be excluded from server implementation/assignment.

Reducing aggregate calculations can improve response times, reduce compute time, and in some data architectures, reduce resource contention.

### 5.14.5 Discoverability

API operations should support discoverability and HATEOAS aspects as described in *5.14.1.4 Level 3 - Hypermedia Controls*.

For design-time definition, this is best achieved by adding OpenAPI Link objects to individual API operations. Operation links should be functional and leverage parameter and context assignments through dynamic expressions as described in *5.12.3 Links*.

When the design-time options do not adequately support the necessary association, as with collection-based resources or batch operation data results, runtime approaches should be used. In this case, a dedicated link property, either as a custom base-derived type object or as a URI-formatted string, should be used to establish the hypermedia controls.

Descriptors, adornments, and extensions may be established outside of a resource operation while aligning to the resource semantics. For example, as a project is updated over time, it will accumulate a tracking history with distinct versions as the modifications occur. These versions can be included as part of the resource model; however, the caller may not always need historical information. Embedding and expansion are valid options but require greater complexity in implementation. Using a hypermedia approach and adhering to a standardized *convention*, historical versions can be provided in a simpler and more efficient way:

- /projects
- /projects/{id}
- / projects/{id}/project_history

By extending the projects' resource semantic structure (as compared to the project data model), returning a project's history data is simpler to implement.

The following conventions should be used to semantically extend resources when there is no other existing convention or practice:

- _history – includes historical versions of the resource

- _permission – an access control model (summary) for the current context, for the resource
- _policy – an access control policy definition for the current context, for the resource
- _schema – a schema model definition for the current context, for the resource
- _status – an informational return, which can include a result, as long-running async support
- _count – an informational return to describe a resource collection size
- _paged – a paged-result set as compared to a simple array for a collection-based resource

In addition to the resource extension convention, each of the previous cases should also establish a data model structure convention. As mentioned in prior sections, well-known and established formats and standards should be used before defining a customized structure. Additionally, existing convention should be leveraged where appropriate when a more formalized data model does not exist (info, error, paged result, etc.).

### 5.14.6 Pagination

Pagination should be achieved through an offset-based or cursor-based approach. Pagination aggregate or result metadata should not be included as the response body. Paginated results should be returned as an array. Offset-based out-of-range conditions should not return error (status) results, instead returning an empty array. ***In the case of the cursor-based approach, invalid cursor values must return an error status.*** If page-level aggregate metadata is required, a common page response object should be used following the “Common Response Bodies” and related guidance in *5.13 Responses*.

Pagination should support high performance data retrieval for common use cases. Common scenarios are:

- User interface paged results
- User interface jump-to behavior
- User interface virtual/infinite scrolling
- User interface result read-ahead
- Parallel result retrieval
- Collection/Set-aware caching
- Result “max” limit thresholds

Pagination operations should provide metadata through resources or as custom headers. Headers should be used to describe constraints such as (page) result size limits (“X-Page-Size-Limit”). When needed, total item count should be returned using the convention ‘_count’.

/items?page_index=1&page_size=100 /items_count

Cursor-based pagination does not allow for parallel iteration. Parallel iteration for offset-based pagination may require the total item result count.

### 5.14.7 Long-Running Asynchronous Processing

***Any/all operations must support a long-running asynchronous* pattern.** However, long-running asynchronous implementation should be reserved for specific scenarios that are inherently different from continuous processing scenarios. An operation that leverages asynchronous processing and requires longer execution time does not necessarily qualify for this style of implementation.

Implementations that are high-volume, queue/message based, or require asynchronous processing of *remote* resources should utilize a long-running asynchronous processing pattern.

This pattern includes the delegation of a worker or queued task for the operation processing. ***As part of the incoming request, either through a defined header (e.g., "X-Request-Id") or operation parameter (e.g., "request_id"), the caller's action must be uniquely identified and also must be validated as unique upon generation if provided by the caller.*** Alternatively, the operation may generate the request identifier for return to the caller. ***In all cases involving a long-running asynchronous operation, a request identifier must be generated for later use in providing an action result.***

***All non-failing long-running asynchronous operations must return a status 202, accepted, to indicate the successful initiation of the request.*** A failure to properly initiate the operation should return a class 4XX error code and message. A successfully accepted, but later failing operation should return the appropriate 4XX status, later, in the eventual result return.

If the operation returns data as part of its action, the URL for that resource should be included in the "Content-Location" header (which differs from the "Location" header used in status 301, redirect scenarios). The value assigned to the "Content-Location" header should include the request identifier. It is recommended that the request identifier is assigned as the query parameter "request_id" as part of the convention.

The long-running asynchronous result should be provided from a separate resource. The result should adhere to an informational data return that indicates status while processing and the result value as a property upon completion.

Consider the scenario of video encoding:

- [POST] /raw_videos (a raw video is submitted for encoding)
- [GET] /raw_videos/{id}/video_status?request_id={request_id}
- [GET] /raw_video/{request_id}/video_status

In the [GET] operations, the subsequent return may leverage a unique key assignment in addition to the "request_id" ***but must rely on the "request_id" parameter value to reconcile to the original caller request.***

### 5.14.8 Batch Operations

Batch operations allow for efficient processing as well as the ability to validate dependently chained operations in a contained and transaction-scoped fashion. Batch operations should follow their non-batch resource operation equivalents. Batch operations can include any non-safe action that intends to modify one or multiple resource aspects. Batch operations may include long-running asynchronous processing.

***Batch operations that are not long-running asynchronous processes must return status 207, multi-status, when any portion of the batch completes successfully. If a failure occurs such that no processing is successfully completed, the action must return a status 4XX error result. Batch operations that are long-running asynchronous processes must return a status 202, accepted, under a non-failing initial acceptance condition.*** Upon completion, the status 207 and return results are expected.

Collection-based [POST] actions to add multiple items against a resource are suitable for batch operations.

Collection-based [PATCH] actions to partially update items within a resource are suitable for batch operations.

Collection-based [PUT] actions to add or update (with full replacement) multiple items against a resource are suitable for batch operations.

Collection-based [DELETE] actions that use filter expressions to define the deletion targets ***or*** the canonically extended [DELETE] with a request content body.

Batch [POST] and [PATCH] operations are not idempotent. Resubmission of a batch [POST] request would result in additions. Resubmission of a batch [PATCH] could violate concurrency requirements.

Batch [PUT] and [DELETE] operations are idempotent. Resubmission in these operations does not change the target state in comparison to the initial request submission.

Batch operation should use an informational result structure as presented in *5.13.1 Common Response Bodies*.

### 5.14.9 File Data

Unstructured data support should be provided using a web-protocol standardized file management approach.

File upload operations should use [POST] and [PUT] operations involving binary data and reserve [PATCH] operations for file metadata assignments. File request content models are defined with the appropriate media type with binary formatting. Consider this simple request body to describe a logo image structure:

```
requestBody:
  content:
    image/png:
      schema:
        type: string
        format: binary
```

Multiple-file uploads, single file uploads with metadata, and combinations therein should adhere to multipart request processing conventions. Multipart request bodies are also defined in OpenAPI via content media type. Consider the upload of a submittal file attachment:

```
requestBody:
  content:
    multipart/form-data:
    schema:
        type: object
        properties:
        submittal_id:
            type: integer
        user_id:
            type: integer
        file_name:
            type: string
        file_data:
            type: string
            format: binary
```

File upload and file data management strategies should distinguish between sizes that are suitable for in-memory processing and for large sized that require streaming mechanisms. File size operation constraints should be enforced to ensure proper treatment and as a safeguard against resource impacts.

File download operations should support web-optimized HTTP conventions in use of [HEAD] and [GET] actions associated with binary data transfers. Large file download operations should incorporate range requests and partial responses. A [HEAD] action is used to provide the resource file metadata required to stream its contents using partial responses including the total content length of the target. ***The requesting caller must include valid range request headers as part of the action. The [HEAD] action preamble must return a status 200, success, if the resource can be resolved and an error status 4XX if the resource metadata cannot be provided. The [GET] action must include the "Range" header indicating the data segment return using the following expression format:***

***bytes={lower bound}-{upper bound} (e.g., the first 2kb segment would be, Range: bytes=0-2047). A successful response must use a status 206, partial content, result. If the requested range by the caller is invalid, a status 416, requested range not satisfiable, must be returned.*** Other failing conditions should return the appropriate class 4XX status result.

### 5.14.10 Entity Flattening (Denormalization)

Denormalizing an entity graph structure is a useful technique in simplifying response payloads. Denormalization strategies may incorporate hypermedia techniques to allow resource expansion at the discretion of the caller while retaining the advantages of a compact-form model result.

When denormalizing an entity model, summarizing properties should be added to sufficiently capture the primary elements required by the caller. ***These properties must include the necessary associative key(s) to resolve the reference resources.*** These associating keys should be incorporated in any operation Link objects in the API definition. The member naming of the subordinate entity should be reflected in the properties representing the newly "flattened" structure. Consider an order item which references a product, and that product includes a property of category. Flattening the product association would take the following form:

```yaml
order_item:
  type: object
  properties:
    id:
      type: integer
      readOnly: true
    quantity:
      type: integer
    product_id:
      type: integer
    product_name:
      type: string
    product_category:
      type: string
```

Entity flattening standardized guidance is provided in *5.11.6 Graph Flattening*.

### 5.14.11 Entity Graph Expansion

In comparison to entity flattening, entity graph expansion is a technique to enrich a resource data model. From a design perspective, graph expansion should not be used to consolidate resource concerns. Graph expansion should be used and limited to the embedding of related resources. Candidates for expansion would be derived from subordinate resources sharing the same path segments. For example, consider the resource path of an order_item in relation to the order to which it belongs: /orders/{id}/order_items. Secondarily, consider the customer associated with the order. Both elements may be included on a potential order data model as a complex object property. The order_items clearly meet the graph expansion criteria while customer does not.

Entity graph expansion should be tailored to the developer experience ***which must include consideration to all other design aspects (later compatibility, performance, discoverability, etc.).***

### 5.14.12 Entity Member Trimming

Data aspect trimming by limiting returned fields is a non-trivial implementation exercise. This endeavor is most beneficial in collection-based, large-set data retrieval scenarios. Before attempting to support expressive or dynamic entity member trimming, semantic resource extension should be explored as an alternative. For example, consider a project data model with 30 useful fields as presented in the collection-based resource: /projects. A semantic extension to trim the information to 10 fields could be expressed as /project_summaries and include a hypermedia control targeting the detailed resource by /projects/{id}. This is a reasonable design-focused accommodation.

When implementing entity trimming strategies, the guidance provided in *5.10.3 Content Negotiation* should be used. The parameter "fields" should be used to encapsulate the trimming expression. The "field" parameter value should use Backus-Naur Form expression grammar. Successfully returned results that have applied trimming should return status 200, success. Status 206, partial content, may be returned indicating a successful result inclusive of the applied trimming.

### 5.14.13 Events

The OpenAPI Specification 3.0 introduces the concept of callbacks and webhooks to enable event functionality in APIs. This allows services to send asynchronous, out-of-band requests to other services in response to certain events, such as user actions, data changes, or system notifications. Event functionality can improve the workflow and efficiency of APIs by reducing the need for polling or manual updates. It can also help create responsive and dynamic systems that can react to events in real time.

#### *5.14.13.1 Callbacks*

Callbacks are defined as part of an operation that expects a callback URL in the request body or parameters. The callback URL is used to send a request in a predefined format and expected responses when an event occurs. For example, a service can send a callback request to notify when a long-running process has completed in order to retrieve the execution results.

Callbacks are initiated by an API operation request operation. The callbacks functions may be executed when an API request is initiated, in progress, or completed, either successfully or with an error. Some technical better practice design concerns when formulating API operation callbacks are:

- Use consistent naming conventions for the callback parameters, such as err, data, res, etc.
- Handle errors gracefully and provide informative feedback to the user or the caller of the API.
- Avoid nested callbacks or callback chains, employ techniques to flatten the code and improve readability.
- Document the expected input and output of the callback function, including the types, formats, and values with clear examples.
- Test the callback function with different scenarios and edge cases to ensure its reliability and robustness.

Additional persistence infrastructure and messaging support may be required to fully establish a comprehensive framework.

#### *5.14.13.2 Webhooks*

Webhooks are a way of implementing event-driven systems design, where a service can notify another service about defined events that occur asynchronously.

As of OpenAPI Specification 3.1.0, there is a new top-level element for describing webhooks that are registered and managed out of band. In comparison to callbacks, this element allows defining webhooks that are not triggered by an API operation, but by some other external event or mechanism. The webhooks element definition is similar to the paths element as described in *5.9 Resource Paths*, but instead of mapping to a path item object, it resolves to a webhook object by name. The webhook object contains a summary, a description, and one or more path items that describe the expected requests for each webhook.

This mechanism of out-of-band messaging without the caller initiation requirement enables a greater richness of design by allowing the API to take on aspects of an event-driven (responsive) system architecture.

When including API webhook functionality, automation, and RPA (robotic process automation) tool suites should contribute to the design requirements.

#### *5.14.13.3 Event Message Data*

A base model for event data should include all necessary information to reconcile the event origin, context, and initiation. Additional metadata may be added to event messages as needed. ***Event message data must not contain sensitive information.*** Timestamp and other event time reference metadata should be included as well as indicating the valid duration of the message when applicable.

A recommended base model for an event message should include:

- event_category – data modification, creation, deletion, addition, informational, error
- event_type – webhook, callback, push
- event_name – fully qualified webhook, callback, or push trigger name
- event_id – a universally unique “fingerprint” of the event firing
- operation_id – associated API operation if applicable
- request_id – the initiating API caller request action if applicable
- correlation_id – a correlation trace/flow/tracking identifier spanning discrete actions

Events are intended to be used and consumed in a variety of system scenarios with the ability to traverse multiple platforms. Events should be purpose-fit, deliberate, and “frugal” in deployment.

Eventing mechanisms incur resource cost and may introduce complexity. ***Event data must correspond to the dynamic aspect of the event context and add meaning information to that end.***

## 5.15 Operating an API

### 5.15.1 Publishing API Definitions

One of the benefits of the OpenAPI Specification is that it produces documentation that contains human-readable instructions for using and integrating with the API it defines. It provides essential information about the API endpoints, methods, resources, authentication protocols, parameters, and headers, as well as examples of common requests and responses. Publishing and properly maintaining API documentation has several benefits when operating a commercial or enterprise API:

- Self-service quick learning for API developers and API consumers to easily understand how to use the API effectively for a given goal.
- Less time and costs spent handling support calls and queries because users can find help and answers to their API documentation questions.
- Better developer experience because the API documentation is clear, well-structured, and consistent.
- Increased awareness and adoption of the API, because the target audience can experiment with the API, see its value, and utility.

Published and well-maintained API definitions, with accompanying documentation, is a formal design concern under operational responsibilities.

### 5.15.2 Policies

Policies are a collection of statements that are executed sequentially on the request or response of an API that are not part of the API definition nor the API implementations. They are used to modify or control the behavior and performance of the API, such as adding security, rate limiting, caching, logging, transformation, and more. Policies are

written in XML and can be applied at different scopes, such as global, product, API, or operation level.

As part of the wholistic API design, within operational requirements, polices are used as a vital and advanced method to achieve functional capabilities through configuration versus code.

Business or service level requirements may be established through policy definition and later monitored or audited to ensure compliance.

### 5.15.3 Instrumentation

Logging, diagnostics, instrumentation, and monitoring are essential components of an API operational design. They enable the API platform team, developers, and operators to track the performance, availability, reliability, and security of an API (platform).

For a deployed API basic logging is the process of recording events and data related to an API request or response with categorization align to the API definition (with API operation granularity).

Diagnostics supports all for the processing and analyzing of real-time logs and other sources of information to identify and resolve issues or errors.

Instrumentation targets the implementation aspect by adding code or tools to an API to measure and collect metrics such as response time, throughput, error rate, etc.

Monitoring is the consolidates and informs the process of observing and reporting the metrics and alerts generated by the instrumentation. Together, these components are a continuation of what was identified in policy and business service levels to ensure the API environment meets its functional and non-functional requirements.

***Operational design requirements must include formal instrumentation definition and conventions.***

# 6. Deliverable and Artifact Definitions

***This section includes a description and the required content for each required API deliverable.***

## 6.1 API Definition Documentation

### 6.1.1 Description

***A comprehensive document outlining the API's structure and functionality.***

### 6.1.2 Content

- ***API name and overview – A high-level summary of the API's purpose and functionality.***
- ***Versioning – Clearly defined versioning information, adhering to semantic versioning (e.g., v1.0, v2.1).***
- ***Authentication – Supported authentication methods.***
- ***Methods – A detailed list of supported HTTP methods (GET, POST, PUT, DELETE, etc.).***
- ***Request and response structure – Definition of expected request parameters, headers, and response formats (JSON, XML).***
- ***Rate limiting and throttling – API usage limits and rate-limiting mechanisms.***
- ***Error handling and status codes – Standardized error responses and HTTP status codes.***
- ***Example requests and responses – Sample API requests and expected responses for developers***
- ***Access control and permissions – Role-based access controls (RBAC) and claim-based permissions.***

- ***Token expiry and session management – Expiration policies and refresh token handling.***
- ***Security headers – Implementation of X-Frame-Options, X-Content-Type-Options, and Strict-Transport-Security headers.***

### 6.2 Testing and Validation Framework

#### 6.2.1 Description

***This document specifies testing methodologies to ensure compliance and performance.***

#### 6.2.2 Content

- ***API test cases for functional, integration, and security testing.***
- ***Validation scripts and automation tools.***
- ***Performance and load testing strategies.***

### 6.3 API Usage Guidelines

#### 6.3.1 Description

***This document provides API usage guidance for API consumers.***

#### 6.3.2 Content

- ***Instructions for using the API and included functionality.***
- ***Examples for normal and batch operations.***
- ***Quick start guides.***
- ***Error handling.***
- ***Optimization techniques.***
- ***AASHTOWare OpenAPI recipe cards.***

### 6.4 Operational and Monitoring Policies

#### 6.4.1 Description

***This document identified any operational considerations for API consumers, including suggested monitoring of Member environments associated with the use of the API.***

#### 6.4.2 Content

- ***Logging and diagnostics requirements.***
- ***API health checks and uptime monitoring.***
- ***Performance monitoring and rate limiting strategies.***

# 7. Appendix A

This appendix provides definitions for key terms and acronyms used throughout the AASHTOWare API Standards and Guidelines. These definitions ensure a clear and consistent understanding of technical concepts, methodologies, and industry best practices referenced in this document.

API (Application Programming Interface) – A set of rules and protocols that allow software applications to communicate with each other.

REST (Representational State Transfer) – An architectural style for designing networked applications using standard HTTP methods.

OpenAPI Specification (OAS) – A widely adopted format for defining RESTful APIs, enabling standardized API documentation and automation.

Endpoint – A specific URL where an API can be accessed to perform operations such as retrieving or updating data.

Payload – The data sent in an API request or response, often formatted in JSON or XML.

OAuth2 – A widely used authorization framework that enables secure access to APIs without sharing user credentials.

Versioning – A method of managing changes to an API while ensuring backward compatibility for existing integrations.

Rate Limiting – A technique used to control the number of API requests a client can make within a given timeframe to prevent system overload.

Pagination – A process of dividing large sets of data into smaller, manageable chunks for efficient retrieval in API responses.

Webhooks – User-defined HTTP callbacks that allow real-time data updates between applications.

## Acronyms

AASHTO – American Association of State Highway and Transportation Officials

API – Application Programming Interface

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation

OAS – OpenAPI Specification

OAuth – Open Authorization

REST – Representational State Transfer

TLS – Transport Layer Security

UUID – Universally Unique Identifier

XML – Extensible Markup Language

This page is intentionally blank.


# 3 - Appendices

This page is intentionally blank.

# AASHTOWARE STANDARDS AND GUIDELINES DEFINITION STANDARD

## S&G Number: 3.015.03.2S

## Effective Date: November 1, 2023

| Document History | | | |
|---|---|---|---|
| **Version No.** | **Revision Date** | **Revision Description** | **Approval Date** |
| 3.0 | 3/15/2018 | Made corrections and edits. Updated SharePoint info. Added info on reference documents, checklists, forms, templates, and aids. Removed references to ALF. | 6/22/2018 Approved by SCOA |
| 3.1 | 3/20/2019 | Clarified that T&AA will notify contractors when the updated Standards and Guidelines Notebook is available. | 7/30/2019 Approved by SCOA |
| 3.2 | 9/30/2023 | Updated the AASHTOWare logo and corrected broken links. | 10/02/2023 Approved by SCOA |

# Table of Contents

# 1. Purpose

The AASHTOWare Standard and Guideline Definition (ASGD) Standard defines the process used to establish, maintain, and publish AASHTOWare standards and guidelines and the Standards and Guidelines (S&G) Notebook.

This standard establishes an internal management process with requirements that impact the Technical and Applications Architecture (T&AA) Task Force and the Special Committee on AASHTOWare (SCOA). The standard does not define requirements that need to be followed by project or product task forces and contractors.

# 2. Responsibilities

Where most of the AASHTOWare standards focus on the responsibilities of the project or product task force and contractor, this standard focuses more on the responsibilities of the Special Committee on AASHTOWare (SCOA) and the Technical and Application Architecture (T&AA) Task Force. SCOA and T&AA are responsible for the majority of the work associated with this standard, while the responsibilities of the task forces and contractors are limited.

The responsibilities of all AASHTOWare participants impacted by this standard are summarized below: Additional details on these responsibilities are provided in the Procedures section of this document.

- The Special Committee on AASHTOWare (SCOA) is responsible for:
  - Approving strategic and tactical plan goals and objectives that drive the need for new and revised standards and guidelines.
  - Approving new and revised standards and the deletion of existing standards.
- The Technical and Application Architecture (T&AA) Task Force is responsible for:
  - Planning, developing, reviewing, revising, deleting, and maintaining AASHTOWare standards, guidelines, and notebook reference documents; and for performing analysis and research associated with these activities.
  - Developing, updating, reviewing, and maintaining checklists, forms, templates, and training/support aids to be used in conjunction with the Standards and Guidelines Notebook.
  - Reviewing all existing standards, guidelines, notebook reference documents, checklists, forms, templates, and training/support aids to ensure that each document is correct, up-to-date, and relevant.
  - Communicating and coordinating with the other AASHTOWare stakeholders to report status, provide information, and resolve reported issues associated with the standards and guidelines, the S&G Notebook, and any of the other notebook-related documents.
  - Approving new, revised, and deleted standards and guidelines. As noted above, standards are also approved by SCOA.
  - Approving minor changes in standards that do not change the intent or the required components of the standard.
  - Preparing and approving new versions of the Standards and Guidelines Notebook.
  - Maintaining the S&G Notebook repository ("*T&AA – S&G Notebook*" SharePoint workspace).
- The T&AA contractor performs many of the above tasks for the T&AA Task Force.
- The AASHTO T&AA PM and SCOA liaison participates in some of the above T&AA tasks.
- The project/product task force members are responsible for:

- Reviewing and reporting issues with all new or revised standards, guidelines, and related documents.
- Reviewing the current version of each standard and guideline as each is used and applied, and reporting any issues resulting from the review or use of a standard or guideline.
- The task force may choose to appoint designees (Technical Advisory Group members, Technical Review Team members, or contractor personnel) to assist in these efforts.

- AASHTO Staff is responsible for:
  - Reviewing and reporting issues with all revised or new standards, guidelines, and related documents.
  - Periodically reviewing the current version of each standard and guideline, and reporting any issues resulting from these reviews.

# 3. Deliverables and Artifacts

The following list provides the required deliverables and artifacts that are planned and created or updated by the T&AA Task Force as a result of the *Procedures* in this document. Definitions and content requirements are provided in the *Deliverable and Artifact Definitions* section of this document.

- *Standard or Guideline*
- *S&G Notebook Reference Document*
- *Standards and Guidelines (S&G) Notebook*
- *AASHTOWare Standard Template*
- *Checklists, Forms, and Templates*
- *S&G Training and Support Aids*
- *T&AA - S&G Notebook Workspace*
- *Next Notebook Shared Folder*

# 4. Procedures

This section describes the procedures used to develop, revise, delete, maintain, store, and publish individual standards and guidelines and the complete AASHTOWare Standards and Guidelines (S&G) Notebook. Many of the procedures are broken down into activities and some activities may be further broken down into tasks.

## 4.1 Begin Update Cycle for Next Version of S&G Notebook

After a new version of the S&G Notebook is published (see *Create and Publish the Standards and Guidelines Notebook*), the next notebook update cycle begins. At this point, prior to any new development and revision activities, the T&AA contractor performs the activities described below.

- Creates a new *Next Notebook* shared folder, "*Notebook FY YYYY,*" as described in the artifact *Next Notebook Shared Folder* definition.
- Copies the contents of the *Notebook* folder in the *Current Notebook* to the *Unchanged Documents* folder in the *S&G Development* library.
- Copies the contents of the *Templates and Other Materials* folder in the *Current Notebook* to the *Templates and Other Materials* folder in the *S&G Development* library.
- Copies the contents of the *Reference* folder in the *Current Notebook* to the *Reference* folder in the *S&G Development* library

- Moves any files in the *Pending* folder needed for the next version of the notebook to the *Changed Documents, Templates and Materials,* or *Reference* folders.
- Copies all folders and content in the *S&G Development* library to the new *Next Notebook* shared folder.

The *Notebook* folder should be empty after the above are completed. The *Changed Documents* folder should also be empty except when files were moved there from the *Pending* folder.

## 4.2 Develop/Revise Standards and Guidelines

This procedure defines the activities that should be performed to develop a new standard or guideline; or to revise an existing standard or guideline. The T&AA chairperson assigns an analyst the responsibility to develop a new standard or guideline; or to revise an existing standard or guideline. This analyst is normally a T&AA Task Force member or the T&AA contractor.

The decision to develop or revise a standard or guideline is normally made to support an objective in the AASHTOWare Strategic Plan, support an objective or project in the T&AA Work Annual Plan, support a finding from the annual QA process, resolve an issue found during the annual review of standards and guidelines (refer to *Maintain the Standards and Guidelines*), and /or resolve an issue reported by a stakeholder.

The following summarizes the activities for creating and revising standards and guidelines. The administrative activities to store, revise, and delete files on the *Next Notebook* shared folder and *S&G Development* library are normally performed by the T&AA contractor.

- If an existing new standard or guideline is to be revised, the following updates are made to the *Next Notebook* shared folder and the *S&G Development* library.
  - The existing Word document for the standard or guideline is copied from the *Unchanged Document* folder to the *Changed Documents* folder. If any secondary content files used to create the existing Word file are stored with the standard or guideline, these are also copied to the *Changed Documents* folder.
  - The existing Word and PDF files for the standard or guideline, and any secondary content files, are deleted from the *Unchanged Documents* folder.
  - The Word version of the standard or guideline is provided to the assigned analyst as the starting document for revisions, along with the secondary content files, when applicable.
- If a new standard or guideline is to be created, the assigned analyst creates the initial Word document by copying the *AASHTOWare Standard Template*. Refer to the *Standard or Guideline* artifact definition for information on the specific content required in each standard or guideline, and for information regarding the location and use of the AASHTOWare Standard Template.
- If a new standard or guideline is developed to replace an existing standard or guideline, the Word and PDF files for the existing standard or guideline that will be replaced, and any secondary content files, are deleted from the *Unchanged Documents* folder of the *Next Notebook* shared folder and the *S&G Development* library.
- Throughout the development cycle, the assigned analyst should provide preliminary versions of the new/revised standard or guideline to the T&AA Contractor to store in the *Changed Documents* folder of the *Next Notebook* shared folder and the *S&G Development library*.
- When the standard or guideline is assigned to the T&AA contractor, all development and update activities should be made directly on the *Next Notebook* shared folder and routinely uploaded to the *S&G Development* library.

- Collect documentation from any prior analysis and documentation on best practices and recommended artifacts that are applicable to the objectives of the proposed new/revised standard or guideline.
- Perform the appropriate industry research required to develop or revise the standard or guideline.
- Review the applicable methods that are currently being used and artifacts being created by the project/product task forces, contractors, AASHTO staff, T&AA members, and/or stakeholder organizations.
- Use the objectives, best practices, methods, artifacts, and industry research, collected in the above steps, to define and/or revise the appropriate procedures needed for the new/revised standard or guideline.
- Develop each procedure with the appropriate level of detail needed to understand how to use the procedure. If needed for clarity or simplicity, divide the procedure into lower level activities and tasks.
- Develop/revise each standard so it is clear on what is expected or required from product/project task force members, contractor staff, SCOA, T&AA, and other stakeholders; and what elements of the standard are optional or may be customized.
- Include the definition and content of the required deliverables and artifacts that shall be produced to comply with each standard. Optional artifacts may also be included. Define these in the standard section defined by the standard template.
- Each standard should clearly describe the procedures and activities that shall be followed; the required major deliverables and artifacts; and the required submittals, approvals and review gates, including who is responsible for producing each deliverable/artifact and the type of review and approval required. The required elements of each standard should be clearly identified and should be summarized at the beginning of each standard. *All new and revised standards will use red italicized text to identify the required elements.* ***New requirements that were not in the prior version are shown in bold italicized text.***
- The procedures, activities, deliverables, and artifacts for each standard that are not required are based on best practices and are recommended. These may be implemented or customized as seen appropriate by the by the project/product task force and contractor. Some standards may also provide additional details on what may be customized and the approach that may be used for customizing certain elements.
- In the case of guidelines, all procedures and artifacts should be defined as recommendations or best practices.
- Define the applicable technical specifications for the standard or guideline. For standards the technical specifications are required, and for guidelines the specifications are recommended.
- Document the purpose/objectives of the standard or guideline and the stakeholder responsibilities in the appropriate sections defined by the AASHTOWare Standard Template.
- When referencing another section within the standard or guideline being created or revised, insert a cross reference as a hyperlink to jump to the referenced location. This activity is normally performed as an edit by the T&AA contractor.
- When referring to another standard, guideline, or reference document outside of the standard or guideline being created or revised, insert a dummy hyperlink with the name of the external standard or guideline. Each dummy hyperlink should point to the top of the document. All dummy hyperlinks will be updated to point to the correct notebook page during the *Create and Publish the Standards and Guidelines Notebook* procedure. This activity is normally performed as an edit by the T&AA contractor.

- When referencing checklists, forms, and templates, insert/revise hyperlinks that point to the existing location (URL) or planned location (URL) of the most recent document on the AASHTOWare web site. This activity is normally performed as an edit by the T&AA contractor.
- When referencing external web sites, the URL should be verified by the assigned analyst. The T&AA contractor will verify the URL again during the creation of the notebook.
- If revising an existing standard or guideline:
  - Increment the version number part of *S&G Number* as described in the *Standard or Guideline* artifact.
  - Update all references to the S&G Number and revision date.
  - Prepare a change summary containing one or more sentence or bullet points that describe the revisions made.
- For a new standard or guideline:
  - Assign a new *S&G Number* and file name as described in the *Standard or Guideline* artifact. The T&AA contractor normally assigns new S&G Number.
  - Add the name of the standard or guideline, S&G Number, and initial date in the document locations defined by the standard template.
  - Create a new document summary (also referred to as a change summary) with several sentences or bullet points describing purpose/objectives of the new standard or guideline.
- Complete/update the cover page including:
  - Standard or guideline name
  - S&G Number and version
  - Effective date for standards and revision date for guidelines. The effective date for each new and revised standard should be the target effective date for the next version the notebook.
  - Document history entry for the new/revised standard or guideline (revision date, summary, approval date, and approval party). The summary entry should include a shorter version of the information in the change summary. The approval date and party are entered after final approval. To keep the cover page to one page, previous document history entries may be removed as needed.
- If a new standard or guideline is developed to replace an existing standard or guideline, the change summary for the new standard/guideline should identify the standard or guideline that will be eliminated and the standard or guideline that replaces it. Both should be identified by name and S&G number.
- If a checklist, form, or template is created, replaced or revised in conjunction with a standard or guideline, the change summary information for the standard or guideline should address the changes checklist, form, or template.
- Change summary information can be documented in any format with one file per change summary or as a combined file with other multiple sets of change summaries. All change summary files should be stored in the *Changed Documents* folder on both the *Next Notebook* shared folder and the *S&G Development* library.
- The change summary file(s) are temporary and will be used to create the overall notebook change summary when the next version of the S&G Notebook is created. The change information is also used in correspondence when requesting review and approval of revised and new standards and guidelines.

### 4.3 Develop/Revise Checklists, Forms, Templates, and Training/Support Aids

In addition to developing and revising content for the S&G Notebook, the T&AA Task Force and/or the T&AA contractor prepares and updates checklists, forms, templates, and training/support aids that are used in conjunction with the S&G Notebook. Most checklists, forms, and templates are used to prepare artifacts and deliverables described in the notebook; where, training and support aids are used to assist stakeholders in using and understanding the S&G Notebook or specific standards/guidelines.

Examples of a checklist, form template and training/support aid are the Contractor Backup Checklist, Review Gate Approval Request Form, Project Work Plan Template, and Summary of S&G Notebook Requirements document.

The need to develop a new checklist, form, or template is normally defined while developing or revising a standard or guideline and a uniform approach to creating an artifact or deliverables is needed. Revisions to existing checklists, forms, and templates are identified in conjunction with revisions to the standard and guideline that references them.

The need to develop a new training or support aid is normally defined when a common compliance problem is identified for one of more standards or when a common usage or understanding issue for one or more standards and/or guidelines is identified. The need to revise an existing aid may be to align the aid with a new or revised standard/guideline, correct known problems/issues, and/or to address newly identified areas of compliance, usage or understanding.

In most cases the need to develop or revise a checklist, form, template, or training/support aid is defined as a result of the annual QA process, annual review of the existing standards and guidelines, T&AA liaison discussion with stakeholders, and/or stakeholder reviews of new/revised standards and guidelines,

All existing checklists, forms, templates, and training/support aids that are used with the current version of the S&G Notebook are stored the *Current Notebook* library in the *Templates and Other Materials* folder. Most checklists, forms, templates, and training/support aids are also stored on the AASHTOWare web site and the AASHTOWare SharePoint workspace.

The following summarizes the activities for creating and updating checklists, forms, templates, and training/support aids.

- All existing checklists, forms, templates, and training/support aids that are referenced within the S&G Notebook, posted to the AASHTOWare web site or a SharePoint site, and/or those distributed to AASHTOWare stakeholders through other means shall be kept up to date with the current version of the S&G Notebook.
- As noted above, at the beginning of the update cycle for a new notebook, all existing checklists, forms, templates, and aids are copied to the *Templates and Other Materials* folder in both the *S&G Development* library and the *Next Notebook* shared folder.
- All new and revised checklists, forms, templates, and training/support aids are maintained in the *Templates and Other Materials* folder of the *Next Notebook* shared folder. Throughout the development cycle, the T&AA contractor should frequently upload preliminary versions to the *Templates and Other Materials* folder of the *S&G Development* library.
- Each new and revised document should include the revision date in the file name as described in the *Checklists, Forms, and Templates* and *S&G Training and Support Aids* artifact definitions.
- The revision date should be within the content of each document, where applicable, such as in the page footers.
- The content of each checklist, form, template, and training/support aid is unique and does not use a standard format. The content is normally based on the content of a

specific artifact or deliverable or on the requirements/use of one or more standards and guidelines.

- If a checklist, form, template, or training/support will not be used with the next version of the notebook, it should be deleted from *Templates and Other Materials* folder of the *Next Notebook* shared folder and the *Templates and Other Materials* folder of the *S&G Development* library.
- Normally, these documents are created, maintained, and revised using Microsoft Word. These could also be created in other formats, such as PowerPoint and Excel.
- Checklists, forms and templates are typically used by the task forces and contractors in the source format (normally Word); where training and support aids are typically used in PDF format.
- If multiple file types exist for a document (such as Word and PDF), all files types should be maintained on both the *Next Notebook* shared folder and the *S&G Development* library.
- In most cases, similar fonts and styles are used to those used in the S&G Notebook.
- As noted above, change summary information for a new, replaced, and revised checklist, form, or template should be prepared with the standard or guideline change summary that references the checklist, form or template.
- Training/support aids are not normally referenced in the S&G Notebook, so change summary information to include in the overall notebook change summary is not needed. However, some type of summary will typically be needed when distributing new and revised training/support aids. Change summary files for training/support aids should be stored on the *Templates and Other Materials* folder in both the *S&G Development* library and the *Next Notebook* shared folder.

## 4.4 Prepare Change Summary for Standard and Guidelines to be Deleted

If a standard or guideline will be deleted from the next version of the notebook, and will be not be replaced, a change summary should be prepared that identifies the standard or guideline by name, number, and version; and provides a brief description explaining why the standard or guideline is being deleted.

The change summary file for the deleted standard/guideline should be stored on the *Changed Documents* folder in both the *Next Notebook* shared folder and the *S&G Development* library.

As with those for new and revised standards and guidelines, deletion change summaries will be included in the overall change summary for the next notebook and with review/approval correspondence.

## 4.5 Review and Approve Standards, Guidelines, and Other Documents

### 4.5.1 Obtain T&AA and Stakeholder Feedback

During the revision or development of a standard or guideline, the lead analyst should initiate reviews by T&AA and AASHTOWare stakeholders, obtain feedback, and make modifications, as required, to address the feedback. Checklists, forms, templates, and training/support aids should also be reviewed.

The following summarizes the activities for reviewing standards, guidelines, checklists, forms, templates, and training/support aids.

- Make presentations and communicate with T&AA Task Force members, T&AA contractor, AASHTO PM, and SCOA liaison, as required, to review new and revised standards and guidelines, provide status information, and collect comments and issues. The T&AA review of the standard or guideline should include review for gaps, overlaps, and proper integration with other standard and guidelines.

- Checklists, forms, and templates that are referenced in a standard or guideline are normally reviewed by T&AA at the same time as the referencing standard or guideline.
- Training and support aids are normally reviewed by T&AA in a similar manner to that of standards and guidelines. Training and support aids are normally reviewed separate from the review or standards and guidelines.
- Address the issues and comments from the T&AA review and revise each standard, guideline, checklist, form, template, and training/support aid, as required. Several T&AA review iterations may be needed before stakeholder review is scheduled.
- Provide heads-up information to SCOA regarding the purpose and status of new, revised, and deleted standards and guidelines and the projected timeline for requesting approval of standards. This information is normally provided at T&AA meetings where joint sessions are held with SCOA.
- After T&AA review is completed, the T&AA Chairperson will distribute each new and revised standard or guideline to AASHTO Staff and the project/product task force chairpersons. Change summary information and any revised/new checklists, forms or templates that are referenced by the standards and guidelines should also be provided. Reviews should be requested from the task forces, contractors, and AASHTO staff; and comments and issues should be solicited for return to T&AA by a specific date. The distribution of the review documents and the return of comments and issues are normally accomplished by email.
- As with T&AA review, training and support aids are normally provided to stakeholders for review independently of the standard and guideline reviews.
- Prior to beginning the above stakeholder review, the T&AA contractor copies all standards, guidelines, checklists, forms, templates, and aids to be reviewed, plus the change summary information, to a folder on the "*T&AA – Executive*" workspace. The T&AA chairperson distributes the documents from this folder. Review copies are typically provided in PDF format.
- The stakeholders should review each standard, guideline, checklist, form, template, and training/support aid for how it satisfies the purpose/objectives, use and understanding of the document, applicability to each task force, and applicability to the AASHTOWare organization. For a standard, the review should also determine if the standard introduces any problems or issues for the stakeholders.
- The T&AA should communicate with the stakeholders, as required, to provide additional information and answer questions. If needed, presentations should be made to assist with communication and understanding.
- When issues and comments are received from stakeholders, T&AA will review them and address those that are warranted. Stakeholder reviews may be repeated, as required, to address major issues.
- During the above activities, T&AA should also advise the stakeholders of any existing standards or guidelines that will be replaced by a new or revised standard and those that will be deleted.
- Any changes to standards, guidelines, change summaries, checklists, forms, templates, and training/support aids made during the review procedure should applied to the appropriate folder(s) (*Changed Documents* and/or *Templates and Other Materials)* on the *Next Notebook* shared folder and uploaded to the *S&G Development* library.
- T&AA and stakeholder comments and feedback should be saved as deemed appropriate by the T&AA Task Force.

Note: Since guidelines do not require SCOA approval; SCOA should be provided an opportunity to review and comment on new and revised guidelines following the stakeholder review.

### 4.5.2 Obtain T&AA Approval of Standard or Guideline

After the above reviews are completed, and the appropriate revisions are made, each revised/new/deleted standard and guideline shall be approved by T&AA. Checklists, forms, templates, and training/support aids do not require T&AA approval. However, since checklists, forms, and templates are referenced by specific standards and guidelines, they can always be reviewed with approval requests.

The T&AA approval activities for standards and guidelines are summarized below.

- All new, revised, and deleted guidelines shall be approved by the T&AA Task Force. SOCA approval is not required for guidelines.
- Other than the minor revisions to standards, as defined below, all other revisions to standards, new standards, and deleted standards shall be approved first by T&AA and then submitted to SCOA for approval.
- Minor revisions to standards which only affect references, hyperlinks, spelling, grammar, format, or readability of the document, and do not change the meaning or the impact on stakeholders, may be approved by the T&AA without SCOA approval.
- The T&AA Task Force Chairperson, at his/her discretion, has the authority to act on behalf of the task force when it is not feasible for the task force to convene in a timely manner to act on S&G Notebook minor changes. This type of approval will only occur during the final preparation of the S&G Notebook.
- Minor revisions to both standards and guidelines will normally include a decimal update to the standard version number, such as “02.1”, instead of a full version number, as in “3.0”. Refer to the *Standard or Guideline* artifact definition.
- Any changes to standards, guidelines, change summaries, checklists, forms, and templates made as a result of the T&AA approval request should applied to the appropriate folder(s) (*Changed Documents* and/or *Templates and Other Materials)* on the *Next Notebook* shared folder and uploaded to the *S&G Development* library.
- If needed, the T&AA approval procedure is repeated, as required, to address major issues.
- T&AA comments and feedback should be saved as deemed appropriate by the T&AA Task Force.

### 4.5.3 Obtain SCOA Approval of Standard

After T&AA has approved a standard, with the exception of minor changes (as noted above), a request to approve each new, revised, or deleted standard shall be submitted to SCOA for final approval. The SCOA approval activities for standards are summarized below.

- The T&AA contractor copies each new or revised standard to be approved, plus the change summaries (including those to be deleted), to a folder on the “*T&AA – Executive*” workspace. New and revised checklists, forms, and templates referenced by the standards are also copied to this folder. The T&AA chairperson distributes the documents from this folder. Approval copies of standards and guidelines are typically provided in PDF format.
- The T&AA chairperson initiates the approval process by preparing a letter or email to the SCOA chairperson requesting SCOA approval of one or more new, revised or deleted standards. The letter/email shall include the change summary information for all revised and new standards to be approved. Standards that will be replaced

should be referenced with the standard that will replace them. If standards are to be deleted without replacement, the letter/email shall request approval for the deletion(s) and the reason for each deletion.

- The letter or email is sent to the SCOA chairperson and copied to the SCOA liaison to T&AA, AASHTO Staff, T&AA members, and the T&AA contractor.
- The T&AA AASHTO PM copies the letter/email and the standards to the appropriate location for SCOA balloting along with any referenced checklists, forms and templates.
- SCOA approves or rejects each standard and the SCOA chairperson notifies the T&AA chairperson of the approval decision. When rejected, the reason for rejection is included in the communication to the T&AA chairperson.
- If a standard is rejected, T&AA reviews the reason for rejection, makes the appropriate changes to the standard and change summary, and repeats the approval process.
- Any changes to standards, guidelines, change summaries, checklists, forms, and templates made as a result of the SCOA approval request should be applied to the appropriate folder(s) (*Changed Documents* and/or *Templates and Other Materials)* on the *Next Notebook* shared folder and uploaded to the *S&G Development* library.
- SCOA comments and feedback should be saved as deemed appropriate by the T&AA Task Force.

## 4.6 Prepare Approved Standards and Guidelines for Notebook Creation

The following activities are performed after new, revised, and deleted standards and guidelines are approved and updates are completed to checklists, forms and templates. The activities are normally performed by the T&AA contractor.

### 4.6.1 Update Standard and Guideline Cover Pages

After each standard or guideline is approved, the T&AA contractor should update the document history on the cover page with the approval date and approval body (SCOA or T&AA). The cover page updates should be applied to the *Changed Documents* folder on the *Next Notebook* shared folder and uploaded to the *S&G* ***Development*** library. At this point the revision date should be removed from the file name of each approved standard and guideline.

### 4.6.2 Remove/Save Prior Versions of Standard and Guidelines

All prior versions of each approved standard and guideline should be removed from the *Changed Documents* folders on both the *Next Notebook* shared folder and the *S&G Development* library. If prior versions need to be saved, they should be stored at alternate location.

Standards and guidelines that were approved for deletion should be removed from the *Unchanged Documents* folders on both the *Next Notebook* shared folder and the *S&G Development* library.

The most recent version of revised/new checklists, forms, and templates should be stored in the *Templates and Other Materials* folder on both the *Next Notebook* shared folder and *S&G Development* library with the revision date in the file name. Prior versions should be removed from this folder. If prior versions need to be saved, they should be stored at alternate location.

Any standard, guideline, and/or other document that is still undergoing development/update, and will be not be included in the next version of the notebook, should be moved to the *Pending* folder on both the *Next Notebook* shared folder and *S&G Development* library.

## 4.7 Review and Update S&G Notebook Reference Documents

S&G Notebook reference documents are created, maintained, and revised to create content for the S&G Notebook, in addition to that of standards and guidelines. Examples of reference documents include the notebook cover page, cover letter, summary of changes, table of contents, S&G overview, and the S&G Glossary. Most updates to notebook reference documents are made specifically for creating the next version of the S&G Notebook and do not require review or approval by T&AA prior to creating the notebook. These updates normally occur after the above procedures for revising, deleting, creating, reviewing, and approving standards and guidelines are completed.

Most of reference documents are created and updated by the T&AA contractor; however, some may be created/updated by a T&AA Task Force member. The administrative activities to store, revise, and delete files on the *Next Notebook* shared folder and *S&G Development* library are performed by the T&AA contractor.

### 4.7.1 Update Reference Documents

The following summarizes the update procedure for reference documents.

- As previously noted, all existing reference documents are copied from the *Current Notebook* library to the *Unchanged Documents* folder in both the *S&G Development* library and *Next Notebook* shared folder at the beginning of the notebook update cycle.
- There are some *Standard Revisions to Reference Documents* (next activity) that are made to specific reference documents with each new notebook; however, all existing reference documents should be reviewed to determine if any additional changes are needed.
- When an existing reference document is updated, it should be deleted from the *Unchanged Documents* folder of the *Next Notebook* shared folder and the *S&G Development* library. The revised document should be stored in the *Changed Documents* folder of the *Next Notebook* shared folder and uploaded to the *S&G Development* library.
- New reference documents are normally not created; however, if a new reference document is created, it should be should be stored in the *Changed Documents* folder of the *Next Notebook* shared folder and uploaded to the *S&G Development* library.
- As described in the *S&G Notebook Reference Document* artifact definition, reference documents are assigned a *S&G Number* and the S&G Number is included at the beginning of the file name.
- Each reference document is generally unique and does not include standard content or use a standard format.
- Most reference documents only undergo the minor revisions required for creating the next notebook. Some simple reference documents, such as section separator pages may never change.
- Where applicable the steps from the *Develop/Revise Standards and Guidelines* procedure should be followed when creating and revising notebook reference documents. For example:
  - If the assigned analyst is a T&AA member, preliminary versions of the new/revised reference document should be routinely provided to the T&AA Contractor to store in the *Changed Documents* folder of the *Next Notebook* shared folder and uploaded to the *S&G Development library*.
  - When the reference document is assigned to the T&AA contractor, all development and update activities should be made directly on the *Next Notebook* shared folder and routinely uploaded to the *S&G Development* library.

- Internal references in a reference document should be inserted as cross reference hyperlinks and external links to other standards, guidelines, and reference documents are inserted as dummy hyperlinks.
- Change summary information should be created for all revised and new reference documents; but is typically very brief. Change summaries should also be created if a reference document will be deleted.
- Cover pages like those used on standards and guidelines may be needed for some reference documents.
- The version number part of the S&G Number is not normally incremented for reference documents except for those that include a cover page.
- Revision dates should be inserted on the cover pages and page footers, where applicable.

- As noted above, most updates to reference documents will not require T&AA review or approval. When approval is needed, cover pages should be updated with the approval date and approval party.
- The more common, routine updates specifically made for notebook updates are normally reviewed by T&AA with the review of the next version of the notebook.
- Given the current purpose and content of reference documents, separate stakeholder and SCOA review is not normally needed, nor is SCOA approval.
- Reference documents do not require SCOA review or approval.
- Reference documents do not typically include a revision date in the file name; however, if a date is included it should be removed prior to creating the notebook. Prior versions should be removed from the *Changed Documents* folder of both the *Next Notebook* shared folder and the *S&G Development library*. If prior versions need to be saved, they should be stored at alternate location.
- The Word and PDF files for references files that will be deleted from the notebook, should be removed from the *Unchanged Documents* folder of both the *Next Notebook* shared folder and the *S&G Development library*.

### 4.7.2 Standard Revisions to Reference Documents

As noted above, each new version of S&G Notebook will include following standard revisions to reference documents.

- Update the effective date on the Notebook Cover Page and Cover Letter.
- Update the Summary of Changes document as follows.
  - Update the effective date.
  - Delete all bulleted change summary information from the previous notebook
  - Add new bullet points that summarizes the revisions made to existing standards and guidelines. Add bullet points to summarize new standards and guidelines and add bullet points to summarize standards and guidelines that have been replaced and deleted. This information should be based on the change summary information created in the above steps; however, some editing may be needed.
  - Insert dummy hyperlinks to each revised/new standard or guideline referenced in the Summary of Changes documents. Each dummy hyperlink should point to the top of the document. All dummy hyperlinks will be updated to point to the correct notebook page during the *Create and Publish the Standards and Guidelines Notebook* procedure.

- Add bullet points to summarize the changes (revisions, deletions, additions) made to all reference documents. If needed, add dummy hyperlinks to each reference document (as described above).
- The order of the Change Summary bullet points should match the order of the revised/new documents in the S&G Notebook.
- Remove all temporary change summary information files from the *Changed Documents* folder of both the *Next Notebook* shared folder and the *S&G Development library*.

- Add/revise entries to the Table of Contents, as required. Each table of contents entry should include a dummy hyperlink (as described above) to the referenced standard, guideline, or reference document. As noted below, all standards, guidelines, and reference documents are ordered by S&G Number.
- Update the Standard and Guidelines Overview document, as required. This document provides an overview of the notebook content with a short summary of each standard, guideline, and reference document. Each standard, guideline, and reference document summarized in the overview includes a dummy hyperlink (as described above).
- Update the Standards and Guidelines Glossary document, as required. The glossary includes definitions of terms used throughout the S&G Notebook.

## 4.8 Create and Publish the Standards and Guidelines Notebook

This procedure is initiated when a new version of the S&G Notebook is to be created. A new notebook may be published at any time; however, the current target date for annual updates to the notebook is September 1. All new and revised standard and guidelines approved during the current notebook update cycle will be included in the new notebook, along with updated reference documents, any new reference documents, and all unchanged standards, guidelines, and reference documents.

Since standards shall be complied with by their effective date, the effective date of all new and revised standards approved for the next notebook must match the new notebook's effective date. Where the effective date of a standard may require a new notebook to be published prior to annual update, guidelines are not binding and normally will not drive the publication of a new notebook.

### 4.8.1 Verify Documents for New Notebook are in Correct Location

The T&AA contractor verifies following:

- The Word (.docx) documents for all new and revised standards and guidelines approved for the next version of the notebook are stored in the *Changed Documents* folder of the *Next Notebook* share folder and *S&G Development* library. The Word documents for all updated reference documents should also be stored in the *Changed Documents* folder.
- The Word (.docx) documents for all existing standards, guidelines, and reference documents that were not revised, deleted or replaced are stored in the stored in the *Unchanged Documents* folder of the *Next Notebook* share folder and *S&G Development* library.
- A PDF document is created for each Word document in the *Changed Documents* and *Unchanged Documents* folders.
- Ensure there is no duplication of documents in the *Changed Documents* and *Unchanged Documents* folders.
- Ensure that the Word files (and other types used) for the latest version of each checklist, form, templates, and training and support aids used with the next version

of the notebook are stored *Templates and Other Materials* folder of the *Next Notebook* share folder and *S&G Development* library. Training and support aids should have both a Word and PDF file.

- Ensure that any secondary content files used to create the Word files are in the folders with the Word document are in the *Reference* folder of the *Next Notebook* share folder and *S&G Development* library.
- Ensure that only files needed for the next version of the notebook are included in the *Reference* folder of the *Next Notebook* share folder and *S&G Development* library.
- All files are named as described in the *Deliverable and Artifact Definitions* section of this document.

If any Word, PDF, or other type of file are missing, in the wrong folder, or note named correctly, the T&AA contractor should resolve these issues before proceeding. Any files not needed for the next version of the notebook should be removed.

### 4.8.2 Create Next Version of Notebook

All Word and PDF files in the *Changed Documents* and *Unchanged Documents* folders are copied to the *Notebook* folder on the *Next Notebook* shared folder. This includes all new, revised, and unchanged standards and guidelines, except those being replaced or deleted, plus the cover page, cover letter, table of contents, overview, separator pages, glossary, and any new reference documents.

The new notebook is created by merging all PDF files in the *Notebook* folder into a single PDF document using a PDF editing tool defined in the *Technical Requirements* second of this standard. The notebook PDF is organized in sequential order of the PDF file names using the S&G Number. The PDF editing tool creates a bookmark in the notebook PDF file at the beginning page of each source PDF file.

After reviewing the notebook document and verifying that all content is correct and in the right order, the new notebook file shall be stored and named as described in the *Standards and Guidelines (S&G) Notebook* artifact definition. All dummy hyperlinks should be edited to point to the correct page in the notebook PDF file. After the dummy links are edited, all hyperlinks in the notebook PDF file should be checked and corrected as required. The new notebook PDF file is stored on the root/top level folder of the *Next Notebook* shared folder and uploaded to the *S&G Development* library

Corrections for hyperlinks and other issues are normally made in the source Word documents and new PDF files are created for each updated Word document. The updated Word and PDF files are updated in *Changed Documents, Unchanged Documents* folders are copied to the *Notebook* folder. These updates are made to the *Next Notebook* shared folder and uploaded to the *S&G Development* library.

The activities to create, review, edit, and correct a new notebook PDF file are repeated until all issues are resolved.

The last step is to update the initial view under document properties in the notebook PDF. The initial view is typically set to:

- Navigation: Bookmarks pane and page
- Page View: Single Page
- Zoom: Automatic or Fit Height
- Always open document to page: 1

### 4.8.3 Move Current Notebook to History

On or before the effective date of the new notebook, the previous version of the notebook should be copied from the *Current Notebook* library to a folder in the S&G

History library. Refer to the *S&G Notebook Workspace* artifact for the naming convention for *S&G History* folders.

After verifying that all content was copied correctly the S&G History library, all subfolders and content is the Current S&G Notebook library should be deleted.

### 4.8.4 Move New Notebook to Current

After the copying the previous version to history, the following root folders plus the new notebook PDF file in the *S&G Development* library are copied to the *Current Notebook* library under the root folder. The new notebook should be in place by the effective date.

- *Notebook*
- *Templates and Other Materials*
- *Reference*
- *SG_Notebook_mmddyyyy.pdf* (current version of S&G Notebook)

After verifying that all content was copied correctly to the *Current Notebook* library, all subfolders and content in the *S&G Development library* should be deleted except for the *Pending* folder.

### 4.8.5 Update AASHTOWare Web Site and AASHTOWare Workspace

The new S&G Notebook and the current version of all checklists, forms, and templates that are referenced in the notebook are posted to the AASHTOWare web site and the *Everyone* SharePoint work space prior to the effective date.

### 4.8.6 Notify Stakeholders

After the new version of the notebook is approved and ready for use, the T&AA chairperson notifies SCOA, AASHTO staff, the project/product chairs, and the contractors that the new notebook is available and provides the URL.

## 4.9 Maintain the Standards and Guidelines

The purpose of this procedure is to ensure that each standard and guideline is correct, up-to-date, and relevant. The T&AA Task Force will perform annual reviews of all standards and guidelines in the current Standards and Guidelines Notebook. These reviews should include, but not be limited, to the following analysis:

- Determine if any hyperlinks embedded in the standards and guidelines are invalid.
- Determine if each standard and guideline is still relevant and up to date with industry directions.
- Determine if the standard is still needed.
- Determine if there are issues in consistency, readability, and/or format with specific standards or guidelines when compared to the majority of the existing standards and guideline.
- Determine if there are any issues like the above with informational or reference documentation included in the Standards and Guidelines Notebook.

T&AA may request assistance from the project/project task forces, contractors, and/or AASHTO staff in reviewing the standards and procedures or in validating issues. In addition, the users of the standards and guidelines should report any issues found while applying the standards and guidelines to ongoing development and maintenance efforts.

If any issues are found during the reviews or reported by stakeholders, T&AA will review these and determine which issues warrant corrective actions. For those issues requiring corrective actions, T&AA will create tasks for the current fiscal year or future work plans. These tasks will include one or more of the following:

- Revise an existing standard or guideline
- Delete or replace an existing standard or guideline
- Create a new standard or guideline
- Create, revise, eliminate or replace an informational or reference documentation that is included in the Standards and Guidelines Notebook.

Tasks to correct hyperlinks, spelling, format, and other minor issues that do not change the meaning or impact of a standard or guideline are normally performed in the current fiscal year.

All tasks to address issues found with the existing standards, guidelines, and other S&G Notebook documentation shall follow the procedures and activities included in this standard for revising, developing, reviewing, deleting, approving, storing and publishing standards and guidelines. Refer to the *Develop/Revise Standards and Guidelines* section.

## 4.10 Standard or Guideline

### 4.10.1 Description

This deliverable/artifact definition is used to define the required content, optional content, format, presentation, and structure for new standards and guidelines. The definition is also used when revising an existing standard or guideline and to bring an existing standard/guideline into compliance with the AASHTOWare Standards and Guidelines Definition Standard.

### 4.10.2 Content

The following describes the content required for each standard and recommended content for guidelines. Each standard and guideline is created and maintained in Microsoft Word. A PDF file for each standard and guideline is created from the source Word file. The file name for each Word and PDF files uses the "*C.NNN.VV.VS file name.ext*" naming convention. This naming convention begins with a *S&G Number*, which is described below, followed by a descriptive file name and file extension, such as "2.020.01.5S Security Standard.docx" or "1.005.02.4S Software Development and Maintenance Process Standard.pdf".

While in development/draft status, the revision date (mmddyyyy) should be appended to the file name as in ""2.050.01.3S Spatial Standard 02012018.docx". The revision date is removed when the standard or guideline is complete.

A Word template, *AASHTOWare Standard Template.docx,* is used to document the content in a consistent format, font, style, and structure. The template is stored in the *S&G Development* library in the *AASHTO Standard Template* folder.

The template defines a document with following sections. Each section is briefly described below. Also, the template includes instructions.

- **Cover Sheet** – The cover sheet includes the following content.
  - AASHTOWare Logo
  - Standard or Guideline Name
  - S&G Number – Each standard and guideline is assigned a S&G Number, which is the "C.NNN.VVS" format; where:
    - C is the number 1-5 which represents the number of the category for the standard or guidelines. The current categories are defined below with the *Standards and Guidelines (S&G) Notebook* definition.

- ◊ NNN is the number 001-099 which represents the standard or guideline number within the category. These are currently numbered in increments of 5 and 10.
- ◊ VV.V is a number 01.0-99.9 which represents the version number of a standard or guideline.
  - For new standards and guidelines, the initial version number is “01.0”.
  - If making a significant change to an existing standard or guideline, increment the version number by 1.0.
  - If making a minor change to correct spelling, grammar, or format, increment the version number by a tenth (0.1), as in “02.1”. This type of change does not change the meaning or impact of a standard or guideline.
- ◊ S is a suffix that indicates the document type, where:
  - S for Standard,
  - G for Guideline, and
  - R for *S&G Notebook Reference Document*

  Examples of S&G numbers are: 1.005.02.4S for the Software Development and Maintenance Process Standard, 2.090.01.2G for the Web & Mobile Data Exchange Guideline, and 0.010.01.0R for the Summary of Changes (reference document).

- □ Effective Date of the Standard or Revision Date for the Guideline or Reference Document
- □ Document History – includes entries for each new version
  - ◊ Version, Date, Revision Description, and Approval Date/Approval Party

- ○ **Table of Contents**
- ○ **Purpose**
  - □ Describe the purpose of the standard or guideline.
  - □ Lifecycle Phases - If applicable, define the phases in the lifecycle model where the major deliverables and artifacts are produced and when the procedures in the standard are performed. This may include a diagram of the applicable lifecycle model(s) with review gates where deliverables and artifacts are approved.
  - □ Applicability and Requirements – Describe the type of projects and MSE efforts that the standard or guideline applies to and note the required components for standard. *Red italicized text should be used denote all requirements in a standard.* ***Bold red italicized text is used to denote new requirements that were not in the previous version.***
- ○ **Task Force/Contractor Responsibilities** – Summarize the task force and contractor responsibilities regarding the standard. If needed, include the responsibilities of T&AA, SCOA, and AASHTO staff.
- ○ **Required/Recommended Deliverables and Artifacts** – Summarize the required deliverables and artifacts that shall be prepared and saved in order to comply with a standard, as well as any optional ones. In the case of a guideline, the artifacts and artifacts should all be designated as recommended.
- ○ **Procedures** – Define the procedures that shall be carried out to comply with a standard or to follow the intent of a guideline.

- **Technical Requirements/Technical Recommendations** - Define the technical requirements that shall be met to comply with a standard, or technical recommendations for a guideline.
- **Deliverable and Artifact Definitions**
  - Description – Provide a brief description of each deliverable and artifact that is a result of the procedures in the standard or guideline.
  - Content - Define the required or recommended content for each deliverable and artifact.
- **Appendices** – This is an optional section that should be used when one or more appendices are needed to document any information that is supplementary to the primary content of the standard or guideline.
- **Document Header and Footers** – The header of each standard and guideline normally includes the standard/guideline name and version. Where the footer contains the page number and revision date.

The author of each new standard and guideline uses the "*AASHTOWare Standard Template.docx"* as the starting point to ensure that the format, fonts, styles, and basic structure (sections and subsections) is similar to other notebook documents and add the applicable content to each section. Except for the Appendices, if one of the standard document sections is not applicable, it is recommended to retain the section heading and note "Note Applicable" or provide an explanation why the section is not used. The Appendices section is normally removed when not used. Other sections may be removed in those cases where the content of the new standard or guideline is unique and doesn't align with the standard content definition. Also, additional sections may be added, when needed.

The source Word (.docx) document for each new or revised standard and guideline is stored in the *Changed Documents* folder of the Next Notebook shared folder and the *S&G Development* library during the update cycle for the next version of the notebook. A PDF file is created for each primary Word document are stored in the same folder. Any secondary files used to create the Word documents, such as inserted images, charts or drawings, are stored normally the same folder as the Word primary file or in the *Reference* folder of the *Next Notebook* shared folder and the *S&G Development* library.

After a standard or guideline is approved and included in the current version of the notebook, the Word, PDF, and secondary files are stored in the *Current Notebook* Library.

# 5. Technical Requirements

The technologies associated with this standard are listed below.

- Microsoft Word is used as described below:
  - Prepare and update the source standard, guideline, and reference/informational documents (*C.NNN.VVS filename.docx*) that compose the content of the S&G Notebook.
  - Convert each notebook content source file (*C.NNN.VVS filename.docx*) to a content PDF file (*C.NNN.VVS filename.pdf*).
  - Prepare and update the *AASHTOWare Standard Template.docx.*
- Nitro Pro (Nitro Software, Inc.) is as described below:
  - Combine all notebook content PDF files (*C.NNN.VVS filename.pdf*) into a single notebook PDF file (*SG_Notebook_mmddyyyy.pdf*).
  - Bookmark the first page of each notebook content file in the combined notebook PDF file.

- Create and update links in the notebook PDF file.
- Establish document properties for the notebook PDF file.
- Minor edits to the notebook PDF file.

- Adobe Acrobat Pro may also be used in lieu of Nitro Pro to perform the tasks listed above.

# 6. Deliverable and Artifact Definitions

This section describes the deliverables and artifacts that are prepared, reviewed, approved, and saved during the review, creation, and update of the AASHTOWare Standards and Guidelines.

## 6.1 S&G Notebook Reference Document

### 6.1.1 Description

In addition to standards and guidelines, the notebook includes content from other documents, which are referred to as S&G Notebook reference documents or more commonly as reference documents. These documents include the Notebook Cover Page, Cover Letter, Summary of Changes, Table of Contents, Notebook Overview, and Standards and Guidelines Glossary.

### 6.1.2 Content

Each reference document includes content which, other than the section separator pages, is unique to that document. As with a *Standard or Guideline*, each of reference document is created and maintained with Microsoft Word and, where applicable, the same fonts, styles, and document format/structure as the standards and guidelines are used. A PDF file for each reference document is created from the source Word file.

Each reference document has an *S&G Number* and both the Word and PDF files use the "*C.NNN.VV.VS file name.ext*" naming convention, as in "0.005.01.0R Cover Letter.docx" and 0.010.01.0R Summary of Changes.pdf". As shown in these examples, the S&G Number for reference documents ends with a "R" suffix and the version number part of the S&G Number does not normally increment for most reference documents.

The exception for incrementing the version number is for reference documents with cover pages. These reference documents increment the version number the same as standards and guidelines and include a cover page with a document history table.

Reference documents that change will normally include the notebook effective date in the cover page, document footers, and other applicable locations.

As with standards and guidelines, during development and revision, the source Word document and PDF file for each new/revised reference document is stored in the *Next Notebook* shared folder and the *S&G Development* library along with any secondary content files. All reference document files used to create the current notebook are stored in the *Current Notebook* library.

## 6.2 Standards and Guidelines (S&G) Notebook

### 6.2.1 Description

The Standards and Guidelines (S&G) Notebook is the official published document that contains all AASHTOWare standards and guidelines. The current version of the notebook is published on the *AASHTOWare* web site at *https://www.aashtoware.org/about/standards-and-guidelines/* and is also available on the *AASHTOWare* SharePoint workspace.

Each version of the notebook is created as a single PDF format document. The standard naming convention for the notebook PDF file is "*SG_Notebook_mmddyyyy.pdf",* where "*mmddyyyy"* is the effective date of the notebook.

### 6.2.2 Content

The notebook PDF document is created by combining the PDF files for all current and approved standards, guidelines, and notebook reference. The PDF documents are combined in sequential order of their file name, as shown below for the September 1, 2017 version of the notebook (*SG_Notebook_09012017.pdf*).

Files used to Create the September 1, 2017 S&G Notebook

- *0.000.01.0R Notebook Cover Page.pdf*
- *0.005.01.0R Cover Letter.pdf*
- *0.010.01.0R Summary of Changes.pdf*
- *0.015.01.0R S&G Table of Contents.pdf*
- *0.020.01.0R S&G Notebook Overview.pdf*
- *1.001R Project Management and Software Engineering.pdf (Section 1 cover page)*
- *1.005.02.4S Software Development and Maintenance Process Standard.pdf*
- *1.010.03.4S Quality Assurance Standard.pdf*
- *2.001.01.0R Technical Standards and Guidelines.pdf (Section 2 cover page)*
- *2.020.01.5S Security Standard.pdf*
- *2.030.03.1S Critical Application Infrastructure Currency Standard.pdf*
- *2.040.04.0S Database Selection and Use Standard.pdf*
- *2.050.01.3S Spatial Standard.pdf*
- *2.060.05.3S Product Naming Conventions Standard.pdf*
- *2.070.04.1S Backup and Disaster Recovery Standard.pdf*
- *2.080.01.4G Mobile Application Development Guideline.pdf*
- *2.085.01.4G Web Application Development Guideline & Architectural Goals.pdf*
- *2.090.01.2G Web & Mobile Data Exchange Guideline.pdf*
- *3.001.01.0R - Appendices.pdf (Section 3 cover page)*
- *3.010.02.3R AASHTOWare Life Cycle Framework.pdf*
- *3.015.02.4S AASHTOWare Standards and Guidelines Definition Standard.pdf*
- *3.020.01.0R Standards and Guidelines Glossary.pdf*

When combining the PDF documents, the first page of each document is bookmarked with the above names and can be referenced by hyperlink. The software tools used for preparing the notebook PDF document, including combining PDF documents, bookmarks, and creating and updating hyperlinks, are listed in the *Technical Requirements* section.

As shown in the content files listed above, after the S&G Notebook Overview, the notebook is divided into three sections based on the category, which is the first number of each document's *S&G Number* and file name. Each section begins with a cover page. The sections/categories are briefly described below.

Notebook Sections/Categories

- 1 - Project Management and Software Engineering

  This section includes category 1 standards. These are process oriented standards that address project management, software development, and quality

assurance practices. There are no current guidelines in this section; however, a guideline may be included in this section with a recommended set of practices.

- 2 – Technical Standards and Guidelines

  This section includes category 2 standards and guidelines. The standards in this section are less process-oriented and include required methods, technologies, deliverables, and artifacts for specific technical topics. The guidelines in this section are similar; however, the methods, technologies, and artifacts are recommended in lieu of being required.

- 3 – Appendices

  This section includes category 3 documents that support the standards and guidelines included the sections 2 and 3. These include the Standards and Guidelines Glossary document and this standard (AASHTOWare Standards and Guidelines Definition Standard). This standard is included in this section to separate it from the standards and guidelines that impact the project/product task force tasks forces and their contractors.

Note: The notebook structure has changed several times with the last major change occurring with the April 2016 version of the notebook. Therefore, most of the prior notebook PDF files in the *S&G History* library will be different from that described above.

## 6.3 AASHTOWare Standard Template

### 6.3.1 Description

The "AASHTOWare Standard Template.docx" is a Microsoft Word document that is used to create the content for new standards and guidelines in a consistent format, font, style, and structure (sections and subsections). The template may also be used to update an existing standard or guideline that did not initially use the template. The template is stored in the *S&G Development* library.

### 6.3.2 Content

Refer to the *Standard or Guideline* artifact definition, for descriptions of the standard sections and content and for how the template is used. The template also includes embedded instructions for most sections.

## 6.4 Checklists, Forms, and Templates

### 6.4.1 Description

Checklists, forms, and templates are documents and files that are used to assist with the preparation of deliverables and artifacts described in the notebook.

Examples of these documents are listed below:

- AASHTOWare Compliancy Backup Checklist
- AASHTOWare Review Gate Approval Request Form
- AASHTOWare Project Work Plan Template

Each checklist, form, and template used with the current notebook is stored in *Templates and Other Materials* folder in the *Current Notebook* library. They are also stored on the *AASHTOWare* web site, *www.aashtoware.org/*, and on the *AASHTOWare* workspace. The current notebook includes a hyperlink that points to the web site location of each checklist, form, and template in the current notebook.

New and revised checklists, forms, and templates are stored in the *S&G Development* library during the notebook update cycle.

### 6.4.2 Content

The content for each checklist, form, and template is unique and is based on the content requirements of a specific deliverable or artifact. Currently, these are all Word (.docx) documents; however, another type of file could be used when applicable for creating a specific deliverable or artifact.

Most checklist, form, and template Word documents use similar fonts and styles as those used for standards and guidelines.

The naming convention for these documents is "*Filename_mmmddyyyy.ext*", where "*mmddyyyy"* is the revision date of the document. The filename should be descriptive with no spaces. The revision date should also be included on page footers and/or other applicable locations.

## 6.5 S&G Training and Support Aids

### 6.5.1 Description

Training and support Aids are documents and files are used in conjunction with the current version of the notebook to assist with the use and understanding of the notebook and/or specific standards or guidelines. Currently, these documents are prepared in Word and are distributed to stakeholders in PDF format. Other types of documents such as presentations and spreadsheets could also be created.

An example of a training/support aid is the "Summary of S&G Notebook Requirements" document.

Each training and support aid is stored in *Templates and Other Materials* folder in the *Current Notebook* library. They may also be stored on the *AASHTOWare* web site, *www.aashtoware.org/*, and on the *AASHTOWare* workspace.

New and revised training and support aids are stored in the *S&G Development* library during the notebook update cycle.

### 6.5.2 Content

There is no specific content for training and support aids; however, the content of each aid should summarize and explain aspects of the current notebook or a specific standard or guideline. Most training and support aid Word documents use similar fonts and styles as those used for standards and guidelines.

The naming convention for these documents is "*Filename_mmmddyyyy.ext*", where "*mmddyyyy"* is the revision date of the document. The filename should be descriptive and, if posted to the web site, should include no spaces. The revision date should also be included on page footers and/or other applicable locations.

## 6.6 T&AA - S&G Notebook Workspace

### 6.6.1 Description

T&AA's AASHTOWare SharePoint site is the repository used to store and access the current version of the S&G Notebook; previous versions of the notebook; all files and documents needed to create the current and previous versions of the notebook; the next version of the notebook; and all new, revised, and existing documents to be included in the next version of the notebook. Checklists, forms, templates, training and support aids, and other reference materials used in conjunction with the current, prior, and next versions of S&G Notebook are also stored in this workspace.

This workspace and its libraries are created by the workspace administrators, which are the T&AA AASHTO Project Manager (PM) or T&AA Contractor. Workspace and library creation is normally performed by the T&AA AASHTO PM, where creating, updating, and maintaining folders and content in the libraries are normally performed by the T&AA

Contractor. With these duties, the T&AA AASHTO PM and T&AA Contractor both require read/write access to the workspace. The T&AA Chair should also have read/write access.

All other T&AA members, AASHTO Staff members, and the SCOA T&AA liaison should be provided with read access to this workspace. Additional access rights and access to others should be granted as needed.

### 6.6.2 Content

The S&G Notebook repository is included the three libraries listed below, each of which is described in the following sub-sections. When displayed in a file explorer view, the libraries are the top-level folders of the workspace.

T&AA - S&G Notebook Workspace Root/Top Level Libraries

- Current Notebook
- S&G History
- S&G Development

#### 6.6.2.1 Current Notebook

The *Current Notebook* library is used to store the complete S&G Notebook document for the current version of the notebook, all source documents and files used to create the current notebook, and checklists, forms, templates, training and support aids, and other reference materials used in conjunction documents with the notebook.

The root/top level of the library contains the complete notebook PDF file and the following folders. Additional folders and subfolders may be created, if needed.

Current Notebook Library Root/Top Level Folders and Files

- Notebook
- Templates and Other Materials
- Reference
- *SG_Notebook_mmddyyyy.pdf* (current version of S&G Notebook)

Each folder and the content within each folder are described below. The content of the current notebook (*SG_Notebook_mmddyyyy.pdf*) is described in the *Standards and Guidelines (S&G) Notebook* definition section.

##### 6.6.2.1.1 Notebook

This folder contains the source Word (.docx) and PDF files for all standards, guidelines, and reference documents included in the current version of the S&G Notebook PDF file. The PDF files are combined for to create the notebook, as shown above in the *Standards and Guidelines (S&G) Notebook* artifact definition. Refer to the *Standard or Guideline* and *S&G Notebook Reference Document* artifact definitions for details on standards, guideline and reference document files.

The Word files for some standards and guidelines include inserted copies of checklists, forms, and/or templates that are stored in the *Templates and Other Materials* folder. Other Word files may include insert files, such as logos, images, charts, or drawings, that are stored in the *Reference* folder.

##### 6.6.2.1.2 Templates and Other Materials

This folder contains document and files used in conjunction with the current version of the S&G Notebook. These are described below:

- ◊ Checklists, forms, and templates - These are documents and files that are used to assist with the preparation artifacts and deliverables described in the notebook. Refer to the *Checklists, Forms, and Templates* artifact definition for examples and additional details.
- ◊ Training and Support Aids – These are documents and files used in conjunction with current version of the notebook to assist with the use and understanding on the notebook and/or specific standards. Refer to the *S&G Training and Support Aids* artifact definition for examples and additional details.

#### 6.6.2.1.3 Reference

This folder contains any other files, that don't fit in the above folders, that needs to be saved with the current version of the notebook. These could include general files such as the AASHTOWare and product logos or presentations and research pertaining to the current notebook. There are no format, content, or file naming specifications for the files in this folder.

### 6.6.2.2 S&G History

The *S&G History* library is used to store previous versions of the notebook PDF file and all source files used to create the notebook. The files for each version of the notebook are stored under a root level folder using the "SG YYYY MMM" naming convention, where "YYYY" is the year of the notebook's effective date and "MMM" is a three-character abbreviation of the month. For example, the April 15, 2016 version of the notebook is stored in folder named "SG 2016 Apr" and the July 1, 2012 version of the notebook is stored in a folder named "SG 2012 Jul".

The root level folders in the library, as of September 1, 2017, are shown below. Although, there are older versions of the notebook, the oldest notebook folder stored in the *S&G History* library is for the April 2004 version.

S&G History Library Root Level Folders (as of 09/01/2017)

- □ *SG 2004 Apr*
- □ *SG 2006 Feb*
- □ *SG 2006 Oct*
- □ *SG 2009 Jul*
- □ *SG 2010 Jul*
- □ *SG 2011 Jul*
- □ *SG 2012Jul*
- □ *SG 2013 Jul*
- □ *SG 2014 Jul*
- □ *SG 2015 Feb*
- □ *SG 2016 Apr*

The notebook folder structure has changed several times since 2004 with the last major change occurring with the April 2016 version of the notebook. With the changes in the notebook structure, there have also been changes in the structure of the *Current Notebook* library. Therefore, the notebook PDF file and most of the history folders will be different from that used for the current version of the S&G Notebook.

The *S&G History* library is updated each time a new S&G Notebook is published. Prior to updating the *Current Notebook* library with the folders and files for the new notebook:

- A new root level folder, such as "*SG 2017 Sep,*" is created in the *S&G History* library for the existing current notebook's effective date.
- The complete file structure of the *Current Notebook* library is copied to the new history folder.

#### 6.6.2.3 S&G Development

The primary purpose of the *S&G Development* library is to store standards, guidelines, reference documents that are being created or revised for inclusion the next version of the S&G Notebook. The library contains also contains the existing (unchanged) standards, guidelines, and reference documents from the current S&G Notebook. The next version of the S&G Notebook is created by combining the PDF files for all new, revised and existing (unchanged) standards, guidelines, and reference documents. Also, some existing files may be targeted for deletion or replacement and will not be included in the next notebook.

The root/top level of the library contains folders shown below plus the next version of the notebook. The first three folders (*Notebook, Templates and Other Materials*, and *Reference*) are equivalent to those in the *Current Notebook* library and are used to replace content in the *Current Notebook* library when a new version of the notebook is created. The other root/top level folders are specific to the *S&G Development* library. Additional folders and subfolders may be created, if needed.

S&G Development Root/Top Level Library Folders and Files

- *Notebook*
- *Templates and Other Materials*
- *Reference*
- *Unchanged Documents*
- *Changed Documents*
- *Pending*
- *AASHTO Standard Template*
- *SG_Notebook_mmddyyyy.pdf* (next version of S&G Notebook)

Each of the above folders and their content are described below.

The content of the next version of the S&G Notebook (SG_Notebook_mmddyyyy.pdf) is the same as the current version, which is described in the *Standards and Guidelines (S&G) Notebook* definition section.

##### 6.6.2.3.1 Notebook

This folder contains the Word and PDF files for all standards, guidelines, and reference documents that are used to create the next version of the S&G Notebook. At the beginning of the notebook development cycle, this folder should be empty.

The folder is populated with the most recent files from the *Unchanged Documents* and *Changed Documents* folder when the next version of the notebook is created. The PDF files for all standards, guidelines, and reference documents in the *Notebook* folder are combined to create the next notebook PDF file.

At the end of the notebook development cycle, when the next version is published, the content of this folder is used to replace the content in the *Notebook* folder in the *Current Notebook* library.

#### 6.6.2.3.2 Templates and Other Materials

This folder contains all files (Word, PDF, and other) for the templates, checklists, forms, and training/support aids that are used in conjunction with the next version of the S&G Notebook. At the beginning of the notebook development cycle, this folder is populated with the existing files in the *Templates and Other Materials* folder in the *Current Notebook* library.

New, revised, and existing files are stored in this folder during the notebook development cycle. In order to track new and revised files, all files names shall include the revision date as described in the *Checklists, Forms, and Templates* and *S&G Training and Support Aids* artifact definitions. Files that will be not be used with the next notebook should be deleted.

At the end of the notebook development cycle, only one copy of each file should be maintained in this folder. When the next version of the notebook is published, the content of this folder is used to replace the content in the *Templates and Other Materials* folder in the *Current Notebook* library.

#### 6.6.2.3.3 Reference

This folder contains any other files, that don't fit in the above folders, that need to be saved with the next version of the notebook. These could include general files such as the AASHTOWare and product logos or presentations and research pertaining to the next notebook.

At the beginning of the notebook development cycle this folder is populated with the existing files in the *Reference* folder in the *Current Notebook* library. These New, revised, and existing files should be maintained in this folder, as needed for the next notebook. Existing files that will not be relevant to the next notebook should be deleted. There are no format, content, or file naming specifications for the files in this folder.

When the next version of the notebook is published, the content of this folder is used to replace the content in the *Reference* folder in the *Current Notebook* library

#### 6.6.2.3.4 Unchanged Documents

This folder contains the Word and PDF files for the standards, guidelines, notebook reference that have not changed since this last published version of the S&G Notebook. At the beginning of the notebook development cycle, this folder is populated with all files from the *Notebook* folder of the *Current Notebook* library.

When an existing file is revised for the next version of the notebook, it should be copied to the *Changed Documents* folder and deleted from this folder. Existing files that will be deleted or replaced in the next notebook are also deleted from this folder.

At the end of the notebook development cycle, this folder should only contain the files from the current notebook that have not changed and have not been deleted or replaced. The content of this folder is copied to the *Notebook* folder when the next version of the notebook is created.

#### 6.6.2.3.5 Changed Documents

This folder contains the Word and PDF files for the standards, guidelines and notebook reference documents that have been revised and created (added) since this last published version of the S&G Notebook. At the beginning of the notebook development cycle, this folder should be empty.

The content of this folder is updated as new standards, guidelines, and reference document are created and existing ones are revised. Revised files originate from the Unchanged Documents folder, as noted above. Word and PDF file are stored in this folder for each revised and new standard, guideline, and reference document. Change summary files are created and stored in the folder that document the changes and additions made since the last published version of the notebook.

Refer to the *Standard or Guideline* and *S&G Notebook Reference Document* artifact definitions for content, format and naming convention details for new and revised standards, guideline and reference document files. As discussed here, standards and guideline in development/draft status, should have the revision date appended to the file name. Draft reference files that undergo T&AA review should also append the revision date to the file name.

At the end of the notebook development cycle, the revision date is removed from the file name of all completed and approved standard, guidelines, and reference documents (Word and PDF). At this point only one version should exist for each standard, guideline, and reference document, and all files are copied to the *Notebook* folder for inclusion in the next version of the notebook.

#### 6.6.2.3.6 Pending

This folder is used to store standards, guidelines, reference documents and other types of document and files that are still undergoing development/update, and will be not be included in the next version of the notebook.

This folder is populated for documents and files in other folders and is not emptied at the end of notebook update cycle. At the beginning of the next update cycle the files are moved back to their orginal location.

#### 6.6.2.3.7 AASHTO Standard Template

This folder contains the AASHTOWare Standard Template which is used for creating new standards and guidelines. The template is a Word template that is used to document the content in a consistent format, font, style, and structure. This folder and file are not moved to the Current Notebook library with a new version of the notebook.

## 6.7 Next Notebook Shared Folder

### 6.7.1 Description

The *Next Notebook* shared folder is a shared folder created, updated, and maintained by the T&AA Contractor on a workstation that is maintained outside of SharePoint and is the primary where location all S&G development, update, and delete activities are performed.

This folder is named "*Notebook FY YYYY*" and is shared with the T&AA AASHTO PM. A new shared folder is created at the beginning of each notebook development cycle. The new shared folder created for the next version of the notebook is is referred to in this standard as the *Next Notebook* shared folder. Prior versions of the shared folder are not addressed by this standard.

### 6.7.2 Content

After a new Next Notebook share folder is created it is initially populated with content from the *Current Notebook* library as defined in the *Begin Update Cycle for Next Version of S&G Notebook* procedure. During the notebook development cycle all documents and files used to create the next version of the notebook are maintained in this folder. New and revised documents and files are uploaded to *S&G Development* library as changes are made and as documents/files are finalized. Also, the notebook PDF file is compiled and edited on the shared folder and uploaded to *S&G Development* library. In addition, all files needed to update *Current Notebook* library are maintained on the shared folder. Additional folders, subfolder and files may also be maintained on the shared folder, as needed.

# Standards and Guidelines Glossary

This document is a glossary of all key terms used throughout the Standards and Guidelines Notebook. Some sections contain definitions that are general and apply to many standards and guidelines, and other sections contain definitions that are primarily used in a single standard.

| Term | Definition |
|---|---|
| AASHTO Staff | AASHTO staff members are responsible for the day-to-day business operations of the AASHTOWare joint development program. An AASHTO staff member serves as a liaison to each project and product task force as well as to SCOA and the T&AA task force. In regards to the Standards and Guidelines, the AASHTO staff liaison to each project and product task force serves as a project manager assisting the task forces in the application of the standards and guidelines, and in the review of new and revised standards and guidelines. The liaisons to SCOA and T&AA also assist with the development, revision, review, and approval of new and revised standards and guidelines. |
| Alpha Test Plan | The Alpha Test Plan includes all of the documentation needed to plan and perform alpha testing and to document the results of alpha testing. The plan identifies the system and system components that will be tested; include the requirements that will be tested; and the test procedures and expected results used to perform and measure the test. |
| Alpha Test Results Report | The Alpha Test Results Report is a required major deliverable that documents the results from Alpha Testing (what was tested, results, problems found, corrections made, outstanding issues, etc.). |
| Alpha Testing | The purpose of alpha testing is to the test whole system with an emphasis on breaking the system, checking the user requirements, and reviewing all documentation for completeness by using the application as if it were in production. Alpha testing is performed after the contractor completes all development testing (unit, build, and system testing). |
| Artifact or Work Product | An artifact is defined as a tangible result (by-product or work product) of a software development, maintenance, or project management activity. Artifacts may be the form of a document, spreadsheet, presentation, data model, installation package, and/or various other types of documentation and software development work products. |
| Beta Test Materials (Beta Test Plan and Beta Test Installation Package) | The Beta Test Materials contains all of the materials needed to release a product for beta testing, including the product, installation procedures, user and system documentation, test instructions, test procedures, and methods to record testing results and report problems. The two primary components of the Beta Test Materials are referred to as the Beta Test Plan and Beta Test Installation Package. |

| Term | Definition |
|---|---|
| Beta Test Results Report | The Beta Test Results Report is a required major deliverable that documents the combined beta testing results of all eta testing agencies, exceptions discovered, and resolutions to the exceptions. The report is submitted as part of the final acceptance of the new or revised product. |
| Beta Testing | The purpose of beta testing is to confirm to the user/tester that all functionality and operability requirements are satisfied, the system will operate correctly in the user's environment, and the system is ready for delivery and implementation. Beta testing also includes the review and validation of all documentation and procedures. Beta testing is performed after alpha testing has been accepted and prior to implementation. |
| Contents List | A contents list must be included with the installation package showing what content is being shipped. The content list should clearly state what platform (computing environment) the installation package was prepared for. |
| Cover letter | The cover letter is sent with the Product Installation Package and includes information like: whom the installation package is being sent to, who is sending the package, what is included in the package, and for what reason. |
| Deliverable | A deliverable is an artifact produced during a project or MSE effort that is required by a AASHTOWare standard (in the S&G Notebook); and shall be planned, reviewed by stakeholders, submitted to the task force, and approved by the task force. A deliverable may be a single document/artifact or may be comprised other multiple documents/artifacts. |
| Deliverable Approval Procedure | This procedure defines the process used by the contractor and task force during a project or product MSE work to submit, approve, and reject deliverables prior to their designated review gates; and to document the approval decision. This is also referred as the procedure for approving deliverables independent of the review gate |
| Deliverable or Artifact Definition | A deliverable or artifact definition is used to define the purpose, format, content, usage, and responsibilities of a deliverable or artifact. |
| Development and Maintenance Document | The Development and Maintenance Documentation is a required artifact for development projects. This documentation, supplemented by the Technical Design Specification, represents the internal documentation for the product, and should describe the logic used in developing the product and the system flow to help the development and maintenance staffs understand how the programs fit together. The documentation should provide instructions for establishing the development environment, and should enable a developer to determine which programs or data may need to be modified to change a system function or to fix an error. The Development and Maintenance Documentation is not required for product MSE work efforts unless an existing version of the document exists. In this case, the documentation must be updated to stay current with the product. |

| Term | Definition |
|---|---|
| Document Template | A document template is Microsoft Word template used to ensure consistency in certain deliverables and artifacts such as work plans, checklists, and forms. |
| Enhancement | An enhancement is a development effort to add new features to an existing AASHTOWare product that is licensed annually; or an effort to modify an existing product. Enhancements are classified as small, medium and larger. |
| Enhancement FDS or SRDS | This deliverable includes the functional design specifications for one or more enhancements. In most cases, the enhancement system requirements are included in the enhancement FDS, and in this case, is frequently referred to as a System Requirements and Design Specification (SRDS). |
| Enhancement System Requirements | Enhancement system requirements are created for each medium and large enhancement and are documented in one or more SRS deliverables. Each enhancement SRS may also be combined with the appropriate functional design specifications to create an enhancement Functional Design Specification (FDS) or System Requirements and Design Specification (SRDS) deliverable. A SRS,FDS or SRDS deliverable is not required for small enhancements and maintenance activities. |
| Exception to Standards | A product or project task force chair may request an exception from following a standard by including the request in the appropriate work plan or by sending a letter to the SCOA chairperson. The request should include all proposed changes and/or exclusions to one or more standards along with the documentation that describes or justifies the reasons for the reason exception(s) and any additional documentation for SCOA consideration. If submitted by letter, the exception request must be submitted to SCOA prior to beginning the phase of the project where the applicable standards are to be used. SCOA will make an approval decision on the exception request and notify the task force chair of the decision. |
| Functional Design Specification (FDS) | The Functional Design Specification (FDS) is a required major deliverable that documents the design of the proposed product using terminology that can be readily reviewed and understood by the task force, technical review teams (TRTs), technical advisory groups (TAGs), and other stakeholders involved in the development process. The FDS translates the requirements in the URS and SRS into design specifications that define how the system requirements will be implemented from a user or business perspective. For developments projects using a waterfall development methodology, the FDS is created as a detailed specification addressing the full scope of the project and all user and system requirements. |

| Term | Definition |
|---|---|
| Guideline | A guideline describes procedures, results, technical specifications and/or technologies that are considered good practices to follow, produce, or use; however, these are not required. A proposed standard or standard process may be initially implemented as a guideline with future plans to implement it as a requirement. |
| Installation Package Checklist | The Installation Package Checklist is used to assist in preparing the Product Installation Package and the completed checklist must be included with the package. |
| Iteration FDS i | The iteration FDS is created for each iteration in an iterative development project. Each iteration FDS includes the system requirements and the functional design specifications for the iteration. As with the iteration system requirements, the iteration functional design focuses the specific scope, objectives, and user requirements of the proposed iteration. The level of detail must be appropriate to proceed with the technical design and construction of the iteration. |
| Iteration System Requirements | Iteration system requirements are created for each development iteration and are documented in the Iteration Functional Design Specification (FDS). Each set of system requirements focuses on the specific scope, objectives, and user requirements of the proposed iteration. The iteration system requirements are developed with the same level of detail as the full scope SRS. Refer to the next section for more information on the Iteration FDS. |
| Iterative Development Methodology | An iterative development methodology is a software development process where the overall functionality to be delivered by a development project is sliced into segments or iterations. A typical iterative process would include a planning phase and high level requirements and design phase, prior to a detailed design, construction, and testing phase for each iteration. After all iterations are completed the process would then conclude with acceptance testing and implementation phases for the composite of all iterations. |
| Large Enhancement | A large enhancement is complex; requires significant funding, effort and/or resources to implement; and requires significant planning, analysis, and design. |
| Maintenance | Maintenance is the technical activity to correct errors and other problems that cause an existing product to operate incorrectly. Maintenance is performed under a product work plan. |
| Maintenance, Support, and Enhancement (MSE) Work Effort | A Maintenance, Support, and Enhancement (MSE) Work Effort refers to the annual maintenance, support, and enhancement work performed for an existing AASHTOWare product. A MSE work effort is performed under a product work plan. As noted above, some large and complex enhancements are developed under a project plan. |

| Term | Definition |
|---|---|
| Management, Monitoring, and Control Procedures | • Management, Monitoring, and Control procedures include procedures for issue management, change management, status reporting, quality management, communication management, configuration management, risk management, and backup and disaster recovery. These procedures are defined in the work plan and are executed throughout the lifecycle of the wok plan. |
| Medium Enhancement | A medium enhancement is more complex than a small enhancement; requires a moderate amount of funding, effort and/or resources; and requires more planning, analysis and design than a small enhancement. |
| Microsoft SharePoint | Microsoft SharePoint is a web application used by AASHTOWare for document management and collaboration. SharePoint is the primary tool used by task forces and contractors to create and maintain product and project document repositories. Task forces, SCOA, T&AA, AASHTO Staff, and contractors also use SharePoint for balloting, meeting scheduling, discussions, and various other sharing and collaboration uses. All AASHTOWare content on SharePoint may be accessed by browser at the AASHTOWare Portal site. |
| Microsoft SharePoint Workspace | Microsoft SharePoint Workspace is desktop application used to access and manage a SharePoint workspace. The data is stored locally and synced with the web version of the workspace. |
| Minor Maintenance | An effort to provide a temporary fix or repair of an existing product module. The temporary fix or repair results must not add to, change nor delete from the functionality of a product module. |
| Preliminary FDS | The preliminary FDS is created for iterative development projects prior to the design and construction of the development iterations. The preliminary FDS is created in lieu of the detailed, full scope FDS listed above; and primarily focuses on design specifications that apply to the system requirements in the preliminary SRS. The preliminary FDS addresses the overall proposed product and only provides limited details on areas of the design that are specific to a single iteration. |
| Preliminary SRS | The preliminary SRS is created for iterative development projects prior to the design and construction of the development iterations. The preliminary SRS is created in lieu of the detailed, full scope SRS listed above; and primarily focuses on high level functionality and on requirements that apply to the overall proposed product, such as user interface, security, accessibility, technical architecture, and internal/external software interface. |

| Term | Definition |
|---|---|
| Product Installation Package | The Product Installation Package contains all procedures, executables, and documentation needed to implement and operate an AASHTOWare product at the customer agency sites. The installation package is distributed to all licensees. The product installation package may include components (if not all) that may be sent electronically. The fact that an item has been electronically sent should be noted on the checklist that is sent with the package. If the entire package is electronically sent, it must still include all items (electronic checklist, contents list ...). |
| Project | An AASHTOWare project refers to work that is performed outside of an annual MSE effect to develop a new product, redevelop/rearchitect an existing product, enhance an existing module, develop specifications, or perform research. Many projects are performed under the auspice of a solicitation with funds collected one-time up-front |
| Project and Product Contractors | A contract software development firm is hired to develop each AASHTOWare software product and is referred to as a project contractor. A product contractor is hired to maintain, support and enhance an existing AASHTOWare product. |
| Project and Product Task Forces | Project and product task forces are made up of business area representatives from the member departments. A project task force manages and oversees the development of an AASHTOWare product; and product task forces manage and oversee the maintenance, support, and enhancement of existing AASHTOWare products. In regards to the Standards and Guidelines, the each task force has the responsibility of:<br>Ensuring that the requirements defined in each standard are complied with.<br>Requesting exceptions to standards (see below).<br>Reviewing and providing comments and problems encountered regarding existing and new standards and guidelines. |
| Project Lifecycle<br>and<br>MSE Lifecycle | An AASHTOWare project refers to work that is performed outside of an annual MSE effect to develop a new product, redevelop/rearchitect an existing product, enhance an existing module, develop specifications, or perform research. Many projects are performed under the auspice of a solicitation with funds collected one-time up-front. Refer to the *Software Development and Maintenance Process Standard* for details on the project and MSE lifecycles. |

| Term | Definition |
|---|---|
| Project or Product Work Plan | A work plan is the formal document that describes the scope and objectives of the work to be performed by the contractor during a specific contract period, requirements or specifications to be met, tasks to be performed, deliverables to be produced, schedule to be met, cost of the effort, required staffing and resources, the technical approach for accomplishing the work, and the approach for managing, monitoring, and controlling the work.<br>A project work plan is created for each AASHTOWare project and a product work plan (or MSE work plan) is created for each annual MSE work effort.The term “work plan” may be used to mean either a project work plan or product work plan. The completed work plan is included in the contract agreement. |
| Project/MSE | Project/MSE refers to both a project and a MSE work effort. |
| Project/MSE Archive Package | The Project/MSE Archive Package is an archive of the final product, project materials, and development artifacts. This package includes the Product Installation Package, all approved deliverables and review gate approval requests, Technical Design Documentation (TDS), Development and Maintenance Document, other artifacts created, source code, build procedures, and any other information needed to setup, configure, change, and rebuild the final product. |
| Project/MSE Closeout | Project closeout is the formal completion of all activities associated with the project work plan. MSE closeout is the formal completion of all activities associated with the product work plan. In both cases, closeout requires approval by the task force. |
| Project/MSE Repository | The project/MSE repository is an online storage area where all project/MSE artifacts, major deliverables, review gate and deliverable approval documentation, and any project/MSE related documentation is stored. The repository must be accessible to the contractor, task force, AASHTO Project Manager, SCOA and T&AA liaisons, and other stakeholders designated by the task force. A SharePoint workspace or a folder in a SharePoint workspace is commonly used as the project repository; however, other technologies may also be used. |
| Project/Product | Project/product is used as a prefix with work plan, task force and contractor and applies to both projects and MSE work efforts. For example, “product/product task force” applies to both product task forces and project task forces. As noted above, without the prefix these also mean either project or product. |
| Reference Document | A reference document is an S&G document that is not a standard or guideline, but is included in the notebook for informational purposes, such the cover page, cover letter, summary of changes, table of contents, introduction, and glossary. Also reference documents such as forms, diagrams, and presentations may be stored in either of the above workspaces. |

| Term | Definition |
|---|---|
| Requirements Traceability Matrix (RTM)<br><br>and Preliminary RTM | The RTM is a required deliverable that describes the backward traceability and forward traceability of the requirements in the URS. The RTM documents that system requirements are be traced to source user requirements. The RTM also documents that each requirement is traced to a design object and a testing procedure. The RTM is referred to as a Preliminary RTM until it is finalized. A RTM is required for projects but is not required for MSE work efforts. |
| Review Gate | Review gates are specific milestones in the lifecycle of both projects and MSE work where the task force shall approve the current status and results of the project or MSE work effort and authorize the project to continue with the next phase or sub-phase of the project life cycle. A standard set of review gates are required for projects and MSE work. One or more deliverables and/or artifacts are submitted with each review gate.The review gates for each project and MSE work effort are defined in the work plan. Standard review gates are defined in the *Common Artifacts Standard*. |
| Review Gate Approval Request | A review gate approval request is a required artifact that the contractor project manager submits to the task force chair at the conclusion of each review gate period. Any unapproved major deliverables associated with the review gate must also be submitted with the request. The request form (see below) is also used to document and communicate the task force decision regarding the approval or denial of a review gate approval request. |
| Review Gate Approval Procedure | This procedure defines the process used by the contractor and task force during a project or product MSE work to submit , approve, and reject review gates and deliverables submitted with the review gates, and to document the approval decision. |
| Review Gate Approval Request Form | The review gate approval request form is a standard form used for submitting review gate approval requests and for documenting the task force approval decision. |
| SharePoint Workspace | A SharePoint workspace is a shared storage area which consists of a set of files to be shared plus other tools (such as calendars, meetings, pictures, forms, and discussions) used for group collaboration. A workspace may be accessed by browser or the SharePoint workspace client. The S&G Notebook workspace is used to store and share the current on the notebook, prior versions, and versions currently being revised. The complete notebooks, as well as the individual standards, guidelines, reference documents, and other files used to compose each version of the notebook are stored is the S&G Notebook workspace. Each task force also uses workspaces for project/MSE repositories and other uses. |
| Small Enhancement | A small enhancement is not complex; requires minimum funding, effort and/or resources to implement; and requires minimum planning, analysis and design. |

| Term | Definition |
|---|---|
| Software Development Methodology | A software development methodology or system development methodology is a framework that is used to structure, plan, and control the process of developing an application or other information system. Several different types of methodologies are used for development including waterfall, iterative or incremental, spiral and RAD (Rapid Application Development). Although any type of methodology may be used if planned and approved in the work plan, the S&G Notebook focuses on methodologies based on the waterfall and iterative methodologies. |
| Special Committee on AASHTOWare (SCOA) | SCOA is the management committee made up of member department representatives that oversees the AASHTOWare joint development program. In regards to the Standards and Guidelines, SCOA is responsible for:<br>• Defining the needs and setting the objectives for AASHTOWare process improvement, and for new or revised standards and guidelines.<br>• Approving all new and revised standards and the deletion of existing standards. |
| Spreadsheet Template | In addition to document templates, some deliverables and artifacts are created with standard Microsoft Excel spreadsheets. |
| Standard | A standard describes mandatory procedures that must be followed, results that must be produced, and technologies and technical specifications that must be used or adhered to during the development, enhancement, and maintenance of AASHTOWare products. AASHTOWare standards are created and implemented in order to ensure a consistent approach is used to develop, change, maintain and deliver software products. |
| Standard Template | This is a document template that is used to ensure consistency in creating standards and guidelines. The template is used to define a standard look and feel across all standards and guidelines including standard cover pages, sections and fonts. This template is stored in the S&G Development workspace. |
| Standard/Guideline Category | A category is assigned to each standard, guideline, and reference document in the S&G Notebook and S&G workspaces to group common documents. The current categories are listed below:<br>• 1 - Project Management and Software Engineering<br>• 2 – Technical Standards and Guidelines<br>• 3 - Appendices |

| Term | Definition |
|---|---|
| Standard/Guideline Number | A number is assigned to each standard, guideline, and reference document in the S&G Notebook and S&G workspaces with the “C.NNN.VV.VS format, where:<br>• C is number 1-5 which represents category of the document (see below)<br>• NNN is the number 001-099 which represents the standard, guideline, or reference document number within the category. These are currently numbered in increments of 5 and 10.<br>• VV.V is a number 01.0-99.9 which represents the version number of a standard, guideline, reference document. Minor changes increment the version number by a tenth, such as “02.1”.<br>• S is a suffix that indicates the document type, where<br>  ▪ S is for standard<br>  ▪ G is for guideline, and<br>  ▪ R is for reference document<br>  ▪ T is for Template |
| System Requirement | A system requirement describes what the proposed product must do in order to fulfill one or more user requirements (how the product will do it). These may describe functionality or impose constraints on the design or implementation (such as performance requirements, security, or reliability). System requirements are documented in the appropriate technical detail for a software developer or integrator to design the proposed product or enhancement. System requirements should also be written where they can be understood by the task force and other stakeholders involved in the development process. |
| System Requirements Specification (SRS) | The SRS is a required deliverable which contains all of the system requirements. The SRS should describe all functional, non-functional, technical, role, and data requirements of the proposed system in sufficient detail to support system design. For developments projects using a waterfall development methodology, the SRS is created as a detailed specification addressing the full scope of the project and all user requirements. |
| Technical and Application Architecture (T&AA) Task Force | The T&AA Task Force is a technical group made up of member department representatives that reports to SCOA. The T&AA task force develops the Standards and Guidelines for AASHTOWare and provides technical assistance to the product and project task forces and contractors in the understanding and application of the standards and guidelines. |

| Term | Definition |
|---|---|
| Technical Design Specification (TDS) | The Technical Design Specification (TDS) translates the system requirements and functional design into precise descriptions of the components, interfaces, and data necessary before coding and testing can begin. The TDS must describe and document the design in the adequate level of detail and terminology to code, configure, build, integrate, and test the proposed product and all components, programs, databases, files, interfaces, security controls, screens, and reports. The TDS must be created for all development projects, but is not required for product MSE work efforts. |
| Test Plan | The test plan is a required major deliverable that describes the testing methodology, what will be tested, testing schedule, and testing deliverables. The test plan may be included or referenced in the project/product work plan or submitted as separate deliverable. |
| The Standards and Guidelines Notebook (S&G Notebook) | The Standards and Guidelines Notebook, also referred to as the S&G Notebook, is the published document and electronic repository of all approved AASHTOWare standards and guidelines. The notebook is available on the *AASHTOWare website*. |
| User Requirement | A user requirement describes what a user or business stakeholder expects from a proposed product (what the user wants to product to do). |
| User Requirements Specification (URS) | The URS is a required deliverable which contains all of the approved user requirements that are to be accomplished in a specified contract period for a specified project or product MSE work effort. The URS is normally incorporated in or referenced by the project/product work plan; however, in some cases, a separate document is created. |
| Voluntary Product Accessibility Template (VPAT) | The Voluntary Product Accessibility Template (VPAT) details how the AASHTOWare product complies with the federal Section 508 standards. |
| Waterfall Development Methodology | A waterfall development methodology is a sequential development process where each phase of development (planning, requirements and analysis, design construction, testing, implementation, etc.) is performed sequentially. |