



Solution Delivery Domain Document

New York State Department of Health (DOH)
Office of Health Insurance Programs (OHIP)

October 23, 2026

Contents

1.0	Introduction	1
1.1	Purpose.....	1
1.2	Objectives	1
1.3	Authority.....	2
1.4	Scope	2
1.5	How This Document Is Organized.....	3
1.6	Companion Domain Standards	3
1.7	Compliance Tier Framework	4
1.8	Compliance and Enforcement.....	6
2.0	Usability and Accessibility Standard	9
2.1	Introduction	9
2.2	Objectives	9
2.3	Legal and Regulatory Framework	9
2.4	Accessibility Standards	10
2.5	Usability Standards	13
2.6	Dashboard and Data Visualization Standards.....	16
2.7	Usability Assessment Tools	18
3.0	Requirements Management Standard	19
3.1	Introduction	19
3.2	Objectives	19
3.3	Requirements Definition.....	20
3.4	Requirements Gathering Gathering	21
3.5	Requirements Documentation.....	21
3.6	Requirements Change Management	24
3.7	Requirements Traceability.....	25
3.8	Verification and Validation	26
4.0	Policy and Business Rules Standard	28
4.1	Introduction	28
4.2	Objectives	29
4.3	MITA Alignment and Business Rules Engine.....	29
4.4	Security and Privacy Requirements	30

4.5	Business Rules Planning and Documentation.....	30
4.6	Rule Transparency and Readability	32
4.7	Rule Traceability.....	33
4.8	Rule Life Cycle.....	33
5.0	Code Standard	36
5.1	Introduction	36
5.2	Objectives	36
5.3	Ownership Rights and Repository Control	37
5.4	Code Leadership and Accountability	37
5.5	Secure Coding Standards	38
5.6	Approved Languages and Tools.....	38
5.7	Dependency Management	39
5.8	Unit Testing	39
5.9	Code Architecture and Design.....	40
5.10	Code Change Control	41
5.11	Code Quality	42
6.0	Test Management Standard	44
6.1	Introduction	44
6.2	Objectives	44
6.3	Test Standards	45
6.4	Test Planning.....	45
6.5	Test Execution.....	45
6.6	Operational Quality Checks.....	54
7.0	Release Management Standard.....	56
7.1	Introduction	56
7.2	Objectives	56
7.3	Release Types	57
7.4	Release Milestones.....	59
7.5	Release Schedule	61
7.6	Change Control and OHIP Interaction.....	62
7.7	Adding and Removing Items Post-Release Interaction	63
7.8	Release Communications	65
7.9	Deployment Activity Coordination for Multiple Integrations/Dependencies.....	65

7.10 Release Notes.....	68
7.11 Go/No-Go Process.....	69
7.12 Root Cause Analysis (RCA) Process	69
7.13 Post-Release Quality Assessment	69
7.14 Post-Release Review Meeting	70
7.15 Key Performance Indicators (KPIs).....	73
7.16 Release Oversight.....	75
8.0 Solution Development Life Cycle (SDLC) Standard	76
8.1 Introduction	76
8.2 Objectives	76
8.3 CMS Streamlined Modular Certification (SMC) Requirements.....	77
8.4 Methodology Requirements	78
8.5 Common SDLC Activities	111
8.6 Decommissioning and Data Archival Phase.....	115
9.0 Business and Technical Architecture Standard	116
9.1 Introduction	116
9.2 Objectives	116
9.3 Business Architecture Artifacts	117
9.4 Application Architecture Artifacts	124
9.5 Technology Architecture Artifacts	129
9.6 Security Architecture Artifacts.....	134
9.7 Architecture Governance and Compliance.....	134
10.0 Contact Information	136
11.0 Revision History.....	136
Appendices	137
Appendix A: Deliverables and Artifacts.....	138
Appendix B: Master Requirements Summary	163
Appendix C: Release Naming Guidelines	223
Appendix D: Requirements Management Guidance	225
Appendix E: MITA Service Definition Packages	231
Appendix F: SMART, INVEST, and Definition of Done (DoD) Criteria	234
Appendix G: Acronyms.....	235

Table of Tables

Table 1 – How this Document is Organized	3
Table 2 – Companion Domain Standards	3
Table 3 – Compliance Tier Framework	4
Table 4 – Domain Requirements Summary	5
Table 5 – Mandatory Quality Controls	48
Table 6 – Defect Severity Assignment	71
Table 7 – Urgency Level	72
Table 8 – Issue Impact	73
Table 9 – Priority Levels	73
Table 10 – Issues Matrix	73
Table 11 – Time/Deployment Based Statistics	74
Table 12 – Impact Statistics	74
Table 13 – SDLC Methodology Selection Guide	79
Table 14 – SDLC Methodology Comparison	80
Table 15 – SDLC Model Quick-Reference Comparison	81
Table 16 – SDLC Methodology Decision Framework	82
Table 17 – Waterfall Methodology Roles and Responsibilities	90
Table 18 – V-Shaped Phase Correspondence Structure	91
Table 19 – Product Backlog Template	94
Table 21 – Agile Methodology Roles and Responsibilities	99
Table 22 – Hybrid Iterative Development Roles and Responsibilities	107
Table 23 – Business Capability Model	117
Table 24 – Business Capability Matrix	119
Table 25 – Business Process Model	120
Table 26 – Use Case Model	122
Table 27 – Conceptual Integration Architecture Model	125
Table 28 – Application Component Model	126
Table 29 – Solution Architecture Model	127
Table 30 – Comparison of Integration Overviews	129

Table 31 – Deployment Model	130
Table 32 – Network Topology	132
Table 33 – Usability and Accessibility Standard Deliverables and Artifacts	138
Table 34 – Requirements Management Standard Deliverables and Artifacts	138
Table 35 – Policy and Business Rules Standard Deliverables and Artifacts	140
Table 36 – Code Standard Deliverables and Artifacts.....	142
Table 37 – Test Management Standard Deliverables and Artifacts	143
Table 38 – Release Management Standard Deliverables and Artifacts	145
Table 39 – SDLC Standard Deliverables and Artifacts.....	149
Table 40 – Business and Technical Architecture Standard Deliverables and Artifacts	161
Table 41 – Design Stage Activities (Part 1).....	225
Table 42 – Design Stage Activities (Part 2).....	226
Table 43 – Design Stage Activities (Part 3).....	227
Table 44 – Development Phase Activities (Part 1).....	228
Table 45 – Development Stage Activities (Part 2).....	228
Table 46 – Implement Phase Activities	229

Table of Figures

Figure 1 – Sample OHIP Master Release Schedule: Year-View.....	62
Figure 2 – Sample OHIP Master Release Schedule (90-Day View).....	75
Figure 3 – Waterfall SDLC	85
Figure 4 – V-Shaped Methodology.....	91
Figure 5 – Agile Scrum SDLC	92
Figure 6 – Hybrid Waterfall + Scrum	100
Figure 7 – Hybrid Waterfall + Iterative Approach.....	102
Figure 8 – Hybrid Iterative Development SDLC	103
Figure 9 – Sample Business Capability Model.....	118
Figure 10 – Sample Business Capability Matrix	120
Figure 11 – Sample Business Process Model.....	121

Figure 12 – Sample Use Case Interaction Diagram	123
Figure 13 – Sample Conceptual Integration Architecture	125
Figure 14 – Sample Application Component Model	126
Figure 15 – Sample Solution Architecture Model	128
Figure 16 – Sample Deployment Model	131
Figure 17 – Sample Network Topology	133

Issued by:	Finalized:
Office of Health Insurance Programs	10/23/26
Solution Delivery Domain Document	Owner: Division of Operations and Systems (DOS) and Chief Technology Office (CTO)

1.0 Introduction

1.1 Purpose

The Office of Health Insurance Programs (OHIP) Solution Delivery Domain Document establishes technical, operational, and quality standards governing how OHIP Medicaid Enterprise Systems (MES) solutions are designed, developed, tested, and delivered. This domain standard ensures all contractors implementing or enhancing OHIP systems do so with consistency, auditability, and in compliance with federal certification requirements.

The OHIP standards establish the mandatory technical, operational, and security requirements for New York State to remain compliant with federal regulations under 42 CFR § 433.112, qualifying the state for enhanced Federal Financial Participation (FFP). Deliverables and artifacts associated with this standard can be found in [Appendix A](#).

1.2 Objectives

The objectives of this Domain Document are to:

- Establish consistent, enforceable standards for the delivery of all MES solutions, ensuring quality, security, and compliance with federal and State requirements.
- Define a flexible Solution Development Life Cycle (SDLC) framework supporting Waterfall, Agile, and Hybrid methodologies tailored to each project's unique parameters.
- Ensure all requirements are clearly defined, documented, traceable, and validated throughout the project life cycle using OHIP-approved tools and practices.
- Govern the design, separation, traceability, and life cycle management of policy and business rules to support auditability and non-proprietary rule ownership.
- Enforce secure, maintainable, and well-documented coding practices with full State ownership of all developed code and assets.
- Establish a minimum set of quality controls and test types to be executed before, during, and after every software release, in alignment with the CMS MES Testing Guidance Framework.
- Provide structured release management processes—including release types, milestone tracking, change control, and communications—to ensure predictable and stable deployments.
- Integrate critical oversight functions—including security, privacy, enterprise architecture, and data governance—into every phase of the solution delivery life cycle.

- Facilitate alignment with CMS Certification processes, including Operational Readiness Review (ORR) and Streamlined Modular Certification (SMC) review.
- Establish a coherent set of business, application, technology, and security architecture artifacts and models—aligned to MITA and industry-standard notations (BPMN 2.0, UML, and ArchiMate 3.1)—that document how MES solutions are structured, integrated, and deployed in support of interoperability, modular procurement, and CMS certification.
- Clarify roles, responsibilities, activities, and required deliverables for each phase to ensure accountability across all project teams and Contractors.

1.3 Authority

The New York State Department of Health (DOH) is the primary authority overseeing the health, safety, and well-being of New Yorkers since 1901. The Department delegates direct operational control and administrative responsibility for the Medicaid Program to the OHIP. Title 11 of the New York State Social Services Law enables OHIP to establish and enforce the standards necessary to administer the State's Medicaid Program. This foundational authority, further supported by the broad regulatory powers vested in the Commissioner of Health by the NYS Public Health Law, enables OHIP to create comprehensive policies and procedures required for the effective, secure, and compliant operation of the MES.

For health information standards, compliance with HIPAA regulations is mandatory, with specific New York enforcement through the NYS Public Health Law. Article II of the NYS Technology Law mandates security policies that comply with both State and federal requirements. The NYS Information Security Breach and Notification Act ensures entities managing sensitive data adhere to privacy and security protocols.

OHIP applies these standards across both hybrid and non-hybrid entities in the Department, ensuring robust protection for all sensitive information including Protected Health Information (PHI), Personally Identifiable Information (PII), Medicaid Confidential Data (MCD), IRS Federal Tax Information (FTI), and other regulated data, aligning with federal and State mandates for data integrity, security, and privacy.

1.4 Scope

This Standard applies to specific contractually negotiated activities between OHIP and existing contracts and to new procurements/contracts identified as part of the evolving MES Roadmap.

The Contractor **MUST** demonstrate how the requirements of this standard will be met within their specific system context. Where system-specific constraints exist, the Contractor **MUST** document those constraints and propose alternative approaches for OHIP approval that achieve an equivalent level of compliance and rigor. Proposed alternatives do not relieve the Contractor of any obligation under this standard unless OHIP provides written approval through the Standards Deviation Request (SDR) process defined in [Section 1.7.1](#).

1.4.1 Multi-System Applicability

OHIP operates a diverse portfolio of systems that vary in architecture, technology stack, and operational maturity. This standard is platform-agnostic in its requirements while acknowledging that implementation approaches can differ across OHIP's portfolio of systems.

1.5 How This Document Is Organized

This document encompasses multiple related OHIP standards. Each component standard is presented as a dedicated section (Sections 2 through 9). An Appendix and Master Requirements Summary Table appears at the end of the document.

Table 1 – How this Document is Organized

Section	Domain Area
Section 2.0	Usability and Accessibility Standard
Section 3.0	Requirements Management Standard
Section 4.0	Policy and Business Rules Standard
Section 5.0	Code Standard
Section 6.0	Test Management Standard
Section 7.0	Release Management Standard
Section 8.0	SDLC Standard
Section 9.0	Business and Technical Architecture Standard

1.6 Companion Domain Standards

The following external OHIP standards directly govern, extend, or bound activities described in this domain standard. Standards previously cited as companions that have been consolidated within this document are now cross-referenced by section number. All references to individual component standards within Sections 2–9 **SHOULD** be read as references to that section of this document.

 **Note:** Internal cross-references between sections of this document (Sections 2 through 9) are provided within each section's introduction. The table below lists only external companion standards that are maintained as separate documents.

Table 2 – Companion Domain Standards

OHIP Standard	Domain	Scope Relationship
OHIP Data Architecture and Naming Standard	Data and Information Management	This standard governs data naming conventions, database design, data conversion, data lakehouse architecture, and content and document management. All solution delivery requirements related to data handling defer to it. It also governs the creation of conceptual, logical, and physical data models (CDM, LDM, PDM), data dictionaries, and modeling tools. The scope of this domain standard begins at the point of physical implementation.
OHIP Security and Privacy Compliance Standard	Security, Privacy and Compliance	Governs encryption, access control, de-identification, BAA execution, and PHI/PII classification. All data standards in this document defer to this standard for security matters.

OHIP Standard	Domain	Scope Relationship
OHIP ITSM Standard	Infrastructure and Operations	Governs general change management, incident management, capacity planning, and escalation procedures.
OHIP Preferred Operational Tools Standard	Infrastructure and Operations	Governs approved analytical, modeling, data quality, and project tracking tools. All tool selections MUST align with the approved list.
OHIP Log Management Standard	Infrastructure and Operations	Governs retention requirements for all pipeline, system, audit, and operational logs.

1.7 Compliance Tier Framework

This standard uses three compliance tiers applied consistently throughout all sections and in the Master Requirements Summary in [Appendix B](#). Every requirement has been individually vetted against enforceability, technical precision, and practicality before receiving its tier assignment. Tier designations appear in throughout the document and in the Tier column of [Appendix B](#).

Table 3 – Compliance Tier Framework

Tier	Type	Definition
MUST	Mandatory	MUST requirements are mandatory. Violation constitutes a material breach. Any deviation requires submission of a formal SDR and written approval from the OHIP IT Standards Team prior to implementation. MUST requirements are binding on all procurement activities and contracts and are subject to audit as part of CMS SMC certification reviews.
SHOULD	Strong Recommendation	SHOULD requirements are strongly recommended. Exceptions MAY be permitted under specific documented circumstances. Any deviations require written justification to the OHIP Project Manager for approval and MUST be documented in the project deviation log. SHOULD requirements are cited as expected practice in procurement activities and contracts. An SDR is not required; however, all deviations MUST be affirmatively justified in writing.
MAY	Permitted Option	MAY requirements denote a permitted option or permissible alternative at the discretion of the Contractor or OHIP. When a MAY option includes stated conditions, those conditions MUST be satisfied before the option MAY be exercised. MAY statements are informational within procurement activities and contracts and do not require prior approval or deviation documentation.

1.7.1 Standards Deviation Request (SDR) Process

A SDR is the formal mechanism by which a Contractor seeks approval to deviate from a **MUST** requirement prior to implementation. An SDR does not apply to **SHOULD** or **MAY** requirements. The SDR process is prospective: it **MUST** be completed and approved prior to the deviating approach is implemented. Retroactive SDRs will not be accepted and do not cure a breach.

The Contractor **MUST** submit a completed SDR to the OHIP IT Standards Team for any proposed deviation from a **MUST** requirement. An SDR is a prospective request: it **MUST** be submitted and

approved by OHIP before the Contractor implements any non-compliant approach. The SDR package **MUST** include:

- **Requirement ID:** the specific requirement identifier from Appendix B
- **Proposed Alternative:** a complete description of the approach to be used instead of the **MUST** requirement
- **Technical Justification:** documented rationale explaining why the **MUST** requirement cannot be met
- **Risk Assessment:** identification of risks introduced by the deviation and their likelihood and impact
- **Mitigation Controls:** specific measures that will reduce or compensate for identified risks
- **Proposed Timeline:** target date for achieving full compliance, if the deviation is temporary

OHIP will review the SDR and issue a written approval or denial within ten (10) business days of receipt of a complete submission. Incomplete submissions will be returned without review. Approved SDRs are logged in the OHIP Deviation Register with the effective date, any conditions attached to the approval, and the anticipated compliance date where applicable. Conditional approvals **MUST** be re-reviewed at the specified review date.

 **Note:** OHIP reserves the right to revoke an approved SDR if circumstances change, if conditions are not met, or if the deviation creates unforeseen risk to program integrity, federal compliance, or data security. Revocation of an SDR is subject to reasonable notice except where urgent risk requires immediate action.

1.7.2 Domain Requirements Summary

The following table summarizes requirement counts across the eight component standards. A complete master listing of all requirements with individual tier assignments appears in [Appendix B](#).

Table 4 – Domain Requirements Summary

Standard	Section	ID Range	MUST	SHOULD	MAY	Total
Usability and Accessibility Standard	2.0	UACC-0001 – UACC-0093	64	29	0	93
Requirements Management Standard	3.0	REQM-0001 – REQM-0084	63	21	0	84
Policy and Business Rules Standard	4.0	RULE-0001 – RULE-0061	51	10	0	61
Code Standard	5.0	CODE-0001 – CODE-0060	55	4	1	60
Test Management Standard	6.0	TMS-0001 – TMS-0076	62	13	1	76
Release Management Standard	7.0	RLM-0001 – RLM-0084	54	28	2	84
SDLC Standard	8.0	SDLC-0001 – SDLC-0170	126	41	3	170

Business and Technical Architecture Standard	9.0	BTA-0001 – BTA-0025	19	5	1	25
Total			494	151	8	653

Table key: **MUST** = mandatory. **SHOULD** = strongly recommended. **MAY** = permitted option.

1.8 Compliance and Enforcement

This section establishes the obligations, rights, and remedies that govern compliance with this standard throughout the contract term. This standard is incorporated into the Contract, and OHIP may update it during the term; the version in effect at the time of performance applies to the requirements it sets. The compliance and enforcement provisions in this section supplement, and do not replace, the remedies and enforcement terms of the governing Contract.

1.8.1 Applicability and Effective Date

The Contractor **MUST** comply with this standard and all applicable laws, regulations, and funding requirements. Compliance is required from the effective date of the Contract or Contract Amendment.

1.8.2 Updates to the Standard

OHIP reserves the right to amend this standard at any time. OHIP will use reasonable efforts to provide at least thirty (30) days' notice before enforcement of substantive changes. The Contractor **MUST** implement changes by the designated effective date.

For changes required by law, funding conditions, or urgent security risks, OHIP reserves the right to designate an earlier effective date. The Contractor **MUST** comply by that date regardless of other project commitments.

Within ten (10) business days of receiving notice of a substantive change, the Contractor **MUST** submit an impact assessment to OHIP and, if applicable, a proposed implementation plan including any requested cost or schedule adjustments. Acceptance of adjustments is at OHIP's sole discretion.

1.8.3 Monitoring, Audit, and Records Retention

OHIP and its designees reserve the right to audit the Contractor's and subcontractors' compliance with this standard upon ten (10) business days' notice, or less for cause or urgent security issues. The Contractor **MUST** provide reasonable access to relevant facilities, systems, records, and personnel upon request.

The Contractor **MUST** retain records evidencing compliance with this standard for at least seven (7) years from creation, or as otherwise required by law or funding requirements, whichever is longer.

1.8.4 Notification of Non-Compliance and Security Events

The Contractor **MUST** notify OHIP in writing within three (3) business days upon identification of any non-compliance with this standard.

Where non-compliance materially impacts the confidentiality, integrity, or availability of OHIP systems or data, or could result in legal or funding exposure, notification **MUST** be made immediately and no later than twenty-four (24) hours after discovery.

All notifications **MUST** describe: the nature and scope of the issue, root cause if known, immediate containment steps taken, and a proposed Corrective Action Plan (CAP) with timeline.

1.8.5 Corrective Action Plan (CAP)

Within ten (10) business days of notice, whether by the Contractor or OHIP of any non-compliance with this standard, the Contractor **MUST** provide a CAP acceptable to OHIP. The CAP **MUST** include defined milestones, responsible parties, and due dates for each remediation step.

OHIP reserves the right to impose or modify a CAP where the Contractor's proposed plan is insufficient.

1.8.6 Remedies and Enforcement

In addition to any other rights and remedies available under the Contract or applicable law, OHIP reserves the right to exercise any or all of the following remedies upon Contractor non-compliance:

- Withhold payments due under the Contract
- Assess service credits or liquidated damages as set forth in the Contract
- Suspend or limit the Contractor's system access
- Require removal of personnel or sub-contractors
- Require re-performance of non-compliant work at no additional cost to OHIP

Remedies are cumulative and can be exercised concurrently. OHIP can also refer matters to oversight bodies or regulators as required by law or policy.

1.8.7 Exceptions and Risk Acceptance

An exception is distinct from a SDR (see [Section 1.7.1](#)). An SDR is a prospective mechanism, it is submitted before a non-compliant approach is adopted. An exception, by contrast, applies where strict compliance is not technically feasible after implementation has begun, or where an existing system or constraint prevents full compliance by the effective date. OHIP will evaluate each exception request on its individual merits and will not grant exceptions that would violate applicable law, federal funding requirements, or CMS certification conditions.

To request an exception, the Contractor **MUST** submit a written request to OHIP that includes: a risk assessment identifying the nature and severity of non-compliance risk; proposed mitigation measures that reduce that risk to an acceptable level; and a proposed timeline either for achieving full compliance or for implementing a permanent alternative solution.

Exceptions are subject to the following conditions:

- Exceptions are time-bound—they expire at the date specified in the approval unless renewed by OHIP in writing.
- Exceptions are non-transferable—they apply only to the specific Contractor, system, and circumstance described in the request.
- Exceptions require OHIP's prior written approval before the non-compliant approach is adopted or continued.

- No exception will be granted that would violate applicable law, federal funding requirements, or OHIP policy.
- OHIP can revoke an approved exception at any time upon material changes in risk, regulatory requirements, or program needs.

 **Note:** *Approved exceptions are tracked in the OHIP Deviation Register alongside approved SDRs. Both represent formal acknowledgment by OHIP of a documented departure from standard requirements. Neither constitutes a waiver of OHIP's rights under the Contract.*

2.0 Usability and Accessibility Standard

2.1 Introduction

The Usability and Accessibility Standard governs how OHIP MES solutions are designed, built, and validated to ensure they are effective, efficient, and equitable for every user, from eligibility workers and claims processors to beneficiaries navigating benefit workflows. Accessibility is grounded in the four POUR principles (Perceivable, Operable, Understandable, Robust) and compliance with Section 508 of the Rehabilitation Act, WCAG 2.2 Level AA, and the Americans with Disabilities Act (ADA), and applicable New York State accessibility policies, including NYS ITS Policy NYS-P08-005 (Accessibility of Information Communication Technology). These requirements are explicitly aligned with CMS SMC guidance and the Medicaid Enterprise Certification Toolkit (MECT)¹. Compliance with this standard is a prerequisite for CMS certification and for OHIP's sustained eligibility for enhanced FFP (FFP).

2.2 Objectives

The primary objectives of this standard are to:

- Ensure all MES solutions comply with federal and state accessibility requirements, Section 508, WCAG 2.2 AA, and the ADA Title II, and NYS ITS accessibility policies, including NYS-P08-005, through the POUR accessibility framework.
- Establish measurable usability benchmarks, including task success rates, time-on-task targets, and System Usability Scale (SUS) scoring thresholds, so that usability is objectively demonstrable and auditable.
- Integrate accessibility and usability as core design requirements from the outset of development, not retrofitted after implementation.
- Define standards for dashboard and data visualization interfaces that ensure information is accurate, current, accessible, and interpretable without specialized training.
- Align all usability and accessibility requirements with CMS SMC guidance and the MECT to support federal certification.

2.3 Legal and Regulatory Framework

The following federal laws and standards establish the legal foundation for the accessibility and usability requirements in this standard. The Contractor is responsible for understanding and complying with all applicable provisions.

Americans with Disabilities Act (ADA)

Although enacted in 1990, the ADA is interpreted by the Department of Justice (DOJ) and federal courts to apply to websites and mobile applications under Title II (state and local government services) and Title III (public accommodations). Compliance is generally measured against the POUR principles and WCAG guidelines.

¹ For more information, refer to the “CMS Streamlined Modular Certification Guidance and the Medicaid Enterprise Certification Toolkit (MECT).”

Under Title II (state and local government services), the Department of Justice (DOJ) and federal courts apply ADA requirements to websites and mobile applications. In 2024, DOJ issued a final rule for Title II entities establishing WCAG 2.1 Level AA as the compliance baseline with specified timelines. OHIP requires WCAG 2.2 Level AA to exceed this baseline and position for forward compatibility.

Section 508 of the Rehabilitation Act

Requires federal agencies, and state programs receiving federal funding, to make electronic and information technology accessible to people with disabilities. Section 508 explicitly incorporates WCAG 2.0 via the Revised 508 Standards; OHIP adopts WCAG 2.2 AA to exceed the federal minimum.²

Section 504 of the Rehabilitation Act

Prohibits discrimination based on disability in any program or activity receiving federal financial assistance. For OHIP, this requires that all MES solutions, including digital services provided to beneficiaries, providers, and state staff, are accessible to individuals with disabilities.

Web Content Accessibility Guidelines (WCAG) 2.2 Level AA³

WCAG 2.2 Level AA, published by the W3C in October 2023, is the current version of the WCAG and the technical benchmark for this standard. WCAG 2.2 is backward compatible with WCAG 2.1, all WCAG 2.1 AA success criteria remain in effect, with nine additional success criteria addressing authentication, dragging movements, consistent help, redundant entry, focus appearance, and other interaction patterns not covered by prior versions. WCAG 2.2 is organized around the four POUR principles detailed in [Section 2.4](#). All MES user-facing components **MUST** meet WCAG 2.2 Level AA success criteria.

 **Note:** Section 508 of the Rehabilitation Act currently incorporates WCAG 2.0 through the ICT Standards. CMS SMC guidance references WCAG 2.1 AA. By adopting WCAG 2.2 AA, this standard exceeds the current federal minimum and positions OHIP for forward compatibility as federal standards are updated.

For additional guidance, refer to the U.S. Web Design System accessibility documentation (<https://designsystem.digital.gov/documentation/accessibility/>) and the ADA.gov regulations (<https://www.ada.gov/law-and-regs/>).

2.4 Accessibility Standards

Accessibility is a federal requirement and a core design principle ensuring OHIP solutions serve all users, including those with disabilities or assistive technology (AT) needs. All accessibility requirements are organized around the four POUR principles defined by WCAG 2.2.⁴

The Contractor **MUST**:

- Comply with Section 508, ADA Title II, NYS ITS accessibility requirements, and WCAG 2.2 AA.⁵

² For more information, refer to "[Section 508 of the Rehabilitation Act, as amended](#)."

³ For more information, refer to the "Web Content Accessibility Guidelines (WCAG) 2.2."

⁴ For more information, refer to "Part 11: Usability" in ISO 9241-11:2018: Ergonomics of human-system interaction."

⁵ For more information, refer to the "[CMS SMC Guidance](#)."

- Submit and maintain a current Voluntary Product Accessibility Template (VPAT), Revised 508 Edition, using the latest ITI template that supports WCAG 2.2 and EN 301 549, and provide an Accessibility Conformance Report (ACR) documenting conformance status for each WCAG success criterion.
- Facilitate both automated and manual accessibility testing.
- Obtain documented proof of accessibility compliance before releasing any OHIP MES solution.

2.4.1 Perceivability

Users must be able to perceive all information and user interface components. All information and interface components must be presentable in ways users can perceive through more than one sense, so content does not depend on a single sense. Example: Provide text alternatives (alt text) for images and captions for videos.

The Contractor **MUST**:

- Adhere to WCAG 2.2 AA perceivability success criteria.⁶
- Provide text alternatives for non-text content.
- Ensure sufficient color contrast and resizable text.
- Caption all video content and offer audio descriptions when applicable.
- Support reflow at a width of 320 CSS pixels without horizontal scrolling and zoom to 400% without loss of content or functionality.
- Avoid content that flashes more than three times per second; honor user motion preferences (prefers-reduced-motion) by providing non-animated alternatives.
- Verify compliance through automated and manual accessibility testing, documented in CMS modular certification packets (e.g., VPAT and accessibility scan reports).

2.4.2 Operability

Users must be able to interact with and navigate the interface. The design must not require interactions that a user cannot perform. Example: Ensure all functionality is accessible via a keyboard.

The Contractor **MUST**:

- Verify that the solution supports complete keyboard operability, proper focus management, and input method alternatives.
- Confirm interactive elements such as forms, menus, modals, and biometric prompts are accessible without the use of a mouse or gestures.
- Ensure that focus indicators meet WCAG 2.2 Focus Appearance and Focus Not Obscured criteria; maintain logical and visible focus order throughout all interactions.

⁶ For more information, refer to “WCAG 2.2 success criteria: 1.1.1 (Non-text Content), 1.2 (Time-based Media), 1.4.3 (Contrast), 1.4.4 (Resize Text), and 1.4.10 (Reflow).”

- Ensure that all functionality is available without gestures requiring path-based input; provide alternatives to dragging movements.
- Provide accessible alternatives to CAPTCHA; avoid reliance on visual puzzles or audio-only challenges.
- Develop solutions with extensions or adjustable durations for time-based tasks.
- Document proof of keyboard and input operability in usability validation and certification artifacts to align with CMS requirements.⁷

2.4.3 Understandability

Users must be able to comprehend both the information presented and how to operate the interface. Content should be readable and predictable. Example: Use plain language, consistent navigation, and helpful error messages.

The Contractor **MUST**:

- Provide consistent navigation patterns for user interfaces.
- Use clearly labeled and readable form fields.
- Develop descriptive error messages with guidance for resolution.
- Associate error messages programmatically to their fields; use ARIA live regions to announce status changes and validation results to assistive technologies.
- Use plain language targeting approximately a 6th–8th grade reading level for consumer-facing content; avoid jargon and idioms.
- Validate Understandability requirements through cognitive usability testing and accessibility compliance scans.
- Demonstrate user guidance and error prevention features to enable CMS certification.⁸

2.4.4 Robustness

Content must be compatible with current and future technologies, including assistive tools. Content must be robust enough to be interpreted reliably by a wide variety of user agents, including screen readers. Example: Use clean, standards-compliant code that works with screen readers and other assistive technologies.

The Contractor **MUST**:

- Provide semantically correct HTML, Accessible Rich Internet Applications (ARIA) roles, labels, and status so that content is programmatically determinable and assistive technology announces status messages.

⁷ For more information, refer to “WCAG 2.2 success criteria: 2.1.1 (Keyboard), 2.2.1 (Timing Adjustable), 2.4.3 (Focus Order), and 2.5 (Input Modalities).”

⁸ For more information, refer to “WCAG 2.2 success criteria: 3.1.1 (Language of Page), 3.2.3 (Consistent Navigation), 3.3.1 (Error Identification), and 3.3.2 (Labels or Instructions).”

- Ensure interactive widgets, biometric interfaces, and modal dialogs expose accessible roles and behaviors to AT.
- Perform manual AT testing using an OHIP-approved matrix that includes, at minimum: JAWS with Chrome, NVDA with Firefox, VoiceOver with Safari (macOS and iOS), and TalkBack with Chrome (Android). Include speech input and switch control operability checks. Document AT test coverage, findings mapped to specific WCAG criteria, remediation actions, and re-test results.⁹

2.4.5 Accessibility of Biometric Interfaces

The Contractor **MUST**:

- Avoid motion-only inputs and allow manual override for gesture-based or timed actions.
- Include clear, accessible instructions, retry prompts, and error recovery options.¹⁰
- Verify biometric accessibility through inclusive usability testing and document results in accessibility assessments submitted to CMS.
- Biometrics **MUST NOT** be the sole authentication factor. Non-biometric fallbacks **MUST** be consistent with NIST SP 800-63B and **MUST NOT** increase cognitive load relative to the biometric option.

The Contractor **SHOULD**:

- Provide non-biometric alternatives such as PIN entry, voice prompts, or knowledge-based questions.
- Ensure all feedback is perceivable to screen readers, keyboard-only users, and low-vision users.

2.5 Usability Standards

The following usability dimensions define the measurable quality attributes that all MES user-facing components **MUST** meet. These standards apply to all components developed, configured, or modified by the Contractor and **MUST** be integrated into design from the outset, not retrofitted after development. For the purposes of this standard, “components” means MES user-facing presentation-layer interfaces, including web pages and portals, mobile application screens, and dashboards. These usability dimensions derive from and align with ISO 9241-11 (usability) and ISO/IEC 25010 (system and software product quality).

 **Cross-Reference:** Usability testing activities must be integrated into the SDLC methodology selected under [Section 8.0](#). Test planning, execution, and acceptance criteria for usability are governed by the applicable testing requirements in [Section 6.0](#) (Test Management Standards).

⁹ For more information, refer to “WCAG 2.2 success criteria: 4.1.2 (Name, Role, Value) and 4.1.3 (Status Messages).”

¹⁰ For more information, refer to “WCAG 2.2 success criterion 2.5.4 (Motion Actuation).”

2.5.1 Effectiveness

The Contractor **MUST**:

- Ensure consumers, providers, and administrative staff can complete tasks such as eligibility determinations, plan selection, claims submissions, or benefit inquiries without confusion or error.
- Design solutions to support a high level of task effectiveness.¹¹

The Contractor **SHOULD**:

- Demonstrate that at least 90–95% of representative users can successfully complete core workflows in controlled testing. Exact targets **SHOULD** be defined per workflow and user segment in an OHIP-approved measurement plan; report results using percentile targets (e.g., P90) by segment.

2.5.2 Efficiency

The Contractor **SHOULD**:

- Evaluate solution designs against CMS usability performance metrics.
- Establish workflow-specific time-on-task targets by role and context in the measurement plan; report using percentiles (e.g., P75, P90). Example: for a trained user in a controlled test scenario, a new eligibility application **SHOULD** meet an OHIP-approved P75/P90 benchmark; targets **SHOULD** be calibrated per release as baseline data becomes available.

The Contractor **SHOULD**:

- Demonstrate task efficiency by targeting time-on-task benchmarks based on user role and context.

2.5.3 Satisfaction

The Contractor **SHOULD**:

- Design solution interfaces with straightforward navigation, responsive design, and user-centered content.
- Test solutions using the SUS or an equivalent tool, with a required minimum Target SUS ≥85 as a minimum; preferred SUS ≥90. If SUS is below 85, the Contractor **SHOULD** provide a remediation plan and demonstrate release-over-release improvement. Results **SHOULD** be reported by user segment.

2.5.4 Learnability

The Contractor **SHOULD**:

- Provide intuitive workflows and clear guidance that minimize the learning curve for solutions.
- Facilitate usability evaluations that include novice user testing.

¹¹ For more information, refer to “ISO 9241-11:2018 (6.2): Effectiveness, and the CMS SMC Guidance.”

The Contractor **SHOULD** ensure solutions support task success after no more than 15 minutes, or as agreed upon by OHIP, for routine functions (e.g., plan selection and address update).

2.5.5 User Experience (UX)

The Contractor **MUST**:

- Ground user interfaces in user-centered design practices.
- Test user interfaces in real-world scenarios.
- Demonstrate end-user involvement and iterative refinement of interfaces, in alignment with CMS requirements.
- Document UX research, personas, and feedback loops as part of solution design artifacts.
- Design all consumer-facing interfaces using a responsive approach, ensuring core workflows can be completed on standard mobile devices without loss of functionality.

2.5.6 Avoidance of Harm from Use

The Contractor **MUST**:

- Ensure solution designs include fail-safes, field validation, and error prevention patterns to reduce the potential for harm.
- Track, report, and mitigate harmful error events during usability testing and include stress testing with users under realistic constraints to align with applicable federal certification and regulatory requirements, including CMS SMC for Medicaid module functions and, for Exchange-related functions, applicable ACA and Qualified Health Plan (QHP) certification requirements.
- Define harmful error categories (e.g., misapplied eligibility determinations, PHI exposure, erroneous notices) with severity levels and remediation SLAs; instrument analytics to detect, monitor, and prevent recurrence.

2.5.7 Maintainability as Related to Usability

Maintainability has usability implications, particularly in terms of how easily the MES component can be updated without introducing confusion for users. Frequent releases and reconfigurations in MES environments demand UI stability and backward compatibility.

The Contractor **MUST**:

- Complete usability regression testing for all changes to MES user-facing components and preserve consistency across updates.
- Provide release notes highlighting user-facing changes and accessibility impacts; perform targeted accessibility regression testing for modified components and document results.

The Contractor **SHOULD**:

- Demonstrate maintainability through version control and documentation of usability impact post-deployment, in alignment with CMS requirements.
- Use design systems and component libraries to facilitate visual and interaction continuity.

2.5.8 Localization and Language Access

The Contractor **MUST**:

- Ensure all public-facing MES interfaces support multi-language presentation in compliance with New York State Language Access requirements (Executive Order 26.1).
- Support English and Spanish at a minimum, with architecture designed to scale to additional languages without requiring core application changes.
- Not rely solely on automated browser translation tools for critical Medicaid eligibility, enrollment, or benefit terminology.
- Ensure multilingual parity for alt text, labels, error messages, transcripts/captions, and downloadable content; support locale-specific number/date formats and allow for text expansion in translated strings.

2.6 Dashboard and Data Visualization Standards

The following standards apply to all interactive dashboards, data visualizations, and reporting interfaces developed for OHIP, including public-facing and internal operational dashboards. Requirements including data validation and output quality, and reporting workflows are governed by the OHIP Data and Information Management Domain Document.

Cross-Domain Compliance Notice

Contractors developing dashboard and data visualization interfaces for OHIP are subject to requirements in two complementary domain standards, both of which apply concurrently:

- This Standard (Solution Delivery, [Section 2.6](#)) governs the user interface and user experience layer: dashboard layout, color usage and contrast, labeling and tooltips, performance targets, data currency indicators, and accessibility compliance with Section 508 and WCAG 2.2 AA.
- The OHIP Data and Information Management (DIM) Domain Document governs the data layer that feeds dashboards and reports: data quality gates and validation for ingested and transformed data, data quality testing, and data lineage and metadata cataloging, per the Data Lakehouse Architecture Standard (Sections 6.6 and 6.14).

 **Note:** Where a specific requirement appears in both standards, the more stringent obligation applies and compliance with both must be demonstrated. Any question of applicability **MUST** be raised with OHIP in writing prior to design. Refer to [Section 1.7.1](#) (SDR) if full compliance with both standards is not technically feasible for a specific interface).

2.6.1 Dashboard Layout and Visual Hierarchy

The Contractor **MUST**:

- Design dashboards with a consistent layout that supports efficient information consumption.
- Place navigation elements, filters, and data currency indicators in consistent locations across all dashboard pages within an application.
- Establish a clear visual hierarchy that directs the user's attention to the most critical information first.
- Present summary-level metrics prominently, with the ability to drill down into supporting detail.

- Ensure that dashboard layouts are responsive and render correctly across standard desktop, tablet, and mobile screen resolutions as specified by OHIP.

2.6.2 Color Usage and Accessibility

The Contractor **MUST**:

- Use color consistently across all dashboards and data visualizations within an application, so that the same color represents the same category or meaning throughout.
- Not use color as the sole means of conveying information.
- Verify that all color-encoded information is also available through text labels, patterns, or other non-color indicators.
- Comply with WCAG 2.2 Level AA contrast ratios for all color combinations used for text, data points, and chart elements.
- Ensure that data visualizations remain interpretable when viewed in grayscale.

2.6.3 Labeling and Tooltip Standards

The Contractor **MUST**:

The Contractor **SHOULD**:

- Use plain language labels for all data elements, axes, and chart components.
- Not display technical field names or database column names to end users.
- Tooltips **SHOULD** be available on focus and activation on desktop and touch devices and announced by assistive technologies. Tooltips **SHOULD** be concise and scannable; generally avoid exceeding approximately 200 characters while allowing multi-line formatting when needed for accessibility.
- Design tooltips that are keyboard-accessible and conform to the accessibility requirements in [Section 2.4](#).
- Include descriptive titles, labeled axes with units of measure, and legends (where multiple data series are presented) for all charts and graphs.

2.6.4 Dashboard Performance Requirements

The Contractor **SHOULD**:

- Design dashboards to meet performance targets under defined test loads: P75 load times of 3–5 seconds and P95 ≤ 8 seconds under normal operating conditions.
- Display an accessible loading indicator with progress messaging; announce loading and completion states via ARIA-live status updates.
- Optimize queries and data structures supporting dashboards to meet the specified performance targets.

The Contractor **SHOULD** implement drill-down and filter capabilities to allow users to navigate from summary views to detailed data without requiring separate page loads where technically feasible.

2.6.5 Data Currency and Transparency

- Display a visible notification to users indicating the data currency issue when data is known to be stale or when a scheduled refresh has not completed.

The Contractor **SHOULD**:

- Display the data refresh date and time on every dashboard page.
- Clearly communicate the update frequency for each data source used in the dashboard (e.g., real-time, daily, weekly, monthly).
- Identify the source system(s) feeding the dashboard at the data-source/dataset level (e.g., NYSOH, eMedNY, WMS); where a displayed value blends multiple sources, indicate the contributing sources. Field-by-field source identification is not required.
- Display a visible notification to users indicating the data currency issue when data is known to be stale or when a scheduled refresh has not completed.

2.6.6 Chart Selection and Branding Standards

The Contractor **SHOULD** select chart and graph types appropriate to the data being presented (e.g., trends over time, part-to-whole comparisons, distributions, rankings) and **SHOULD NOT** use visualization types that distort or misrepresent the underlying data, including truncated axes without clear indication, 3-D effects on quantitative charts, and dual axes without distinct labeling. All dashboards, visualizations, and report interfaces **SHOULD** conform to OHIP/DOH branding requirements, including approved logos, color palettes, and typography, as provided by OHIP.

The Contractor **SHOULD**:

- Ensure all dashboards, visualizations, and report interfaces conform to OHIP/DOH branding requirements, including approved logos, color palettes, and typography, as provided by OHIP.

2.7 Usability Assessment Tools

The Contractor **SHOULD**:

- Utilize the SUS Questionnaire and Scoring Guide to measure overall user usability perception, gather and convert raw scores, and interpret the final usability score.
- Utilize the VPAT (Revised 508 Edition) to verify WCAG 2.2 AA and Section 508 compliance.

 **Note:** For WCAG 2.2 AA and Section 508 compliance verification tools, refer to the VPAT requirement in [Section 2.4](#) of this standard.

3.0 Requirements Management Standard

3.1 Introduction

The Requirements Management Standard governs how solution requirements, including but not limited to business, functional, and non-functional, are gathered, documented, analyzed, traced, validated, and maintained throughout the life cycle of every OHIP MES project. Rigorous requirements management is foundational to every other delivery discipline: testing cannot verify what has not been defined; architecture cannot be designed without validated scope; and CMS certification cannot be demonstrated without traceable evidence that the system was built to meet the outcomes it claims to support.

In a multi-vendor MES environment, requirements management carries additional weight. Gaps, ambiguities, or inconsistencies in requirements documentation propagate across dependent systems and Contractors, creating integration failures, rework, and schedule risk that are difficult and expensive to recover from. This standard establishes the structures of the Requirements Management Plan, the Requirements Traceability Matrix (RTM), and a formal requirements life cycle that prevent those failures by ensuring requirements are a living, continuously validated contract between OHIP and its Contractors throughout every phase of delivery.

All activities for requirements under this standard **MUST** be performed using OHIP-approved tooling, documented in approved formats, and submitted to OHIP for formal review and approval at each life cycle milestone. The RTM is not a project artifact completed once; it is a controlled record that **MUST** be current, accurate, and accessible to OHIP at all times. Deliverables and artifacts associated with this standard can be found in [Appendix A](#).

 **Note:** This standard is organized around six core disciplines: Requirements Gathering, Requirements Documentation, Requirements Analysis, Requirements Change Management, Requirements Traceability, and Verification and Validation. These disciplines are informed by recognized industry guidance, including ISO/IEC/IEEE 29148 (Systems and software engineering—Life cycle processes—Requirements engineering) and PMI requirements management practice. Such references are informative; compliance obligations are defined solely by the requirements of this standard.

3.2 Objectives

The primary objectives of this standard are to:

- Establish a structured, repeatable process for gathering, documenting, analyzing, and managing requirements across all OHIP MES projects.
- Define clear standards for creating and maintaining the Requirements Management Plan and RTM as controlled, version-managed deliverables with OHIP approval at each baseline.
- Centralize requirements documentation and management in OHIP-approved tooling, ensuring a single, authoritative source of record accessible to OHIP and Contractors at all times.
- Ensure all requirements maintain continuous forward and backward traceability, linked from their business and regulatory sources—including applicable CMS regulations and certification criteria, ACA and QHP requirements for Exchange-related functions, and federal and State requirements—through design artifacts, source code, test cases, and implementation evidence, to support audit readiness, CMS SMC, and other applicable certification and compliance reviews.

- Provide consistent processes for requirements change management that reduce ambiguity, prevent scope creep, and support efficient collaboration across multiple Contractors and OHIP stakeholders.
- Ensure all requirements are validated against OHIP business needs and CMS certification criteria before development begins, reducing rework and ensuring the delivered solution meets its intended outcomes.

3.3 Requirements Definition

Requirements under this standard are classified into three types. The Contractor **SHOULD** ensure that all three types are identified, documented, and traced for every MES project:

- **Business Requirements:** Business capabilities and processes that capture the goals, objectives, and outcomes that the Medicaid program aims to achieve. Business requirements reflect the strategic priorities of the agency and define what the organization intends to accomplish with the solution.
- **Functional Requirements:** Specifications of what the system **MUST** do to support business objectives. Functional requirements describe the specific behaviors, features, and interactions the system provides, serving as the reference for developers and testers.
- **Non-Functional Requirements (NFR):** Quality attributes defining how the system **MUST** perform, including performance, reliability, security, usability, scalability, and technical constraints. NFRs shape the solution's design and architecture.

These three types are the primary classification for all requirements and are intentionally comprehensive: specialized requirement categories—including technical, interface, integration, reporting, data retention, security, accessibility, and privacy requirements—are classified within them according to whether they describe system behavior (functional) or system qualities and constraints (non-functional). To preserve visibility into these categories, the Contractor **SHOULD** further categorize each requirement in the OHIP-approved requirements repository using OHIP-defined attributes, including at a minimum: requirement category (e.g., interface, reporting, data retention, security, accessibility), regulatory or compliance source where applicable (e.g., CMS regulation, SMC outcome, ACA/QHP requirement, State regulation, OHIP policy), and the originating business area. Attribute-level categorization enables filtered traceability and compliance reporting without fragmenting the primary classification model.

3.3.1 Requirement Quality and Readiness

Every requirement **MUST** meet the following quality criteria before proceeding to development:

- **Testable:** includes defined acceptance criteria and an identified test method.
- **Traceable:** linked forward to design, code, and tests, and backward to its source and rationale.

Every requirement **SHOULD** also meet the following quality criteria:

- **Atomic:** each requirement addresses a single capability or constraint.
- **Unambiguous:** the requirement has one clear interpretation and is consistent with other requirements.
- **Necessary:** justified by a documented business need or regulatory driver.
- **Feasible:** achievable within known constraints.

- **Prioritized:** ranked using the OHIP-approved prioritization method.

 **Cross-Reference:** For Definition of Ready (DoR) and Definition of Done (DoD) criteria applicable to requirements and user stories, refer to [Appendix F](#).

3.4 Requirements Gathering

Requirements gathering is the systematic process of identifying and collecting requirements from all relevant stakeholders, regulatory sources, and existing system documentation. The Contractor **MUST**:

- Conduct structured requirements elicitation activities including stakeholder interviews, workshops, document analysis, and observation to identify business, functional, and NFRs.
- Engage OHIP business owners, subject matter experts, and end users throughout the gathering process to ensure requirements accurately reflect operational needs.
- Identify and document requirements from all applicable regulatory and compliance sources, including CMS certification conditions, SMC outcomes and metrics, ACA and QHP requirements for Exchange-related functions, federal and State regulations, and OHIP policies.
- Review and incorporate requirements from existing system documentation, including current system specifications, known deficiencies, and enhancement requests.
- Load all gathered requirements into the OHIP-approved requirements repository.

To support multi-vendor coordination, the Contractor **MUST** participate in cross-vendor dependency sessions and backlog refinement cadences established by OHIP to identify interface requirements, sequencing needs, and shared service dependencies early in the gathering process.

The Contractor **SHOULD** review and update requirement attributes for the Business Area/Functional Area and the Business Process Titles as applicable. The Contractor **SHOULD** review and update requirement attributes with the applicable CMS Certification Outcomes, Conditions, and Metrics. Business Owners and the OHIP Enterprise Architecture (EA) team confirm all updates.

3.5 Requirements Documentation

The Contractor **MUST** produce and maintain two core deliverables that govern the requirements management process: the Requirements Management Plan and the RTM. Together, these deliverables provide a comprehensive framework for capturing, managing, and validating requirements throughout the project life cycle. Each is described below.

3.5.1 Requirements Management Plan

The Requirements Management Plan serves as the blueprint for how requirements are identified, documented, analyzed, and managed throughout the project. The Contractor **MUST**:

- Draft the Requirements Management Plan explaining the Contractor's approach to complying with this standard.
- Establish communication protocols to ensure stakeholders are informed and engaged at every stage.
- Define roles and responsibilities (RACI) for requirements gathering, analysis, approval, and maintenance, including cross-vendor coordination responsibilities.

- Specify the OHIP-approved prioritization method and backlog refinement cadence.
- Define the governance for requirements change control, including decision authority and escalation pathways.
- Submit the Requirements Management Plan to OHIP for review and approval before requirements elaboration begins.

3.5.2 Requirements Traceability Matrix (RTM)

The RTM links each requirement to its source and tracks its status through the development life cycle, ensuring every requirement is accounted for and verified against the final deliverables. This section defines the RTM as a controlled deliverable; the traceability obligations the RTM must evidence, and the life-cycle stages through which the RTM evolves, are defined in the Requirements Traceability section of this standard. The Contractor **MUST**:

- Develop and maintain the RTM as a controlled, version-managed deliverable.
- Ensure each RTM entry identifies the requirement, its origin (e.g., stakeholder request, regulatory requirement), and its relationships to design elements, test cases, and project objectives.

Include direct references to:

- o Design artifacts (e.g., process flows, use cases) and Architecture Decision Records (ADRs) where applicable.
 - o Interface specifications and API specifications for cross-system requirements.
 - o Test cases, procedures, and results (functional, integration, regression, security, accessibility, and performance), with links to objective evidence (logs, screenshots, or reports).
- Provide an initial RTM version during the Design Stage and update it throughout all SDLC phases for every Release.
- Baseline each iteration of the RTM upon OHIP approval. Previous data **MUST** be updated in each RTM iteration to maintain current requirement traceability.
- Submit each RTM iteration to OHIP for review and approval.
- Demonstrate 100% forward and backward traceability for all baselined requirements prior to production deployment. Zero open Severity 1 or Severity 2 defects tied to release-scoped requirements unless explicitly risk-accepted in writing by OHIP.

3.5.3 Requirements Management Tools

The Contractor **MUST** use the Department-designated Requirements Management tooling for all requirements management activities. The OHIP Infrastructure and Operations Domain Document, Preferred Operational Tools Standard, lists the designated tools. If any additional or alternate technologies are needed to support solution development, the Contractor **MUST** propose the requirements and seek approval from OHIP.

The Department-designated Requirements Management tooling **MUST** provide:

- Unique, immutable requirement IDs.
- Versioning with full change history and audit trail.

- Role-based access controls.
- Export and reporting features sufficient to produce audit-ready evidence for CMS certification.

3.5.4 Business Analysis Quality Assurance (BAQA)

The Contractor **SHOULD** establish and operate a Business Analysis Quality Assurance (BAQA) program that delivers consistent, traceable, and implementation-ready requirements and maintains RTM readiness across all delivery increments. The BAQA program **SHOULD** define the methods, artifacts, review gates, and metrics used to validate business needs and prepare work for execution. As part of the Requirements Management Plan ([Section 3.5.1](#)), the Contractor **SHOULD** include a BAQA Plan that describes the standards, templates, tools, review cadences, metrics, and escalation paths that govern BAQA; the plan **SHOULD** be applied during Scope Prioritization and maintained throughout all phases of the SDLC.

The Contractor **SHOULD** establish standards, consistent with those defined by the Department, for use cases, acceptance criteria, and nonfunctional requirements, and **SHOULD** maintain end-to-end traceability from business objectives through requirements, test cases, and Releases. To support high-quality requirements, the Contractor **SHOULD** use structured elicitation, rapid prototyping, and usability testing for high-risk items, as well as asynchronous validation when subject matter expert (SME) availability is limited.

The Contractor **SHOULD** implement peer reviews of requirements and scenarios prior to RTM commitment. No work item may enter an Iteration unless it meets the entry criteria and passes BAQA peer review; any exception **SHOULD** include a documented mitigation plan tracked to closure. At a minimum, entry criteria **SHOULD** include:

1. Linkage to business objectives and applicable policy and process references
2. Behavior-Driven Development (BDD) acceptance criteria
3. Identified dependencies and impacts
4. Prototype or annotated artifacts for high-risk items
5. Product Owner and BAQA lead approval

The Contractor **SHOULD** track and report RTM readiness rate, requirements defects that escape to delivery, rework due to requirements gaps, peer review findings, and corrective actions. Regular BAQA status, including audit results and continuous improvement activities, **SHOULD** be provided to project leadership and made available to the Department upon request. The Contractor **SHOULD** conduct periodic audits of requirements quality and traceability and capture lessons learned, updating standards, checklists, and templates accordingly. The Contractor **SHOULD** maintain tools and repositories that support version control and comment history and provide end-to-end linking traceability across objectives, use cases, tests, and Releases, aligned with the OHIP Infrastructure and Operations Domain Document, Preferred Operational Tools Standard.

3.5.5 Requirements Analysis

Requirements analysis transforms gathered requirements into detailed, testable specifications. The Contractor **MUST**:

- Elaborate all Solution Requirements to a testable and verifiable level or identify acceptance criteria for requirements that cannot be tested through automated means.
- Prioritize Solution Requirements in collaboration with OHIP Business Owners.

- Assess requirements for completeness, consistency, feasibility, and potential conflicts with other requirements or system constraints.

NFRs (NFRs)

The Contractor **SHOULD**:

- Identify dependencies between requirements, including cross-Contractor and cross-module dependencies in the multi-vendor MES environment, informed by the cross-vendor dependency sessions conducted under the Requirements Gathering discipline.
- Collaborate with the OHIP Certification team to ensure that the correct CMS Outcomes and Metrics IDs are traced to applicable requirements.
- Resolve requirements conflicts, ambiguities, or gaps before development begins. Unresolved requirements issues **MUST** be escalated to OHIP for determination.

Non-Functional Requirements (NFRs)

- NFRs **MUST** be quantified with measurable targets, defined test methods, and pass/fail criteria. At minimum, address:
- Performance and scalability (e.g., P75/P90/P95 response times, throughput, concurrent user loads).
- Availability and reliability (e.g., service level objectives, RTO/RPO).
- Security and privacy (e.g., authentication and authorization requirements, encryption, audit logging, HIPAA minimum necessary).
- Accessibility and usability (e.g., WCAG 2.2 AA conformance, SUS targets per [Section 2.0](#)).
- Interoperability (e.g., X12 5010, FHIR, HL7 as applicable to the module).
- Data quality (accuracy, completeness, timeliness).

Data, Interfaces, and Migration

The Contractor **MUST** define and trace requirements for external interfaces (ICDs and data contracts), data transformations, and reporting dependencies. Data migration and conversion requirements **MUST** specify reconciliation criteria and acceptance thresholds.

3.6 Requirements Change Management

Requirements change management governs how modifications to baselined requirements are proposed, evaluated, approved, and incorporated. Changes to requirements after baseline approval carry significant downstream impact on design, development, testing, and certification artifacts. The Contractor **MUST**:

- Submit all proposed changes to baselined requirements through a formal change request process approved by OHIP.
- Changes to baselined requirements are governed by a Requirements Change Control Board (RCCB) or equivalent governance body with defined membership, decision authority, and escalation pathways. Emergency changes **MUST** follow a documented expedited path and be ratified at the next scheduled RCCB meeting.
- Document for each proposed change: business justification; impact analysis across scope, schedule, and cost; operational impact analysis identifying all currently operating functionality, interfaces, batch processes, reports, dashboards, and business rules that the change could

affect, determined using the traceability links maintained under the Requirements Traceability discipline; affected traceability links; NFR impacts; security and privacy impact assessment; cross-vendor and cross-module dependencies; and required test updates, including regression test coverage for every operational function identified as potentially affected.

- Obtain OHIP approval before implementing any change to a baselined requirement.

The Contractor **SHOULD**:

- Update the RTM, design artifacts, test cases, and all downstream traceability links to reflect approved changes.
- Maintain a change log that records all requirement changes, including the original requirement, the nature of the change, the approval decision, and the date of implementation.
- Perform and document a production impact assessment before approving any requirement change to a solution component in production operation, identifying affected operating functions, interfaces, batch processes, reports, and business rules using RTM traceability, and defining the regression test scope that demonstrates existing functionality is not degraded prior to deployment.

For any requirement change proposed after a solution component has entered production operation, the Contractor **MUST** perform and document a production impact assessment before the change is approved for implementation. The assessment **MUST** identify the operating functions, interfaces, batch processes, reports, and business rules potentially affected—using RTM traceability as the verification source rather than developer recollection—and **MUST** define the regression test scope that will demonstrate, prior to deployment, that existing functionality is not degraded. Changes deployed without the documented production impact assessment and completed regression coverage constitute non-compliance with this standard.

Once implemented, any system modifications **MUST** be reviewed, and the RTM **MUST** be updated as appropriate to ensure accuracy.

 **Cross-Reference:** Requirements changes that affect system scope, schedule, or budget are also subject to the change control processes defined in the OHIP Program and Project Management Domain Document.

3.7 Requirements Traceability

The RTM section above defines the RTM deliverable itself; this section defines the traceability discipline that the RTM must demonstrate and the stages through which it matures. Requirements traceability ensures that every requirement can be traced forward from its business source through design, development, testing, and implementation, and backward from any implementation artifact to the requirement it satisfies. The Contractor **MUST**:

- Maintain continuous forward and backward traceability for all requirements throughout the development life cycle.
- Link requirements to the business area and the business need each is designed to meet.
- Link requirements to data models, analysis results, source code, test cases, test procedures, test results, and documentation produced as a result of the requirement.

- Maintain traceability even when a single requirement corresponds to multiple deliverables or artifacts within the system implementation life cycle. Common data, reporting, and shared services often reuse the same interface requirements, system requirements, networks, and user interfaces.
- Develop design artifacts that include traceability to Solution Requirements. Where applicable, design artifacts **MUST** be traced to requirements in the repository.
- Trace test cases to requirements in the repository. Each test case **MUST** reference the specific requirement(s) it validates.
- Ensure data lineage is documented from requirement through logical and physical data models to ETL/API implementations to reports and dashboards, including validation and reconciliation evidence where applicable.

3.7.1 Requirements Traceability Matrix (RTM) Life Cycle

The RTM evolves through three stages, Design, Development, and Implementation (DDI). Each iteration of the RTM Deliverable **MUST** further elaborate and update traceable attributes for each stage. The Contractor **MUST**:

- **Design Stage:** Identify and build out all requirements. Elaborate Solution Requirements to testable specifications. Establish traceability between requirements, design artifacts, and CMS certification elements. Submit the first RTM baseline for OHIP approval.
- **Development Stage:** Update the RTM with development traceability, including source code references, unit test mappings, and integration test cases. Provide Release information for each requirement. Confirm NFR testability with OHIP. Submit the updated RTM for OHIP approval.
- **Implementation Stage:** Validate test results and confirm traceability between test cases and requirements. Facilitate an RTM walkthrough meeting demonstrating full traceability, requirements coverage, accuracy, and completeness. Update each requirement's status to implemented or baselined. Submit the final RTM update for OHIP approval.

The detailed activity tables for each stage (DDI), including specific responsibilities and project documentation requirements, are provided in [Appendix D](#).

3.8 Verification and Validation

Verification confirms that the system was built correctly (i.e., meets its specifications). Validation confirms that the system meets the business needs it was intended to address. The Contractor **MUST**:

- Verify that each requirement has been implemented as specified by reviewing design artifacts, code, and test results against the documented requirement.
- Validate that the implemented solution meets OHIP's business needs through structured acceptance testing with OHIP business owners and end users.
- Demonstrate adequate testing coverage for all requirement types, including functional, non-functional, integration, regression, and user acceptance testing (UAT).
- Collaborate with OHIP staff responsible for CMS certification to ensure correct CMS Outcomes and Metrics IDs are traced to applicable requirements at each life cycle stage.
- Review, update, and confirm traceability between test cases and requirements, validating the testing status for each requirement in the RTM.

No requirement **MUST** be considered complete until it has been verified through testing (or an approved alternative verification method) and validated by OHIP business stakeholders.

Release Readiness Exit Criteria

Prior to production deployment, the Contractor **MUST** demonstrate and document:

- Requirements coverage: 100% of release-scoped requirements traced forward and backward with passing verification evidence.
- Acceptance criteria: met for each requirement with objective evidence.
- Security and privacy: verification passed per applicable controls; no critical findings outstanding.
- Accessibility and usability: conformance demonstrated per [Section 2.0](#).
- Performance and scalability: NFR targets met under defined loads and environments.
- UAT: OHIP business owner sign-off obtained.
- Defects: no open Severity 1 or Severity 2 defects in release scope unless explicitly risk-accepted in writing by OHIP with a mitigation plan and timelines.

The Contractor **SHOULD**:

- Provide the resolution plan for any outstanding requirements concerns identified during verification or validation.
- The Contractor **SHOULD NOT** proceed to production deployment with requirements in an unverified or unvalidated state unless a specific exception is approved by OHIP in writing.

 **Cross-Reference:** Testing standards, test management processes, and test environment requirements are governed by [Section 6.0](#) (Test Management Standards) of this domain document.

4.0 Policy and Business Rules Standard

4.1 Introduction

The Policy and Business Rules Standard governs how the logic that drives program eligibility determinations, claims adjudication, benefit calculations, enrollment processing, and all other policy-driven system behaviors is defined, structured, managed, and maintained across the OHIP MES. OHIP operates a unified entry point for multiple public health programs, including Medicaid (MA), Child Health Plus (CHP), the Essential Plan (EP), and Qualified Health Plans (QHP) through the New York State of Health (NYSOH) marketplace. Business rules are the codified expression of federal regulation, State statute, and agency policy that govern all of these programs, and they are among the most consequential software components in operation, directly affecting the coverage, payments, and services that beneficiaries, providers, and enrollees receive.

A defining principle of this standard, grounded in the Medicaid Information Technology Architecture (MITA) framework, is that business rules **MUST** be separated from core application code.¹² The MITA framework states that *“states should ensure the use of business rules engines to separate business rules from core programming, and should provide information about the change control process that will manage development and implementation of business rules. States should be able to accommodate changes to business rules on a regularized schedule and on an emergency basis.”* Rules that are embedded in application logic become opaque, difficult to audit, and costly to change when statutes or regulations are updated. By requiring rules to be externalized, whether through a dedicated business rules management system, a configurable rules engine, or an equivalent mechanism approved by OHIP, this standard ensures that OHIP retains full transparency and control over how policy is implemented, and that rule changes driven by annual regulatory updates, court decisions, or legislative amendments can be made efficiently without requiring full application redeployment. Static documents and spreadsheets **MUST NOT** serve as the authoritative repository or system of record for business rules. Spreadsheets and similar artifacts **MAY** be used only as supplementary analysis or communication aids, and any rule content they contain **MUST** originate from, and be reconcilable to, the rules maintained in the OHIP-approved repository.

This standard establishes requirements across the full rule life cycle: from initial planning and documentation through transparency, traceability, human readability, decoupling from application code, versioning, change management, and eventual archival. Every rule managed under this standard **MUST** be traceable to its authoritative legal or policy source, authored in business-friendly language that non-technical stakeholders can read and validate, and available to OHIP in both human-readable and machine-readable formats at all times. Deliverables and artifacts associated with this standard can be found in [Appendix A](#).

¹² For more information, refer to the “Business Rules Engines” subsection of Chapter 2 in the [Medicaid Information Technology Architecture Framework, Part III](#).

4.2 Objectives

The primary objectives of this standard are to:

- Assert OHIP's ownership and visibility over all business rules by requiring that rules reside in a State-managed repository, exposed via standard APIs, and available for demonstration to OHIP upon request at any time.
- Enforce the MITA principle of separating business rules from core application code, enabling rules to be deployed, versioned, and updated independently of application releases to support agile response to regulatory and legislative change.
- Ensure complete traceability of every policy rule from its authoritative source, statute, regulation, or official policy document, through to its implementation, tested behavior, and production deployment.
- Require rules to be written in clear, business-friendly language validated by OHIP stakeholders before implementation, ensuring rules accurately reflect policy intent and are auditable by non-technical reviewers.
- Establish a rigorous life cycle for rule creation, versioning, change management, and archival that supports audit readiness, CMS certification evidence, and long-term regulatory compliance.
- Protect the integrity and reusability of business rules across the MES by prohibiting the duplication of rule logic across systems and mandating standardized API exposure for cross-application consumption.

Empower OHIP business users to understand, review, and validate business rules without requiring technical expertise, ensuring that rule logic remains transparent and accountable to program policy owners.

4.3 MITA Alignment and Business Rules Engine

The Contractor **MUST** separate business rules from core application code and make rules available in both human-readable and machine-readable formats across every program OHIP administers through the MES and NYSOH, including Medicaid, CHP, the Essential Plan, and QHP processing. For Medicaid functions, this requirement implements the MITA framework, which mandates separating business rules from core programming. For Exchange-related functions, the same externalization requirement applies as an OHIP obligation, consistent with applicable federal Exchange program requirements under the ACA. All of these programs operate under frequently changing statutes, regulations, and policies, and standardized, externalized rule management enables rapid changes to system logic without modifying application code.

The Contractor **MUST**:

- Externalize all policy and business rules from application code. Rules **MUST NOT** be embedded within the application's programming code base.
- Implement a business rules management approach, whether through a dedicated rules engine, a configurable rules management system, or an equivalent mechanism, that separates rule logic from application logic and is approved by OHIP.
- Expose all business rules through standardized, versioned APIs so multiple MES applications can consume the same logic without duplication. API contracts **MUST** define request/response

schemas, error codes, versioning strategy, and backward compatibility approach. Requests and responses **MUST** be deterministic and auditable.

- Support rule changes on both a regularized schedule (e.g., annual regulatory updates, rate changes) and on an emergency basis (e.g., court orders, legislative amendments with immediate effective dates).
- Ensure the rules management approach empowers OHIP business users to review, understand, and validate rule logic without requiring programming expertise.

The Contractor **SHOULD** use tools listed in the OHIP Infrastructure and Operations Domain Document, Preferred Operational Tools Standard. If an alternative tool is required, the Contractor **SHOULD** propose the alternative and obtain OHIP approval before implementation.

4.4 Security and Privacy Requirements

The Contractor **MUST** comply with the OHIP Security, Privacy, and Compliance Domain Document, Security and Privacy Compliance Standard. When developing, implementing, or enhancing business rules and processes, the Contractor **MUST** ensure full compliance with the applicable security and privacy standards and baselines, including:

- OHIP Security, Privacy, and Compliance Domain Document, Security and Privacy Compliance Standard
- OHIP Application Programming Interface (API) Secure Development Baseline
- OHIP Artificial Intelligence (AI) Security and Privacy Baseline (where AI-assisted rule logic is employed)

The Contractor **SHOULD**:

- Comply with the OHIP Security, Privacy, and Compliance Domain Document when developing, implementing, or enhancing business rules.

4.5 Business Rules Planning and Documentation

The Contractor **MUST** provide and obtain OHIP approval for a Policy and Business Rules Implementation Plan and Business Rules Documentation before rule development begins.

4.5.1 Policy and Business Rules Implementation Plan

The Contractor **MUST** develop and submit a Policy and Business Rules Implementation Plan for OHIP review and approval. The plan **MUST** describe the Contractor's approach to defining, implementing, testing, deploying, and maintaining business rules in compliance with this standard, including:

- The rules management architecture and technology approach (rules engine, BRMS, or equivalent).
- The process for separating business rules from application code.
- The change control process for rule creation, modification, and deployment.
- Roles and responsibilities for rule authoring, review, testing, and approval.

The approach for accommodating both scheduled rule changes and emergency rule deployments.

- Rule API standards, including versioning, request/response schema, error handling, and performance targets.
- The approach for effective dating, including support for retroactive application and parallel operation of multiple rule versions when policy requires grandfathering or staggered effective dates.
- Testing strategy, including unit and decision-table coverage targets, regression testing, backtesting against representative historical data to validate expected impact and detect anomalies before release, and performance testing for rule APIs.
- The Contractor **MUST** use de-identified or masked data for all rule testing, adhering to OHIP security and privacy standards.

The Plan **SHOULD** also include:

- A current-state rules inventory and gap analysis, performed before rule development begins, that: identifies all existing business rules in scope, regardless of where they currently reside (application code, rules engines, configuration, spreadsheets, documents, or undocumented operational practice); traces each existing rule to its authoritative legal or policy source, explicitly flagging rules whose source cannot be identified for OHIP disposition; identifies conflicts, duplicates, and obsolete rules across systems; maps each current-state rule to its target state in the OHIP-approved repository (migrate, re-author, consolidate, or retire); and assesses the impact of the migration on currently operating functionality, interfaces, and downstream consumers of rule outcomes.

4.5.2 Business Rules Documentation

The Contractor **MUST** produce and maintain a consolidated Business Rules Documentation deliverable that combines the Business Rules Traceability Matrix, and Business Rules Specification into a single controlled artifact. The complete, current inventory of rules and rule artifacts maintained in the State-managed repository constitutes the Business Rules Catalog (BRC) and **MUST** be exportable and reviewable by OHIP on demand; requirements artifacts such as use cases and user stories remain governed by the Requirements Management Standard (Section 3.0), and rules **MUST** trace to them through the rule-to-source mapping required by this section. This deliverable **MUST** be accepted by OHIP before rule creation begins and updated throughout the rule life cycle.

The Business Rules Documentation **MUST** include, at minimum:

- A catalog of all business rules with unique identifiers, descriptions, and current status (draft, approved, active, deprecated, archived).
- Traceability mapping linking each rule to its authoritative source (statute, regulation, policy document) with specific section, subsection, and paragraph references.
- Rule specifications explaining the rule principles used, rule structure, versioning approach, constraints, decision logic, and any standards or conventions applied.
- Artifact locations, process flows, reconciliation of source material, test specifications, and templates used for rule development.
- A mapping of rule IDs to source citations, document hyperlinks, and change requests from the business.

The Business Rules Documentation **SHOULD** include, at minimum:

- Rule specifications **SHOULD** be expressed using structured decision models (e.g., decision tables or an equivalent decision-modeling notation) where the rule logic involves multiple conditions or outcomes, to support unambiguous review and decision-table test coverage.

4.6 Rule Transparency and Readability

4.6.1 Rule Transparency

In addition to the repository obligations established in the Objectives, business rules **MUST NOT** be proprietary and **MUST NOT** be implemented in a manner that prevents OHIP from reading, exporting, or independently operating its rules. Upon OHIP's request, the Contractor **MUST** demonstrate all rule activities, artifacts, and executions.

4.6.2 Human Readability and Maintainability

The Contractor **MUST**:

- Author rules in clear, business-friendly language that promotes understanding by non-technical stakeholders. Rules **MUST** be readable and comprehensible by OHIP policy staff, not only by developers.
- Obtain OHIP agreement on rule readability and understanding before implementation. Rules **SHOULD NOT** be implemented without OHIP's confirmation that the rule logic is correctly understood.
- Use consistent, descriptive identifiers for rules and decision tables.
- Generate both human-readable exports (e.g., Word, Excel, or PDF) and machine-readable formats (e.g., XML or JSON) for all rules.
- Identify the applicable Federal and/or State regulation, statute, or official policy that each rule implements, where applicable.

The Contractor **SHOULD**:

- Break complex logic into reusable sub-rules or functions.
- Embed explanatory notes for assumptions, business context, and known limitations.

4.6.3 Substantive and Procedural Rule Separation

The Contractor **MUST** maintain a clear distinction between substantive rules and procedural rules. Rules that make outcome determinations **MUST NOT** be written together with rules that determine procedural logic.

For example, rules that determine program eligibility (Medicaid, CHP, EP, or QHP) **MUST NOT** be mixed with rules that determine application screen flow or step ordering. The Contractor **MUST NOT** mix procedural logic into substantive logic rules, and **MUST NOT** mix user interface visibility logic into substantive logic rules.

4.6.4 Determinism and Policy Guardrails

Business rules **MUST** produce deterministic, explainable outcomes, the same inputs and effective date **MUST** always produce the same result. The Contractor **MUST** define default behavior for missing or ambiguous inputs.

4.7 Rule Traceability

Every business rule **MUST** be traceable to its authoritative source. The Contractor **MUST** maintain and ensure:

- End-to-end links from each policy rule to its originating legislation (State and federal statutes or agency policy documentation).
- Rule-to-statute mapping referencing the specific section, subsection, and paragraph in the legal text.
- Policy document references linking official policy manuals or guidance documents with version and publication date.
- A mapping of rule IDs to source citations, document hyperlinks, and change requests, incorporated in the Business Rules Documentation or directly within the business rules management system.
- Forward traceability from rules through test cases and test results to implementation evidence, supporting CMS certification and audit readiness. The Contractor **MUST** link all test cases and results to rule IDs in the Business Rules Documentation and include objective evidence (screenshots, logs, or reports).



Cross-Reference: For requirements traceability at the solution level, refer to [Section 3.0](#).

4.8 Rule Life Cycle

The following requirements govern the full life cycle of policy and business rules from creation through deployment, versioning, change management, and archival.

4.8.1 Independent Rule Deployment

The Contractor **MUST** ensure that rule deployment is not tied to application deployment. Rules **MUST** be deployable independently of any application that invokes them. The Contractor **MUST**:

- Implement robust rule versioning that allows applications to invoke the correct ruleset validated for use with that application version.
- Enable rules that do not impact the application interface to be changed and deployed without requiring application redeployment.
- Enable rules that change on a regular basis (e.g., annual poverty level updates, rate table changes) to be updated in the rules management system without modifications to calling applications.
- Ensure rules are independently testable and deployable, decoupled from application release cycles, so that a rule change can be validated and promoted without requiring an application-tier release.

4.8.2 Rule Decoupling and Reuse

The Contractor **MUST** ensure that all policy and business rules are application-agnostic. Rules **MUST NOT** be part of the code base of any specific application. The Contractor **MUST**:

- Expose all rules through a standardized API to allow multiple applications to consume the same rule logic.
- Not recreate the same policy and business rules for multiple applications. Where the same rule applies across programs or modules, a single rule definition **MUST** be consumed by all applications through the standard API.
- Decouple rules so that changes to one application's user interface, workflow, or data model do not require changes to the business rules that serve multiple consumers.
- Remediate, at no additional cost to the Department, any rules not authored in the prescribed application-agnostic, streamlined manner.

4.8.3 Rule Change Management

The Contractor **MUST**:

- Submit all proposed changes to production business rules through a formal change request process approved by OHIP.
- Document the regulatory or policy basis for each rule change, the affected rules, the impact on downstream applications and reports, and the proposed effective date.
- Follow the designated OHIP approval process for releasing any rule change to production.
- Support two categories of rule changes: scheduled changes (e.g., annual regulatory updates deployed on a regularized cadence) and emergency changes (e.g., court orders or legislative amendments requiring immediate deployment).
- For emergency changes, the Contractor **MUST** follow the Department's established emergency change management process.
- Update the Business Rules Documentation, traceability records, and test cases to reflect all approved changes.
- Implement Department-initiated rule changes through the standard OHIP change and approval process.

 **Cross-Reference:** Rule changes that affect system scope, schedule, or budget are also subject to the change control processes defined in the OHIP Program and Project Management Domain Document.

4.8.4 Rule Versioning

The Contractor **MUST**:

- Preserve all prior rule versions in the repository to support audit, point-in-time analysis, and gap analysis for future rule change impact.
- Ensure that the current production rule version and all historical versions are always accessible for review and testing to OHIP.

The Contractor **SHOULD**:

- Maintain version control for all business rules, including version number, effective date, expiration date (if applicable), and the regulatory basis for each version.

4.8.5 Rule Effective Dating

The Contractor **MUST** support parallel operation of multiple rule versions when policy requires grandfathering, phase-in periods, or staggered effective dates. The system **MUST** enable time-travel queries that reproduce past determinations using the rule version and effective date that was in effect at the time of the original decision.

4.8.6 Rule Monitoring and Observability

The Contractor **MUST** instrument rule executions with metrics (volume, latency, error rates) and structured logs that include the rule version and effective date used for each execution. The Contractor **MUST** implement alerts for anomalous patterns (e.g., a sudden spike in denial rates or processing errors after a rule change) and provide dashboards and reports for OHIP oversight, including post-deployment impact reviews.

Immutable, time-stamped audit logs of all rule changes and deployments **MUST** be maintained, including approver identity and release artifacts.

4.8.7 Rule Archival

The Contractor **SHOULD** ensure that:

- Rules in production are only archived after OHIP approval and only when the data created or determined by the rules has also been archived.
- Rules implement change-date logic and remain in the rules management system as long as data derived from the rules remains in use. This allows for reference and gap analysis when evaluating future rule change impact on existing data.

5.0 Code Standard

5.1 Introduction

The Code Standard governs how all custom software developed under OHIP MES contracts is written, reviewed, secured, tested, documented, and maintained. Code is the foundation on which every other capability in the Medicaid Enterprise rests, and the quality, security, and ownership of that code directly determines OHIP's ability to operate independently of any single Contractor, respond to emerging threats, sustain performance under production load, and transition operations when contracts change.

This standard asserts two foundational principles that govern every requirement within it. First, NYS owns all custom code developed with State funds, unconditionally and from the moment of creation, not at contract end, not after transition. All source code, configurations, scripts, and related artifacts **MUST** reside in a State-owned repository to which OHIP has full access at all times. Second, code quality is a continuous discipline, not a pre-release checkpoint. Security, maintainability, test coverage, dependency currency, and code structure **MUST** be enforced through automated tooling integrated into the development workflow, not through manual review after the fact.

In the OHIP multi-vendor MES environment, code quality standards also serve an integration function. When multiple Contractors contribute code to interdependent systems, inconsistent coding practices, inadequate test coverage, or poor documentation creates technical debt that propagates across modules and surfaces as integration failures, security vulnerabilities, and operational instability. The requirements in this standard establish a shared baseline that protects the integrity of the entire ecosystem, not just individual components.

Deliverables and artifacts associated with this standard can be found in [Appendix A](#).

5.2 Objectives

The primary objectives of this standard are to:

- Assert unconditional State ownership and control over all code, configurations, and software artifacts developed for OHIP, ensuring the State's rights are established from project initiation and protected throughout the contract life cycle.
- Standardize development on a curated set of approved, modern programming languages and tools, with active community support, strong security track records, and alignment with Medicaid industry practice, to reduce technical debt and support long-term maintainability.
- Enforce a secure development life cycle through mandatory secure coding standards, comprehensive unit test coverage thresholds, automated static analysis, peer review, and zero-tolerance policies for unresolved high-severity security findings before production promotion.
- Promote the creation of modular, layered, well-documented solutions that expose open APIs, support reuse across OHIP modules, and reduce the cost and risk of future integration, enhancement, and Contractor transition.
- Establish continuous code quality as an operational requirement through automated code maturity tooling, dependency scanning, runtime monitoring, and measurable quality targets aligned with ISO/IEC 5055:2021.
- Ensure complete auditability of code changes through mandatory version control traceability, cryptographic commit signing, and linkage of every change to an approved work item or requirement.

5.3 Ownership Rights and Repository Control

5.3.1 Ownership Rights

All software development **MUST** comply with 45 CFR 95.617, ensuring the State obtains unrestricted rights and software enters the public domain as required. For commercial solutions, FFP funds may be used for licensing and services, with the State retaining unrestricted rights to all data, configurations, and any customizations developed with federal funds.

The Contractor **MUST** ensure that OHIP retains:

- Project-developed code and code ownership rights from project initiation.
- All ownership rights to every stage of developed software (including source code, modifications, and documentation) developed with State funds.

5.3.2 Repository Control

The Contractor **MUST**:

- Store all source code, scripts (e.g., Infrastructure-as-Code, shell scripts, and build scripts), and low-code artifacts (e.g., COTS/SaaS configurations) in a State-owned version control repository at all times.
- Restrict repository access to authorized personnel according to the principle of least privilege.

5.4 Code Leadership and Accountability

OHIP will designate a Lead Maintainer and Backup Maintainer for all codebases or modules for every project. The Contractor **MUST** support this structure by ensuring that:

- The Lead Maintainer oversees code quality and function for the assigned codebase.
- All changes (e.g., merge/pull requests) are reviewed and approved by the Lead Maintainer or Backup Maintainer before merging.
- If the Lead Maintainer becomes unavailable, the Backup Maintainer fills the role without disruption to the review process.
- The Lead Maintainer **MUST** use the Department-designated code-quality and security (SAST) scanning tools and retain the resulting reports as audit evidence for CMS certification.

5.5 Secure Coding Standards

Before code development begins, the Contractor **MUST** adopt the secure coding standard prescribed by the Department (e.g., OWASP Secure Coding Practices,¹³ NIST 800-218,¹⁴ SEI CERT Coding Standards¹⁵) and obtain OHIP approval. The Contractor **MUST**:

- Apply the approved coding standard consistently. Compliance **MUST** be enforced through automated Static Application Security Testing (SAST) tooling integrated into CI/CD pipelines.
- Ensure all code adheres to best practices for validation, authentication, error handling, and other secure coding requirements in the adopted standard.
- Repositories **MUST** implement automated secret detection to prevent committed credentials, tokens, certificates, or cryptographic material from entering the codebase.
- Applications **MUST** satisfy security verification requirements aligned with an OHIP-approved application security verification standard (e.g., OWASP ASVS). The applicable verification level **MUST** be agreed with OHIP before development begins.

5.6 Approved Languages and Tools

5.6.1 Language Approval Criteria

The Contractor **MUST** obtain OHIP approval for the selection of all programming, scripting, and markup languages before development begins. To be approved, languages **MUST**:

- Be selected from the OHIP Approved Technology Stack. The Contractor **MUST** submit a SDR for any language, framework, or runtime not currently on the approved list.
- Remain within vendor-supported life cycle periods for all production deployments. Languages or runtimes approaching end-of-life **MUST** be identified and a migration plan submitted to OHIP.
- Have available linters, be easy to maintain, and publish best practices.

Approved languages **SHOULD**:

- Have active community releases within the last twelve (12) months and a roadmap that extends a minimum of twenty-four (24) months from project initiation.

5.6.2 Preferred Programming Languages

The following is OHIP's preferred set of programming languages:

- **Python**, Data Engineering, Data Science and AI, Analytics, Application Development, and Back-end Development.
- **JavaScript or TypeScript**, Front-end Development and Application Development.

¹³ Refer to [OWASP Secure Coding Practices](#).

¹⁴ Refer to [NIST 800-218 Secure Software Development Framework](#).

¹⁵ For more information, refer to the [SEI CERT Coding Standards](#).

- **Go, Rust, Java (not Java EE Pre-Spring Boot or EJB 2.x), C#,** Back-end Development, Big Data/Data Engineering, and Enterprise Software Development.

5.6.3 Utility Languages

In addition to programming languages, the following utility languages are commonly used within the OHIP MES ecosystem:

- Git, SQL/PL-SQL, YAML/TOML, Shell Scripting, JSON, XML, HTML/XHTML, CSS, Markdown, HCL.
- Based on the project or directive, other languages not listed above **MAY** be considered for OHIP approval with documented justification

5.7 Dependency Management

The Contractor **MUST**:

- Establish a Patch Management Plan and process to upgrade underlying platforms and dependencies proactively, rather than waiting for major releases.
- Obtain OHIP approval for the Patch Management Plan for all third-party libraries, frameworks, and tools prior to code development.
- Deploy all third-party libraries, frameworks, and tools with the latest supported version and keep them up to date.
- The Contractor **MUST** generate and maintain a machine-readable Software Bill of Materials (SBOM) in SPDX or CycloneDX format for all production releases. SBOMs **MUST** be updated for every release and provided to OHIP upon request.
- The Contractor **MUST** perform automated dependency vulnerability scanning as part of the CI/CD pipeline. Builds with known Critical or High vulnerabilities in dependencies **MUST NOT** proceed to production without documented risk acceptance from OHIP.
- The Contractor **MUST** remediate dependency vulnerabilities within the following timelines: Critical, 7 calendar days; High, 30 calendar days; Medium, 90 calendar days; or as otherwise defined by OHIP. Exceptions **MUST** be documented with a mitigation plan and OHIP approval.

5.8 Unit Testing

The Contractor **MUST**:

- Achieve unit test coverage of 100 percent for authentication and authorization modules, at least 80 percent of statements, and at least 75 percent of branches across the entire codebase. All builds **MUST** pass existing tests before merging.
- Achieve unit test coverage of 100 percent for authentication and authorization modules, at least 80 percent of statements, and at least 75 percent of branches across the entire codebase. All builds **MUST** fail if any test fails or coverage drops below these thresholds.
- Coverage thresholds **MUST** be automatically enforced through CI/CD pipeline quality gates. Coverage reports **MUST** be generated with every build and accessible to OHIP.

5.9 Code Architecture and Design

5.9.1 Secure by Design

The Contractor **MUST** develop and obtain OHIP approval for a code security design prior to solution development. The design **MUST** demonstrate how confidentiality, integrity, and availability will be maintained in all API modules and **MUST** be developed with Zero Trust Principles.

5.9.2 Modular, Layered Design

The Contractor **MUST** design all solutions with modular architecture separating the presentation layer, business logic, and data layer. Services **MUST** be loosely coupled and follow modular architecture principles with explicit separation of concerns (e.g., microservices, SOA, modular monolith, or domain-oriented services) to prevent tight coupling and support independent deployment, testing, and maintenance.

5.9.3 Open Interfaces

Open APIs **MUST**:

- Be exposed and well-defined for all business modules and integrations.
- Be documented to enable interoperability and data exchange in a standardized manner.
- Be utilized by the presentation layer and be enabled for access by third-party tools.
- Be properly secured per the OHIP Security, Privacy, and Compliance Domain Document, Security and Privacy Compliance Standard and the OHIP Infrastructure and Operations Domain Document, Application Integration Standard.
- APIs **MUST** follow a versioning strategy (e.g., semantic versioning or URL-based versioning) with a documented backward compatibility and deprecation policy approved by OHIP.
- API contracts **MUST** be defined using a machine-readable specification format (e.g., OpenAPI/Swagger) and maintained in the State-owned repository alongside the code they describe.

5.9.4 Reuse and Interoperability

- The Contractor **MUST** evaluate existing OHIP Medicaid technologies, shared services, and enterprise components before developing new capabilities. Where existing enterprise services are available (e.g., IAM, integration platform, monitoring), the Contractor **MUST** consume them rather than build alternatives.
- The Contractor **SHOULD** design solutions to promote sharing and reuse across programs and other states.

5.9.5 Code Documentation

Code **MUST** be self-documenting. All public functions, classes/modules, and packages **MUST** include structured documentation comments (e.g., Python docstrings, Javadoc, XML comments, JSDoc, or Go comments) sufficient to generate reference documentation via standard tooling.

Repositories **MUST** have and maintain markdown files (e.g., README.md) that explain the code, including how to build the code and how to test the code.

Repositories **MUST** maintain architecture and integration documentation sufficient to support independent operation, maintenance, and transition by a successor Contractor. This includes, at minimum: system context diagrams, dependency maps, deployment architecture, and integration flows.

5.10 Code Change Control

5.10.1 Version Control and Traceability

The Contractor **MUST**:

- Manage all code changes in a version-controlled repository (e.g., Git).
- Use version control features to track every modification with accountability to an individual user account.
- Link each code change to a work item or requirement for full traceability.
- Production and release branches **MUST** enforce branch protection controls preventing direct commits, force pushes, and bypass of required reviews or pipeline checks.
- Release artifacts **MUST** be traceable to source commits and generated through controlled CI/CD pipelines. Manual builds for production deployment are prohibited.

The Contractor **SHOULD**:

- Designate a repository administrator for every project.

5.10.2 Peer Review Requirement

The Contractor **SHOULD NOT** merge code with the main branch(es) without at least one (1) qualified reviewer's approval. The designated code owner **SHOULD** review and approve changes others make to their component. The Contractor **SHOULD** configure repository controls (branch protection) to enforce this review and approval before any merge to protected branch(es).

5.10.3 Integrity of Changes

The Contractor **MUST** enable cryptographic signing of commits to verify authorship. Signed tags or checksums **MUST** be used for release artifacts to prevent tampering. All deployments **MUST** originate from the version-controlled source; direct changes to production environments are prohibited.

5.10.4 Developer Environments

The Contractor **MUST** develop code in an environment that is isolated from test and production environments.

Where a code change requires a corresponding database schema change, model change, or data fix, including changes to stored procedures, views, indexes, constraints, or schema definitions, the Contractor **MUST** also comply with the database change control requirements of the OHIP Data and Information Management Domain Document, Section 4.5.3 (Database Change Control) and Section 4.5.4 (Data Fix Governance). The code change and the associated database change **MUST** be coordinated and packaged together in the same Release to maintain environment consistency and traceability. Neither change **MUST** be promoted to production independently of the other without documented OHIP approval.

 **Cross-Reference:** Code change control requirements in this section govern the development-layer process. All code changes destined for production must also satisfy the release-level change control and approval requirements defined in [Section 7.6](#) (Change Control and OHIP Interaction) and [Section 7.7](#) (Adding and Removing Items Post-Release). A code change **MUST NOT** be promoted to production outside of an approved release package except where explicitly authorized as a HotFix or Expedited Release under [Section 7.3](#).

5.11 Code Quality

5.11.1 Comprehensive Logging

The Contractor **MUST** provide code that is structured with contextual logs for all errors and security-relevant events, which include, at minimum:

- Timestamp (ISO 8601 with UTC offset)
- Correlation/Trace ID
- Component or module name
- Severity level
- Staff or system identity (if applicable)
- Proactive error warnings on WARN, ERR, or CRIT
- Error codes/messages and relevant parameters

The Contractor **SHOULD** follow OWASP's Logging Cheat Sheet for event taxonomy and format guidance.

Logs **MUST NOT** contain unmasked PHI, Personally Identifiable Information (PII), credentials, secrets, tokens, or cryptographic material. All sensitive data **MUST** be masked or redacted prior to logging.

 **Cross-Reference:** Enterprise log management requirements including retention, sensitive data restrictions, and SIEM integration are governed by the OHIP Infrastructure and Operations Domain Document, Log Management Standard.

5.11.2 Log Management

The Contractor **MUST** ensure all logs are centralized. Log formats **MUST** be uniform and easily consumable by analytic tools. Logs **MUST** be retained for a period compliant with the OHIP Infrastructure and Operations Domain Document, Log Management Standard and other relevant policies or regulations to enable forensic investigations and to prevent unauthorized access or modification of log records.

5.11.3 Performance Observability

Performance of code and APIs **MUST** be measurable. The Contractor **MUST** implement end-to-end distributed tracing to capture request flows across services, using tools approved by OHIP.

5.11.4 Continuous Code Maturity

The Contractor **MUST**:

- Remove unused or obsolete code, libraries, features, and documentation from projects to reduce complexity and potential attack surface.
- Use the Department-designated automated tools to scan for code smells, complexity, and structural weaknesses.
- Set targets for key quality factors, Security, Reliability, Performance Efficiency, and Maintainability, as defined in ISO/IEC 5055:2021.
- Evaluate quality metrics continuously and act (refactoring, additional testing, or reviewing) to improve code quality to comply with defined goals.
- Provide OHIP with access to all code and code artifacts within a State-owned, central repository from project initiation.
- The Contractor **MUST** track, prioritize, and remediate identified technical debt items according to severity classifications and remediation timelines defined by OHIP.
- Technical debt items **MUST** be visible in the OHIP-approved project management tool.

6.0 Test Management Standard

6.1 Introduction

The Test Management Standard establishes the testing governance framework that applies to all software releases across the OHIP MES. Testing in an MES environment carries stakes that go beyond typical software quality assurance: a defect that escapes into production can affect eligibility determinations, disrupt claims processing, or compromise the PHI of millions of Medicaid beneficiaries. At the same time, OHIP's eligibility for enhanced FFP is contingent on demonstrating to CMS that its systems meet defined Medicaid Enterprise outcomes, a demonstration that depends entirely on the completeness and traceability of testing evidence.

This Standard operationalizes the CMS MES Testing Guidance Framework and, for SMC modules, the CMS Medicaid Streamlined Certification Guidance, translating federal expectations into specific, enforceable requirements for every phase of the testing life cycle. It establishes a mandatory baseline of thirty quality control types, from static analysis and unit testing through security scanning, performance testing, UAT, and synthetic production monitoring, and defines the planning, execution, transparency, defect management, and reporting standards that govern how those controls are applied, tracked, and evidenced.

A central principle of this Standard is that OHIP must retain full visibility into, and independent ability to conduct, quality assurance activities. The Contractor provides the testing infrastructure and executes the required controls, but OHIP must be able to independently verify test results, review artifacts on demand, and conduct its own UAT in an environment that is isolated from Contractor access. Testing is a shared accountability, not a delegated one.

6.2 Objectives

The primary objectives of this Standard are to:

- Define and enforce a comprehensive, mandatory set of quality control types and testing standards that apply to all software releases, grounded in the CMS MES Testing Guidance Framework and CMS SMC requirements.
- Establish minimum quality gates, including test coverage thresholds, defect resolution criteria, and required test artifact deliverables, that must be met before any release receives OHIP production deployment approval.
- Ensure OHIP retains full transparency and independent oversight of all Contractor testing activities, with the ability to review test artifacts on demand, conduct independent UAT in an isolated environment, and verify testing completeness before signoff.
- Create a robust defect management life cycle that captures, categorizes, triages, and resolves defects with defined severity classifications, resolution SLAs, and root cause analysis (RCA) requirements for critical findings.
- Establish continuous operational quality monitoring, including synthetic monitoring, SLA alerting, and post-go-live testing, to ensure production systems maintain performance and reliability between releases.
- Ensure all testing activities and metrics are documented and structured to satisfy CMS certification evidence requirements, enabling OHIP to demonstrate system quality and outcome achievement during SMC reviews.

6.3 Test Standards

6.3.1 Medicaid Enterprise System (MES) Testing Guidance Framework

All testing activities **MUST** adhere to the CMS MESs Testing Guidance Framework (hereafter, MES Testing Guidance Framework), the CMS-published guidance that defines testing expectations for MES modules and informs certification review.¹⁶ This framework establishes the required testing approach, documentation, and quality expectations for any MES component, independent of certification status.

Modules that pursue SMC **MUST** also meet the requirements outlined in the CMS Medicaid Streamlined Certification Guidance.¹⁷ These certification expectations build upon, and do not replace, the foundational testing requirements defined in the CMS Medicaid Testing Guidance Framework.

Testing scope, rigor, and control selection **MUST** be proportionate to business, security, operational, and integration risk. The Contractor **MUST** document the risk-based rationale for test depth and coverage in the Test Management Plan.

6.4 Test Planning

The Contractor **MUST** provide OHIP with a Test Management Plan Deliverable to comply with the quality control requirements of this Standard and CMS certification requirements. The Test Management Plan Deliverable **MUST** be approved by OHIP.¹⁸

The Contractor **MUST** provide complete, traceable, and reproducible testing evidence demonstrating comprehensive and passing tests for all quality control types required by this standard. The Test Management Plan is OHIP's (and CMS's) initial quality check to ensure that the Contractor has planned appropriate quality control activities—including the regression testing scope, automation approach, and execution cadence required by Table 5—prior to UAT and production. OHIP **MUST** approve the Contractor's Test Management Plan as part of the contract execution. Any subsequent changes to this plan **MUST** be approved by OHIP before any testing begins.

The Contractor **MUST** maintain bidirectional traceability between requirements, test cases, test results, defects, and release artifacts. Traceability **MUST** be demonstrable on demand for CMS certification review.

6.5 Test Execution

Test execution validates that the system functions as intended and meets internal and external stakeholder needs. This phase validates system functionalities and identifies defects before deployment. Each of these components plays a vital role in the testing process, contributing to the design of a robust and reliable system. Throughout Test Execution, the Contractor **MUST** develop and maintain comprehensive test artifacts including, but not limited to:

- **Test cases:** Test cases include the detailed, step-by-step instructions that outline how to test specific features or functionalities of the system. Each test case includes input data, execution conditions, and expected results. Test cases ensure that the system behaves as intended under

¹⁶ For more information, refer to the CMS Medicaid Enterprise System [Testing Guidance Framework](#).

¹⁷ Refer to [Streamlined Modular Certification Guidance for Medicaid Enterprise Systems](#).

¹⁸ Per Expectation 2 of Test Planning of the MES Testing Guidance Framework.

specified conditions and are derived from system requirements to confirm that each requirement is met.

- **Test scenarios:** Test scenarios provide a high-level description of what needs to be tested within the system, focusing on workflows. They offer a broader perspective than test cases and are designed to validate end-to-end processes or interactions within the system, ensuring that various components function together seamlessly.
- **Test scripts:** Test scripts are automated sequences of instructions that execute the test cases. These scripts are often developed using testing frameworks or tools, enabling efficient and repeatable testing processes.
- **Test results:** Test results document the outcomes of executing test cases, highlighting whether a test has passed or failed based on the expected results. They provide vital insights into the system's performance and identify areas that require corrective actions or further analysis.
- **Test reports:** Test reports compile and summarize the outcomes of testing activities, presenting detailed information on the status, progress, and effectiveness of tests performed. They provide stakeholders with a comprehensive view of the testing process, including metrics on test coverage, defect density, and resolution rates, enabling informed decision-making regarding system deployment. The Contractor **MUST** submit a Test Results Report.

6.5.1 Test Management Tools

The Contractor **MUST** use OHIP approved test management tools for:

- **Test Planning and Scheduling:** Plan and schedule test cycles, assign testing resources and responsibilities, and align test milestones with the Integrated Master Schedule and the OHIP Enterprise Release Calendar. The tools **MUST** provide visibility into planned versus actual test progress to support early identification of testing schedule risk.
- **Test Case Management:** Create, organize, and maintain test cases that are well-structured, documented, and easily accessible. The tools **MUST** provide traceability to test case requirements and defects.
- **Test Execution Tracking:** Track the progress of test executions by test step, providing insights into which tests have passed, failed, or been skipped.
- **Defect Management:** Report, trace, and resolve defects to identify and fix issues early in the development process.
- **Reporting and Analytics:** Provide detailed reports and dashboards, offering valuable insights into the status of testing efforts, trends, and areas for improvement.
- **Collaboration and Communication:** Facilitate communication and collaboration among team members, for transparency regarding testing activities.
- **Pipeline Integration and Scanning:** Tools **MUST** support integration with the CI/CD pipeline to automate the execution of Software Composition Analysis (SCA), Infrastructure as Code (IaC) security scanning, and API contract testing as mandatory quality gates on every build, with results visible in the test management dashboard.
- Test management platforms **MUST** maintain immutable audit history for test execution, defect modification, status changes, approvals, and evidence updates.

- Testing tools **MUST** implement role-based access control aligned with least privilege principles and segregation of duties between test execution, defect management, and test approval.

The Contractor **SHOULD** define and maintain, within the Test Management Plan, a test environment strategy covering every test level required by this standard, including unit, system integration, regression, performance, security, accessibility, and UAT. The strategy **SHOULD** identify which environments support which test levels, how configuration parity with production is maintained for environments used in pre-deployment verification, how environments are provisioned, refreshed, and controlled, and how test data is managed in each environment in accordance with Section 6.5.3 and OHIP security and privacy standards. Dedicated environment requirements for UAT are defined below.

6.5.2 User Acceptance Testing (UAT) Environment

The Contractor **MUST** provide one (1) or more UAT environments for OHIP's exclusive use. Each dedicated environment must be capable of testing all functionality planned for production deployment. To avoid complications that arise from shared environments with Contractors and development teams, OHIP requires at least one (1) UAT environment for dedicated OHIP user testing. In certain instances, multiple UAT environments may be needed. For example, if multiple projects or code streams are being run at the same time or when different configurations need to be tested and evaluated in parallel. With the increased adoption of Agile and Hybrid methodologies, this can allow more flexibility testing in environments dedicated to multi-phase implementations. Production PHI or PII **MUST NOT** be used in UAT or test environments unless explicitly authorized by OHIP and protected in accordance with OHIP security and privacy standards. The Contractor **MUST** use de-identified, masked, or synthetic test data.

6.5.3 Test Data Management

The Contractor **MUST** establish and maintain a test data management approach that addresses: provisioning of test data sets, use of synthetic or masked data for all non-production environments, refresh cadence, retention policies, and traceability of test data sets to the test cases they support.

Test data sets used for verification **MUST** be documented and reproducible to support re-testing and CMS audit review.

6.5.4 Test Transparency

At OHIP's request, test activities and artifacts **MUST** be provided or demonstrated prior to providing signoff for production releases. To maintain trust in the quality process, OHIP **MUST** be allowed to review the test process and procedures.

6.5.5 Mandatory Quality Controls

The Contractor **SHOULD** maximize the use of automation and automated reporting to reduce cost and SDLC time. Where manual testing is required, the Contractor **SHOULD** document the justification and obtain OHIP approval.¹⁹ Projects may require more quality control checks than mandated in Table 5.²⁰ At a minimum, the Contractor must perform the quality controls and tests in Table 5. The application of

¹⁹ It is highly recommended to perform as many automated tests as possible inside code repositories to catch defects early and to reduce the need for additional environments.

²⁰ For example, separating development integration testing versus system integration testing, specific production feature testing, or special Blue/Green testing.

mandatory controls can be tailored for specific release types (as defined in [Section 7.0](#)), subject to a documented, risk-based assessment and formal OHIP approval.

Table 5 – Mandatory Quality Controls

Control	Descriptor
Coding Standards and Linters	Leverage automated tools to enforce adherence to coding standards, which are critical for ensuring code consistency, maintainability, and quality across the development team. These tools check for style conformity, complexity constraints, and adherence to industry best practices. Examples include ESLint for JavaScript, Pylint for Python, and SonarLint for multi-language support. By integrating these tools into the development process, teams can quickly detect and rectify deviations from established coding norms, thus minimizing defects and streamlining code reviews.
Peer (Code) Reviews	Structured peer (code) review processes involve team members evaluating each other's code changes through diffs and pull requests. This collaborative effort helps ensure that code is not only functionally correct but also readable and in compliance with defined standards, catching errors that automated tools might not detect to enhance the overall quality and security of the software.
Static Code Analysis	Implement comprehensive static code analysis as an integral part of the development workflow. This involves using specialized tools to automatically examine source code for common issues such as code smells, security vulnerabilities, and redundant or dead code paths. Tools provide actionable insights and detailed reports, which help developers proactively remedy potential weaknesses, ensure security compliance, and enhance the maintainability of the codebase.
Sanity (Smoke) Testing**	Perform focused sanity (smoke) testing on system components or bug fixes to verify that the most critical functionalities operate as expected after deployment or a significant change. These preliminary tests act as a quick assessment to catch any fundamental issues before more detailed testing phases. While sanity testing can be manual, automating these checks using tools enhances efficiency and provides fast feedback on code changes, facilitating continuous integration and deployment practices.
Unit Testing*	Develop comprehensive unit tests that serve to validate the functionality of individual components of the codebase, such as functions or classes. Using testing frameworks, such as JUnit for Java or pytest for Python, these tests are automated to ensure that each unit behaves as expected independent of other parts of the application.
System Integration Testing (SIT)*	Execute system integration tests to assess the interoperability and interaction between integrated modules and with external systems. These automated tests, conducted using frameworks, such as TestNG, pytest, and Cypress, are crucial for ensuring seamless communication and data exchange across system components. By simulating real-world interactions, integration tests verify that different modules work together as intended, thereby reducing integration risks and enhancing system reliability and performance.

Control	Descriptor
System/End-to-End Testing*	Implement system/end-to-end testing to evaluate the entire application in a production-like environment. These tests assess compliance with functional and NFRs by simulating user scenarios that span across multiple system components. Tools such as Selenium and Cypress are employed to automate these tests, providing comprehensive coverage and ensuring that the system functions correctly in real-world conditions. This testing phase is essential for verifying that all integrated components operate harmoniously and adhere to specified requirements before final deployment.
Solutions Exploration Testing	Undertake solutions exploration testing during the early stages of feature development to assess initial system prototypes. This approach allows teams to iterate on design and functionality based on feedback, facilitating the discovery of innovative solutions and minimizing the risk of developing features that do not align with user needs or business objectives.
Regression Testing*	Conduct detailed regression testing to ensure that new code changes do not introduce regressions in existing functionalities. Automated regression test suites are run after each code change to verify that the software continues to comply with its functional requirements and that previous defect fixes remain intact. Regularly updating and executing these tests is paramount to maintaining system stability and quality as development progresses.
Interface Testing*	Perform interface testing to verify the integrity and accuracy of data exchanges across system interfaces and APIs. Automated tools such as Rest Assured and Karate facilitate these tests by ensuring that data flows are consistent, secure, and compliant with expected standards. Interface testing is crucial for validating seamless integration with third-party services and internal modules.
Data Conversion and Migration Testing*	Execute data conversion and migration testing to verify the accurate transformation and migration of data from legacy systems to new platforms. Automated scripts using tools, such as QuerySurge and Talend, are employed to ensure all data fields are accurately transferred and that business rules are consistently applied during the conversion process. Resulting exception and error reports provide insights into any anomalies, ensuring data integrity and successful migration.
Performance, Stress, and Load Testing*	These tests will determine benchmarks, upper performance limits, and maximum operating capacity. Performance Testing evaluates how the solution will perform in terms of responsiveness and stability under a particular workload. Stress Testing evaluates the extent to which a solution continues to perform when subjected to extreme workloads, or when a component has been compromised; and Load Testing simulates normal conditions and user load to determine how the solution will behave.
Security Testing ²¹	Conduct comprehensive security reviews throughout any solution implementation. This includes testing key safeguards, verifying the audit processes are in place and conducting “ethical hacking.” Tests MUST also be performed to demonstrate that the solution complies with applicable security requirements.

²¹ For more information, refer to the OHIP Security, Privacy, and Compliance Domain Document, Security and Privacy Compliance Standard.

Control	Descriptor
Security and Vulnerability Testing ^{22*}	In addition to Security testing, the Contractor MUST implement ongoing security and vulnerability testing through a combination of automated SAST and Dynamic Application Security Testing (DAST), penetration testing, and vulnerability scans using tools, such as SonarQube, Burp Suite, and Metasploit Framework. These proactive measures identify potential security threats and vulnerabilities across the application, securing data and ensuring compliance with security best practices and standards.
Accessibility Testing ²³	Perform rigorous accessibility testing to verify that the system complies with legal standards (e.g., WCAG, Section 508) and is inclusive to all users, including those with disabilities. Automated tools such as WAVE, Siteimprove, and VoiceOver are used to identify accessibility issues and improve the user experience by ensuring all functionalities are perceivable, operable, understandable, and robust for diverse users.
Usability Testing*	Usability testing involves observing real users interacting with the user interfaces (UI) to identify user experience (UX) issues and areas for improvement. It provides valuable insights into user behavior, preferences, and potential challenges, allowing developers to refine the design and functionality. By collecting feedback and pinpointing areas of friction, usability testing aims to enhance the overall experience and ensure the software meets user expectations effectively. This iterative process helps ensure that the final product is intuitive, efficient, and satisfying for its intended audience.
UAT*	OHIP validates the functionality in UAT of the solution, which enables OHIP to accept or reject changes.
Parallel Testing*	Used when transitioning from a legacy system to a new system and it aims to find out whether the legacy system and the new system are behaving the same or differently and to ensure that the two (2) systems produce consistent results
Post-Go-Live Testing	Perform throughout ongoing operations and support, such as when any defects involving code designed, developed, or implemented by the Contractor. Post-Go-Live testing is to be performed on any Releases, upgrades, or enhancements applied to OHIP environments.
Business Continuity and Disaster Recovery (BC/DR) ²⁴	Regularly validate business continuity and disaster recovery plans to ensure backup, failover, and recovery procedures are effective and aligned with organizational goals. Automated tools are used to simulate real-world scenarios, confirm data integrity, and test readiness for potential disruptions.
ORR	Prepare for ORR with CMS to assess system configuration, processes, and preparedness based on submitted artifacts. These evaluations are comprehensive, ensuring the system is fully operational, compliant, and ready for production environments.

²² Required with continuous (minimum monthly) testing by [ARC-AMPE](#).

²³ For more information, refer to Section 508 [Test for Accessibility](#) and Department of Justice [Final Rule](#).

²⁴ Required for [CMS Enhanced Funding](#).

Control	Descriptor
Certification Review ²⁵	Engage in a rigorous certification review process led by CMS to demonstrate that the system complies with all required metrics and standards before official deployment. This final assessment validates that the solution delivers on its promises and fulfills all contractual obligations.
Synthetic Monitoring	Utilize synthetic monitoring for continuous, automated production checks that monitor system availability and performance using pre-defined metrics and health endpoints. Tools assist in identifying potential issues before they impact users.
Synthetic Alerting	Implement robust synthetic alerting mechanisms that provide real-time notifications when production checks fail. These alerts, coupled with predefined escalation processes, enable timely intervention, minimize downtime, and maintain system reliability by efficiently routing issues to the appropriate support, operations, and security teams.
Service Level Agreement (SLA) Monitoring and Reporting	Automate the monitoring and reporting of SLAs by tracking key indicators such as uptime, response time, and throughput. Regular reporting ensures compliance and provides transparency into performance metrics, supporting accountability and data-driven decision-making.
SLA Alerting and Escalation	Establish real-time SLA alerting solutions to promptly notify teams when thresholds are breached. This setup includes defined escalation workflows to expedite issue resolution, maintain service standards, and ensure customer satisfaction through proactive management of service-level expectations.
API Contract Testing	Automated tests that verify two (2) separate systems agree on the format and content of their API communication. The Contractor MUST implement automated API contract tests for all inter-module and inter-vendor integrations to detect breaking changes before they reach SIT, using OHIP-approved contract testing tooling integrated into the CI/CD pipeline.
SCA / Dependency Scanning	Automated scanning of all open-source libraries and third-party dependencies for known CVEs and licensing violations. The Contractor MUST run SCA tools on every build, enforce zero tolerance for critical CVEs in production-bound artifacts, and remediate high CVEs within the timeframes defined in the OHIP Security, Privacy, and Compliance Domain Document, Security and Privacy Compliance Standard.
Infrastructure as Code (IaC) Security Scanning	Automated security scanning of all infrastructure templates (e.g., Terraform, CloudFormation, Ansible) before provisioning. The Contractor MUST scan all IaC artifacts for security misconfigurations (e.g., overly permissive access roles, publicly accessible storage) as a mandatory gate in the CI/CD pipeline before infrastructure is provisioned.

* Per Expectation 3 of Test Execution of the MES Testing Guidance Framework.

** Per Expectation 1 of Operational Monitoring of the MES Testing Guidance Framework.

The Contractor **SHOULD**:

- Implement structured peer code review processes before merging code changes.
- Implement comprehensive static code analysis as part of the development workflow.

²⁵ For more information, refer to [CMS Guidance](#) for Streamlined Modular Certification.

- Implement synthetic alerting mechanisms providing real-time notifications when anomalies are detected.
- Automate SLA monitoring and reporting for key indicators including uptime and response times.
- Establish real-time SLA alerting and defined escalation paths for threshold breaches.
- Utilize synthetic monitoring for continuous automated production checks.

6.5.6 Defect Rate

The Contractor **MUST**:

- Track and maintain the number of issues and defects by type and per Release that are discovered during any test type.
- Provide a RCA after each Release on major or critical defects.
- Track and report escaped defects, defects identified in production that were not detected during pre-production testing. Escaped defect rates **MUST** be reported per release and analyzed for root cause.
- The Contractor **MUST** provide recommendations on quality control activities that may be added or enhanced to prevent similar defects in future Releases, especially following UAT or production-escaped defects.

6.5.7 Defect Management Plan

The Contractor **MUST** submit a Defect Management Plan to be approved and updated to reflect the type of testing being executed. The Defect Management Plan provides the approach, activities, and roles for handling defects. This plan also includes activities associated with the defect management strategy, reporting, conducting the triage, and performing actions to resolve defects. The plan's primary goal is to clearly identify the necessary defect management activities, durations, and stakeholder expectations and responsibilities. The Defect Management Plan **MUST** also define how the defect rate, RCA, and escaped defect obligations of Section 6.5.6 are operationalized, including the tracking method, reporting cadence, and responsible roles. This Contractor's Defect Management Plan **MUST** include, at minimum, processes to:

- Identify and document defects, their descriptions, and the number of defects/bugs in the Backlog by non-production or production environment (Defect Log).
- Assess the impact severity level and priority of defects on each business process.
- Implement practical activities to reduce the occurrence and impact of defects in business operations.

The Contractor **SHOULD**:

- RCA defects using a standardized taxonomy that includes, at minimum: requirements deficiency, design deficiency, coding error, configuration error, integration failure, infrastructure issue, data quality issue, or operational/process failure.
- Manage the details and resolution status with dates and times of defects.
- Coordinate response teams for effective defect resolution.
- Align with other planning efforts.

The defect severity classification (Severity 1–4) and associated incident resolution timeframes are defined in [Section 7.14.1](#). The business priority assessment framework, including urgency levels and priority matrices, is defined in [Section 7.14.2](#). The Contractor's Defect Management Plan must incorporate and reference these classifications.

6.5.8 Defect Report

A Defect Report captures information about a specific defect or issue within a solution and serves as a communication tool between testers, developers, and other stakeholders. The following list includes high-level subject areas that **SHOULD** be covered in the Contractor's Defect Report:

- **Unique Identifier:** A unique number or code that distinguishes this defect from others in the tracking system, which allows for easy reference and tracking.
- **Title or Summary:** A brief, descriptive name that summarizes the defect in a few words, which helps quickly identify the nature of the defect.
- **Description:** A detailed account of the defect, explaining what was expected versus what occurred to provide context and help developers understand the issue.
- **Steps to Reproduce:** A clearly defined, sequential list of actions taken that led to the discovery of the defect to help developers replicate the issue reliably.
- **Environment Details:** Information about the test environment where the defect was found, including hardware, software versions, and network conditions to ensure the defect can be reproduced in a comparable setup.
- **Severity and Priority:** Indication of the defect's impact on the system and how soon it is expected to be addressed. Severity classifications (Severity 1–4) are defined in [Section 7.14.1](#) Defect Severity Assignment; business priority levels are defined in [Section 7.14.2](#). Severity reflects the technical impact; priority reflects the business urgency.
- **Attachments or Screenshots:** Additional files, such as logs, screenshots, or videos, that provide visual evidence or detailed error messages that are not easily described textually.
- **Expected versus Actual Results:** A comparison highlighting the discrepancy between the intended system behavior and the observed outcome, providing clarity on the defect.
- **Status:** The status of the defect, such as "New," "In Progress," "Resolved," or "Closed," indicating its life cycle stage.
- **Assigned To:** The name or role of the person or team responsible for addressing the defect, ensuring accountability
- **Date Identified:** The date the defect was discovered, which helps in tracking and prioritizing issue resolution over time.
- **Date Resolved:** The date the resolution to the defect was successfully moved into production. For defects resolved in the DDI phase, this is the date the resolution was successfully validated in a lower environment.
- **Details/Notes:** Details of how the defect was resolved, and any other important information not included elsewhere in the report.

A defect report enables efficient and effective defect resolution, enhancing the overall quality and reliability of the software product. This report not only facilitates communication between team members

but also contributes to improving processes by identifying recurring issues or weaknesses in development practices.

The Contractor **SHOULD**:

- Ensure every Defect Report captures at minimum: unique ID, title, description, steps to reproduce, environment, severity/priority, attachments, expected vs actual results, status, assignee, dates, and resolution notes.

6.5.9 Testing and Quality Metrics for CMS

Testing and quality metrics **SHOULD** be available for CMS26. Contractor metrics **SHOULD** include, but are not limited to:

- Percentage of requirements covered by and traced to test cases.
 - Percentage of software code covered by test cases.
 - Current list of defects with data, such as defect title, description, test case reference, requirements reference, severity, open date, planned resolution date, and status, etc.
 - Visual representations (charts or graphs) showing the distribution of defects by severity level, in addition to the rate of opening and closing defects.
 - Statistics showing defect age.
 - Statistics for defect reopening.
 - Release-over-release trend analysis for defect density, defect escape rate, recurring root cause categories, and test coverage progression.
-
- Make testing and quality metrics available for CMS, including requirement coverage, code coverage, defect list, severity charts, age, and reopening statistics.

6.5.10 Test Artifact Retention

Testing artifacts, including but not limited to test plans, test cases, execution results, defect records, evidence screenshots and logs, and traceability exports, **MUST** be retained according to OHIP retention requirements and remain accessible for CMS audit or certification review.

6.5.11 Evidence Integrity

Testing evidence submitted for CMS certification or OHIP approval **MUST** be timestamped, immutable once submitted, and reproducible. Evidence **MUST** include sufficient context (environment, data set, date, tester identity) to independently validate the test outcome.

6.6 Operational Quality Checks

The Contractor **MUST** establish and maintain operational quality checks to ensure continued system performance, stability, and user satisfaction after each Release. These checks **MUST** provide visibility

²⁶ Per Expectation 4 of Test Execution of the MES Testing Guidance Framework.

into production issues, system reliability, and overall system health to support continuous improvement and reduce defects in future Releases.

The Contractor **MUST**:

- Monitor and track user-reported issues—including issues reported by members/consumers, providers, State and county staff, and Contractor operational staff—which includes the following:
 - o Track, categorize, and maintain all issues and defects reported by any end user post-Go-Live.
 - o Analyze severity, frequency, and trends across Releases.
- Conduct ongoing system health monitoring, including and not limited to the following:
 - o Develop, implement, and maintain system health metrics that support continuous monitoring of critical system functions.
 - o Establish automated alerts or reporting mechanisms to notify stakeholders when predefined thresholds or conditions occur.
- The Contractor **MUST** provide recommendations for quality control enhancements, such as expanded test scenarios, strengthened validation steps, or additional regression coverage, to prevent recurrence of production issues. Recommendations **MUST** be documented and tracked in the OHIP-approved project management tool.
- The Contractor **MUST** evaluate system health trends and provide actionable recommendations to mitigate risks or performance degradation, reported at a cadence defined by OHIP.

OHIP will independently conduct post-release quality assessments as defined in [Section 7.13](#), including Go-Live quality analysis and HotFix/Expedited Release analysis. The Contractor's operational quality monitoring must support and provide data for these OHIP assessments.

 **Cross-Reference:** Release readiness exit criteria—including requirements coverage, defect thresholds, security and privacy verification, accessibility conformance, performance validation, and UAT sign-off—are defined in [Section 3.8](#) (Verification and Validation). The Contractor **MUST** satisfy all exit criteria before production deployment.

7.0 Release Management Standard

7.1 Introduction

The Release Management Standard governs planning, scheduling, coordination, execution, and post-deployment activities associated with every software release across the OHIP MES. In a multi-vendor Medicaid enterprise where individual modules are deeply interdependent for sharing interfaces, data flows, and common infrastructure an uncoordinated deployment can cascade into system-wide failures that affect beneficiary access, provider payments, and State compliance obligations. Release management is the discipline that prevents that.

This Standard applies to five distinct release types Enterprise, Major, Minor, Expedited, and HotFix. Each release type will have defined scope, testing requirements, approval steps, and communication obligations. It governs every phase of the release life cycle from initial scope definition and schedule management through deployment coordination, Go/No-Go decision-making, post-deployment Hypercare, RCA, and performance reporting via KPIs. For the System and Data Integrator (SDI) operating at the center of the OHIP MES, this Standard is particularly critical: as the integration hub coordinating multiple Contractor deployments, the SDI is responsible for ensuring that cross-system dependencies are identified, communicated, and resolved before any module goes to production.

A defining characteristic of this Standard is its emphasis on predictability and accountability. OHIP and its Contractors must operate on visible, documented schedules with defined freeze dates, structured change control processes for post-freeze modifications, and mandatory escalation paths when milestones are at risk. Every release leaves behind a documented record, Release Notes, an Implementation Completion Report, and where applicable a RCA, creating an auditable history of what changed, when, why, and what was learned.

7.2 Objectives

The primary objectives of this Standard are to:

- Establish a structured, predictable release process across five defined release types that reduces deployment risk, ensures consistent quality gates, and protects the stability of production systems shared across multiple MES modules.
- Prevent cross-system conflicts and integration failures by requiring early identification of dependencies, coordinated scheduling through the OHIP Enterprise Release Calendar, and mandatory Go/No-Go processes before any multi-module deployment proceeds.
- Enforce consistent quality requirements, including mandatory SDLC phases, scope freeze management, testing completion, and formal approval steps, as non-negotiable preconditions for production deployment approval across all release types.
- Provide stakeholders at all levels with clear, timely, and standardized communications on release scope, impacts, deployment status, and risk throughout the release life cycle, from initial planning through post-deployment stabilization.
- Establish a continuous improvement loop through mandatory post-release reviews, RCA for emergency releases, and KPI reporting that tracks release health, defect rates, and deployment efficiency over time.

- Define clear accountability for release coordination, particularly for the SDI role, ensuring that multi-vendor deployments are actively managed, escalation paths are documented, and no module deploys in isolation from its system dependencies.

7.3 Release Types

OHIP utilizes five primary release types, with Major and Minor releases aligning under an overarching Enterprise Release umbrella to meet various end-user needs on different time and urgency scales.

The Contractor, in collaboration with OHIP State Business Leaders, **MUST** create an initial schedule. OHIP reserves the right to adjust schedules for sequencing and dependency management.

7.3.1 Enterprise Release

The Enterprise Release Schedule follows a predictable cycle for solution updates organized around Major and Minor releases. A major release is a significant update that introduces substantial changes or new features to the solution, aiming to enhance its functionality in a meaningful way. These major updates serve as the cornerstone of the release strategy because they usually involve fundamental improvements or transformations in the solution's architecture.

After a major release, the next scheduled updates are minor releases. These minor releases contain smaller changes or enhancements, possibly including bug fixes, security patches, or incremental feature improvements. This release cycle ensures that the solution is continually evolving while maintaining stability and user familiarity. OHIP will work closely with the Contractor to identify if and when multiple minor releases are required between each major release.

To ensure regular progress and continual improvement, the organization commits to having at least one of these releases every quarter. This quarterly schedule guarantees that the solution remains up to date with improvements and fixes, providing a reliable and steady flow of enhancements to users.

The Contractor will receive guidance through all phases of Enterprise releases and through all release types. The Contractor retains the ability to deploy Expedited and HotFix-type releases, but these **MUST** be reported as “Standalone” releases, rather than part of the existing Enterprise suite.

7.3.2 Major Release

A major release in solution development, more generally, signifies a key update that brings substantial enhancements and transformations to an application or solution. It serves as a pivotal moment in the life cycle of the solution, often marking the introduction of new features, significant architectural improvements, or changes that impact usability, performance, and scalability.

The purpose of a major release is to address evolving user needs, incorporate technological advancements, and support strategic business goals. It provides users with enhanced capabilities, streamlined workflows, and improved efficiencies, contributing to a more robust and versatile solution. Major releases typically involve comprehensive changes that can reshape the user experience or expand the functionalities of the solution, positioning it to better meet future demands and challenges.

For major releases the Contractor **MUST**:

- Plan and develop such releases with OHIP
- Undergo thorough reviews by technical leads, project managers, and stakeholders

- Deploy on an OHIP-approved cadence.
- Test in lower environments before receiving OHIP approval for deployment to production.²⁷
- Adhere to all SDLC requirements.²⁸
 - o Complete SIT, UAT, Regression Testing, and other test types deemed necessary by OHIP before approval to deploy.

The Contractor will receive guidance throughout the testing phases from OHIP, and approval is contingent upon meeting criteria for functionality, performance, and business alignment.

7.3.3 Minor Release

Minor releases are deployed more frequently and contain bug fixes. They may also contain enhancements if approved by OHIP.

For minor releases the Contractor **MUST**:

- Test in lower environments before being prepared for deployment to production.
- Adhere to all SDLC requirements.
- Conduct appropriate reviews for targeted changes, with approval focused on ensuring enhancement meet specified objectives.
- Be subjected to a full testing component and complete the SIT, UAT, and Regression cycle processes before approval to deploy can be received from OHIP.

7.3.4 Expedited Release

Expedited releases are intended to contain critical bug fixes, which **MUST** be released sooner than the next Major/Minor release, but less urgently than an immediate HotFix. Expedited releases **MUST** be approved by OHIP. To implement expedited releases, the Contractor **MUST**:

- Test in lower environments before being prepared for deployment to production.
- Complete an agreed-upon set of testing and validation before deployment and **MAY** not follow the SDLC requirements.
- Be subject to a limited amount of testing, which **MAY** include and is not limited to internal Contractor validation, OHIP data testing and validation, and a cycle of manual regression and automated smoke tests.
- Document and communicate the enhancements regarding the fixes applied.

²⁷ For more information, refer to [Section 6.0](#).

²⁸ For more information, refer to [Section 8.0](#).

7.3.5 HotFix Release

HotFix releases are intended to contain critical bug fixes, which **MUST** be released immediately to resolve high-severity impact and **MUST** be approved by OHIP.

- Due to the high severity of impact being resolved, HotFix releases do not follow the SDLC process but are expected to complete an agreed-upon set of tests adequate for the release timeline.
- HotFix testing **MUST** be as comprehensive as teams are able to complete without significantly delaying the deployment.
 - o HotFixes are intended to primarily consist of Severity One (1) and Severity Two (2) items, with Severity Three (3) being approved only after additional review by OHIP and Business Leadership.
 - o Following deployment confirmation, additional testing and validation **MUST** be completed in the next numbered software release.
 - o For all HotFix and Expedited releases, the Contractor **MUST** complete a retrospective peer code review within one (1) business day of deployment. If the HotFix modified authentication, authorization, encryption, or interface logic, a mandatory security review **MUST** be completed within two (2) business days. All emergency code changes **MUST** be merged back into the mainline development branch and validated in the next scheduled release.

Routine and Scheduled Maintenance. Routine maintenance activities—including security patching, platform and dependency updates, certificate renewals, and infrastructure maintenance—are not a separate release type. Maintenance that modifies deployed software or configuration **MUST** be managed as a Minor Release, or under the Expedited or HotFix process where its urgency meets the criteria for those types. Maintenance that does not modify deployed software or configuration (e.g., infrastructure-level maintenance) **MUST** nonetheless be scheduled on the OHIP Enterprise Release Calendar as a maintenance window, with the notification and communication obligations of this Standard applying to any activity with potential user-facing impact, including planned downtime. No maintenance activity **MAY** be performed outside a scheduled and communicated window except under the HotFix provisions of this Standard.

7.4 Release Milestones

Each software release is composed of a series of SDLC milestones, which define the beginning, progression, and end of a release cycle. The Contractor **MUST** track and report dates, status, and risks for the items outlined in this section in a manner and cadence to be determined during project initiation.

7.4.1 Scope Phase

Before a release can be started, it **MUST** first be appropriately sized. The scope phase is used to determine how many items (and of which type) a release can contain while still maintaining quality and consistency. The Contractor **MUST** work with OHIP PMO stakeholders to help ensure initial scope is within the abilities of testing teams, and other supportive organizations.

At various times during the solution release cycle, scope **MUST** be either “Frozen” or “Open.” For example, during SIT, there are generally numerous code changes due to defects being identified throughout the test cycle. Scope is expected to be dynamic to accommodate these unexpected changes

and revisions. During the UAT phase, most or all unexpected defects are expected to be included, and the test scope is expected to remain somewhat static.²⁹

Production PHI/PII **MUST NOT** be used in non-production environments unless explicitly authorized by OHIP and protected in accordance with OHIP security and privacy standards. Environment promotion **MUST** follow a defined sequencing (e.g., Development → SIT → UAT → Staging → Production) with no environment bypassed without documented OHIP approval.

7.4.2 Deployment/Go-Live

Deployment/Go-live activities are tracked as a multi-step path beginning with approval and ending with confirmation of deployment and validation completion. The Contractor **MUST**, at a minimum, track and report on:

- Production Readiness Review meeting and communications date and result.
- Go/No-go meeting and communications date and result.
- Final Deployment Approval.
- Deployment report, including activities (e.g., start, progress updates, and completion update).
- Go-Live date when the updated code is considered ready and available to end-users.

Every production deployment **MUST** include a documented rollback plan that defines: rollback triggers (criteria under which rollback is initiated), rollback decision authority, execution steps, validation procedures to confirm successful rollback, and estimated restoration timelines. Rollback procedures **MUST** be validated during pre-production Deployment Procedure Reviews.

Post-deployment validation **MUST** include, at minimum: smoke testing of critical business transactions, synthetic transaction validation for key workflows, interface and batch processing verification, monitoring and alerting confirmation, and data reconciliation where applicable. Validation results **MUST** be documented and provided to OHIP before Go-Live is declared.

7.4.3 Post-Release Cycle

Validation and observation continue in the Maintenance and Operations (M&O) phase of the SDLC. The Contractor **MUST** conduct solution testing in a test environment, at minimum, for the following circumstances:

- Releasing an update to the solution after Go-Live.
- Fixing a defect in a solution after Go-Live.
- Change is proposed to another MES solution or solution that might affect the solution.
- Change is proposed to infrastructure software, such as a patch, fix, or new version of database software, which might be used by the solution.

²⁹ For more information, refer to [Section 6.0](#).

These changes are governed by a technical change control process. A change is initiated by a Contractor, OHIP Information Systems (IS), or another party. In the change request, they document the anticipated impact on other solutions, and the stages and types of testing required.³⁰

For additional details about the post-release testing process, refer to [Section 6.6](#).

7.5 Release Schedule

To provide an Enterprise-level release and deployment experience, OHIP has outlined an overarching schedule structure, which acts as a vehicle for disparate Project, Application, or Contractor releases.

7.5.1 Enterprise Schedule Creation

The OHIP Enterprise Schedule is established by OHIP in consultation with existing, impacted Contractors. It **MUST** be evaluated against any moratorium dates and other Project/Application Release Management Plans for any overlap or intrusion of blackout dates.

The Enterprise Schedule **MUST** be reviewed by OHIP and distributed to OHIP and Contractor leadership resources upon approval.

7.5.2 Contractor Schedule Creation

The Contractor **MUST** retain control and responsibility for release schedule creation, reporting, and code delivery, but are expected to adhere to constraints (e.g., type, milestones, and naming) as well as all following reporting and communication processes ascribed by OHIP.

When creating schedules, the Contractor **SHOULD** be aware of and strive to align with the OHIP Enterprise Schedule, while being mindful of potential downstream code dependencies. Any potential dependency or conflict is to be brought to the attention of OHIP so analysis and recommendations can be provided to avoid unexpected schedule or quality impact.

7.5.3 Sample Enterprise Release Calendar

Figure 1 shows the 2025 Enterprise Release Schedule. Purple rectangles that start with “NY” represent Major/Minor releases. Grey rectangles that begin with “Project” illustrate the “container” relationship between aligned and standalone Contractor releases.

³⁰ Refer to the OHIP Program and Project Management Domain Document, Technical Change Control Standard for more information.

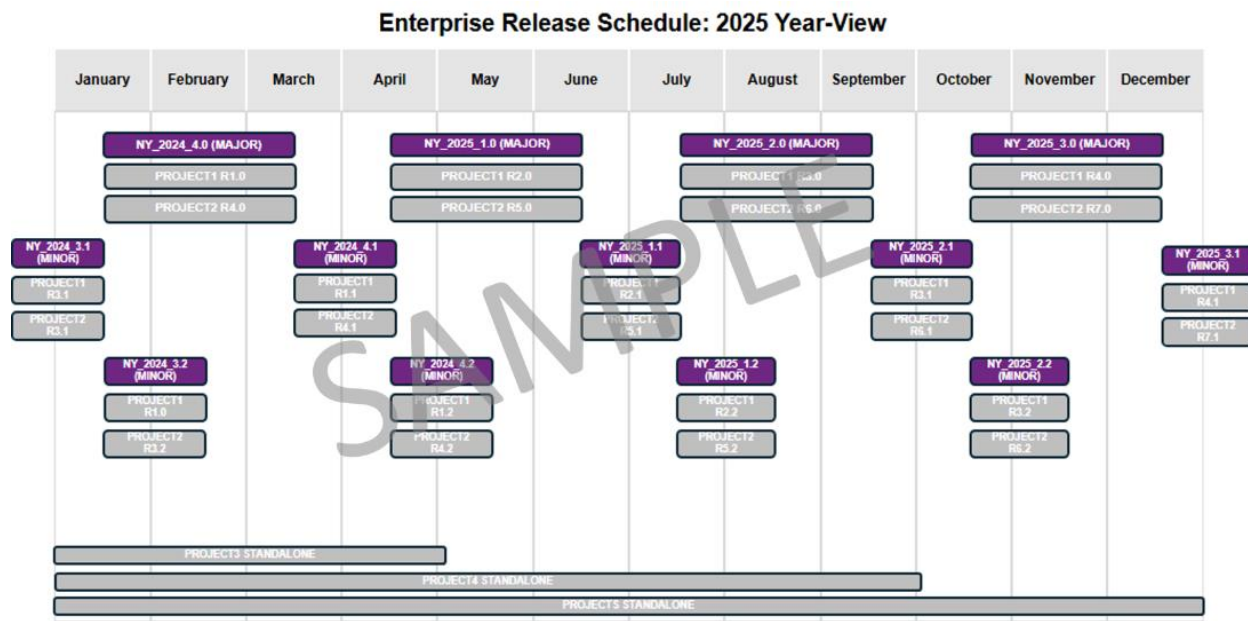


Figure 1 – Sample OHIP Master Release Schedule: Year-View

7.5.4 Moratoriums and Blackout Windows

Moratorium and blackout dates are compiled by OHIP for the current year with Contractor input and generally are on federal and state holidays. The Moratorium Calendar is evaluated regularly to help ensure accuracy. Currently, OHIP does not observe “hard” moratorium dates, meaning moratorium information is provided for guidance and to inform teams that moratoriums *might* be required during deployment work or immediately after completion. Moratoriums must be scheduled and approved in advance.

7.5.5 Blackout Collision Process

In the event of a collision between critical schedule milestones and moratorium dates, OHIP will perform an analysis to determine whether work can proceed as planned or will require additional communication efforts. If work is approved to proceed, additional resources may be required to be available, either on standby or in an active available state. If OHIP is unable to approve, OHIP will work with the Contractor to determine the next available date. The forward-facing schedule will be evaluated at this time to determine if any future dates or dependent projects require adjustment or realignment.

7.6 Change Control and OHIP Interaction

Code-layer change control, including version control, peer review, commit integrity, and developer environment isolation, is governed by [Section 5.10](#) (Code Change Control). This section governs release-level change control and OHIP approval processes. Both layers must be satisfied for any change promoted to production.

The release-level change control process governed by this section applies to all changes promoted to production environments, including application code changes, configuration changes, infrastructure changes, and database schema or data fix changes. For database schema changes and data fixes specifically, the OHIP Data and Information Management Domain Document, Sections 4.5.3 and 4.5.4,

govern the database-layer change control process and **MUST** be satisfied concurrently with the release-level requirements of this section. A change request submitted under this section that involves a database schema or data modification **MUST** reference the corresponding Data and Information Management change control record. OHIP release approval **MUST NOT** be granted for database changes that have not completed the Data and Information Management database change control process.

Change Control and Release Management are closely related disciplines to ensure releases are accurately prepared, tracked, and documented. While Release Management focuses on maintaining an SDLC, change control provides controls and governance over how software releases **MUST** be documented and approved before (and after) deployment. Every software release **MUST** have at least one, and often multiple, change requests submitted for tracking and approval purposes.³¹

Release-level change control decisions, including release prioritization, scope freeze exceptions, deployment conflict resolution, and emergency deployment authorization, **MUST** be governed by a Release Governance Board (RGB) or equivalent governance body established by OHIP. The RGB **MUST** include representation from OHIP PMO, Business Owners, Security, Enterprise Architecture, Testing Leadership, Operations, and impacted Contractors. The RGB's decision authority, meeting cadence, and escalation path **MUST** be documented in the OHIP-approved Release Management Plan.

7.7 Adding and Removing Items Post-Release Interaction

To further support testing teams and overall release stability, the Contractor **MUST** provide planned release freeze dates prior to the start of any testing. Any code changes made after the release freeze date due to unplanned change requests or production incidents can potentially cause instability in the environment, unpredictable test results, and ultimately lead to escaped defects. Because of this, any unplanned scope change proposals submitted after the release freeze date **MUST** be approved in a review meeting which evaluates risk from development, quality assurance, and business perspectives.

7.7.1 Approval Steps Required

The following approval steps **MUST** be adhered to for adding or removing items after scope freeze:

1. Issue is reported and identified as being critical for removal or addition after the scope freeze date.
2. After determination of the need for addition/removal, Development and Test teams analyze, document, and formalize an estimate of time-to-code and test and evaluate whether the item needs to be addressed via HotFix or Expedited release versus an addition to the current release.

The following approval steps **SHOULD** be adhered to for adding or removing items after scope freeze:

1. Issue priority is evaluated and updated to Severity 1 or Severity 2. Severity 3 items should very rarely be added to a release post-scope freeze and require additional review.

³¹ Refer to the OHIP Program and Project Management Domain Document, Project Change Control Standard for more information.

If the upcoming release is determined to be the required vehicle, then the Contractor **MUST** complete the following steps:

1. Analysis documentation is updated in a standardized MS Word, Excel, or Outlook template, or in a format agreed upon by OHIP and the Contractor.
2. The analysis **MUST** be archived in such a way as to be easily retrievable upon request by OHIP.
3. Dev/Test timeline and justification **MUST** be presented to OHIP Leadership, Business stakeholders, and System PMO via email for review prior to the meeting.
4. Issue is discussed and scoped/confirmed in meeting with the OHIP Business and any other relevant stakeholders.
5. During or immediately following the meeting, the reason for addition/removal is documented.
6. An informational email is sent to the UAT Manager with the OHIP Business and PMO point of contacts in carbon copy (CC). This email **MUST** contain a brief description of the issue and include both the above justification and the test analysis from Step Three (3) as an attachment.
7. If the addition requires an increase in existing test resources, or inclusion of alternate teams (other than what was initially scoped), this **MUST** be noted clearly and directly in the body of the email with overall quantity and resource name(s) included.
8. A patch item is created, and the Contractor begins working to promote the change in preparation for deployment with the release.

 **Note:** The process described in [Section 7.6](#) is required for both the addition and removal of items.

7.7.2 Post-Scope Freeze Items Template

Once approval has been received to add or remove an item, the Contractor **MUST** document the change and send it to OHIP PMO. The documentation **MUST** include:

- **Issue Details:** Issue Description, Scheduled Release, Component.
- **Addition or Removal Criteria:** Details as to why this defect was deemed critical enough for post-scope freeze addition or removal (e.g., provide number of cases impacted by component).
- **Impact:** Details about impacted population, business severity, and overall scope.
- **Approved by:** Name and role of the person who approved this defect/issue.
- **Development Time Required:** Number of hours and/or resources required for development work to complete.
- **Number of Test Scenarios:** Number of scenarios that require testing, which **MUST** include whether time travel or batch run are required.
- **Estimated testing time:** Number of hours and resources required to complete testing.
- **UAT Time:** Number of hours and resources required to complete UAT.

The documentation **SHOULD** include:

- **Trade-off Incident:** Is there another incident being swapped in-or-out in place of the item in question? This field is not mandatory but should be populated if there is a replacement item.

7.8 Release Communications

A standardized and accurate communications process is critical to successfully executing Release Management.

For all release types, the Contractor **MUST** include:

- Description of the issues: Why is the release needed?
- Impact of the release: Is this a temporary fix? Does it affect other functionality, etc.?
- Planned schedule: When is the release occurring?
- RCA: What caused the issue (post-release analysis), and what can be done to prevent reoccurrence?

The Contractor **MUST** propose content for each of these communications not specified in the RFP.

7.8.1 Release Implementation Completion Report

To help ensure documentation of Release Implementation, the Contractor **SHOULD** submit the Release Implementation Completion Report following deployment of the Release.

The Contractor **SHOULD**:

- Propose content for release communications not specified in the RFP.

7.9 Deployment Activity Coordination for Multiple Integrations/Dependencies

OHIP plays a central role in documenting release readiness planning, individual/collective Go-Live approaches, and developing and maintaining the communication plan and distribution lists to use during the release deployment cycle as some releases may impact other modules. The definition of a requesting module and integrating module are included below:

- **Requesting Module:** The Contractor responsible for consolidating and monitoring all connected releases for a particular release cycle.
- **Integrating Module:** The Contractor responsible for performing project or application work with outside dependencies.

For deployment activity coordination, the Contractor of the Requesting Module **MUST**:

The Contractor **SHOULD**:

- Initiate deployment activity planning as soon as a release is identified as having dependencies, integrations, or connected project activities.
- Develop and maintain Deployment Plans from Integrating Modules into a comprehensive Release Management Plan and update it throughout the release life cycle.
- Lead, facilitate, and coordinate all deployment activities associated with the release, including integrating module inputs and cross-team dependencies.
- Follow OHIP-provided processes that define deployment indicators, status reporting requirements, communication expectations, and update cadences during the deployment cycle.

- Collaborate with OHIP and request assistance or feedback when necessary to support effective deployment activity coordination.

7.9.1 Activities

During a deployment cycle, the following coordination processes **MUST** be observed. The items below define what, when, and how for many key moments during deployment work.

- **Distribution Lists:** Updates **MUST** be provided to a consistent group of stakeholders.
- **Deployment Update Methods:** Updates **MUST** be provided to stakeholders by the Requesting Module using approved distribution channels
- **Deployment Update Cadence and Key Indicators:** Updates **SHOULD** be provided to the Requesting Module by the Integrating Module Project leads and consolidated and distributed to the "Master Distribution List" by the Requesting Module at the following intervals:
 - o Go-Live approval received
 - o Deployment start
 - o Key milestone completion and/or status triggers (refer to the next bullet, "Status Callout Triggers")
 - o End-of-day (if work crosses multiple days)—could also be referred to as "deployment work-pause"
 - o Beginning-of-day (if work crosses multiple days)—could also be referred to as "deployment work-resume"
 - o Deployment activity completion
 - o Post-deployment validation starts (can be combined with the previous item if validation start directly coincides with deployment end)
 - o Post-deployment validation completion
- **Status Callout Triggers:** Certain events outside established cadences or indicators **MUST** trigger an update to be documented and/or provided to the "Master Distribution List" by Integration Modules. The following events **MUST** trigger a status callout:
 - o An issue that has a significant risk of delaying internal downstream or other Project milestones is identified
 - o A project milestone has been missed or rescheduled
 - o Unanticipated staffing concerns that potentially impact the timeline
 - o Moratorium dates that cannot be worked over
 - o Deployment work is significantly ahead of schedule
 - Each coordinated deployment **MUST** rely on a Release Management Plan to provide clear, documented steps that will prevent collisions and ensure critical activities are reported accurately.

7.9.2 Status Readout

After Initial Integration Planning occurs, a reoccurring status readout meeting **MUST** be established to facilitate the following activities:

- Review previous and current releases
- Review the scope of the upcoming release
- Assess and mitigate risk
- Update the Release Management Plan
- Communicate changes
- Revisit the long-term Roadmap

Within project Go-Live or cutover, a "final" status readout meeting may be scheduled to address release retrospective release health.

During a release cycle in which multiple projects, and/or applications deploy with dependencies, a Go/No-Go process **MUST** be employed to ensure clear communication and avoid potential collisions or issues.

The Contractor **SHOULD**:

- Establish a recurring status readout meeting after initial integration planning.

7.9.3 Release Plan

During a release cycle in which multiple projects, Contractors, and/or applications are deploying with dependencies, there is always risk of potential collision, delays, or unforeseen circumstances impacting projects, which need to be addressed by individual project leaders and the coordinated group. To obtain OHIP approval for a release, the Contractor **SHOULD** submit a Release Plan to describe the approach for how a product, enhancement, or feature will be delivered to end users. It ensures that all elements of the Release are thoughtfully organized, coordinated across teams, and executed with minimal disruption.

7.9.4 Post-Deployment Hypercare

Once coordinated releases have been completed, there **MUST** be a period of heightened awareness and rapid response to deployment-related and potentially deployment-related issues. Before commencing operations, the Requesting Module **MUST** schedule a brief meeting with the Hypercare Coordinator and develop the Hypercare Plan, which **MUST** be approved by OHIP.³²

Comprehensive Hypercare requirements, including planning and coordination activities, execution and support obligations, success criteria, and exit/extension procedures, are defined in [Section 8.5.28.5.2](#). The requirements in this section govern the release-coordination aspects of Hypercare; [Section 8.5.2](#) governs the full Hypercare life cycle.

³² For more information, refer to [Section 8.0](#).

7.9.5 Escalation Process

Any project that requires coordination between multiple projects, Contractors, and/or applications has the potential to experience conflict and/or communication barriers.

In the event of coordination or communication challenges, teams need to be able to escalate to resolve conflicts and continue work in an effective manner. In the event of issues, attempts **MUST** be made to resolve both internally and directly with the impacted/impacting parties. If these attempts fail, then escalation **MUST** occur.

Some examples of escalation-worthy problems include, but are not limited to, the following:

- A key stakeholder regularly misses key status update meetings.
- A key stakeholder does not respond to emails or does not respond in a timely fashion.
- A key stakeholder refuses to provide critical schedule or progress information when requested.

Requesting Module Path

- If resolution attempts are unsuccessful, the Requesting Module **MUST** contact OHIP to attempt to facilitate coordination activities.
- If OHIP cannot resolve the issue, the OHIP escalation path **MUST** be followed.
- If the escalation is due to OHIP, the escalation path **MUST** be followed, beginning with the PMO Lead and moving upward.

Integrating Module Path

- If resolution attempts are unsuccessful, the Integrating Module **MUST** contact OHIP to attempt to facilitate coordination activities.
- An OHIP escalation path **MUST** be followed if OHIP fails to garner a solution.
- If the escalation is due to OHIP, the OHIP escalation path **MUST** be followed, beginning with the PMO Director and moving upward.

7.9.5.1 Initial Reporter Tasks

If a milestone is at risk or missed, OHIP **MUST** be notified immediately. The Contractor **MUST** send this notification to OHIP Project Managers. If a date is missed, OHIP will directly contact the leadership teams in charge of the milestone. The Contractor **MUST** identify the impact of the missed milestone, the anticipated date the milestone **MUST** be met, and a RCA for the missed milestones within one week of notification. The Contractor **SHOULD** propose the format of the notification unless specified in the procurement.

7.10 Release Notes

To help ensure key stakeholders have a clear understanding of what is included in upcoming software deployments, OHIP requires the Contractor to distribute “Release Notes” to the identified stakeholders.

- **Delivery Timeline:** Release Notes are to be delivered before completion of software deployment.
- **Scope:** Release Notes **MUST** be created and distributed with all versioned software deployments.
- For HotFix and Expedited releases, the Contractor **MUST** provide Release Notes containing all known production-impacting changes at the time of deployment. Where complete information is

unavailable at deployment time, the Contractor **MUST** provide an updated version within two (2) business days of deployment.

- The recommended delivery format is to send an email with a multi-tab MS Excel attachment. The release “contents” portion **SHOULD** also be included in the change request for deployment to help ensure key stakeholders have a clear understanding of what is included in upcoming software deployments.

7.11 Go/No-Go Process

To help ensure adequate checks and balances, and to increase awareness and accountability across multiple offices, the Contractor **MUST** utilize a Go/No-go process in collaboration with the state prior to deploying new code in a Production environment.

 **Cross-Reference:** Release readiness exit criteria—including requirements coverage, defect thresholds, security and privacy verification, accessibility conformance, performance validation, and UAT sign-off—are defined in [Section 3.9](#) (Verification and Validation). All exit criteria **MUST** be satisfied before the Go/No-Go decision proceeds.

7.12 Root Cause Analysis (RCA) Process

To help ensure clear, consistent communication, adequate follow-up, and completion of preventative measures, the following process **SHOULD** be followed upon deployment of any HotFix or Expedited Release. Immediately after a HotFix or Expedited Release deployment, communication **SHOULD** be sent to the identified distribution group. This communication **SHOULD** identify a key stakeholder responsible for RCA completion and provide an estimated time for the first update. The Contractor **SHOULD** propose the contents of the RCA unless specified in the RFP.

RCA for HotFix and Expedited releases **MUST** be completed and submitted to OHIP within five (5) business days of deployment. Each RCA **MUST** include: a description of the root cause using a structured methodology (e.g., Five Whys or equivalent), corrective actions with assigned owners and due dates, and verification that corrective actions are effective. OHIP **MUST** track corrective action completion and recurrence. Recurring root causes **MUST** be escalated to the Release Governance Board.

7.13 Post-Release Quality Assessment

The Contractor’s obligations for operational quality monitoring post-deployment are defined in [Section 6.6](#). OHIP’s post-release quality assessments described in this section leverage and build upon the Contractor’s quality data.

OHIP will conduct Go-Live quality assessments for every Major/Minor release, which details unexpected or “emergency” release reporting and analysis, and deficiencies in the release and deployment process.

7.13.1 Post Release HotFix and Expedited Release Analysis

OHIP Go-Live quality analysis includes, and is not limited to, the following:

- Count of HotFix and Expedited Releases per Major/Minor release
 - o Historical trending visualization

- Count of HotFix and Expedited Releases generated by a previous HotFix (versus a planned release)
- HotFix and Expedited Release content analysis
 - o Severity
 - o Create-date
 - o Component
- RCA Completion and follow-up items review

7.14 Post-Release Review Meeting

Following each Major and Minor release, OHIP will schedule and host a Post-Release Review Meeting to review findings and discuss successes, issues, and areas for improvement. This meeting **SHOULD** occur within two (2) weeks of the release Go-Live date. The Contractor **MUST** ensure Contractor Release Management Leadership and any additional relevant stakeholders participate in the Post-Release Review Meeting. The meeting **MUST** also have representation from OHIP Executive Leadership and the PMO.

7.14.1 Defect Severity Assignment

The Defect Triage Process **MUST** include a comprehensive review of defect severity and the subsequent mitigation plans.

Defect management planning, defect reporting requirements, and defect life cycle processes are governed by [Section 6.5.6](#) and [Section 6.5.7](#). This section defines the severity classification framework and triage process that must be applied within that defect management life cycle.

A high-level outline is described below:

- Discuss the initial severity of each defect in triage.
- Triage team revises original Severity/Revise up or down as needed with business approval.
 - o Business determines the business urgency.
 - o Additional defects added to the scope after the scope freeze date **MUST** be of the Severity 1 and Severity 2 categories.
 - o Any Severity 3 item proposed for post-scope-freeze inclusion **MUST** be accompanied by a documented business justification and impact analysis and **MUST** receive explicit OHIP and Release Governance Board approval before inclusion. Severity 4 items **MUST NOT** be added after scope freeze.
 - o Any defect tagged to a HotFix/expedited release that did not go through triage **MUST** be addressed to capture the correct severity.
- The Contractor provides impact analysis of any related functional area (e.g., batches, no touch, and notices, etc.) and fixes/retests the complexity analysis.
- The defect is scheduled for an active release or put in "backlog."
- If the active release scope freeze date is closed, the business **MUST** make a crucial decision regarding the need for and feasibility of scope changes.
- The Contractor **SHOULD** fully analyze the impact on DEV, SIT, and UAT, estimating the effort needed to implement the fix, and present it to the business.

- The Contractor **MUST** confirm the release label for the defect.
- The result of the analysis and details on testing effort (e.g., number of test scenarios and need for time travel) and timelines are presented to the UAT manager for awareness.

Table 6 – Defect Severity Assignment

Severity	Definition	Incident Resolution Timeframe
Severity 1	<p>A security, privacy, or system incident MUST be assigned as a Severity 1 if the incident is characterized by any of the following:</p> <ol style="list-style-type: none"> A technical service and shared services are inaccessible and/or unusable, with no acceptable workaround. An outage that presents significant health, safety, or financial risk to NYS and/or its citizens. A significant portion of Department users are unable to access the solution, or access is materially impaired across one or more business units. Controls are severely compromised (e.g., privileged account breach and widespread infiltration), posing an immediate and serious threat to Department assets. Large-scale exposure to confirmed breach of sensitive personal information (e.g., Personally Identifiable Information [PII], PHI [PHI], and Federal Tax Information [FTI]), raising significant regulatory and reputational risks, or any incident that is actively escalating and could cause the above impacts if not contained immediately. 	Four (4) hours from when the incident began occurring
Severity 2	<p>A security, privacy, or system incident MUST be assigned as a Severity 2 if the incident is characterized by any of the following:</p> <ol style="list-style-type: none"> An outage that is affecting solution technical services and shared services (e.g., connectivity for a physical location, downed operations affecting multiple clients, or production system operating in a degraded mode). A fix is required to be implemented within a certain timeframe. Although a workaround exists, its use has significant business impacts. Non-recoverable data loss Unusual or unauthorized access to key system or data, with the potential to escalate if not contained quickly. Likely compromise of personal information (e.g., PII, PHI, and FTI) that demands swift action to reduce further exposure. 	Eight (8) hours from when the incident began occurring

Severity	Definition	Incident Resolution Timeframe
Severity 3	<p>A security, privacy, or system incident MUST be assigned as a Severity 3 if the incident is characterized by any of the following:</p> <ol style="list-style-type: none"> Although a technical service or shared service is usable, it is not functioning in accordance with applicable specifications. There are no substantial adverse impacts on business operations. A substantial adverse impact includes repercussions to business operations and functionality defined as Critical and High (refer to Table 7 and Table 8). Suspicious activity or limited breach of less critical systems, requiring thorough investigation to prevent deeper infiltration. Potential exposure of data on a smaller scale or in a limited context, warranting prompt corrective measures. 	Three (3) calendar days from when the incident began occurring
Severity 4	<p>A security, privacy, or system incident MUST be assigned as a Severity 4 in the event that any incident that does not comply with the requirements of Severity 1, 2, or 3, or is characterized by any of the following:</p> <ol style="list-style-type: none"> A technical service and shared service requires minimal attention. The issue is not substantially averse to the business operations of the solution (e.g., daily functionality is not impacted, or a workaround is possible) and does not require immediate resolution. A date for correction MUST be agreed to between the parties. Minimal threats to Department assets, often related to isolated or easily contained incidents. Little to no risk to personal data, with any concerns typically handled through standard support process. 	Six (6) calendar days from when the incident began occurring

The Contractor **SHOULD**:

- Conduct a defect triage process including severity review, business urgency assessment, impact analysis, and scheduling decisions.

7.14.2 Priority Assessment

Priorities for issues are assigned based on Urgency Level and Issue Impact. Urgency Level indicates the level denoting how quickly the issue needs to be resolved.

Table 7 – Urgency Level

#	Urgency Level	Description
1	Critical	The issue needs to be addressed immediately, as the project cannot move forward until the issue is resolved.
2	High	The issue needs to be addressed. While it does not have an immediate severe impact on the project, it will if it is not resolved.
3	Medium	The issue MUST be addressed but does not immediately impact the project.

Issue Impact: Based on the risk matrix, Table 8 describes the severity of the issue's impact.

Table 8 – Issue Impact

#	Issue Impact	Description
1	Critical	The issue has a significant impact on the scope, timeline, and/or budget.
2	High	The issue has a moderately high impact on the scope, timeline, and/or budget.
3	Medium	The issue has an impact on the scope, timeline, and/or budget.

Priority rating of issues is based on the priority levels in Table 9.

Table 9 – Priority Levels

#	Priority Level	Description
1	Critical Priority	The issue MUST be a complete outage or severe degradation of the Contractor's System that affects a large portion of the State staff or external users with no viable workaround, posing an immediate business, financial, safety, or regulatory risk.
2	High Priority	The issue is a significant degradation or partial outage of the Contractor's System that impacts many State staff or external users, and a workaround may exist but is cumbersome or risky.
3	Medium Priority	The issue has a limited impact on non-critical functions of the Contractor's System or a small number of State staff or external users, with a workaround available with manageable inconvenience.
4	Low Priority	The issue causes a minor disruption, involves cosmetic defects, or consists of informational requests reported as incidents, with no meaningful business impact and an easy or no-impact workaround.

Table 10 highlights an issues matrix, based on urgency and impact.

Table 10 – Issues Matrix

	Impact		
Urgency	Medium	High	Critical
Critical	High Priority	Critical Priority	Critical Priority
High	Medium Priority	High Priority	Critical Priority
Medium	Medium Priority	Medium Priority	High Priority

7.15 Key Performance Indicators (KPIs)

To support accurate reporting on release status, health, and efficiency, the Contractor **SHOULD** create and update the KPIs listed below on a monthly basis. The Contractor **SHOULD** compile, validate, analyze, and distribute these metrics to a key list of stakeholders identified by OHIP. The method of creation and distribution remains at the sole discretion of the Contractor; however, the creation methodology and distribution vehicle **SHOULD** be documented and provided to OHIP for review before acceptance. Where possible, usable queries **SHOULD** be provided.

To support independent validation, the Contractor **MUST** provide OHIP with unrestricted access to the underlying data sources used to generate KPI metrics.

7.15.1 Time or Deployment-Based Statistics

Time or deployment-based statistics **MUST** be provided for the overall scope of work or application. These metrics **MUST** be broken out by severity as shown in Table 11 below.

Table 11 – Time/Deployment Based Statistics

Area Measured	Sample KPI
Number of releases	<ul style="list-style-type: none">Total releases per month/quarter and year by release type (e.g., major, minor, and break/fix)
Release success	<ul style="list-style-type: none">Code delivered per scheduled timeWhether scope has or had to be adjusted (e.g., items moved in or out of the release)
MTTR (Mean-Time-To-Repair) items	<ul style="list-style-type: none">Time from release at fault to incident reportTime from report to validated fix

7.15.2 Impact-based Statistics

Impact-based statistics **MUST** be provided for the overall scope of work or application. These metrics **MUST** also be broken out by severity. Sample KPIs are highlighted below.

Table 12 – Impact Statistics

Area Measured	Sample KPI
Release-caused incidents	Broken out by cause (e.g., missed activity, missed requirement, or missed testing, and faulty code, etc.)
Release-caused incidents	Severity 1 and Severity 2 Incidents
Release-caused incidents	Incident-related solution downtime
Emergency releases	<ul style="list-style-type: none">Total CountTime from release to detectionTime from detection to deploymentAdditional Emergency releases generated by the prior Emergency release
RCAs	<ul style="list-style-type: none">Number missingTime to deliveryPercent that accurately identified cause
Estimated deployment downtime	<ul style="list-style-type: none">External facingInternal facing
Actual deployment downtime	<ul style="list-style-type: none">External facingInternal facing

In addition to the metrics defined above, the Contractor **MUST** track and report: escaped defect rate (defects discovered in production per release), HotFix-to-planned-release ratio (percentage of releases that are unplanned), and release-over-release trend analysis for defect density, deployment success rate,

and mean time to recovery. These metrics **MUST** be reported monthly and included in the post-release review.

7.16 Release Oversight

The Contractor **SHOULD** provide regular communication and feedback when its releases impact other information systems in OHIP. The Contractor implementing the change **SHOULD** coordinate with OHIP and other Contractors' release schedules to align activities and changes.

7.16.1 Master Release Schedule View

The first level of reporting covers a rolling ninety (90)-day view and presents multiple scope of work and application releases. The report is structured in such a way as to provide both a calendar view and a text table that provides Health, Release, Go-Live Date, and Status Comments columns.

The health indicator shows whether the release is on track for successful deployment or if potential risks have been identified. This analysis is outlined further in the following sections.

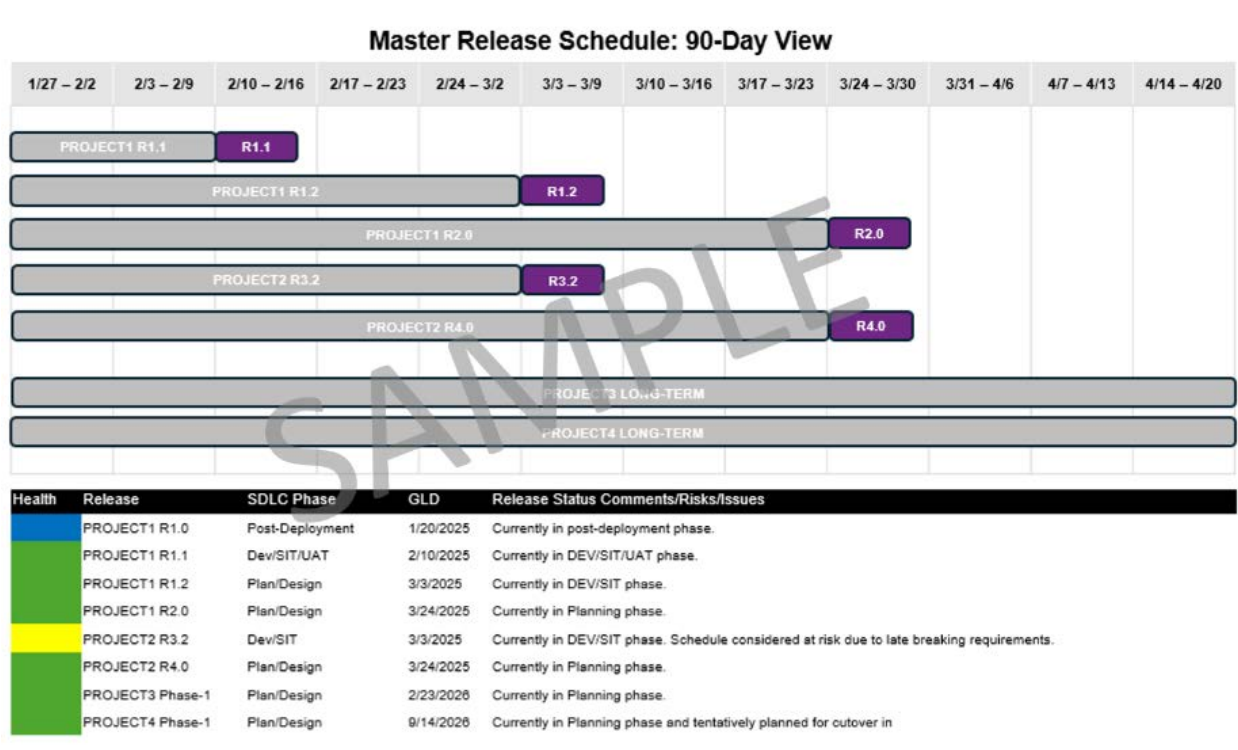


Figure 2 – Sample OHIP Master Release Schedule (90-Day View)

7.16.2 Project Release Metrics

The project release metrics section of the report punches in an additional layer and provides a view of the current related releases, i.e., Major->Minor->Minor, Phase 2A->Phase 2B, etc. This view provides a textual analysis of the current and recent releases and notes in more detail than the initial slide, as well as providing a table-view breaking down each of the related releases and adding detail such as scope, scope change, count of incidents by severity, and a summary of any Hot-Fixes generated by the release.

8.0 Solution Development Life Cycle (SDLC) Standard

8.1 Introduction

The SDLC Standard is the governing framework for how OHIP MES solutions are planned, designed, built, tested, deployed, stabilized, and retired. It recognizes that no single development methodology is appropriate for every project in a large-scale Medicaid modernization program: a commercial-off-the-shelf implementation has fundamentally different delivery characteristics than a complex custom integration or a multi-year platform replacement. This Standard addresses that reality by providing a structured but flexible framework, supporting Waterfall, Agile Scrum, V-Shaped, and three (3) variants of Hybrid methodology, while establishing the overarching governance requirements that apply equally across all of them.

What does not vary across methodologies is the obligation to deliver with quality, transparency, and auditability. Regardless of how a Contractor organizes its sprints, phases, or iterations, it must maintain an approved SDLC Governance Plan, document its approach in an OHIP-approved format, ensure OHIP staff have continuous access to all SDLC artifacts, and align every phase of delivery to the CMS SMC evidence requirements that govern OHIP's federal funding eligibility. The Hybrid Iterative Development (Iteration-Based) methodology introduced in [Section 8.4.4.3](#) specifically addresses the delivery patterns of large-scale, multi-year MES implementations, providing a structured ten-phase life cycle that bridges iterative development with the rigorous End-to-End SIT and UAT required for Medicaid systems.

This Standard also governs the activities that span and follow the primary development life cycle: CMS Readiness Review gates, Organizational Change Management (OCM) alignment, Training, Hypercare, Transition-In for incumbent-to-new Contractor handoffs, and Legacy Decommissioning and Data Archival for releases that retire existing systems. Together these requirements ensure that OHIP's contractual protections extend not just to what is built, but to how it is transitioned, stabilized, and ultimately retired.

8.2 Objectives

The primary objectives of this Standard are to:

- Define a flexible, adaptive SDLC framework supporting Waterfall, Agile Scrum, and three (3) Hybrid methodologies, including the Bundle-Based Hybrid designed for large-scale MES implementations, with structured selection criteria and governance requirements that apply universally across all approaches.
- Establish consistent, repeatable processes for planning, executing, and delivering high-quality MES solutions on time and within scope, with OHIP approval gates at each major life cycle milestone and continuous traceability from requirements through to production evidence.
- Integrate critical oversight functions, including CMS SMC Readiness Reviews, security and privacy compliance, enterprise architecture governance, and data governance, into every SDLC phase as mandatory gates rather than optional checkpoints.
- Clarify roles, responsibilities, activities, and required deliverables for each SDLC phase and methodology, ensuring clear accountability across all Contractor and OHIP participants throughout the project life cycle.
- Protect OHIP's federal funding eligibility by aligning all SDLC phases to CMS SMC requirements, and establishing a mandatory pre-Go-Live CMS Readiness Review gate to verify certification evidence before any production deployment is approved.

- Govern the full solution life cycle beyond deployment, including OCM alignment, Knowledge Transfer, post-deployment Hypercare, Contractor Transition-In, and Legacy Decommissioning, ensuring OHIP's operational and legal obligations are met after a solution goes live, not just before.

8.3 CMS Streamlined Modular Certification (SMC) Requirements

The State Medicaid Director Letter #22-001, issued by the Centers for Medicare & Medicaid Services (CMS) on April 14, 2022, provides updated guidance for the certification of MES, particularly highlighting the SMC process. This approach emphasizes an outcomes-based certification framework, replacing the previous Medicaid Enterprise Certification Toolkit (MECT) and Medicaid Eligibility and Enrollment Toolkit.

Compliance with the CMS SMC processes is required to secure enhanced federal funding for MES projects. Certification under this framework ensures robust solution functionality, security, and interoperability, which are essential for sustaining program improvements and operational transparency. The full OHIP MES team, including the Contractor, **MUST** understand and integrate the CMS-approved outcomes and metrics into deliverables from the outset. All MES solutions developed **MUST** generate required data and uphold compliance standards as set by both federal and State requisites. Close collaboration and seamless communication between OHIP, its Contractors, and the State CMS Officer via the State's liaison are required. Aligning metrics and outcomes reinforces a shared vision, while timely submission of necessary documentation and the active participation of all critical State and Contract resources in CMS review sessions demonstrate ongoing project health and compliance readiness. Regular updates, delivered through structured reports and demonstrations, highlight project progress and help prevent misalignments early and are required by CMS for any project receiving enhanced FFP.

In addition to State resources, the Contractor **MUST** also assist in developing and submitting evidence that supports certification readiness. Operational readiness and certification review sessions with CMS require the State to present detailed solution functionality and performance validations. The active engagement of all required OHIP MES resources in these sessions, coupled with meticulous documentation, facilitates the CMS reviewers' assessment of solution readiness for live operations.

Maintaining certification is equally important and requires continuous reporting efforts post-certification. The Contractor **MUST** support OHIP in submitting operational reports that validate sustained outcome achievements, ongoing compliance with funding conditions, and effective operational support for certified solutions. The Contractor **MUST** identify and address performance issues promptly as specified by OHIP to support continuous quality improvement and risk mitigation.

In summary, the Contractor **MUST** assist with facilitating and supporting the Department's adherence to CMS SMC processes. As of the publish date of this standard, the deliverables in this section are required for CMS SMC. The Contractor **MUST** adhere to any new deliverables or processes if updates are made by CMS after the publication date of this standard.

Deliverables: SMC Intake Form, CMS Monthly Project Status Report, Security Assessment Report, Test Management Plan (and testing results), Deployment Plan, Defect and Risk List, CMS Readiness Review Package, Risk Register (if applicable), CMS Readiness Approval Sign-Off.³³

- The Contractor **MUST** track and remediate all gaps identified during the CMS Readiness Review before the Go-Live gate is approved. Any unresolved gaps **MUST** be documented in a Risk

³³ Refer to the [CMS SMC Certification Guide](#) for more information on CMS SMC deliverables.

Register with OHIP-approved mitigation plans, and OHIP **MUST** provide written authorization acknowledging acceptance of residual risk before deployment proceeds.

- The Contractor **MUST** provide a consolidated CMS artifact package, including Requirements Traceability Matrices, VPAT documentation, security assessment results, and test results reports, to OHIP at least ten (10) business days prior to the scheduled Go-Live date. OHIP **MUST** review and formally approve the package before the Contractor proceeds to deployment.
- The Contractor **MUST** conduct a CMS Certification Readiness Review as a mandatory gate prior to Go-Live approval. During this review, the Contractor **MUST** demonstrate that all system functionality slated for deployment maps to the defined CMS MES Outcomes as documented in the approved SMC Intake Form.

As a mandatory phase gate prior to Go-Live approval, the Contractor **MUST** conduct a formal CMS Certification Readiness Review. This gate ensures that federal certification requirements are met before production deployment and protects OHIP's eligibility for enhanced FFP. This gate applies universally to all SDLC methodologies.

8.3.1 CMS SMC Readiness Review Gate

As a mandatory phase gate prior to Go-Live approval, the Contractor **MUST** conduct a formal CMS Certification Readiness Review. This gate ensures that federal certification requirements are met before production deployment and protects OHIP's eligibility for enhanced FFP. This gate applies universally to all SDLC methodologies.

- The Contractor **MUST** conduct a CMS Certification Readiness Review as a mandatory gate prior to Go-Live approval. During this review, the Contractor **MUST** demonstrate that all system functionality slated for deployment maps to the defined CMS MES Outcomes as documented in the approved SMC Intake Form.
- The Contractor **MUST** provide a consolidated CMS artifact package, including Requirements Traceability Matrices, VPAT documentation, security assessment results, and test results reports, to OHIP at least ten (10) business days prior to the scheduled Go-Live date. OHIP **MUST** review and formally approve the package before the Contractor proceeds to deployment.
- The Contractor **MUST** track and remediate all gaps identified during the CMS Readiness Review before the Go-Live gate is approved. Any unresolved gaps **MUST** be documented in a Risk Register with OHIP-approved mitigation plans, and OHIP **MUST** provide written authorization acknowledging acceptance of residual risk before deployment proceeds.

Deliverables: CMS Readiness Review Package, Risk Register (if applicable), CMS Readiness Approval Sign-Off

8.4 Methodology Requirements

This section provides the requirements that apply regardless of the SDLC methodology and then specifies the requirements for each SDLC methodology supported in OHIP implementations, Waterfall (including V-Shaped), Agile Scrum, Hybrid approaches, and Spiral. DevOps practices may be applied as a delivery acceleration layer within any approved methodology.

The following tables provide guidance for selecting the appropriate SDLC methodology and comparing key characteristics across all supported approaches. Methodology selection **MUST** be documented, justified, and approved by OHIP prior to project initiation (see [Section 8.4.1](#)).

Table 13 – SDLC Methodology Selection Guide

Methodology	Best Suited For	Development Cycle	Testing Approach	Flexibility
Waterfall	Stable, well-defined requirements; COTS implementation; fixed-scope contract; regulatory reporting solutions	Sequential, phase-gated	Phase-end UAT	Low: Changes costly after design
Agile Scrum	Evolving requirements; user-facing applications; innovation-driven features; short delivery cycles needed	Sprint-by-sprint iterative	Per-sprint demo + UAT	High: Backlog reprioritized continuously
V-Shaped	Stable, fully defined requirements; COTS implementations; infrastructure modernization; projects requiring mandatory phase-by-phase CMS certification evidence	Sequential, phase-gated with bilateral test design	Test strategy designed per phase before development; verification at each phase gate before progression	Low: Changes after phase entry require full phase re-entry
Hybrid: Waterfall + Scrum	Large releases with both stable architecture and iterative UI/UX; mixed functional/non-functional scope	Waterfall architecture, Scrum sprints	Per-sprint + full UAT	Medium: Architecture stable, features evolving
Hybrid: Waterfall + Iterative Design	Solutions requiring deep design exploration before development; complex integrations; greenfield platforms	Waterfall development, iterative design gate	Phase-end UAT	Medium: Design flexible, development fixed

Methodology	Best Suited For	Development Cycle	Testing Approach	Flexibility
Hybrid Iterative Development (Bundle-Based)	Large-scale, multi-year MES implementations; complex multi-module projects; extensive stakeholder dependencies (e.g., MECM)	Bundle-based incremental (Design→Develop→SIT→Exploration cycles)	Per-bundle Exploration + full End-to-End SIT + UAT	High: Continuous stakeholder engagement per bundle
Spiral	High-risk modernization; uncertain or evolving requirements; complex multi-vendor integration; projects where technical risk must drive iteration scope	Risk-driven iterative cycles (Planning → Risk Analysis → Engineering → Evaluation)	Per-cycle testing + End-to-End SIT + UAT	High: Risk analysis determines scope of each cycle

Table 14 below provides a side-by-side comparison of all five supported SDLC methodologies across key delivery dimensions to support planning and governance decisions.

Table 14 – SDLC Methodology Comparison

Dimension	Waterfall	Agile Scrum	V-Shaped	Hybrid: Waterfall + Scrum	Hybrid: Waterfall + Iterative	Hybrid Iterative	Spiral
Planning Horizon	Full project upfront	Sprint (2–4 wks)	Full project upfront	Release + Sprint	Release + Design iteration	Phase + Bundle	Per-cycle; cycle count determined by risk profile
Requirements	Comprehensive upfront	Progressive (backlog)	Comprehensive upfront; locked per phase	Upfront NFR + progressive	Upfront + iterative design gates	Phased scope prioritization	Progressive; elaborated per cycle based on risk analysis
Development Unit	Phase	Sprint	Phase with bilateral test gate	Sprint within Release	Waterfall phase	Bundle (Design→Dev→SIT→Exploration)	Spiral cycle (Planning → Risk Analysis → Engineering → Evaluation)

Dimension	Waterfall	Agile Scrum	V-Shaped	Hybrid: Waterfall + Scrum	Hybrid: Waterfall + Iterative	Hybrid Iterative	Spiral
Key Governance Gate	Phase sign-off	Sprint Review	Phase entry sign-off + test strategy approval	Release Go/No-Go	OHIP design approval	Bundle close + End-to-End SIT	Risk review + Cycle Evaluation with OHIP
User Validation	End-of-phase UAT	Per-sprint demos	Phase-end testing; UAT at final phase	Per-sprint + UAT	Per-iteration demo + UAT	Solution Exploration per bundle + full UAT	Per-cycle Evaluation + full UAT
Stakeholder Feedback	Milestone-based	Continuous	Milestone-based	Continuous	Design iteration	Per-bundle structured	Periodic (cycle Evaluation reviews)
Risk Management	Upfront planning	Continuous (sprint retro)	Upfront planning; test-per-phase verification reduces late defect risk	Sprint retro + release planning	Design phase gates	Monitor & Control throughout all phases	Primary driver—formal risk analysis gate at each cycle
CMS SMC Evidence	Release-based artifacts	Sprint-based evidence	Phase-based verification artifacts; strong traceability for certification	Release-based	Release-based	Iterative evidence + release submission	Per-cycle risk and evaluation artifacts + release submission
Preferred Project Size	Small–Medium	Any	Small–Medium; stable scope; COTS or infrastructure	Medium–Large	Large	Large-scale multi-year	Medium to Large; high-risk or uncertain scope

Table 15 – SDLC Model Quick-Reference Comparison

Methodology	Flexibility to Implement Changes	MVP Delivery Speed	OHIP Involvement	Risk Management	Documentation Complexity	CMS Certification Readiness
Waterfall	Low	Low	Low	Low	High	High
V-Model	Very Low	Low	Low	Medium	Very High	Very High
Agile Scrum	High	High	High	Medium	Medium	Medium
Spiral	Very High	Medium	High	Very High	High	High

Methodology	Flexibility to Implement Changes	MVP Delivery Speed	OHIP Involvement	Risk Management	Documentation Complexity	CMS Certification Readiness
Hybrid: Waterfall + Scrum	Medium	Medium	High	Medium	High	High
Hybrid: Waterfall + Iterative Design	Medium	Low-Medium	Medium-High	Medium	High	High
Hybrid Iterative Development	High	Medium	Very High	High	High	High
Kanban (M&O Only)	Very High	Medium-High	High	Low-Medium	Low	Low

How to Read This Table

- **Flexibility to Implement Changes:** How easily the methodology accommodates scope, design, or requirements changes after work has begun. Higher is more flexible.
- **MVP Delivery Speed:** How quickly the methodology delivers a usable increment of working functionality to stakeholders. Higher means faster initial delivery.
- **OHIP Involvement:** The level and frequency of OHIP stakeholder engagement required throughout the life cycle. Higher means more frequent touchpoints and feedback loops.
- **Risk Management:** How proactively the methodology identifies, assesses, and mitigates project and technical risks. Higher means risk is addressed earlier and more systematically.
- **Documentation Complexity:** The volume and formality of documentation artifacts produced. Higher means more extensive documentation, which supports audit and compliance but requires more effort.
- **CMS Certification Readiness:** How naturally the methodology generates the evidence artifacts required for CMS SMC. Higher means certification evidence is produced as a byproduct of normal methodology execution.

Table 16 below provides a decision framework for methodology selection based on project characteristics. When multiple methodologies appear suitable, the Contractor **SHOULD** evaluate the project against the factors below and propose the methodology with the strongest alignment. Hybrid approaches that combine elements from multiple methodologies are encouraged when a single methodology does not fully address the project's needs.

Table 16 – SDLC Methodology Decision Framework

Project Factor	Waterfall / V-Shaped	Agile Scrum	Spiral	Hybrid Approaches	DevOps Practices
Project Size	Small to Medium	Small to Large	Medium to Large	Medium to Large	Any (applied as layer)
Complexity	Low to Medium	Low to High	High	Medium to High	Any

Project Factor	Waterfall / V-Shaped	Agile Scrum	Spiral	Hybrid Approaches	DevOps Practices
Requirements Stability	Stable, well-defined	Evolving, likely to change	Uncertain, evolving with risk	Mixed: Stable architecture, evolving features	Any
Risk Tolerance	Low: Predictable outcomes needed	Moderate: Sprint-by-sprint visibility	High: Risk analysis drives each iteration	Moderate to High	Moderate: Automation reduces deployment risk
Client / OHIP Involvement	Phase milestones	Continuous (sprint demos, backlog prioritization)	Periodic (risk review gates)	Continuous to periodic depending on approach	Continuous (monitoring, feedback loops)
Regulatory / CMS Certification	Strong: Clear phase-gate evidence	Moderate: Sprint evidence requires aggregation	Strong: Risk documentation supports audit	Strong: Governance gates at each phase/iteration	Supports: Automated evidence collection
Time-to-Market	Longer: Sequential delivery	Faster: Incremental delivery	Variable: Depends on risk complexity	Moderate to Fast	Faster: Automated pipelines
OHIP Example Use Cases	COTS implementation, regulatory reporting, fixed-scope contracts	User-facing applications, innovation features, short delivery cycles	High-risk modernization, uncertain integration scope, complex vendor coordination	MECM (Hybrid Iterative), mixed NFR/functional scope (Waterfall + Scrum)	Complement to any SDLC for CI/CD, automated testing, infrastructure management

8.4.1 Overarching Requirements

Architecture governance applies across all methodologies. Before commencing development activities, the Contractor obtains OHIP Enterprise Architecture approval of the solution architecture artifacts defined in [Section 9.0](#), and routes significant architecture changes occurring after that approval through OHIP's established enterprise architecture governance and change control processes prior to implementation. Both the initial approval and subsequent significant architecture changes are reviewed through the OHIP Enterprise Architecture Review Board (EARB), described in [Section 9.7](#).

The Contractor **MUST** adhere to the following requirements, which apply to all SDLC methodologies:

- The SDLC methodology **MUST** be thoroughly documented, detailing key processes, and deliverables.
- The SDLC Governance Plan **MUST** also identify how the project's SDLC change control procedures interface with the release-level change control process ([Section 7.6](#)) and, where database changes are in scope, the database change control process governed by the OHIP Data and Information Domain Document, [Section 4.5.3](#).
- OHIP **MUST** approve all methodologies, including any tools utilized within the SDLC.

- Proof of concept or prototype demonstrations and solution exploration activities **MUST** occur monthly or as determined by OHIP.
- Commitment and resource requirements from OHIP staff **MUST** be clearly identified and approved before project initiation.
- OHIP staff **MUST** have access to all SDLC-produced artifacts.
- Any proposed changes to existing solution architecture **MUST** follow SDLC change control processes and be approved by OHIP.
- The Contractor **MUST** develop and maintain an SDLC Governance Plan, approved by OHIP, documenting: the selected methodology; the project's approach to implementing this standard; phase gate criteria; SDLC change control procedures; and the CMS SMC evidence collection approach. The plan **MUST** be reviewed and updated at each major phase transition.

Any proposed deviation from the requirements of this standard, including methodology changes, phase gate bypasses, or deliverable modifications, **MUST** be submitted as a formal SDR to OHIP. The SDR **MUST** include: a description of the deviation, business justification, risk assessment, proposed compensating controls, and the anticipated duration. OHIP approval **MUST** be obtained before the deviation is implemented.

8.4.1.1 DevOps Practices Within SDLC Methodologies

DevOps practices, including Continuous Integration (CI), Continuous Delivery (CD), infrastructure as code, automated testing pipelines, and automated deployment, represent a delivery acceleration and quality improvement layer that may be applied within any OHIP-approved SDLC methodology. DevOps is not a standalone SDLC methodology and does not replace the structured phase gates, governance reviews, and OHIP approval requirements defined in this standard. Rather, DevOps practices complement the selected SDLC methodology by automating repetitive tasks, reducing deployment risk through pipeline automation, and enabling faster feedback cycles.

The Contractor **MAY** propose the adoption of DevOps practices as part of their SDLC Governance Plan. Any such proposal **MUST** include:

- A description of the specific DevOps practices to be implemented (e.g., CI/CD pipeline configuration, automated testing integration, infrastructure as code provisioning, automated deployment orchestration).
- A mapping of how DevOps automation integrates with the project's selected SDLC methodology without bypassing phase gates, OHIP approval requirements, or Go/No-Go processes defined in [Section 7.11](#).
- A description of how DevOps tooling complies with the OHIP Infrastructure and Operations Domain Document, Preferred Operational Tools Standard and the OHIP Security, Privacy, and Compliance Domain Document, Security and Privacy Compliance Standard, including access controls for automated pipelines.
- Documentation of how automated deployment processes will generate the audit trail and evidence artifacts required for CMS SMC.

All code developed, tested, and deployed through DevOps pipelines **MUST** comply with [Section 5.0](#) and [Section 6](#). CI/CD pipelines **MUST** enforce segregation of duties: developers **MUST NOT** independently approve their own code for production deployment. Production deployment approval **MUST** require independent review by a qualified reviewer who did not author the change. CI/CD administrative privileges **MUST** be restricted to authorized personnel and auditable. Automated pipelines **MUST NOT**

promote code to production without satisfying the release-level change control and Go/No-Go requirements defined in [Section 7.11](#).

The Contractor **SHOULD**:

- Conduct proof of concept or prototype demonstrations monthly, or on an OHIP-determined cadence.

8.4.2 Waterfall SDLC

The Waterfall methodology is a traditional and linear approach that progresses sequentially through development phases. Each phase is completed before the next begins, with little overlap or iteration between stages. For example, all requirements are defined before solution design, and development is completed before testing, etc. Each step requires OHIP approval before proceeding.

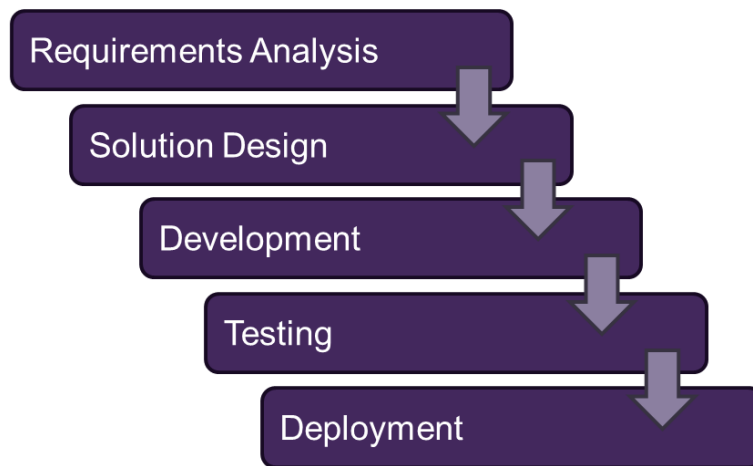


Figure 3 – Waterfall SDLC

Figure 3 outlines industry-accepted phases for a Waterfall SDLC. The subsequent sections provide a high-level understanding of common activities occurring in each phase.

8.4.2.1 Requirements Analysis Planning

 **Cross-Reference:** Requirements management processes, including the Requirements Management Plan and RTM, are fully governed by the OHIP Requirements Management Standard ([Section 3.0](#) of the Solution Delivery domain). This section defines requirements activities specific to the Waterfall methodology.

In the Requirements Analysis phase, the Contractor **MUST**:

- Gather and define the complete set of detailed requirements for the intended solution.
- Conduct Work Sessions with end-users to review, solicit feedback, validate, and document detailed requirements. During these sessions the Contractor **MUST** obtain validation of both functional and NFRs, in addition to the elaboration of requirements, identification of new detailed requirements, and other details necessary for development.

- Document detailed requirements within the RTM³⁴ based on results from the Work Sessions.
- Obtain OHIP approval on the validated requirements within the RTM.
- Use the validated detailed requirements as a fixed point of reference for future phases, such as solution design, development, testing, and implementation.

Deliverables: Requirements Management Plan and RTM³⁵

8.4.2.2 Solution Design

In this phase, the solution's architecture and design are defined based on the requirements gathered in the previous phase. The outputs of this phase serve as a blueprint for building the solution. During this phase, the Contractor **MUST** develop and perform the following:

- Architectural Design: Define the overall solution architecture, including technology stack, solution components, and their interactions. This activity **MUST** include creating diagrams to depict how different components of the solution interact, providing a high-level structure of the solution.
- Interface Design: Design user interfaces and interactions between system components. This work **MUST** inform the Interface Design Document. For user interfaces, this **MUST** include defining the layout and navigation of the solution from a user's perspective. For solution interfaces, this **MUST** include definition of communication protocols and data exchange formats between different software components or solutions.
- Batch Processing Design: Design all batch jobs and batch processes that are part of the solution, regardless of triggering mechanism (interface-triggered, scheduled, or event-driven). This activity **SHOULD** include documentation of each job's purpose and scope; execution schedule or triggering mechanism; dependencies and job sequencing; error handling, restart, and recovery procedures; and the roles and responsibilities of all stakeholders. This work **MUST** inform the Functional Design Document, which serves as the authoritative source of batch job documentation. Batch jobs triggered by an interface **MUST** also be referenced in the Integration Plan.
- Database Design: Design the database schema, including tables, relationships, constraints, and indexing strategies to ensure data is stored effectively and can be accessed efficiently. This work **MUST** also inform the Data Management Plan.
- Documentation: Prepare design documents, diagrams (such as Unified Modeling Language (UML) [UML] diagrams), and specifications that **MUST** be used in the Development phase.
- Design Review and Validation: Conduct design reviews with all relevant stakeholders to ensure the design complies with the requirements and is feasible to implement. This activity **MUST** include checking for design consistency, completeness, and adherence to requirements and standards.

³⁴ Refer to [Section 3.0](#) for more information on the Requirements Management Plan and the RTM.

³⁵ Refer to [Section 3.00](#).

Deliverables: Design documents (including Functional Design Documents) and data models.³⁶³⁷

The Contractor **SHOULD**:

- Document, within the Integration Plan, the design of all interfaces and processes involved in integrations during the Design phase, serving as the consolidated single point of reference for all integrations, including: an overview of integration points and interfaces; an interface catalog with summary characteristics; the detailed interface specification for each interface; batch control processes triggered by an interface (purpose, scope, roles, and responsibilities), with full batch job detail maintained in the Functional Design Document; and assumptions, constraints, and dependencies that may impact the integration design.
- Provide, within the Integration Plan, the detailed record and technical specification of each interface/integration, including: a description of the interface/integration (e.g., federal, state, county, private entities); detailed requirements; the approach and processing involved; communication protocols; endpoint configurations to connect to partner systems; orchestration details; the test schedule and cadence; and error handling.

8.4.2.3 Development

This phase involves translating the detailed design documentation into source code. Developers write, debug, and test the code to build the solution. All coding **MUST** be done according to the corresponding, approved detailed design documents and in accordance with [Section 5.0](#). During this phase, the Contractor **MUST**:

- **Code Implementation:** Write the code in OHIP's-designated repository based on design specifications and follow coding standards for programming language, tools, and environments.
- **Code Review and Refinement:** Identify relevant stakeholders and perform peer reviews with stakeholders to check code quality, compliance with standards, and proper implementation of requirements. This collaborative review process helps identify and correct issues early in the Development phase to ensure the code is free of errors.
- **Oversight Review:** Review to ensure code is in compliance with OHIP MES Standards.

The Contractor **SHOULD**:

- **Version Control Management:** Adhere to OHIP's version control requirements to manage and track changes to the code.
- **Documentation:** Document the code, including inline comments and developer guide documentation.
- **Integration:** As development progresses, individual modules are integrated to form larger components or subsystems. The Contractor **SHOULD** focus on ensuring that different parts of the solution work together as intended, often using integration tests to validate interoperability.

³⁶ Refer to the OHIP Business and Technical Architecture Standard for more information on Functional Design Documents and Technical Design Documents.

³⁷ Refer to the OHIP Data and Information Management Domain Document, Data Architecture and Naming Standard for more information on data models.

- **Data Migration:** Plan, select tools, and execute migration of data from legacy systems to new systems³⁸ in accordance with the OHIP Data and Information Management Domain Document, Data Conversion Standard, which governs source-to-target mapping, data cleansing strategy, reconciliation, rollback procedures, and post-conversion data retention. The Data Conversion and Migration Plan required as a deliverable of this phase **MUST** demonstrate alignment with OHIP Data and Information Management Domain Document, Data Conversion Standard.

Deliverables: Source Code and Data Conversion and Migration Plan³⁹

8.4.2.4 Testing

 **Cross-Reference:** Testing requirements in the Waterfall SDLC are fully governed by the OHIP Test Management Standard ([Section 6.0](#) of the Solution Delivery domain). The obligations in this section represent the SDLC-layer entry and exit conditions. Duplication between this Standard and the Test Management Standard is intentional—each standard governs its own layer of the life cycle.

For the testing phase, the Contractor **MUST** follow the requirements in the OHIP Test Management Standard. Testing in the Waterfall methodology is multi-faceted and plays a crucial role in ensuring that the developed solution complies with specified requirements, functions correctly, and that defects are resolved to DOH's approval. During this phase, the Contractor **MUST** develop and perform the following:

- **Verify and validate the Solution:** Ensure it has been built according to design specifications and fulfills the intended business requirements and user needs, confirming that the completed product aligns with defined expectations.
- **Perform Quality Assurance (QA) Activities:** Conduct systematic testing to identify and resolve defects and bugs to DOH's approval before the solution goes live, thereby improving the overall quality and reliability of the final product.
- **Mitigate Risks:** Identify and resolve issues and defects early in the testing process, reducing the likelihood of system failures or operational disruptions and enabling corrective actions prior to deployment.

By the end of the Testing phase, the Contractor **MUST** have a fully validated, high-quality, and deployable product that is ready for the deployment phase, thereby minimizing the risk of post-deployment issues.

Deliverables: Test Management Plan and Test Results Report⁴⁰

8.4.2.5 Deployment

 **Cross-Reference:** Deployment and Release requirements are fully governed by the OHIP Release Management Standard ([Section 7.0](#) of the Solution Delivery domain). This section defines the SDLC-layer deployment activities and their relationship to the Hypercare period described in [Section 8.5.2](#).

³⁸ For more information, refer to the OHIP Data and Information Management Domain Document.

³⁹ Refer to the OHIP Data and Information Management Domain Document, Data Conversion Standard for more information on the Data Conversion and Migration Plan.

⁴⁰ Refer to [Section 6.0](#) for definitions of the Test Management Plan and Test Results Report.

The Contractor **MUST** follow the requirements in [Section 7.0](#) when deploying into a Production environment. In the Deployment phase, the tested solution is transitioned from the Development environment to the Production environment to be used by end-users. Deployment planning includes scheduling the rollout, defining roles, identifying potential risks, and preparing contingency plans to address any issues that might occur during the transition. During the Deployment phase for a Waterfall SDLC, the Contractor **MUST** develop and perform the following:

- **Deployment Planning:** Develop a detailed Deployment Plan that includes the necessary steps for approval, installation, changes, and configuration. The Deployment Plan must include defining roles and responsibilities, identifying potential risks, and developing contingency plans to address any issues that may arise during this phase.
- **Deployment Procedure Reviews:** Practice the deployment in a controlled Pre-Production or Staging environment.
- **Execution:** Complete the installation and configuration of the solution in the Production environment.
- **Configuration Management:** Manage configuration settings and environment variables.
- **Verification and Validation:** Perform final checks to ensure the solution is correctly functioning, configured, and complies with CMS certification requirements. These final checks take place upon deployment to the Production environment.
- **Documentation:** Prepare deployment guides, installation manuals, and user documentation continuously throughout the phase.

Deliverables: Deployment Plan, Hypercare Plan, and Deployment Procedure Review Results

In summary, a Waterfall Methodology's structured approach allows for systemic progression through well-defined phases, each built upon the success of its predecessor. Once the Deployment process is complete, the Contractor **MUST** provide support during the Hypercare period ([Section 8.5.2](#)), which includes supporting operations in the new Release and executing the transition to the M&O team(s).

8.4.2.6 Waterfall Methodology Roles and Responsibilities

The success of a solution project relies heavily on the coordination and contributions of a diverse team whose role brings unique experience and perspectives to ensure the project maintains its trajectory from start to finish. Table 17 summarizes several key role examples in the Waterfall Methodology.

 **Note:** The OHIP Product Owner role defined below is consistent across all SDLC methodologies. See also [Section 8.4.3.10](#) (Agile roles) and [Section 8.4.4.3](#) (Hybrid Iterative roles) for methodology-specific role definitions.

Table 17 – Waterfall Methodology Roles and Responsibilities

Role	Brief Description	Key Responsibilities
Project Manager	The Project Manager is responsible for ensuring that cost, scope, quality, and time are managed appropriately.	<ul style="list-style-type: none"> • Develop the project plan to ensure the project aligns with strategic business goals and remains in scope • Facilitate communication to manage obstacles and dependencies with all relevant stakeholders • Support tracking progress against the project plan to identify and manage risks and issues, implementing actions when necessary • Provide regular updates to relevant stakeholders • Divide project tasks among team members
OHIP Product Owner	The Business Owner is responsible for maximizing the value of the solution. They understand the business needs and consumer expectations and approve the solution when it meets those expectations.	<ul style="list-style-type: none"> • Collaborate with the project team to define and prioritize business requirements that align with organizational objectives • Facilitate communication with other business stakeholders to translate activities • Ensure the project delivers business value and achieves anticipated benefits upon implementation • Provide feedback during testing
Project Team	The Project Team consists of developers and testers who work together to deliver the product. They are self-organizing and cross-functional, meaning they have all the skills necessary to create the product.	<ul style="list-style-type: none"> • Participate in project planning and goal setting • Collaborate to design, develop, and test the product • Ensure quality through testing and quality assurance activities to comply with requirements • Adapt to changes and continuously improve processes

8.4.2.7 V-Shaped Methodology (Verification and Validation)

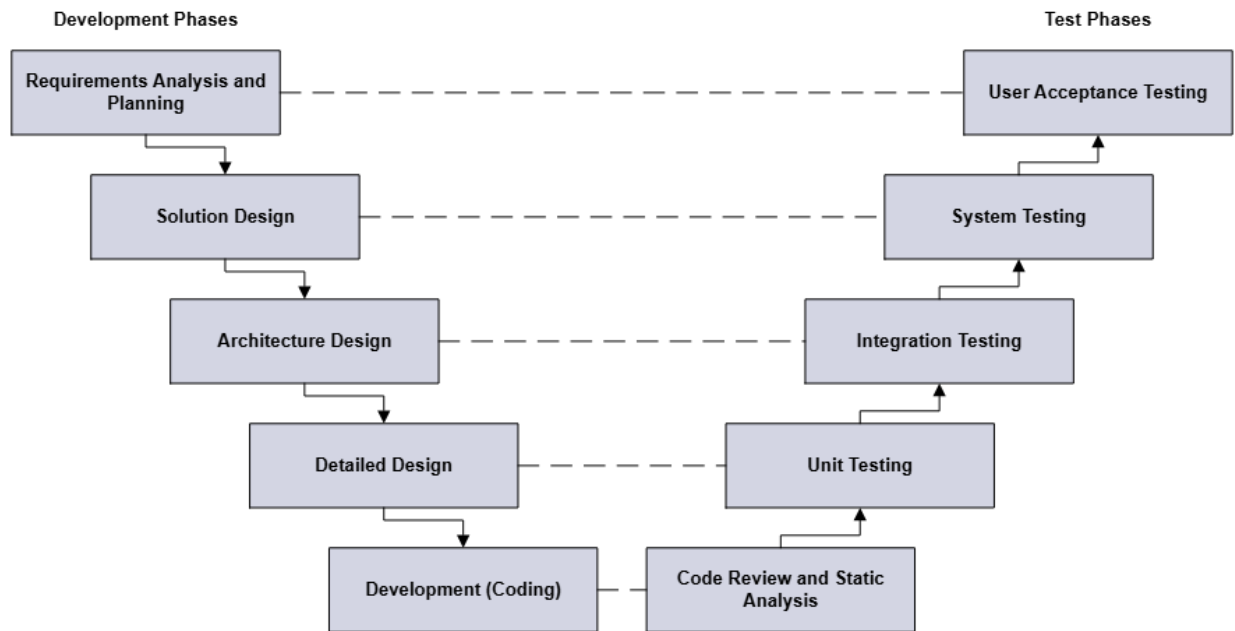


Figure 4 – V-Shaped Methodology

The V-Shaped methodology is a structured variant of the Waterfall SDLC in which every development phase on the left axis of the V has a directly corresponding, pre-defined test phase on the right axis. Rather than treating testing as a downstream activity, V-Shaped requires that the test strategy, test cases, and acceptance criteria for each phase be fully designed before development of that phase begins. This bilateral structure ensures that requirements, architecture, design, and code are each verified against their phase-specific entry criteria before progression and validated against business outcomes at the completion of the corresponding test phase.

The V-Shaped model is best suited for OHIP MES projects with stable, fully defined requirements, mandatory phase-by-phase verification evidence for CMS SMC, and a defined test qualification path from unit testing through acceptance testing before any phase can be closed. It is particularly appropriate for COTS implementations, infrastructure modernization projects, and integration efforts where the architecture is fixed, and scope is not expected to change during delivery.

The phase correspondence structure under V-Shaped is as follows:

Table 18 – V-Shaped Phase Correspondence Structure

Development Phase	Corresponding Test Phase
Requirements Analysis & Planning	UAT
Solution Design	System Testing
Architecture Design	Integration Testing
Detailed Design	Unit Testing
Development (Coding)	Code Review and Static Analysis

When the V-Shaped methodology is selected, the Contractor **MUST**:

- Follow all requirements defined in [Section 8.4.2](#) (Waterfall SDLC) of this Standard, which governs all phase-gated, sequential development activities.
- Design and document the full test strategy, test cases, and acceptance criteria for each development phase before that phase's development work commences. Test artifacts for each phase **MUST** be submitted to OHIP for review and approval as part of the phase entry gate.
- Maintain bilateral traceability in the RTM, linking each requirement forward to its development artifact and backward to its corresponding test case and test result, throughout all phases.
- Resolve all test findings from a given verification phase before the corresponding development phase is considered closed and before the next development phase begins.

Deliverables: All deliverables specified in [Section 8.4.2](#) (Waterfall SDLC), plus a V-Shaped Test Strategy Document submitted at project initiation that maps every development phase to its corresponding test phase, test types, entry/exit criteria, and the responsible testing team.

8.4.3 Agile Scrum SDLC

The Agile Scrum SDLC is characterized by repeated cycles of requirements validation through user stories, followed by small cycles of development (coding), testing, and releases, known as Sprints. Sprints continue until the end-product is complete (refer to Figure 5, below.) This methodology relies on strong communication, from both the Contractor and OHIP, to ensure that all stakeholders are aligned and that any issues are promptly addressed. This iterative development process continues until the solution is complete and ready for deployment, keeping the end-user's needs at the forefront and allowing for changes in response to feedback or shifting priorities.

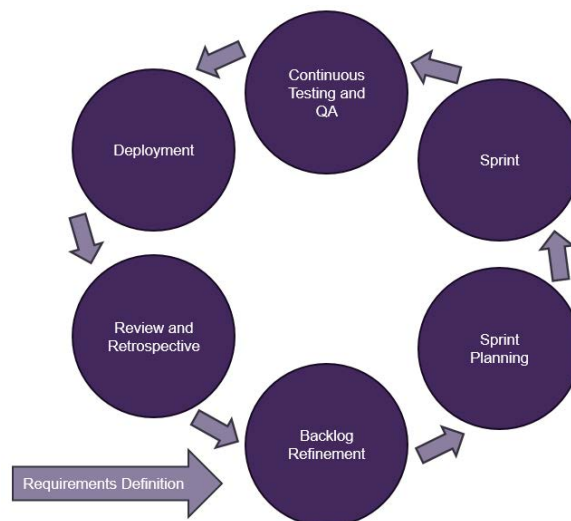


Figure 5 – Agile Scrum SDLC

8.4.3.1 Requirements Definition

 **Cross-Reference:** The OHIP Requirements Management Standard ([Section 3.0](#)) governs the overall requirements framework, including RTM maintenance and OHIP approval processes. Agile-

specific requirements artifacts (Product Backlog, User Stories, Sprint Backlog) defined in this section supplement that framework. RTM obligations defined in [Section 3.0](#) apply regardless of methodology.

In the Requirements Definition phase, the Contractor **MUST**:

- Adhere to [Section 3.0](#).
- Understand and articulate the needs and expectations of users and stakeholders.
- Identify key stakeholders, including the Product Owner, sponsors, and end-users and validate stakeholder engagement throughout the project.
- Define roles and responsibilities, particularly the Product Owner's role in prioritizing requirements and making decisions.
- Capture high-level requirements, breaking them down into user stories or features that can be prioritized in the Product Backlog.
- Continually refine the Product Backlog to ensure that the evolving solution aligns with business goals and user needs.

Product Backlog: Development and Ongoing Maintenance

The Contractor **MUST** maintain a Product Backlog that is a prioritized list of all the work (such as new features, bug fixes, and technical tasks), known to be needed in the product and serves as the single source of requirements for any changes to the product. The Product Backlog is dynamic and evolves as insights are gained and conditions develop. Product Backlog elements **MUST** include:

- **Epic:** Describes a large body of work including many features that will typically span multiple releases. An Epic represents a container to capture multiple features necessary to enable a significant capability.
- **Feature:** Describes a coherent chunk of functionality that produces a tangible business outcome. Features are comprised of multiple user stories and are implemented within a single Release.
- **User Story:** Expresses the value that a business persona needs to derive from the product, as discussed below.

User stories are short, simple descriptions of a feature, told from the perspective of the user who wants new functionality. A user story follows the template: *As a [type of user], I want [an action] so that [a benefit/a value]*. The following components are required for a user story:

- **Persona:** Identifies the type of user who will benefit from the feature, encouraging empathy by focusing on potential users.
- **Action:** Describes a business function or activity that **MUST** be performed, or an observable user experience that the system **MUST** offer.
- **Benefit/Value:** Explains why the user requests the feature, focusing on the value or benefit that will result from fulfilling this need and helps the team understand the purpose and importance of the story.

Table 19 below provides a Product Backlog template that organizes the attributes for each item.

Table 19 – Product Backlog Template

ID	User Story	Priority	Estimation	Acceptance Criteria	Description	Status
001	As a user, I want to...	High	8 SP	Given..., When..., Then...	Context info	Ready
002	As an administrator, I can...	Medium	5 SP	Given..., When..., Then...	Context info	Pending

User Story Requirements

The Contractor **MUST** create effective user stories by adhering to established Agile best practices and criteria. At minimum, the Contractor **MUST**:

- Keep user stories concise while providing sufficient context for development and planning.
- Ensure each user story articulates clear user or business value.
- Use user stories as a mechanism for productive discussion and refinement during team collaboration.
- Reassess user stories as part of backlog refinement to maintain quality and relevance.
- Meet quality criteria, including SMART, INVEST, and Definition of Done (DoD) to support clarity, feasibility, and alignment with project objectives.⁴¹

Deliverables: Product Backlog Report

8.4.3.2 Backlog Refinement

The Contractor **MUST** work with the Product Owner to implement backlog refinement as a continuous process for maintaining healthy and actionable backlogs. The Product Owner may choose to enhance user story development by encouraging participation from business stakeholders in the following ways:

- Inviting business stakeholders to backlog refinement sessions.
- Orchestrating workshops and ad/hoc sessions to gather direct business stakeholder input so user stories reflect the nuances of a full range of business scenarios and customer journeys.
- Maintaining on-going business stakeholder/Product Owner collaborations.

The Contractor **MUST** conduct Product Backlog Refinement activities that include, but are not limited to:

- **Clarification of backlog items:** Collaborative discussion of each item to refine requirements and user stories.
- **Prioritization of work:** Alignment with the Product Owner to order items based on urgency, value, stakeholder input, and strategic objectives.
- **Effort estimation:** Use of industry standard Scrum estimation techniques to refine level of effort for backlog items.
- **User story decomposition:** Breaking large user stories or epics into smaller, manageable, and actionable items that can be completed within a single Sprint.

⁴¹ Refer to Appendix B: SMART, INVEST, and Definition of Done (DoD) Criteria.

- **Addition of necessary detail:** Inclusion of context, acceptance criteria, and supporting information to ensure each user story is sufficiently defined and ready for inclusion in an upcoming Sprint.
- **Routine refinement sessions:** Regular one 1- to 2-hour sessions, conducted on an ongoing basis, to maintain a current, accurate, and prioritized Product Backlog that supports upcoming Sprints.

8.4.3.3 Release Planning

In Agile, a Release is when the results of several development iterations (called Sprints) are combined and launched into the Production environment. These iterations work together to deliver a complete business function or set of functions. The Contractor **MUST** implement well-scoped and vetted releases using the following:

- **Release Planning:** Submit a Release Plan⁴² for OHIP approval, prior to initiating a Release.
- **Sprint (0):** Each Release **MUST** commence with Sprint (0), which is dedicated as the planning phase for the entire Release. During Sprint (0), high-level business requirements for the Release are defined as user stories, development and testing environments are prepared, stakeholders are aligned, and the Roadmap to guide the Development Team through the Release is created.

Deliverables: Release Plan, and the Initial Product Backlog

8.4.3.4 Sprint Planning

For Sprint Planning, the Contractor **MUST** perform the following activities:

ID	Sprint Goal	Task ID	Task Name	Estimation	Assigned To	Status
001	Enhance user onboarding	T-001	Wireframe design	3 hours	John Doe	In Progress
001		T-002	Develop account creation	5 hours	Jane Smith	Done
002	Backend optimization	T-003	Optimize database queries	8 hours	John Doe	To-Do
002		T-004	Load testing	6 hours	Jane Smith	In Progress

Deliverables: Sprint Backlog Report

The Contractor **MUST** maintain and refine both the Product Backlog and the Sprint Backlog to ensure they remain accurate, actionable, and aligned with Agile delivery expectations and Sprint objectives. Backlog refinement **MUST** occur continuously throughout the Sprint, including during daily standups or other short, focused sessions. The Contractor **MUST** perform the following activities:

- Maintain an up-to-date Product Backlog by refining items for clarity, relevance, detail, estimates, and priority to ensure readiness for future Sprints.

⁴² Refer to the [Section 7.0](#) for more information.

- Engage stakeholders and facilitate collaborative refinement by coordinating involvement in discussions and workshops so user stories accurately reflect business needs and customer journeys.
- Decompose selected PBIs (PBIs) into actionable tasks that are well defined, understood by the team, and free of blockers.
- Update and adjust tasks, reallocate resources, or refine estimates based on actual progress and newly discovered information.
- Monitor progress throughout the Sprint by tracking task status (e.g., To Do, In Progress, Done) using appropriate tools to maintain transparency.
- Ensure alignment with the Sprint Goal by regularly reviewing remaining work and confirming that all tasks contribute to achieving the agreed-upon objective.
- Support continuous improvement by making iterative adjustments to both backlogs to enhance clarity, reduce ambiguity, and stay aligned with evolving project needs.

The Contractor **SHOULD**:

- Maintain a continuously refined Product Backlog by updating backlog items for relevance, clarity, detail, estimates, and prioritization to ensure they are ready for selection in upcoming Sprints.
- Provide a Sprint Backlog with selected PBIs, Sprint Goal, detailed tasks, and progress tracking.
- Implement continuous backlog refinement maintaining accuracy, stakeholder engagement, decomposed tasks, and Sprint Goal alignment.

Deliverable: Sprint Velocity Chart

8.4.3.5 Sprints

The Contractor **SHOULD** conduct daily standup meetings (also known as Daily Scrum Meetings) that include all team members. Daily Scrum Meetings monitor progress, identify impediments, and adjust plans as needed. The Contractor **SHOULD** ensure team members communicate what was done, what **SHOULD** be done, and any blockers to the solution during the daily standup meetings. The Contractor **SHOULD** adhere to coding and test management standards, while conducting unit testing and Continuous Integration (CI) to validate the functionality. The Contractor **SHOULD** update the RTM) at the close of each Sprint to reflect completed, in-progress, and deferred user stories and their test case linkages, in accordance with [Section 3.0](#) of this document.

Deliverables: Sprint Burndown Chart and Sprint Backlog

8.4.3.6 Continuous Testing and Quality Assurance (QA)

 **Cross-Reference:** Testing requirements in the Agile SDLC are fully governed by the OHIP Test Management Standard ([Section 6.0](#)). The mandatory quality controls, defect management requirements, and test artifact requirements defined in [Section 6.0](#) apply in full. The obligations in this section define how test activities integrate into the Sprint cadence.

The Contractor **SHOULD** follow the requirements in the OHIP Test Management Standard and implement continuous testing and Quality Assurance (QA) throughout the development process to ensure the solution meets specified requirements and functions as expected across various environments.

The technical discipline operating within each Sprint, including Test-Driven Development (TDD), BDD, peer code review, automated static analysis, dependency scanning, and CI/CD pipeline execution, is

governed by the OHIP Code Standard ([Section 5.0](#)) and the OHIP Test Management Standard ([Section 6.0](#)) of this document. Those standards define the specific thresholds, tooling, and quality gates that apply during Sprint development activity. The requirements in this section govern the testing and QA cadence, transparency, and evidence obligations specific to the Agile Scrum life cycle. Both sets of requirements apply concurrently and are complementary

To support continuous testing and ongoing quality validation, the Contractor **MUST**:

- Execute automated tests, including security oversight, unit tests, integration tests, and end to end tests where applicable, while also conducting manual and exploratory testing to validate system behavior.
- Perform automated regression testing to confirm that new changes do not negatively affect existing functionality.
- Validate each user story against its acceptance criteria to ensure quality, completeness, and readiness for acceptance.

The Contractor **SHOULD**:

- Develop and maintain automated unit tests for application components, meeting the coverage thresholds defined in [Section 5.0](#) (Code Standard), to support early defect identification. TDD **SHOULD** be applied where appropriate.
- Apply Behavior Driven Development (BDD) when appropriate to ensure alignment among developers, QA personnel, and other stakeholders on expected system behaviors, using a shared language (e.g., Gherkin) to define acceptance criteria prior to development.

8.4.3.7 Review

On a cadence agreed upon with OHIP, usually after each Sprint, the Contractor **MUST** conduct a collaborative review session with business stakeholders to demonstrate completed functionalities. The Contractor **MUST** collect feedback from stakeholders to support the Product Backlog refinement and reprioritization efforts. In this activity, the Contractor **MUST** ensure Sprint Reviews have:

- On-going alignment between the solution being built and the necessary business functionality and user experiences required.
- The agreed-upon acceptance criteria for each product increment.
- Where the solution is pursuing CMS SMC, the Contractor **MUST** retain Sprint Review demonstration materials, including recordings, screenshots, and documented outcomes, as required CMS certification evidence, organized and accessible for CMS review sessions.
- To minimize administrative overhead and support Agile velocity, the Contractor **MUST** establish an automated or semi-automated evidence collection approach. Sprint Review artifacts **MUST** be indexed and linked directly within the work item record (e.g., Jira ticket or DevOps user story) at Sprint close, rather than assembled manually as a separate project management task. The Contractor **MUST** document the evidence indexing approach in the Test Management Plan for OHIP approval prior to the first Sprint Review, and **MUST** ensure all indexed evidence is accessible and retrievable for CMS review sessions without requiring manual compilation.

8.4.3.8 Retrospective

The Contractor **SHOULD** close out each Sprint with a Retrospective session to assess the team's performance, processes, and collaboration by identifying what went well, what did not, and areas for

improvement. The Contractor **SHOULD** develop action items and strategies for improving future Sprints to support continuous improvement ahead of the next Sprint.

8.4.3.9 Deployment

 **Cross-Reference:** Deployment and Release requirements are fully governed by the OHIP Release Management Standard ([Section 7.0](#) of the Solution Delivery domain). This section defines how deployment activities integrate into the Agile delivery life cycle and their relationship to the Hypercare period described in [Section 8.5.2](#).

Once the Sprint is complete, the Contractor **MUST**:

The Contractor **SHOULD**:

- Follow the requirements within the OHIP Release Management Standard.
- Prepare the deployment package, which includes the final code, configuration files, and other related documentation, by ensuring that all tests are passed, acceptance criteria are met, and any defects are resolved to DOH's approval.⁴³
- Provide the necessary documentation, such as release notes and user manuals prior to deployment in the Production environment.
- Where applicable, develop and execute a Data Migration and Conversion Plan as part of Agile Sprint and Release deployment activities. All data migration execution activities **MUST** comply with the OHIP Data and Information Management Domain Document, Data Conversion Standard, which is the authoritative standard governing how data migration is performed, including source-to-target mapping, cleansing, reconciliation, exception handling, rollback, and post-conversion retention. This section governs when and how migration fits within the Agile delivery life cycle; the OHIP Data and Information Management Domain Document, Data Conversion Standard governs how the migration itself is executed.

Data Migration Sequencing

Medicaid legacy data migration is complex and rarely fits within a standard 2-to-4-week Sprint cadence. Where data migration scope is significant, the Contractor **SHOULD** structure data engineering and conversion activities as a dedicated parallel workstream with its own sprint cadence, commonly referred to as "data sprints", rather than bundling migration execution inside application feature sprints. Data sprint outputs **MUST** feed into and satisfy the deployment gate criteria for the relevant application Release before production promotion is permitted. This approach **MUST** be documented in the Data Migration and Conversion Plan, approved by OHIP prior to execution, and executed in full compliance with the OHIP Data and Information Management Domain Document, Data Conversion Standard.

Deliverables: Release Notes and User Manuals

⁴³ Refer to [Section 7.0](#).

8.4.3.10 Agile Scrum Methodology Roles and Responsibilities

When utilizing an Agile Methodology, clearly defined roles and responsibilities are crucial for fostering collaboration, ensuring efficient workflows, and delivering high-quality products. Examples of key roles in an Agile methodology are summarized in Table 21 below.

 **Note:** The OHIP Product Owner role defined below is consistent across all SDLC methodologies. See also [Section 8.4.2.6](#) (Waterfall roles) and [Section 8.4.4.3](#) (Hybrid Iterative roles) for methodology-specific role definitions.

Table 20 – Agile Methodology Roles and Responsibilities

Role	Brief Description	Key Responsibilities
Scrum Master	The Scrum Master is responsible for ensuring that the Scrum team adheres to Scrum principles and practices. They act as a coach and facilitator, helping the team to stay on track and removing any obstacles that MAY hinder progress.	<ul style="list-style-type: none">• Facilitate daily Scrum meetings (also known as “daily standups”).• Lead Sprint Planning and retrospective meetings.• Manage obstacles by communicating with all relevant stakeholders.• Guide the team on Agile best practices to self-organize for greater efficiency.• Promote continuous improvement within the team.
OHIP Product Owner	The Product Owner is responsible for maximizing the value of the product that results from the Scrum team’s work. They understand the business needs and consumer expectations, and they manage the Product Backlog.	<ul style="list-style-type: none">• Support defining and communicating the product vision and strategy to all stakeholders involved.• Manage and prioritize the Product Backlog.• Liaise with stakeholders to gather feedback throughout the project and incorporate it appropriately to align with business objectives.• Ensure the team focuses on delivering value and meeting product goals by balancing business value and technical feasibility.

Role	Brief Description	Key Responsibilities
Scrum Team	The Scrum Team consists of developers and testers who work together to deliver the product increment. They are self-organizing and cross-functional, meaning they have all the skills necessary to create the product.	<ul style="list-style-type: none"> Participate in Sprint Planning and goal setting. Collaborate to design, develop, and test the product to align with the objectives and goals of the project. Ensure quality through testing and other forms of quality assurance to comply with the Definition of Done. Adapt to changes, continuously improving their processes. Ensure accurate capacity planning.

8.4.4 Hybrid SDLC Approaches

A Hybrid methodology stands as a versatile approach that combines elements from various SDLC methodologies to accommodate diverse project requirements. OHIP defines two (2) preferred Hybrid approaches: Waterfall + Scrum and Waterfall + Iterative Design. The Contractor is also free to propose alternate hybrid approaches for OHIP review and approval.

8.4.4.1 Waterfall + Scrum

HYBRID: WATERFALL + SCRUM

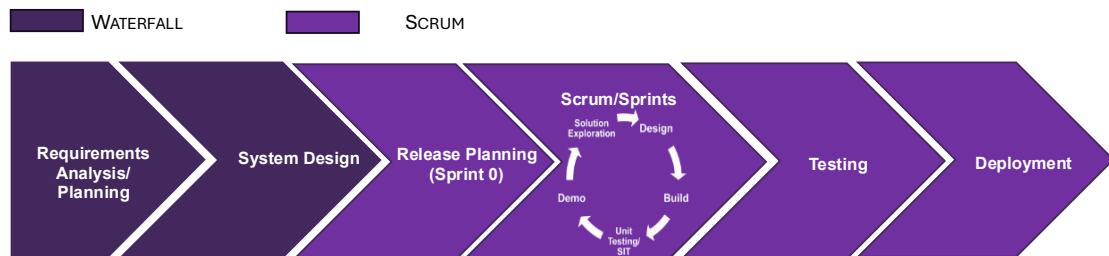


Figure 6 – Hybrid Waterfall + Scrum

The **Waterfall + Scrum** Hybrid (Figure 6) approach depicted above engages the following requirements:

- **Waterfall:** The requirements analysis/planning and Solution Design segments of the Waterfall approach are defined in [Section 8.4.2](#) (Waterfall SDLC) in this Standard to gather requirements and complete technical designs, where:
- **Requirements Analysis/Planning** follows the traditional waterfall methodology, focused on NFRs, and
- ***Error! Reference source not found.* Solution Design** focuses on articulation of NFRs design components (e.g., platform architecture, application architecture, security architecture, data architecture, interoperability, and scalability and performance design elements).

- The methodology activities provide increased end-user engagement and feedback concerning the usability and functionality of the solution being development. Using this methodology, the Contractor **MUST** conduct these activities:
 - o **Release Management:** The Contractor **SHOULD** perform an initial Release Planning (Sprint 0) phase to determine what functionality **SHOULD** be included in the Release. Within each Release, the Contractor **SHOULD** execute a full SDLC sequence. Each Release can consist of one or multiple sprints until the Release goal is achieved. Within each sprint, the Scrum team will execute Design, Build, Unit Testing, SIT, and Demo, while the Solution Exploration team is responsible for Solution Exploration testing. After the iterations of Sprints required to develop and fully vet the planned release functionality, the Release is deployed into the designated MES Production Environment.
 - o **Design, Build, Unit Testing and SIT:** The Contractor **SHOULD** conduct testing activities in cyclical, time-boxed iterations of functional design and development, where user interfaces, functionality, and outputs are designed and coded. During the sprints, the Contractor **SHOULD** conduct Refinement Sessions to identify end-users' needs, preferences, feedback, and challenges they **MAY** be currently facing and turn them into system functionality to improve their experiences. OHIP's Product Owner reviews the Backlog and is responsible for providing business direction and input in the Refinement Sessions.
 - o **Demos:** At the end of each sprint, the Contractor **SHOULD** conduct demos to showcase new functionality to the OHIP Product Owner, allowing them to provide feedback and approval.
 - o **Solution Exploration:** OHIP staff **SHOULD** be able to perform feature testing during Solution Exploration before commencing UAT. Solution Exploration feedback is limited to two (2) categories: (1) usability and design observations, which the Contractor **SHOULD** log, triage, and address within the current or next sprint; and (2) defects against approved acceptance criteria, which **SHOULD** be tracked in the defect management tool and addressed per the OHIP Defect Management Plan. New functional requests identified during Solution Exploration are explicitly out of scope for the current sprint and **SHOULD** be formally routed to the Product Backlog for OHIP Product Owner prioritization in a future sprint or release.
 - The team conducts activities within the sprint (such as Design, Build, Unit Testing, and Demos) for the User Stories in the Sprint Backlog, while simultaneously executing Solution Exploration for User Stories completed in the previous sprint and conducting requirement gathering sessions for the upcoming sprint(s).
 - o **Testing:** The Contractor **MUST** execute all required production readiness and pre-deployment testing.
 - o **Deployment:** The Contractor **MUST** conduct several critical tasks to promote the approved code package to the production environment. Once the production environment is prepared, the team deploys the code package and validates the deployment by conducting post-deployment testing.

Each Release under the Waterfall + Scrum Hybrid **SHOULD** commence with Sprint (0) as defined in [Section 8.4.3.3](#) (Release Planning) of this Standard. Sprint (0) establishes the Release planning

framework, defines high-level user stories, prepares environments, and aligns all stakeholders before iterative development begins.

Deliverables: Refer to [Section 8.4.2](#) (Waterfall SDLC) and [Section 8.4.3](#) (Agile Scrum SDLC).

8.4.4.2 Waterfall + Iterative Design

HYBRID: WATERFALL + ITERATIVE DESIGN

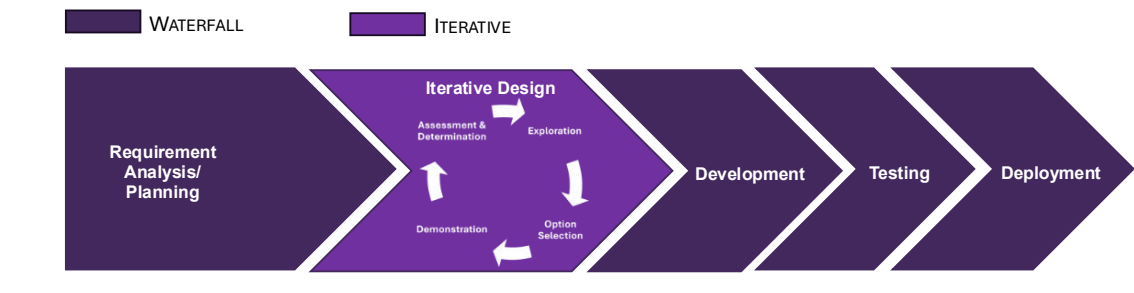


Figure 7 – Hybrid Waterfall + Iterative Approach

When the Hybrid: Waterfall + Iterative Design Methodology is chosen, where Waterfall is used for Requirement Analysis/Planning, Development, Testing, and the Deployment phase, and where the Iterative Design is an Agile approach (as shown in Figure 7), the Contractor **MUST** complete all activities required by each component of the methodology.

The Contractor **MUST** follow:

- All requirements defined in [Section 8.4.2](#) (Waterfall SDLC) for the waterfall portion of the Hybrid approach depicted in Figure 3.
- Employ an iterative, user-centered process to validate and refine the solution design before development begins. The Iterative Design Phase **MUST** be repeated as needed until OHIP stakeholders approve the design approach. The steps below indicate the high-level activities the Contractor **MUST** conduct for in this phase:
 - o **Step 1: Exploration:** Identify, provide rationale, and document multiple design alternatives that align with OHIP defined business needs and user requirements. The Exploration activity **MUST** include, and not be limited to, the following considerations: solution constraints, architectural alignment, usability expectations, and integration needs.
 - o **Step 2: Option Selection:** Upon evaluation of the design alternatives, the Contractor **MUST** recommend a best-fit option based on established criteria (i.e. feasibility, cost, complexity, and operational impact). The rationale supporting the recommended option **MUST** be clearly documented.
 - o **Step 3: Demonstration:** Present the recommended design option to OHIP stakeholders using workflows, models, or demonstrations of a configured commercial-off-the-shelf (COTS) solution. Demonstrations **MUST** clearly illustrate how the design satisfies user needs and aligns with the overall solution requirements.

- o **Step 4: Assessment and Determination:** Solicit and analyze OHIP stakeholder feedback on the demonstrated option. The Contractor **MUST** obtain OHIP approval on the design to proceed into Development. If OHIP approval is not received, the Contractor **MUST** incorporate the feedback into the next iteration, beginning again with Step 1: Exploration.

Deliverables: All deliverables indicated in [Section 8.4.2](#) (Waterfall SDLC), an Alternative Solutions Matrix

8.4.4.3 Hybrid Iterative Development (Iteration-Based) SDLC

The Hybrid Iterative Development (Iteration-Based) methodology is a structured yet iterative SDLC approach designed for large-scale, multi-year MES implementations that require both rigorous governance across the full project life cycle and incremental delivery of functionality through time-boxed Iteration cycles. This methodology is currently in use on the MECM project and represents OHIP's preferred approach for complex, multi-module implementations with extensive stakeholder dependencies and phased delivery expectations.

The methodology progresses through eleven sequential phases, Scope Prioritization, Iterative Design and Development (Requirements, Design, Develop, SIT, and Solution Exploration), End-to-End SIT, UAT, Deployment, Go-Live, and Closure, with Monitor and Control activities maintained continuously throughout all phases. The core of the methodology is the Iteration cycle, in which each Iteration begins with Requirements definition for the capabilities in scope, followed by Design, Develop, SIT, and Solution Exploration, each delivering a tested and validated increment of functionality toward the overall Release goal.

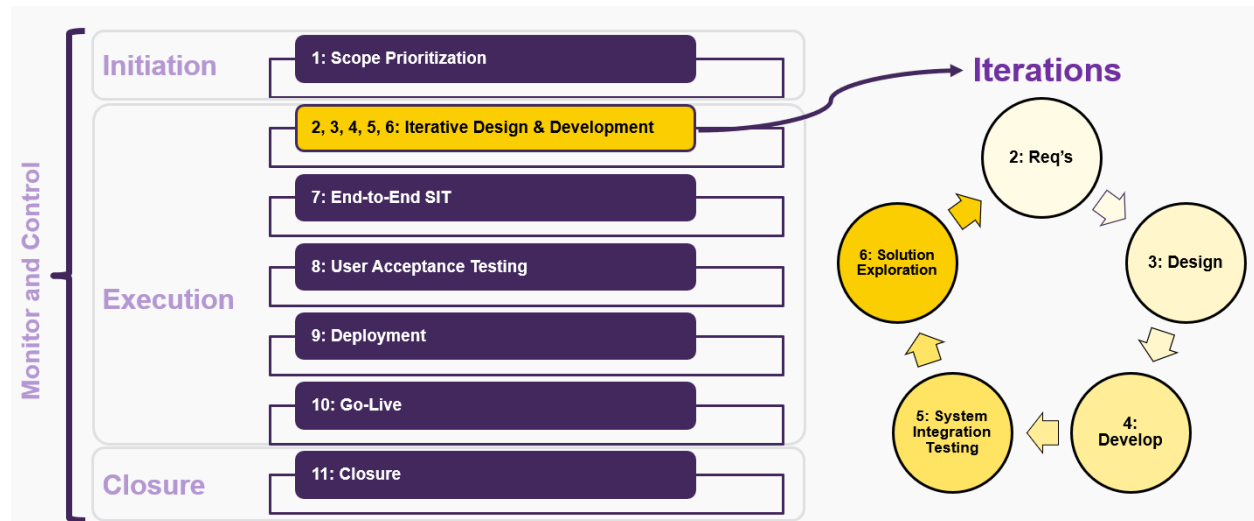


Figure 8 – Hybrid Iterative Development SDLC

Figure 8 illustrates the full 11-phase life cycle with Monitor and Control running vertically, and the Iteration cycle detail (Req's → Design → Develop → SIT → Solution Exploration) on the right.

Phase 1: Scope Prioritization (Initiation)

The Scope Prioritization phase establishes the overall project scope, identifies and prioritizes the capabilities to be delivered across all Iteration cycles, and aligns stakeholders on the delivery strategy. This phase is the foundation for all subsequent planning and execution activities.

The Contractor **MUST**:

- Conduct a formal Scope Prioritization workshop with OHIP business and technical stakeholders to define and rank all capabilities for delivery.
- Map all prioritized capabilities to their planned Iteration cycles and expected release dates.

Phases 2–6: Iterative Design and Development (Execution)

The core of the Hybrid Iterative Development methodology is the Iteration cycle. Each Iteration delivers a tested and validated increment of functionality and consists of five sequential activities: Requirements, Design, Develop, SIT, and Solution Exploration. Multiple Iteration cycles are executed, with each cycle building upon the verified output of the previous one. Requirements are elaborated at the beginning of each Iteration for the capabilities in scope for that cycle, ensuring that design and development activities are grounded in fully defined and OHIP-approved requirements.

During the Requirements and Design activities of each Iteration, the Contractor conducts Joint Application Design (JAD) sessions with OHIP and designated stakeholders to collaboratively define and validate requirements and solution design before development begins. JAD is distinct from Solution Exploration: JAD shapes the solution before it is built, whereas Solution Exploration provides OHIP structured access to verify the developed, SIT-tested functionality afterward.

Iteration Activity 1: Requirements

 **Cross-Reference:** Requirements management processes, including the Requirements Management Plan and RTM, are fully governed by the OHIP Requirements Management Standard ([Section 3.0](#) of the Solution Delivery domain). This subsection defines how requirements activities integrate into the Hybrid Iteration cadence.

Each Iteration begins with a focused requirements phase in which the Contractor elaborates detailed requirements for the capabilities assigned to that Iteration. Requirements defined during this activity must be traceable to the capabilities prioritized in Phase 1 and must be approved by OHIP before the Iteration's Design activity begins.

The Contractor **MUST**:

- Adhere to [Section 3.0](#) for all requirements activities in each Iteration.
- Elaborate and document detailed functional and NFRs for all capabilities included in the current Iteration before Design begins.
- Update the RTM to reflect all requirements baselined for the current Iteration and obtain OHIP approval before proceeding to Design.

Deliverables: Requirements Management Plan (initial Iteration only), RTM (updated per Iteration)

Iteration Activity 2: Design

The Contractor **MUST**:

- Produce functional design documents and interface specifications for all capabilities included in the Iteration before any development work begins.
- Obtain OHIP review of all design artifacts before the Iteration's Develop activity begins.

Iteration Activity 3: Develop

The Contractor **MUST**:

- Develop all Iteration capabilities in accordance with the OHIP Code Standard, completing code implementation, peer review, unit testing, and component integration.
- Update the RTM to link each developed component to its source requirements before the Iteration's SIT activity begins.

Iteration Activity 4: SIT

 **Cross-Reference:** Testing requirements are fully governed by the OHIP Test Management Standard ([Section 6.0](#) of the Solution Delivery domain). The obligations in this subsection define how testing integrates into the Hybrid Iteration cadence.

The Contractor **MUST**:

- Execute SIT for each Iteration to validate that the Iteration's developed components function correctly with all previously completed Iterations and existing system components.
- Resolve all critical and high defects identified during Iteration SIT before progressing to Solution Exploration.
- Follow [Section 6.0](#) for all SIT activities.

Iteration Activity 5: Solution Exploration

Solution Exploration provides OHIP staff with structured access to the developed and SIT-tested Iteration functionality before it is incorporated into the End-to-End SIT phase. This activity enables early defect identification and ensures ongoing alignment between delivered functionality and business expectations.

To protect Release scope integrity, Solution Exploration operates within strictly defined boundaries. Feedback gathered during Solution Exploration is limited to two (2) categories:

1. **Usability and Design Observations:** Issues relating to interface behavior, workflow clarity, labeling, or user experience which the Contractor **MUST** log, triage, and resolve within the current or subsequent Iteration cycle
2. **Defects:** Functional behaviors that deviate from approved acceptance criteria which the Contractor **MUST** log in the defect tracking tool and address per the OHIP Defect Management Plan.

New functional requests or scope additions identified during Solution Exploration are explicitly out of scope for the current Iteration and **SHOULD** be formally routed to the Product Backlog for OHIP prioritization in a future Iteration or Release. The Contractor **SHOULD** NOT begin development on any item identified during Solution Exploration unless it has been formally accepted into the Product Backlog and approved by the OHIP Product Owner. OHIP **SHOULD** designate a Solution Exploration Coordinator responsible for enforcing these boundaries during each Iteration's exploration period.

- The Contractor **MUST** provide OHIP with access to the Iteration's fully SIT-tested functionality in a dedicated Solution Exploration environment.
- The Contractor **MUST** provide high-level testing scripts and acceptance criteria to guide OHIP's Solution Exploration activities for each Iteration.

- The Contractor **MUST** track all OHIP feedback received during Solution Exploration as candidate user stories or defects in the Requirements Management tool.
- The Contractor **MUST** incorporate all OHIP feedback from Solution Exploration into the current or a subsequent Iteration before the End-to-End SIT phase begins.

Deliverables: Iteration Summary Document (per Iteration), Solution Exploration Summary (per Iteration), Updated RTM

Phase 7: End-to-End SIT (Execution)

Following the completion of all Iteration cycles, the End-to-End SIT phase validates the integrated solution across all components, interfaces, and data flows. This phase ensures that the cumulative output of all Iterations functions as a complete, coherent system.

- The Contractor **MUST** conduct End-to-End SIT after all Iteration cycles are complete, validating the integrated solution across all components and interfaces.
- The Contractor **MUST** resolve all critical and high defects identified during End-to-End SIT before UAT commences.
- The Contractor **MUST** follow all requirements in the OHIP Test Management Standard for End-to-End SIT activities.

Phase 8: UAT (Execution)

UAT follows successful End-to-End SIT and represents OHIP's formal validation of the integrated solution against business requirements and acceptance criteria prior to deployment approval.

- The Contractor **MUST** support OHIP in conducting UAT in accordance with the OHIP Test Management Standard and the OHIP Release Management Standard.
- The Contractor **MUST** not proceed to Deployment until OHIP has formally signed off on UAT completion.

Phase 9: Deployment (Execution)

 **Cross-Reference:** Deployment and Release requirements are fully governed by [Section 7.0](#). This subsection defines how deployment integrates into the Hybrid Iterative life cycle.

- The Contractor **MUST** follow all requirements in the [Section 7.0](#) for Deployment activities in the Hybrid Iterative Development methodology.
- The Contractor **MUST** follow all requirements in [Section 8.3.2](#) for post-deployment stabilization.

Phase 10: Go-Live (Execution)

- The Contractor **MUST** utilize the Go/No-Go process defined in [Section 7.0](#) prior to Go-Live.
- The Contractor **MUST** submit a Release Implementation Completion Report following Go-Live in accordance with [Section 7.0](#).

Phase 11: Closure

The Closure phase formally concludes the project, captures lessons learned, confirms all deliverables have been accepted, releases project resources, and transitions the solution to the M&O team.

- The Contractor **MUST** conduct a formal Closure phase that includes: lessons learned documentation, final RTM baseline confirmation, formal release of project resources, and transition to the M&O team.
- The Contractor **MUST** submit a Closure Report to OHIP for review and approval documenting the outcomes of all project phases, any open items, and the transition plan.

Deliverables: Lessons Learned Report, Closure Report, Final RTM Baseline

Monitor and Control (All Phases)

Monitor and Control activities run continuously throughout all eleven phases of the Hybrid Iterative Development methodology. These activities provide OHIP with consistent visibility into project health, risks, issues, and schedule adherence.

- The Contractor **MUST** maintain Monitor and Control activities throughout all phases, encompassing: schedule tracking against the approved project baseline, risk and issue identification and management, SDLC change control for any scope or design changes, and regular structured status reporting to OHIP.
- The Contractor **MUST** submit Monitor and Control status reports to OHIP on a cadence defined at project initiation, including at minimum: schedule health, active risks and issues with mitigations, Iteration status, and any proposed changes to the approved plan.

Deliverables: RAID Log

Hybrid Iterative Development Roles and Responsibilities

The Hybrid Iterative Development methodology requires clearly defined roles across both the project governance layer and the Iteration execution layer. The following table summarizes key roles; additional roles should be defined in the project's approved SDLC documentation.

 *Note: The OHIP Product Owner role defined below is consistent across all SDLC methodologies. See also [Section 8.4.2.6](#) (Waterfall roles) and [Section 8.4.3.10](#) (Agile roles) for methodology-specific role definitions.*

Table 21 – Hybrid Iterative Development Roles and Responsibilities

Role	Brief Description	Key Responsibilities
Delivery Lead	Overall accountability for the Hybrid Iterative Development project on behalf of the Contractor.	Own the project schedule, Iteration plan, and Monitor and Control reporting. Serve as primary escalation point to OHIP PMO. Ensure all phase gates are met before proceeding.
OHIP Product Owner	OHIP representative responsible for business direction and acceptance decisions.	Prioritize capabilities across Iterations. Participate in Solution Exploration and provide structured feedback. Approve Iteration close artifacts and phase sign-offs.

Business Analyst	Responsible for requirements definition, RTM maintenance, and stakeholder alignment.	Develop and maintain the RTM across all Iteration cycles. Facilitate Solution Exploration and track feedback as candidate user stories or defects.
Development Lead	Technical lead responsible for all Design and Develop Iteration activities.	Ensure code quality, peer review completion, and unit test coverage per the OHIP Code Standard. Own the Build activity output for each Iteration.
QA Lead	Responsible for all Iteration SIT, End-to-End SIT, and UAT support activities.	Develop and execute test plans for each Iteration SIT cycle and End-to-End SIT. Follow the OHIP Test Management Standard throughout. Report defect status at each Iteration gate.
Solution Exploration Coordinator	Coordinates OHIP Solution Exploration activities within each Iteration.	Prepare testing scripts and environment access for OHIP. Facilitate OHIP feedback sessions. Track and triage all feedback items for incorporation into the current or subsequent Iteration.
Project Manager	Day-to-day schedule, risk, and issue management.	Maintain Monitor and Control reporting. Manage the Risk and Issue Register. Coordinate cross-team dependencies and escalate to the Delivery Lead as needed.

Deliverables: Iteration Summary Document (per Iteration), Lessons Learned Register

The Contractor **SHOULD**:

- Obtain OHIP acceptance of the Iteration's functionality at the conclusion of Solution Exploration before that functionality advances to the End-to-End SIT phase.

8.4.4.4 Proposing an Alternative Methodology

The methodology used for any solution items not specified in this Standard **MUST** be agreed upon by OHIP and documented prior to the Contractor starting any work.

- The Contractor **MUST** submit a written Alternate Methodology Proposal to OHIP that includes: a description of the proposed methodology; justification for why none of the methodologies defined in [Sections 8.4.2–8.4.4](#) are suitable; a mapping of the proposed methodology's phases to the deliverables in [Appendix A](#); and an assessment of any CMS SMC certification implications.
- The Contractor **MUST** obtain written OHIP approval of the Alternate Methodology Proposal before commencing any project activities under the alternate methodology.

OHIP **SHOULD** consider the proposed methodology's alignment with CMS MES Testing Guidance Framework expectations and SMC certification requirements when evaluating the proposal.

Kanban for M&O and Steady-State Support Activities

Kanban is a recognized Agile workflow methodology suited to M&O (M&O) and steady-state support activities where work arrives continuously rather than in defined Sprint cycles, including ongoing defect resolution, minor enhancement requests, infrastructure patching, configuration changes, and production support queue management. Unlike Sprint-based Agile, Kanban operates on a continuous flow model with work-in-progress limits and no fixed iteration timeframes, making it appropriate for environments where the volume and priority of incoming work is unpredictable.

The Contractor **MAY** propose Kanban as the primary workflow methodology for M&O and steady-state support activities that fall outside of active DDI Releases. Any such proposal **MUST** be submitted as an Alternate Methodology Proposal per the requirements of this section and **MUST** include: a description of the Kanban board structure and work-in-progress limits; the process for triaging, prioritizing, and escalating incoming work items; how SLA compliance will be tracked and reported to OHIP; and how Kanban activities will interface with the Release Management Standard ([Section 7.0](#)) when work items require formal release packaging for production deployment.

Kanban **MUST NOT** be proposed as the primary delivery methodology for any new DDI effort, major feature development, or project receiving CMS SMC. Those activities require a structured methodology with defined phase gates as specified in Sections [8.4.2](#) through [8.4.4.3](#).

Rapid Application Development and Incremental Model Acknowledgment

Rapid Application Development (RAD) is an iterative methodology that prioritizes rapid prototyping, user feedback, and quick iteration cycles over extensive upfront planning. RAD concepts, including functional prototyping, iterative user feedback sessions, and accelerated design-build-test cycles, may be valuable for specific OHIP use cases such as dashboard prototyping, user interface exploration, and proof-of-concept development where rapid visualization of business requirements accelerates stakeholder alignment.

The Contractor **MAY** propose RAD elements within an approved SDLC methodology for designated prototyping or UI/UX exploration activities. Any such proposal **MUST** be submitted as an Alternate Methodology Proposal per the requirements of this section and **MUST** include: a description of which project activities will use RAD techniques; how RAD prototyping outputs will be transitioned into the project's formal design and development process; and how regulatory compliance, security requirements, and CMS certification evidence will be maintained for deliverables originating from RAD activities.

A pure RAD methodology, in which rapid prototyping replaces structured requirements analysis, formal design documentation, and phase-gated testing, **MUST NOT** be proposed as the primary delivery methodology for any OHIP MES project. OHIP's regulatory, compliance, and federal certification obligations, including the CMS SMC requirements defined in Section 8.3, require the structured governance and traceability provided by the methodologies defined in Sections 8.4.2 through 8.4.4.3.

Incremental Model

The Incremental model divides a project into smaller increments, each delivering a portion of the total functionality. Each increment follows a mini-life cycle of planning, design, development, and testing before being integrated with previously completed increments. This approach is conceptually similar to the Hybrid Iterative Development methodology defined in [Section 8.4.4.3](#), which organizes delivery into repeated Iteration cycles with progressive integration.

Contractors seeking an incremental delivery approach **SHOULD** evaluate the Hybrid Iterative Development methodology ([Section 8.4.4.3](#)) or the Hybrid Waterfall + Scrum approach ([Section 8.4.4.1](#)) before proposing a standalone Incremental methodology. If neither approved hybrid approach meets the project's needs, the Contractor **MAY** propose an Incremental methodology as an Alternate Methodology Proposal per the requirements of this section.

8.4.4.5 Spiral SDLC

The Spiral methodology is a risk-driven iterative SDLC approach that combines elements of both Waterfall's structured planning and iterative development's adaptability. Each cycle (or spiral) progresses through four (4) quadrants: Planning, Risk Analysis, Engineering, and Evaluation. What distinguishes the Spiral model from other iterative approaches is that risk analysis is the primary driver of each iteration, the scope, depth, and approach of each cycle are determined by the risks identified and the strategies developed to mitigate them.

The Spiral methodology is best suited for OHIP MES projects with significant technical or integration uncertainty, high-risk modernization efforts involving multiple interdependent systems, projects where requirements cannot be fully defined upfront due to technical unknowns, and complex vendor coordination scenarios where risk must be continuously assessed and managed. It is not appropriate for small, well-defined projects with stable requirements where Waterfall or V-Shaped would be more efficient.

Spiral Phases

Each spiral cycle consists of four (4) phases:

- **Planning:** Define objectives for the current cycle, identify constraints, and establish the scope of the iteration based on the risk profile. Requirements for the current cycle are elaborated during this phase.
- **Risk Analysis:** Identify, assess, and prioritize risks for the current cycle. Develop risk mitigation strategies and determine whether to proceed, prototype, or adjust scope. The Contractor **MUST** document all identified risks and mitigation strategies in a Spiral Risk Register.
- **Engineering:** Execute the design, development, and testing activities for the current cycle. All engineering activities **MUST** comply with [Section 5.0](#) and [Section 6.0](#).
- **Evaluation:** Conduct a formal review of the cycle's output with OHIP stakeholders. Gather feedback, validate that risk mitigations were effective, and confirm readiness to proceed to the next cycle or to End-to-End testing.

Multiple spiral cycles are executed, with each cycle building upon the verified output of the previous one. The number of cycles is determined by the project's risk profile and is documented for OHIP approval.

Spiral Methodology Requirements

When the Spiral methodology is selected, the Contractor **MUST**:

Submit a Spiral Project PlanThe Contractor **SHOULD**:

- Define and submit to OHIP for approval the anticipated number of cycles, the scope and objectives of each planned cycle, the risk analysis approach, and the criteria for determining when iterative cycling is complete and End-to-End SIT may begin.
- Conduct a formal Risk Analysis phase at the beginning of each spiral cycle. The risk analysis **SHOULD** be documented in a Spiral Risk Register that identifies each risk, its probability, potential impact, mitigation strategy, and residual risk after mitigation.
- Submit the Spiral Risk Register to OHIP for review at the beginning of each cycle. OHIP **SHOULD** approve the risk mitigation approach before the Engineering phase of that cycle begins.

- Conduct a formal Evaluation review with OHIP at the conclusion of each spiral cycle. The Contractor **SHOULD** present the cycle's deliverables, demonstrate that planned risk mitigations were executed, and obtain OHIP concurrence before proceeding to the next cycle.
- Maintain the RTM per the OHIP Requirements Management Standard ([Section 3.0](#)) and update it at the conclusion of each spiral cycle.
- Follow the OHIP Release Management Standard ([Section 7.0](#)) for deployment of all Spiral methodology outputs to production.
- Follow the OHIP Release Management Standard and Test Management Standard for all deployment and testing activities.

Deliverables: Spiral Risk Register (updated per cycle), Cycle Evaluation Reports, RTM (updated per cycle), Test Management Plan

8.5 Common SDLC Activities

The SDLC activities outlined here are not compatible with specific methodologies. Regardless of the SDLC approach, the Contractor **MUST** include measures to ensure that both technical staff and end-users have access to training resources tailored to their roles. For newly deployed solutions, the Contractor **MUST** provide enhanced support for a defined period to promote operational stability and encourage user adoption.

8.5.1 Training

The Contractor **MUST** develop Training Plan(s) that addresses the distinct needs of both technical resources and end-users, as the requirements of each group differ.

- **Technical Resources:** Training for technical resources **MUST** focus on the architecture, integration, security, and maintenance of the implemented solution. The technical training plan **MUST** ensure that staff can support solution operations efficiently, troubleshoot issues proactively, and adapt to evolving federal and State requirements.
- **End-Users:** Training for end-users **MUST** focus on those who interact with the solution in their day-to-day business activity workflows. Training materials, user manuals, and help content **MUST** comply with the accessibility requirements defined in [Section 2.0](#), including WCAG 2.2 AA conformance for any digital training assets. Training for the end-user community **MUST** be role-based, scenario-driven, and accessible, focusing on useability, navigation, and task execution within the solution interface. It **MUST** include hands-on practice, job aids, user manuals, and support channels to reinforce learning.
- **Go-Live Gate:** The Contractor **MUST** provide documented evidence to OHIP that all required end-user and help desk training has been executed and that participant sign-off sheets, training completion records, or equivalent documentation have been collected and are accessible for OHIP review before the Go-Live gate is approved.
- **UAT Gate:** All relevant training materials, user manuals, and job aids **MUST** be drafted and approved by OHIP before UAT commences. This ensures that end-users participating in UAT are testing against finalized workflows and documentation, and that UAT findings reflect real operational gaps rather than documentation deficiencies.

- **OCM Alignment:** Technical readiness does not equate to operational readiness. The Contractor **MUST** ensure that SDLC milestones are tightly synchronized with OCM and training activities to prevent Go-Live failures caused by unprepared end-users or help desk staff.

8.5.2 Hypercare

 **Cross-Reference:** The Hypercare requirements in this section represent the SDLC-layer obligations for post-deployment stabilization. Additional operational coordination requirements for Hypercare (including the Hypercare Plan approval, Requesting Module responsibilities, and escalation processes) are governed by [Section 7.9.4](#). These standards are complementary, not duplicative.

Hypercare is a designated period of heightened, time-bound support intended to stabilize new or changed functionality, ensure system performance, and reinforce user confidence and maintain operational continuity. Upon successful Go-Live, the Contractor **MUST** plan for and provide Hypercare services, at OHIP-designated locations and/or approved remote sites, for a period mutually agreed upon by OHIP.

8.5.2.1 Hypercare Planning and Coordination of Activities

The Contractor **MUST** submit a Hypercare Plan for OHIP review and approval. This deliverable **MUST** define the scope, duration, roles and responsibilities, coverage hours, applicable environments, escalation paths, communications cadence, tooling, entry/exit criteria, and extension criteria. It also **MUST** define the plan to remediate defects and the approach to execute rollback if the deployed functionality does not meet required performance, stability, or acceptance criteria.

For coordination, the following Contractor responsibilities applies:

- Provide reliable and knowledgeable staff to support configuration and customization services during the Hypercare period.
- Resolve incidents and provide knowledge transfer to OHIP.
- Submit a Hypercare Completion Report, which contains all incidents and defects identified during the Hypercare period and the resolutions that occurred during or after Go-Live for OHIP review and approval.

8.5.2.2 Execution and Support Activities

When executing Hypercare support activities, the Contractor **MUST**:

- Monitor performance in real time, maintaining dashboards for availability, latency, throughput, error rates, and key business transactions.
- Operate continuous incident and problem management, including Sev 1/Sev 2 immediate response and expedited triage; ensure Sev 3/Sev 4 items are tracked and managed to closure per the approved plan.
- Integrate automated and manual testing into the CI/CD pipeline to detect and prevent regressions; execute targeted regression suites, smoke tests, and data validation checks post deployment.
- Be responsible for correcting data errors introduced by the data conversion process; any necessary data backfills **MUST** be performed at no additional cost during the warranty period.

- Maintain open communications with stakeholders via daily standups and status reports during Hypercare, including issue lists, aging, SLA compliance, risks, mitigations, and next day plan.
- Provide Knowledge Transfer (including runbooks and training) are completed prior to production Go-Live.⁴⁴
- When executing Hypercare support activities, the Contractor **SHOULD**:
- Conduct refresher sessions (as needed) to educate stakeholders, reinforce key workflows, and address any gaps observed at Go-Live.

In the event roll back is required, the Contractor **MUST**:

- Execute the rollback plan defined per [Section 7.5.2](#), including: immediate notification to OHIP, execution of documented rollback procedures, validation that rollback was successful, RCA initiated within twenty-four (24) hours, and a CAP submitted to OHIP within five (5) business days.

8.5.2.3 Success Criteria and Exit/Extension Activities

The Contractor **MUST**:

- Track and report KPIs, which includes Contract-specific SLAs for incident response and resolution by severity and report compliance daily during Hypercare.
- Meet SLAs as defined in the Contract.
- Develop Hypercare Exit Criteria for OHIP review and approval.
- Exit Hypercare only when the OHIP approved exit criteria are met.
- Request Hypercare extension only if OHIP approved criteria are met (e.g., breach of SLA thresholds, recurrence of high severity incidents, material performance degradation).

 **Note:** New or unrelated changes do not, by themselves, trigger extended Hypercare; such changes must follow normal change control unless approved by OHIP.

Deliverable: Hypercare Plan, Hypercare Completion Report

8.5.3 Transition-In Phase

Beyond the core SDLC methodologies described in this Standard, the successful implementation and long-term value of any IT solution depends on a range of critical activities that bridge the gap between project delivery and sustained business operations. This section addresses consideration for managing the pivotal event when transitioning solution development, implementation, and M&O (M&O) activities from an incumbent contractor to a new contractor, which necessitates a Transition-In Phase.

⁴⁴ It is expected that operations readiness activities, including and not limited to production Standard Operating Procedures, on-call schedules, and escalation paths are finalized and in place before Cutover activities commence.

The Transition-In Phase⁴⁵ focuses on the seamless transition of operational systems maintenance from an incumbent to a new Contractor working on transformational enhancements and current operations of designated MES business functions.

Overall transition governance, including the master Transition-In Plan and Transition-Out Plan, the knowledge transfer framework, staffing continuity, and the timing and cost provisions that apply across all domains, is established by the OHIP Program and Project Management Domain Document, Section 2.18 (Transition Management), which is the authoritative source for the transition framework. This section governs the SDLC and solution-handover dimension of Transition-In: the technical and operational activities by which a new Contractor assumes development, implementation, and M&O of designated MES functions. The plans and artifacts produced under this section implement and inform the master Transition-In Plan governed by that document. Operational and service-continuity activities for systems already in production, and all Transition-Out execution, are governed by the OHIP Infrastructure and Operations Domain Document.

During the Transition-In phase, the new Contractor **MUST** conduct the following activities:

- Submit a Transition-In Management Plan (TIMP) for OHIP review and approval that details the solution-level execution of the master Transition-In Plan (governed by the Program and Project Management Domain Document, Section 2.18), describing the approach to managing Transition-In activities across System Architecture, project management, M&O, and DDI at both enterprise and service/function levels.
- Coordinate closely with the Incumbent Contractor and OHIP, through weekly joint planning sessions and aligned expectations, plans, and timelines, documenting status in Transition-In Weekly Status Reports for OHIP review.
- Develop, maintain, and regularly review the transition-in schedule within the TIMP, identifying all tasks, milestones, dependencies, and sequencing necessary for Transition Cutover.
- Execute the TIMP, progressively assuming responsibilities across all transition areas while maintaining operational stability and meeting SLA requirements.
- Conduct and document knowledge transfer activities, including review of materials, walkthroughs, shadowing, validation of operational processes, and confirmation of understanding.
- Assess and prepare to evolve solution architecture and fully assume M&O responsibilities under the TIMP, culminating in a final readiness review with OHIP and demonstrating completion of all Transition In requirements for final approval and handoff.
- Define the approach for DDI responsibilities during Transition-In within the TIMP, including alignment with ongoing enterprise modernization activities.
- Conduct a formal Code Quality Assessment of all incumbent Contractor codebases prior to assuming M&O responsibility. The assessment **MUST** verify that inherited code meets the minimum quality thresholds defined in [Section 5.0](#) of this document, specifically: unit test statement coverage of ≥80%, branch coverage of ≥75%, 100% coverage for authentication and authorization modules, zero unresolved Severity 1 or Severity 2 static analysis findings, and compliance with the approved secure coding standard. Where incumbent code fails to meet these

thresholds, the Contractor **MUST** document all gaps in a Transition-In Code Quality Gap Report, submitted to OHIP within thirty (30) days of the assessment. OHIP **MUST** review the gap report and determine, in writing, whether the new Contractor is required to remediate identified gaps, whether the incumbent Contractor is responsible for remediation prior to handoff, or whether an agreed exception with a time-bound remediation plan applies. The new Contractor **MUST NOT** assume contractual responsibility for code quality deficiencies that are documented in the gap report and attributed to the incumbent prior to OHIP's written determination.

Deliverables: TIMP, Transition-in Weekly Status Reports, Transition-In Execution, and Transition Cutover

8.6 Decommissioning and Data Archival Phase

For Releases intended to replace, modernize, or subsume existing legacy functionality, the Contractor's SDLC **MUST** include a formal Decommissioning and Data Archival Phase. This phase ensures the controlled retirement of legacy systems, the preservation of all Medicaid-required historical data, and the protection of OHIP's legal and compliance obligations after new functionality is in production. All data migration, archival, and retention activities performed as part of this phase **MUST** comply with the OHIP Data and Information Management Domain Document, Data Conversion Standard, which is the authoritative standard governing execution of data migration and post-conversion retention. This section governs the life cycle gate and authorization requirements for decommissioning; the OHIP Data and Information Management Domain Document, Data Conversion Standard governs how the underlying data activities are performed.

- The Contractor **MUST** develop and submit a Legacy Sunsetting Plan for OHIP approval. This plan **MUST** define the timeline for legacy system read-only transition, the schedule for interface severance with dependent systems, and the criteria and date for final legacy infrastructure shutdown. The plan **MUST** be submitted to OHIP for review no later than the UAT gate of the Release replacing the legacy functionality.
- The Contractor **MUST** ensure all historical legacy data required for Medicaid compliance, federal audit trails, and legal holds is successfully migrated to the OHIP Data Lakehouse or another OHIP-approved archival system before legacy systems are decommissioned, in accordance with the retention and life cycle requirements defined in the OHIP Data and Information Management Domain Document, Data Conversion Standard and Data Lakehouse Architecture Standard. The Contractor **MUST** provide documented evidence of data migration completeness and accuracy, including record count reconciliation, referential integrity validation, and sample-based content verification. Migration validation results **MUST** be submitted to OHIP for approval before any legacy system is placed in read-only mode or decommissioned. Data completeness and integrity **MUST** be validated and documented in a Data Archival Verification Report submitted to OHIP.
- The Contractor **MUST** obtain formal written sign-off from the OHIP Product Owner and the OHIP Security Officer before executing any destructive actions against legacy systems, including interface severance, data deletion, or infrastructure shutdown. Destructive actions **MUST NOT** proceed without documented OHIP written authorization.

Deliverables: Legacy Sunsetting Plan, Data Archival Verification Report

9.0 Business and Technical Architecture Standard

9.1 Introduction

The Business and Technical Architecture Standard governs the architecture artifacts and models that the Contractor produces to describe how OHIP MES solutions are structured, integrated, and deployed. It spans four architecture domains business, application, technology, and security; and establishes the modeling notations, content requirements, and documentation expectations for each. These artifacts provide a consistent, vendor-neutral view of OHIP's solutions that supports interoperability, modular procurement, investment planning, and CMS certification.

The artifacts in this section are grounded in the MITA framework and in widely adopted, industry-standard modeling notations, Business Process Model and Notation (BPMN) 2.0 for process models, UML for use case models, and ArchiMate 3.1 for application, technology, and deployment models. To ensure architectural consistency across the portfolio, the Contractor references a shared definitions catalog maintained in OHIP's Enterprise Architecture Repository and applies standardized vocabulary for terms, capability names, and component descriptions across all domains.

This section is closely related to other sections of this document and to companion standards. Architecture artifacts are developed during the Design and Implementation phases of the SDLC described in [Section 8.0](#), and the business and use case models traceably support the requirements defined under [Section 3.0](#) (Requirements Management Standards). Data architecture, including the Conceptual, Logical, and Physical Data Models, data dictionaries, and data naming conventions, is governed by the companion OHIP Data and Information Management Standard and is not duplicated here; this section begins at the point where those data artifacts inform business, application, and technology architecture. Security architecture is developed simultaneously with the other domains and relies on their artifacts, as described in [Section 9.6](#).

9.2 Objectives

The primary objectives of this Standard are to:

- Establish a consistent set of business, application, technology, and security architecture artifacts and models that describe how MES solutions are structured, integrated, and deployed.
- Require industry-standard modeling notations (BPMN 2.0, UML, and ArchiMate 3.1) and a shared definitions catalog so that architecture artifacts are interoperable, comparable, and vendor-neutral across OHIP's system portfolio.
- Support MITA alignment and CMS certification by mapping business capabilities, processes, and maturity to OHIP's strategic roadmap, state self-assessment, and transition planning artifacts.
- Provide end-to-end traceability from business capabilities and use cases through application, technology, and deployment architecture, enabling impact analysis, modular procurement, and investment planning.
- Ensure all integrations and interfaces depicted in architectural models are fully and consistently specified to support secure, reliable interoperability across MES modules.
- Integrate security architecture with the business, application, and technology domains so that data sensitivity, access control, and secure deployment are addressed from the outset of design.

9.3 Business Architecture Artifacts

The Contractor **MUST**:

- Reference a shared definitions catalog maintained in OHIP's Enterprise Architecture Repository for business architecture artifacts.
- Match the standardized vocabulary for terms, capability names, and component descriptions across all domains to ensure architectural consistency.

9.3.1 Business Capability Model

A Business Capability Model defines *what* an enterprise requires to execute its business strategy effectively. It identifies and organizes high-level business functions (e.g., Member Management, Provider Management, and Claims Processing) into structured domains and sub-capabilities. This model provides a stable, business-centric view of operations, independent of organizational structure or technology.

For example, in MITA, capabilities are grouped into modules and used to drive architecture alignment, investment planning, and system modernization.

Table 22 – Business Capability Model

Solution Design	Business Architecture
Model Type and Notation	A Hierarchical Functional Model using informal structured notation. For example, MITA presents capabilities as indented lists, tree structures, or functional blocks.
Applicable Phase	Requirements Review and Design Phase
Applicable Deliverables	<ul style="list-style-type: none">• Business Architecture Models• State Self-Assessment

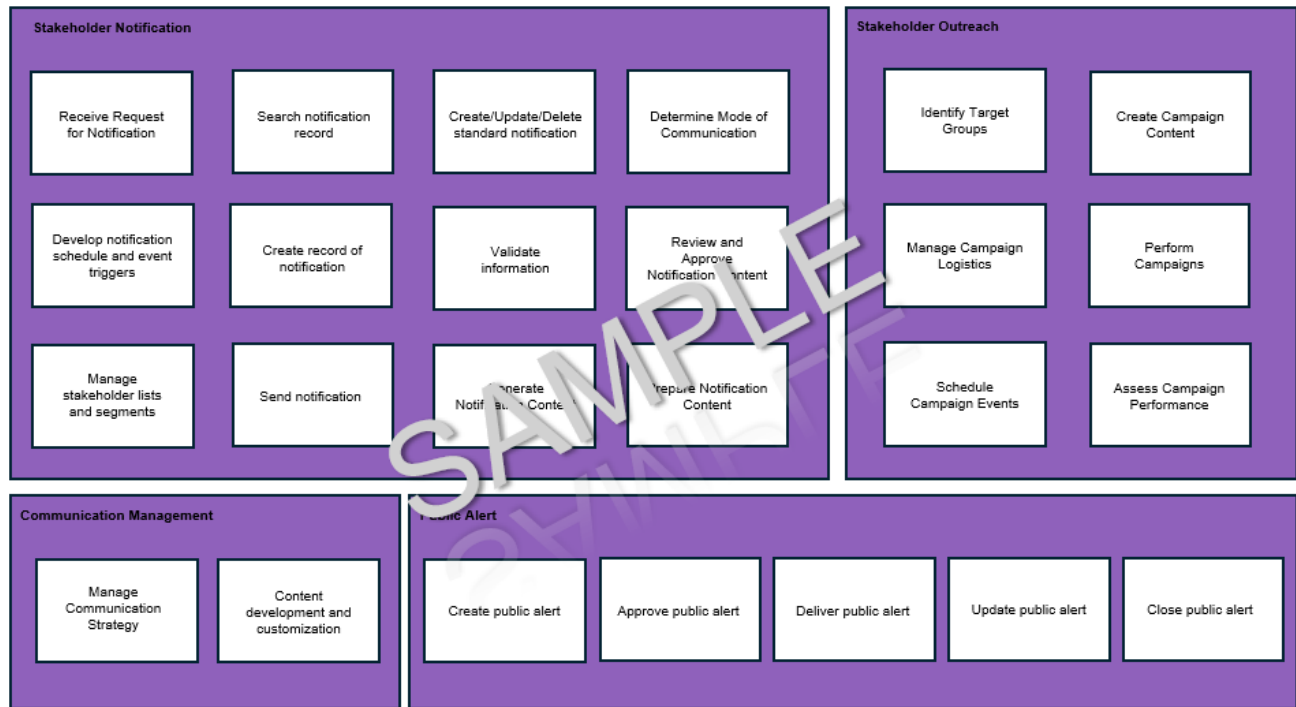


Figure 9 – Sample Business Capability Model

The following data elements provide a structured way to capture and convey the relevant information in a Business Capability Model:

- **Capability Name:** A succinct and descriptive title for the capability that clearly indicates the function or activity it represents
- **Capability Description:** A detailed explanation of what the capability entails, including its scope, purpose, and any key activities involved
- **Category/Domain:** The business domain or category under which the capability falls, such as Finance, Marketing, Operations, etc.

9.3.1.1 Guidance for Creating a Business Capability Model

To develop a Business Capability Model, the Contractor **MUST**:

- Understand the organization's strategic goals and objectives.
- Identify the “why” behind creating the capability model.
- Identify the high-level capabilities that define the essential functions or services the business needs to offer.
 - o Utilize industry standards and frameworks as references, such as the APQC Process Classification Framework or the Business Architecture Guild’s BIZBOK Guide, to ensure comprehensiveness.

- Break down high-level capabilities into more detailed, granular sub-capabilities as needed for clarity and understanding, stopping at a level where it is meaningful for providing insights and making decisions.
- Structure the capabilities into a logical hierarchy or model. Typically, capabilities are organized into core, supporting, and strategic groups.

9.3.2 Business Capability Matrix

The Business Capability Matrix maps each business capability to maturity levels and specifies the expected behaviors, processes, and outcomes at each level of capability development. This model helps assess an agency's "As-Is" and "To-Be" operational maturity and supports the development of a strategic Roadmap to reach desired future states.

Table 23 – Business Capability Matrix

Solution Design	Business Architecture
Model Type and Notation	Refer to the Maturity Assessment Matrix using tabular notation. Each row represents a capability. Each column corresponds to a maturity level, with descriptive criteria in each cell.
Applicable Phase	Requirements Review and Design Phase
Applicable Deliverable	State Self-Assessment

There is a link between business processes and business capabilities as shown in the MITA Business Capability Matrix and sample Business Capability Matrix below:

CM – Case Management					
Manage Population Health Outreach					
Capability Question	Level 1	Level 2	Level 3	Level 4	Level 5
Business Capability Quality: Accuracy of Process Results					
How accurate are the results of the processes?	Manual processes result in greater opportunity for human error. Accuracy is low.	Automation and standardized business rules definitions reduce error and improve accuracy above Level 1.	SMA adopts MITA Framework, industry standards and information exchange with intrastate agencies and entities improving accuracy to 90% or higher.	SMA adopts MITA Framework, industry standards and information exchange with regional agencies and entities improving accuracy to 99% or higher.	SMA adopts MITA Framework, industry standards and information exchange with national agencies and entities improving accuracy to 99% or higher.
Business Capability Quality: Utility or Value to Stakeholders					
Does the business process satisfy stakeholders?	Stakeholders lack confidence in information negatively affecting stakeholder satisfaction with the process.	Automation and standardized business rules provide clear and consistent information. Stakeholder satisfaction is greater than Level 1.	SMA adopts MITA Framework, industry standards and information exchange with intrastate agencies and entities improving stakeholder satisfaction to 90% or higher. SMA uses survey or questionnaire for information collection.	SMA adopts MITA Framework, industry standards and information exchange with regional agencies and entities improving stakeholder satisfaction to 95% or higher.	SMA adopts MITA Framework, industry standards and information exchange with national agencies and entities improving stakeholder satisfaction to 98% or higher.

Figure 10 – Sample Business Capability Matrix

9.3.2.1 Guidance on Creating a Business Capability Matrix

To create a Business Capability Matrix, the Contractor **MUST**:

- Identify and define the high-level capabilities (e.g., Consumer Management, Product Development, Supply Chain Management, etc.).
- Leverage existing frameworks (e.g., reference models, Business Capability Map by the Business Architecture Guild, etc.) to capture all necessary capabilities.
- Group related capabilities into categories or business domains (e.g., Sales, Marketing, and IT, Operations).
- Assess the current maturity level for each capability identified.
- Apply an industry-specific or customized maturity model (e.g., Initial, Developing, Defined, Managed, and Optimizing).
- Determine the performance of each capability (e.g., poor, adequate, and excellent) and the desired Future State.
- Prioritize capabilities based on strategic importance, performance gaps, and consumer impact.
- Design a table or chart that lists capabilities in rows with their respective attributes in columns, such as current maturity, target maturity, strategic importance, and performance.
- Use visual cues, such as color coding, to indicate levels of maturity, performance ratings, or prioritization levels for a clearer and more instinctual understanding.

9.3.3 Business Process Model

Business Process Models are a visual representation of the sequence of activities, tasks, or steps in a business process. The purpose of a Business Process Model is to show the workflow of the business processes in scope for a solution. This artifact is also referred to as Swimlane Diagram, Process Workflow, or as the TOGAF Process Flow Diagram artifact. Business Process Models describe the activities needed to achieve a business objective and the roles responsible for performing those tasks.

Table 24 – Business Process Model

Solution Design	Business Architecture
Model Type and Notation	SHOULD Use BPMN 2.0, Business Process Elements: Start, Intermediate, and End Events, Business Processes, Activities, Gateways, Pools, and Lanes. For complex process modeling, refer to Object Management Group - BPMN 2.0 standards
Applicable Phase	Requirements Review and Design Phase
Applicable Deliverable	<ul style="list-style-type: none">• Functional Design Document(s)• Activity Diagrams

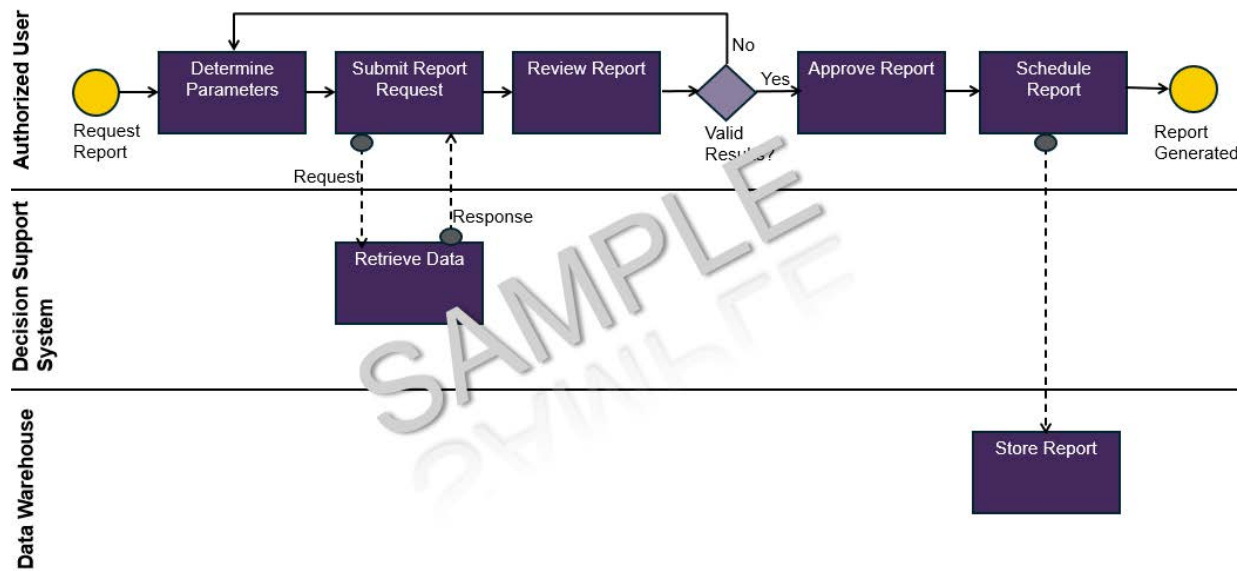


Figure 11 – Sample Business Process Model

A business process model typically includes:

- **Activities:** These are the basic units of work performed during a business process. Activities range from simple tasks to more complex operations. Each activity is depicted as a rectangle with rounded corners.
- **Sub-processes:** A sub-process is a set of activities that have a logical grouping within a larger process, which are expanded to show its details or collapsed to simplify the process model.
- **Start/End Events:**
 - o **Start Event:** This signifies the beginning of a process. It is represented by a single circle and can have various triggers, such as a message, timer, or condition.
 - o **End Event:** This indicates where a process path concludes. It is depicted as a double circle and can include various results, such as message sending or error results.
- **Gateways:** Gateways control the divergence and convergence of sequence flows in a process. They dictate the branching, forking, merging, and joining of paths.
 - o **Exclusive Gateway:** This allows only one path to be taken (usually depicted with an "X" or a diamond).
 - o **Parallel Gateway:** This allows multiple paths to be executed simultaneously (represented by a "+" inside a diamond).
 - o **Inclusive Gateway:** This splits paths, based on conditions, allowing one or more paths to be taken (depicted as a diamond with a circle inside).
 - o **Complex Gateway:** This manages complex conditions and rules for diverging/converging paths.
- **Lanes and Pools:**
 - o **Pools:** These represent major participants in a process, such as different organizations or major units. Each pool acts as a container for processes, depicted as containing one or more lanes.

- o **Lanes:** These are sub-divisions within a pool, often used to represent roles, departments, or specific functions within a broader organizational context.
- **Message Flows:** Message flows depict the exchange of information between participants in different pools. They are represented as dashed lines with an open arrowhead, indicating the direction of communication.
- **Conversations:** In BPM, a conversation represents a set of logically related message exchanges between participants. It is typically depicted as a hexagon and helps illustrate communication between different entities involved in a business process.

These elements form the core components of Business Process Model and Notation (BPMN), allowing organizations to document and analyze their processes effectively.

9.3.3.1 Guidance on Creating a Business Process Model

To create an effective Business Process Model, the Contractor **MUST**:

- Select the business process being modeled. Clearly define the start and end points.
- Define the boundaries of the process to avoid scope creep.
- Establish the goal of modeling this process (e.g., improve efficiency, identify bottlenecks, train, etc.).
- Select a modeling tool (e.g., Visio, Lucidchart, Bizagi, etc.).
- Define a consistent process modeling notation, such as BPMN (Business Process Model and Notation). Ensure all stakeholders understand the symbols used.
- Break down the process into main activities or tasks. A task is a single unit of work.
- Arrange tasks in the order they are performed.
- Assign responsibilities for each task, roles could include teams, departments, or individual job titles.
- Include decision branches (e.g., Yes/No and If/Else scenarios) and the conditions that guide the different paths.
- Define what inputs start the process and what outputs are produced at the end.
- Show how data moves through the system if relevant.
- Present the model to stakeholders and gather feedback.
- Check the model for logical consistency, completeness, and correctness.
- Incorporate feedback and make necessary revisions to the model.

9.3.4 Use Case Model

A Use Case Model is a type of model in software development and business analysis used to document the interactions between external actors (e.g., users and a system or application). It describes the system's functional requirements by identifying cases, actors, and scenarios.

Table 25 – Use Case Model

Solution Design	Business Architecture
-----------------	-----------------------

Format	UML Behavioral – Use Case Elements: Actor, Use Case, Boundary, Use Case relationships
Applicable Phase	Requirements Review, Design and Implementation Phase
Applicable Deliverable	Functional Design Document(s)

Figure 12 below is a sample use case model showing interactions among applications:

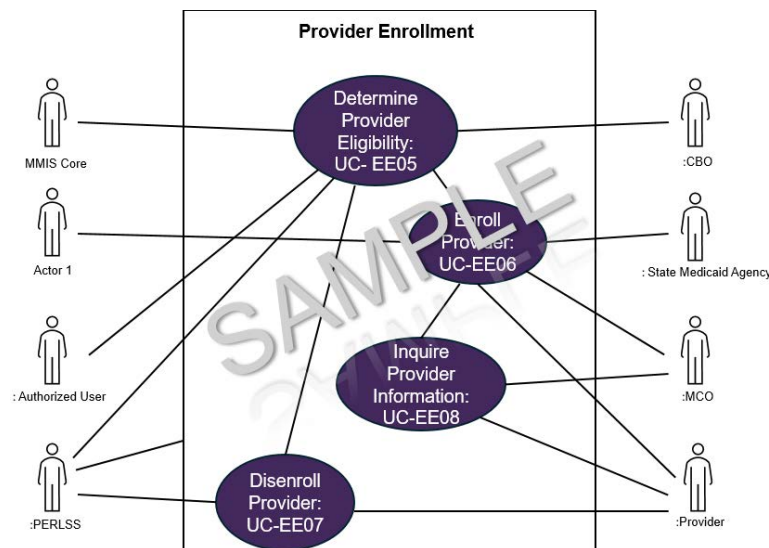


Figure 12 – Sample Use Case Interaction Diagram

A Use Case Model typically includes the following elements:

- **Actors:** The users or external systems that interact with the system
- **Use Cases:** The specific actions or tasks the system **MUST** perform to fulfill user or system requirements
- **Scenarios:** The specific conditions or paths a use case takes during execution
- **Relationships:** The relationships between use cases and actors, such as dependencies or extensions

9.3.4.1 Guidance for Creating a Use Case Model

To create an effective Use Case Model, the Contractor **MUST**:

- Clearly define the boundaries of the system being modeled.
- Represent the system with a rectangle and label it with the system's name.
- Determine the external entities and roles played by users or other systems interacting with your system. These **MAY** include human users, hardware devices, or other software systems.
- Differentiate between primary actors, who initiate interactions with the system, and secondary actors, who participate passively.
- Specify roles and responsibilities for each actor.
- Identify the goals of each actor.

- o **Special Requirements:** NFRs related to the use case, such as performance or security.

The Contractor **SHOULD**:

- Create a list of use cases (such as the functional requirements) that satisfy the goals of each actor. Use case names **SHOULD** be phrased as actions, such as “Log In,” “Place Order,” or “Generate Report.”
- Prioritize use cases by determining the importance and urgency of each use case.
- Write detailed descriptions for use cases, which **MUST** include, at minimum, the following:
 - o **Title:** The name of the use case.
 - o **Actors:** The primary and secondary actors involved.
 - o **Preconditions:** Conditions that **MUST** be true before the use case starts.
 - o **Main Flow:** The typical sequence of interactions (steps) between actors and the system executing the use case.
 - o **Alternate Flows:** Variations or exceptions in executing the use case.
 - o **Postconditions:** The state after the use case is completed.
 - o **Special Requirements:** Non-functional requirements related to the use case, such as performance or security.
- Represent actors using stick figures and place them outside the system boundary.
- Draw ovals inside the system boundary to represent each use case and label them with their names.
- Use lines to connect actors to the use cases they interact with.
- Indicate relationships, such for use cases that are always called by other use cases and for optional, conditional use cases.

9.4 Application Architecture Artifacts

The Contractor **MUST**:

- Develop a Conceptual Architecture Diagram that provides a high-level, end-to-end view of product components, service layers, data integrations, and hosting or platform layers by consolidating content from the Application Component Model, Solution Architecture Model, Conceptual Integration Architecture Model, and the Deployment Model.
- Create a Conceptual Application Integration Architecture Model (Contractor-neutral) and Solution Architecture Model (the Contractor’s specification of products) to show the integration of applications.

9.4.1 Conceptual Application Integration Architecture Model

A Conceptual Application Integration Architecture Model represents the integration of different systems, applications, or components into a single, cohesive system. It typically includes the different systems, applications, or modules that are being integrated, along with their key interfaces and connections. It represents the overall architecture of the integrated system and identifies any external factors or constraints that **MAY** affect its design or operation.

To create this model type, the Contractor **MUST** analyze the individual system(s) or component(s) to be integrated and identify their commonalities and differences. The Contractor **MUST** then define the requirements and relationships needed to integrate the different components into a cohesive system.

Table 26 – Conceptual Integration Architecture Model

Solution Design	Application Architecture
Format	ArchiMate 3.1 Components
Applicable Phase	Design and Implementation Phase
Applicable Deliverable	Technical Architecture Model(s)

Figure 13 illustrates this artifact for a fictional solution that would add the yellow servers to cloud and on-premises hosting. This example does not represent the current OHIP architecture, which has more nodes, connections, and layers of boundaries.

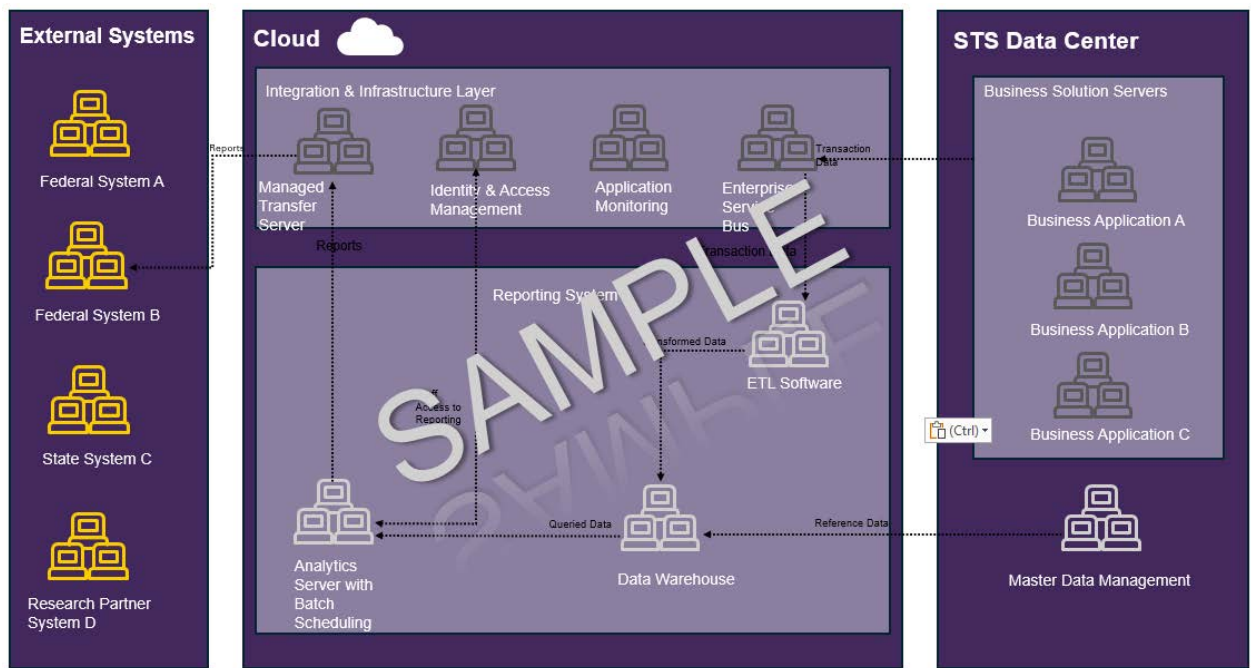


Figure 13 – Sample Conceptual Integration Architecture

A Conceptual Integration Architecture Model typically includes the following data elements:

- **Systems, Applications, or Components:** These are the individual entities that are being integrated. Each system or application is typically represented as a separate block or node within the model.
- **Interfaces:** Interfaces represent the points of interaction between systems, applications, or components. They define how data or services are exchanged.
- **Connections or Flows:** These illustrate the communication paths between interfaces, and represents data flows, service calls, API calls, or message exchanges.

9.4.2 Application Component Model

An Application Component Model is an architectural artifact used to encapsulate various aspects of application components as part of a solution proposal or documentation.

Table 27 – Application Component Model

Solution Design	Application Architecture
Model Type and Notation	ArchiMate 3.1 Application Components Elements: Application Components, Boundaries, Interfaces
Applicable Phase	Design and Implementation Phase
Applicable Deliverable	Technical Architecture Model(s)

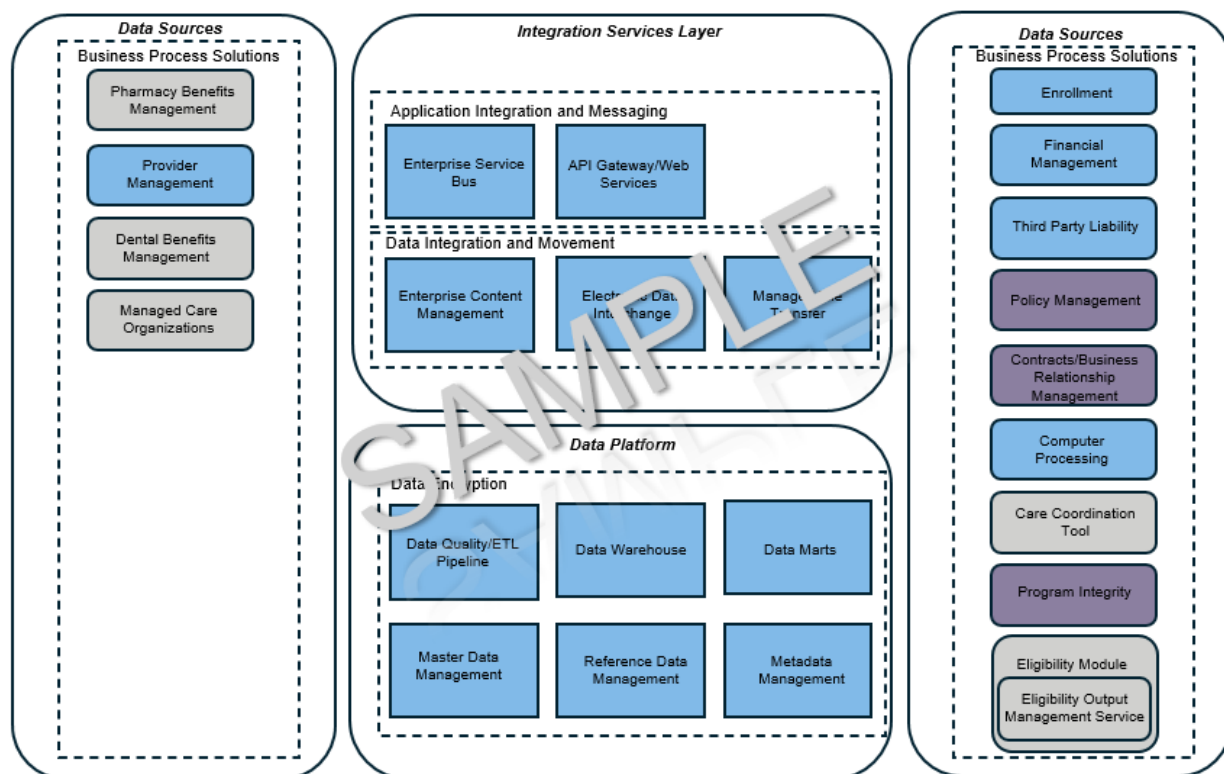


Figure 14 – Sample Application Component Model

Note: This artifact does not show the connections between applications. Figure 14 is not a representation of OHIP's current architecture.

The data elements of an Application Component Model typically include:

- **Component Name:** A unique identifier for the application component, often reflecting its purpose or functionality within the solution.

- **Description:** A brief overview of what the component does, including its role in the overall architecture and any key functions it supports.
- **Functionality:** Detailed enumeration of the specific functions or services the component provides, often aligned with business requirements.
- **Interfaces:** Information on how the component interacts with other components or systems, including APIs, protocols, and data formats used.
- **Dependencies:** A listing of other components, systems, or resources that this component relies on to function correctly.

The diagram and/or accompanying chart **MUST** include:

- Name of layer, module, or component (use Contractor-neutral business names that summarize the functionality).
- Functions to be performed by the application.
- Application status (in production, procurement, planned, etc.)
- Phase when the application **MUST** be implemented.
- Other descriptive or strategic information as needed.

9.4.3 Solution Architecture Model

The Solution Architecture Model, sometimes referred to as a Solution Architecture Diagram, is a logical overview of all application components in the Contractor's solution. This model outlines the servers and interfaces needed in the technology architecture. It also serves as a high-level diagram of the information flowing between applications. If the model does not include sufficient detail, additional diagrams or charts with further details **MUST** be provided.

Table 28 – Solution Architecture Model

Solution Design	Application Architecture
Format	ArchiMate 3.1 Components - nodes and links within boundaries. Elements: Application Components, Boundaries, Actors
Phase	Design and Implementation Phase
Applicable Deliverable	Technical Architecture Model(s)

The simplified, fictional sample in **Error! Reference source not found.15** shows suitable boundaries and applications with business and product names that do not represent current OHIP architecture.

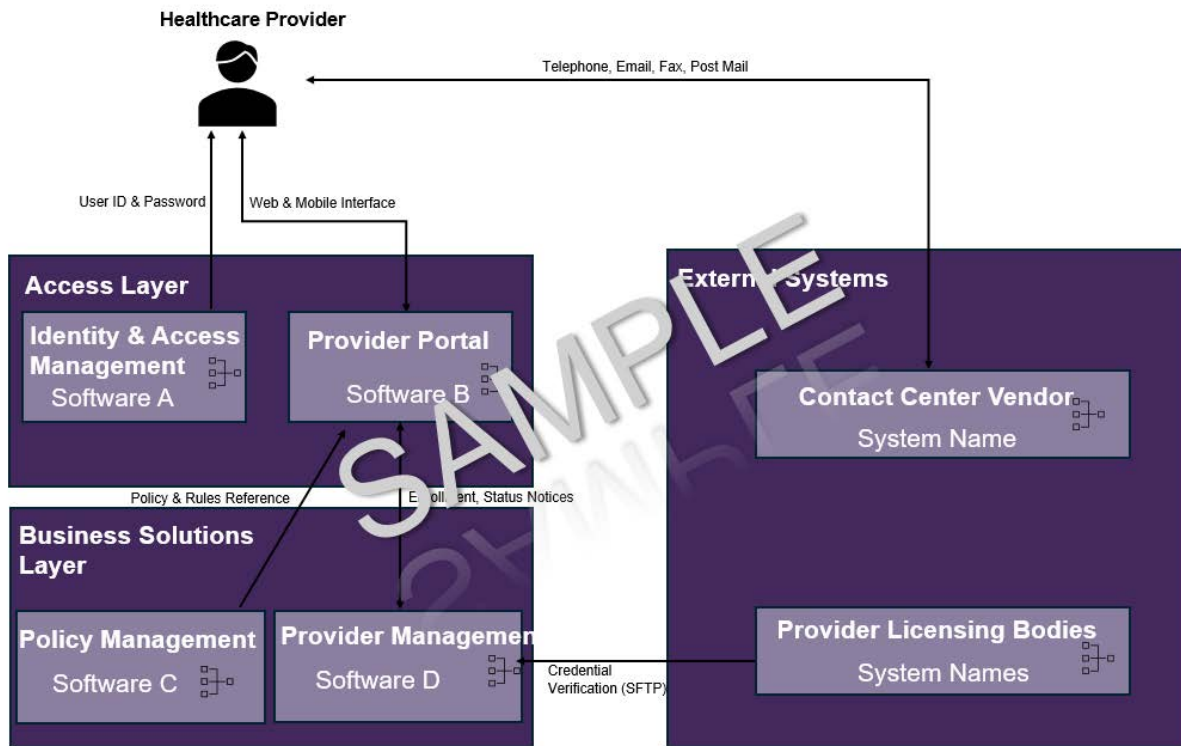


Figure 15 – Sample Solution Architecture Model

A Solution Architecture Model typically includes the following elements:

- **Application Components**
 - **Name and Description:** Each application's name and a brief description of its purpose and functionality
 - **Functionality:** Detailed explanation of what the application component does
- **Servers/Containers**
 - **Server/Container Names/Types:** Identification of the physical or virtual servers hosting applications
 - **Specifications:** High-level specifications, such as CPU, memory, and storage, if relevant to the logical architecture
 - **Deployment Architecture:** Which components are hosted on which servers
- **Interfaces**
 - **Interface Descriptions:** Definition of how different application components interact with each other
 - **Protocols and Standards:** Information about the data exchange methods such as REST, XML, JSON, etc.
 - **Data Flow:** Direction and trigger of data flow between components

9.5 Technology Architecture Artifacts

The following artifacts support Technology Architecture, but are not classified as models in this Standard:

- Technology architecture descriptions in the Concept of Operations.
- Interface Control Design Document.
- System Architecture Design Document.
- System Design Package.
- Cloud Design Package.
- Audit, Balance, and Control⁴⁶ (a stand-alone artifact if architects prefer to call it out explicitly).

9.5.1 Integration Overviews

Table 30 compares the artifacts that demonstrate how applications and technology **MUST** be integrated. Although these artifacts show distinct information, they **SHOULD** be aligned.

Table 29 – Comparison of Integration Overviews

Artifact Name	Conceptual Application Integration Architecture Model	Solution Architecture Model	Deployment Model	Network Topology Model
Section	Section 9.4.1	Section 9.4.3	Section 9.5.2	Section 9.5.3
Scope	Solution Design	Solution Design	Solution Design	Solution Design
Domain	Technology (and Application)	Application	Technology	Technology
Bounds	Organizations, application layers, clouds, platforms	Organizations, app layers	Security zones, cloud environment and hosting locations	Security zones, cloud environment and hosting locations
Nodes	Application and technology components	Application components, user roles	Servers	Network hardware and software, locations, servers, devices
Links	Message flows and integration technology	Message flows	Message protocols	Network connections

All integrations and interfaces shown in architectural models **SHOULD** include, at minimum:

- Interface name

⁴⁶ A document (or a set of procedures) that defines how to reconcile and verify every data exchange between systems. This can be thought of as the “proof sheet” for an interface—control totals, sequence checks, hash-totals, reconciliation reports, exception handling rules, etc.—all designed to prove that “what went out is exactly what came in,” with nothing lost, nothing duplicated, and no silent failures.

- Unique identifier
- Source and target systems
- Data format(s) (JSON, XML, HL7/FHIR, EDI, etc.)
- Protocol/transport mechanism
- Direction of data flow
- Triggering event or frequency
- Ensure all integrations and interfaces shown in architectural models include, at minimum: interface name, unique identifier, source and target systems, data format(s), protocol/transport mechanism, direction of data flow, and triggering event or frequency.

9.5.2 Deployment Model

A Deployment Model describes a logical technology architecture in the form of hardware and software nodes (such as servers) arranged into security zones (such as clouds, platforms, or tiers) and linked by communication paths (interface technology). The model shows how applications **SHOULD** be integrated securely to implement an IS solution, such as the translation of the Conceptual Integration Architecture to physical technology architecture and the translation of the Solution Architecture Model from applications to technology architecture.

Table 30 – Deployment Model

Solution Design	Technology Architecture
Format	(SHOULD Use Format) ArchiMate 3.1 Components
Applicable Phase	Design and Implementation Phase
Applicable Deliverable	Technical Architecture Model(s)

In Figure 16, server nodes and zone boundaries are shown. The communication path technology is not indicated on the links, as this is an early (conceptual) iteration of the model. Color is used to indicate layers of applications. This generic, sample model does not represent current OHIP architecture.

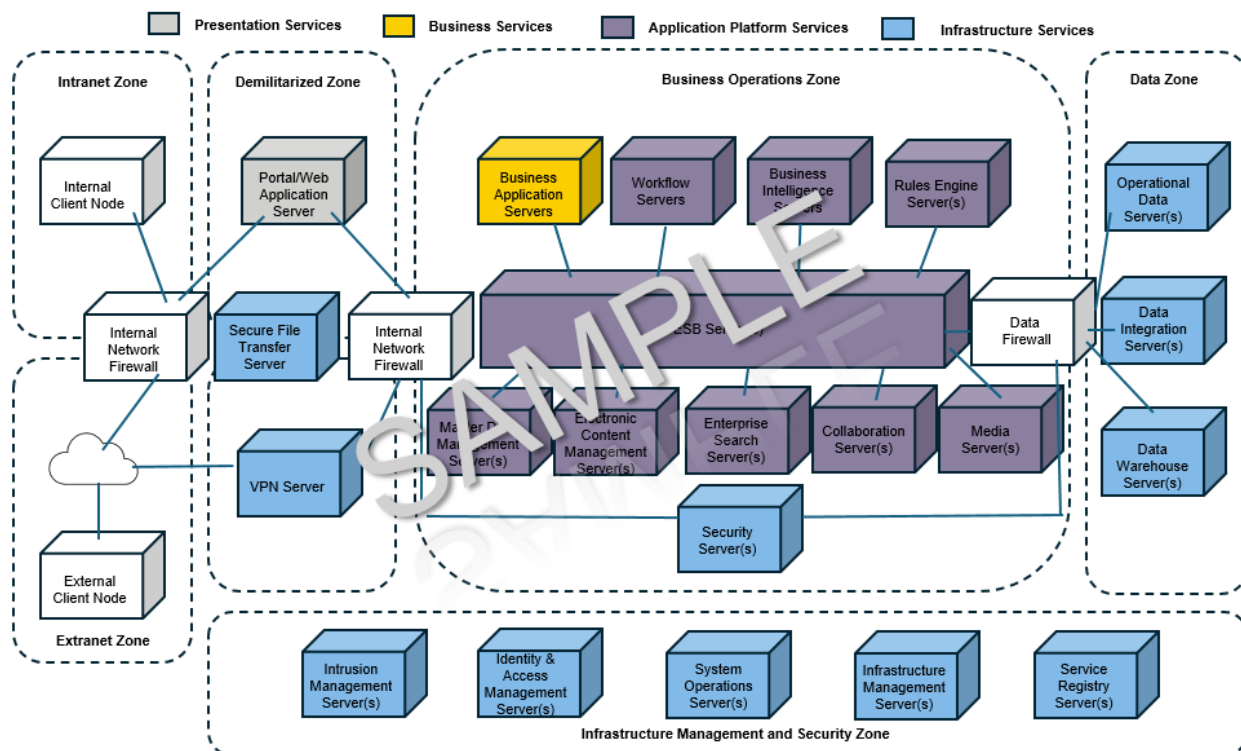


Figure 16 – Sample Deployment Model

A Deployment Model typically includes the following data elements:

- **Nodes**
 - o **Hardware Nodes:** Represent physical or virtual servers, including details such as CPU, memory, storage, and operating system.
 - o **Software Nodes:** Represent applications, services, middleware, databases, or other software components running on the hardware.
- **Security Zones**
 - o **Zones/Tiers:** Logical or physical groupings of nodes based on security policies or function, such as presentation, application, and data tiers.
 - o **Cloud Platforms:** Differentiation between public, private, hybrid, or multi-cloud environments.
 - o **DMZ (Demilitarized Zone):** A network area that provides an extra layer of security for external connections.
- **Communication Paths**
 - o **Interfaces:** Protocols and technologies used for communication between nodes, such as HTTP/S, TCP/IP, or WebSockets.
 - o **Network Topology:** The arrangement and connections between nodes, often illustrated as lines representing data flow.
- **Security Considerations**
 - o **Firewall Rules:** Guidelines and configurations for data flow between security zones.

- o **Encryption Methods:** Protocols used to secure data in transit, such as Transport Layer Security/Secure Sockets Layer (TLS/SSL).
- **Data Flow**
 - o **Information Exchange:** Details about data being exchanged between nodes, including format and frequency.
- **Scalability**
 - o **Load Balancers:** Tools, such as application delivery controllers (ADCs) that direct traffic.
 - o **Auto-Scaling Policies:** Rules for dynamically adjusting resources based on demand.
- **Redundancy and Failover**
 - o **Backup Nodes:** Secondary nodes to take over in case of failure.
 - o **Cluster Configurations:** Grouping of nodes for high availability systems.
- **Monitoring and Management**
 - o **Monitoring Tools:** Software used to observe system performance.
 - o **Management Interfaces:** Channels for administrators to control and configure the deployment.

9.5.3 Network Topology Model

A Network Topology Model shows the networking hardware and software that connects the technology infrastructure for a solution within the enterprise network. The Network Topology diagram connects the servers and other nodes on the Deployment Model.

Technology and security details such as IP addresses, subnets, or port numbers **MAY** be shown on the diagram.

Table 31 – Network Topology

Solution Design	Technology Architecture
Format	ArchiMate 3.1 Components
Related	The Deployment Model and Network Topology are aligned views on the same technology.
Applicable Phase	Design and Implementation Phase
Applicable Deliverable	Technical Architecture Model(s)

Figure 17 shows the Conceptual Design of a solution. Once the solution design is complete, the physical network topology **MUST** specify actual rather than representative clouds and network equipment, with IP addresses and other details. This sample does not represent the current OHIP architecture.

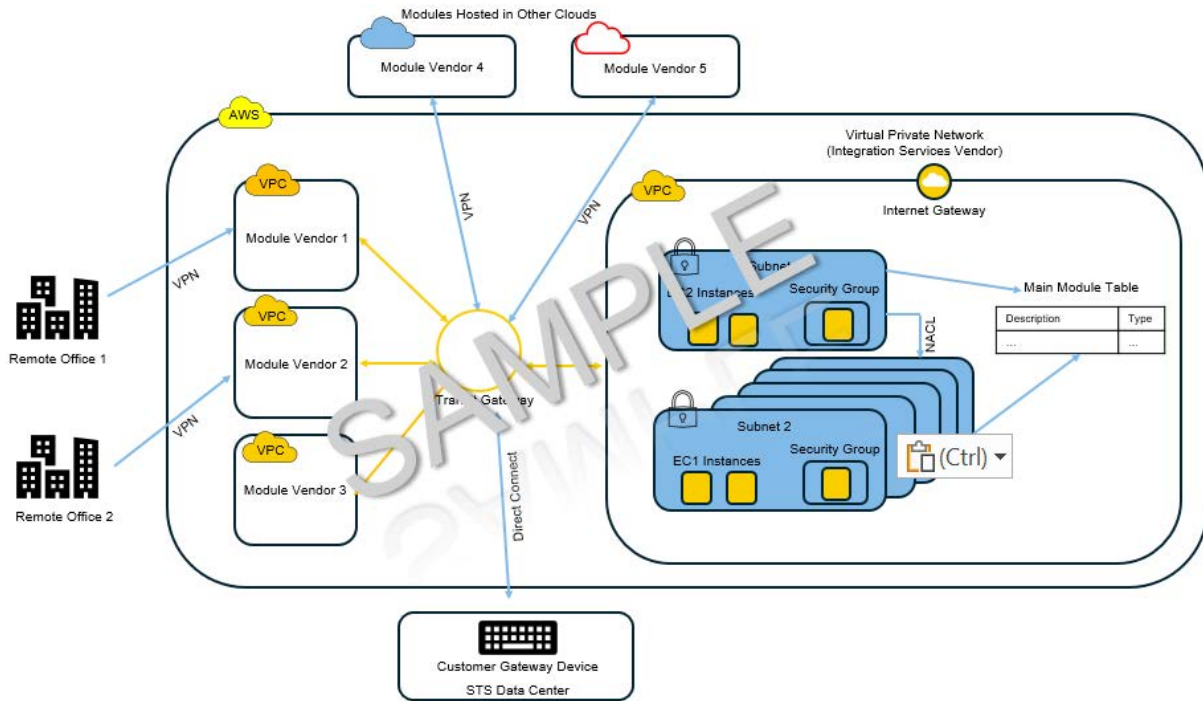


Figure 17 – Sample Network Topology

The Network Topology Model typically includes the following data elements:

- **Node:** Represents any device within the network, such as computers, routers, switches, or servers. Nodes are the fundamental building blocks of network topologies.
- **Link:** A connection between two nodes. Links can be physical (such as cables) or logical (such as wireless connections) and are essential for data transmission within the network.
- **Interface/Port:** The point of connection on a node, typically where links are terminated. Interfaces control data flow to and from the node.
- **Subnet:** A logically visible subdivision of an IP network. Subnets help organize and manage a network more efficiently by breaking it into smaller, manageable sections.
- **Network Segment:** A section of a computer network that is bounded by routers, switches, or hubs. It includes multiple nodes and can represent a portion of a larger network.
- **Topology Type:** Defines the structure of the network layout. Common types include:
 - o **Bus Topology:** All nodes connected to a single communication line.
 - o **Star Topology:** All nodes connected to a central node.
 - o **Ring Topology:** Nodes connected in a closed loop.
 - o **Mesh Topology:** Nodes interconnected, either fully or partially.
 - o **Hybrid Topology:** Combination of two or more topologies.
- **Routing Table:** Information stored in nodes, especially routers, that dictates the next hop for data packets to reach their specified destinations.
- **Network Protocols:** Rules and conventions for data transmission. Protocols define many aspects of data handling and transmission, including addressing and error checking.

- **Device Attributes:** Characteristics and properties of the nodes, including operating system, processing capabilities, IP addresses, and MAC addresses.
- **Configuration Data:** Settings and parameters that dictate how nodes and links operate within the network, including VLANs, QoS settings, and security policies.

When creating a Network Topology Model, the Contractor **SHOULD** use commonly recognized symbols to represent:

- Boundaries of secured areas of the network, including clouds, virtual private clouds (VPCs), subnets, data center environments
- Infrastructure nodes such as servers, user devices, data centers
- Network hardware and software nodes such as firewalls, routers, switches, gateways, load balancers, etc.
- Network connections between any of the nodes
- The Internet
- Use commonly recognized symbols in the Network Topology Model to represent secured-area boundaries (clouds, VPCs, subnets, data centers), infrastructure nodes, network hardware and software nodes, network connections, and the Internet.

9.6 Security Architecture Artifacts⁴⁷

9.6.1 Security in Other Domains⁴⁸

Security architecture is developed simultaneously with the other architectural domains and relies upon their artifacts.

- Data sensitivity is defined in the data dictionaries of the Conceptual, Logical, and Physical Data Models.
- Security access rights are defined for application user roles.
- Technology architecture, including the Deployment Model and Network Topology Model, is designed to ensure security.

9.7 Architecture Governance and Compliance

The architecture artifacts defined in this section are living governance instruments, not one-time deliverables. To ensure the delivered solution remains consistent with the approved architecture throughout the life cycle, the Contractor operates an architecture governance process spanning decision capture, compliance verification, reference-architecture alignment, and traceability. This process operates

⁴⁷ [NYS17-003-Notification-Standard-for-Certain-Types-of-Regulated-Data](#) describes state regulations for data reporting on sensitive data.

⁴⁸ The OHIP Security and Privacy Compliance Domain Document provides details on compliance requirements with NIST SP 800-53, ARC-AMPE and other Security Architecture requirements.

within OHIP's enterprise architecture governance and change control framework and complements the architecture approval gate defined in [Section 8.4.1](#).

The venue for these approvals is the OHIP Enterprise Architecture Review Board (EARB). Operated by the Office of the Chief Technology Officer, the EARB is the body through which the OHIP Enterprise Architecture approvals required by this standard are exercised: it reviews proposed solution designs and significant architecture decisions and issues a disposition of Approved, Approved with Conditions, Rejected, or Deferred, with conditions tracked to closure. This section names the review venue and its intake vehicle; it does not create approvals beyond those this standard already requires.

Submissions are made on the NYSDOH Medicaid Architecture Decision Record (ADR)/Solution Design Document (SDD) Intake Form (see Appendix A), which captures the problem statement, the proposed solution, the options considered including “do nothing,” consequences and trade-offs, a review-rubric self-assessment, security and data considerations, integration impacts, and any exceptions requested. The form is a design-decision intake, not a change-control instrument: an SDD submission supports the initial architecture approval gate of Section 8.4.1, and an ADR submission supports significant architecture decisions and design-option changes arising after that approval. Release-level change control ([Section 7.6](#)) and the Standards Deviation Request process ([Section 1.7.1](#)) remain separate mechanisms with their own triggers.

The Contractor **MUST**:

- Establish and maintain Architecture Decision Records (ADRs) that document significant solution and technical architecture decisions, including the decision, its context and rationale, the alternatives considered, and the resulting consequences, and reference the applicable ADRs from the RTM and design artifacts.
- Conduct architecture compliance reviews at OHIP-defined milestones to verify that the implemented solution conforms to the architecture artifacts approved under this section, documenting and remediating any deviations through OHIP's enterprise architecture governance and change control processes.
- Demonstrate alignment of the solution architecture with the OHIP Business Reference Architecture and the MITA framework, documenting and obtaining OHIP approval for any justified departures.
- Maintain traceability from solution architecture components to the requirements they satisfy (through the RTM) and to the deployed solution, so that each significant architecture element can be traced to its driving requirement and its implementation.

Management of technical debt arising from architecture decisions is governed by [Section 5.0](#), CODE-0060; the Contractor records architecture-related technical debt accordingly rather than maintaining a separate register.

10.0 Contact Information

Submit all inquiries and requests for future enhancements to the policy owner at:

Division of Operations and Systems (DOS) and Chief Technology Office (CTO)
11 Corporate Woods Boulevard
Albany, NY 12211
Telephone: TBD
Email: TBD

Statewide technology policies, standards, and guidelines may be found at the following website:

<https://its.ny.gov/policies>

11.0 Revision History

The following table records the revision history of this Consolidated Domain Standard. The seven component standards each maintained their own revision histories prior to consolidation. This table captures the life cycle of the domain document from initial publication onward.

Date	Version	Author	Description of Change
06/25/26	1.0	NYS TAS	Initial publication of the OHIP Solution Delivery Domain Document. Consolidates: Usability and Accessibility Standard, Requirements Management Standard, Policy and Business Rules Standard, Code Standard, Test Management Standard, Release Management Standard, and the SDLC Standard and incorporates compliance tiers (MUST/SHOULD/MAY) for all requirements.
08/17/26	1.1	NYS TAS	Named the Enterprise Architecture Review Board (EARB) as the venue for the Enterprise Architecture approvals this standard already requires; referenced the NYSDOH Medicaid ADR/SDD Intake Form and defined submission criteria (Sections 8.4.1 and 9.7; new requirements BTA-0024 and BTA-0025, both MUST). Appendix A (Table 40) artifact added; Appendix G acronyms updated (EARB added, SDD added, orphaned ARB entry replaced); Section 1.7.2 and Appendix B counts updated.
10/23/26	1.2	NYS TAS	Domain Document approved by DOH.

Appendices

Appendix A: Deliverables and Artifacts

The following tables identify the Deliverables and Artifacts associated with this Domain Document. Deliverables are contractually binding work products that the Contractor **MUST** submit to OHIP for formal review and approval. Artifacts are supporting work products (e.g. documentation, models, charts) that the Contractor **MUST** create, maintain, and retain, as determined by OHIP. The Contractor **MUST** provide Artifacts to OHIP upon request, or as otherwise directed, to support oversight, audits, and program needs.

The Contractor **MUST** produce and submit all applicable Deliverables and Artifacts identified in the format, structure, and cadence specified by OHIP or the Contract. OHIP may provide templates for the Contractor's use.

OHIP may modify the Deliverables and Artifacts required by this Appendix at any time to align with evolving program needs. Upon notice of changes, the Contractor **MUST** comply with all modifications and implement them promptly.

Table 32 – Usability and Accessibility Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Web Content Management Plan	The Contractor must develop a Web Content Management Plan documenting its approach to meeting WCAG 2.2 AA accessibility standards across web content, including content authoring standards, accessibility testing workflows, remediation procedures, and ongoing compliance monitoring.
Artifact	System Usability Scale (SUS) Questionnaire and Scoring Guide	The Contractor must use the SUS Scoring Tool and Guide to measure overall user useability perception, provide a quantifiable usability score, and enable fast, low-cost usability testing. The tool explains how to convert raw questionnaire responses into a standardized score, and how to interpret the resulting number.
Artifact	WCAG 2.2 AA Voluntary Product Accessibility Template (VPAT)	The Contractor must use the VPAT ® Revised 508 Edition to report the extent to which their information and communication technology (ICT) products conform to the accessibility standards specified in WCAG 2.2 A and AA levels as well as the US Federal Accessibility Standard for Information and Communication Technology (ICT) Products.

Table 33 – Requirements Management Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Requirements Management Plan	The Contractor must develop a Requirements Management Plan, which describes the framework for how requirements will be gathered, documented, classified, standardized, maintained, validated, and tested throughout the term of the awarded Contract. The Requirements Management Plan must outline the approach and methods for how requirements traceability will be achieved and maintained. The

Type	Name	Description
		<p>Requirements Management Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Requirements Identification</i> – A process for gathering and capturing requirements from stakeholders, including workshops, interviews, surveys, and document analysis • <i>Requirements Documentation</i> – The methods and tools used to document requirements in a clear, consistent, and accessible format. This typically includes the use of requirements specifications, user stories, or use cases. • <i>Requirements Analysis</i> – Techniques for analyzing requirements to ensure they are complete, feasible, and aligned with project objectives. This may involve validating requirements against business objectives and performing impact analysis. • <i>Requirements Prioritization</i> – A systematic approach to prioritizing requirements based on factors such as stakeholder importance, cost, risk, and project impact • <i>Requirements Traceability</i> – The establishment of traceability matrices or other tools to track the relationships between requirements and project artifacts • <i>Requirements Change Control</i> – Procedures for managing changes to requirements, including how changes are documented, evaluated, and approved • <i>Verification and Validation</i> – Processes for ensuring that requirements are correctly implemented and deliver the intended value. This includes reviews, inspections, and testing to confirm that deliverables meet defined requirements • <i>Requirements Communication</i> – Strategies for communicating requirements information to stakeholders, ensuring they are aware of requirement statuses, changes, and any associated decisions • <i>Business Analysis and Quality Assurance (BAQA) Plan</i> – Description of standards, templates, tools, a review cadence, metrics, and escalation paths to govern the BAQA program • <i>Assumptions, Constraints, and Dependencies</i> – Describe of any assumptions, constraints, and dependencies that may impact requirements management

Type	Name	Description
Deliverable	Requirements Traceability Matrix (RTM)	The Contractor must maintain a Requirements Traceability Matrix (RTM) that serves as a comprehensive tool for tracking the relationship and lineage of requirements throughout the SDLC and project life cycle, ensuring that each requirement (functional, non-functional, business, etc.) is addressed from inception through to implementation and testing of the solution. The Contractor must be able to generate a report of changes to the RTM. This deliverable facilitates project transparency and accountability by clearly mapping requirements to their corresponding design, development, and verification artifacts. The RTM must be maintained in the Department's preferred tool.
Deliverable	Requirements Traceability Matrix Update Report	The Contractor must develop a Requirements Traceability Matrix Update Report to document RTM changes as requirements from the initial RTM are elaborated on. The report must have traceable relationships with other components (e.g., data, reporting, shared services, interfaces, etc.) as they are defined, and as complex one-to-many relationships are created.

Table 34 – Policy and Business Rules Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Business Rules Documentation	<p>The Contractor must develop Business Rules Documentation that serves as the single authoritative source for all business rules. Business Rules Documentation may include, but is not limited to:</p> <ul style="list-style-type: none"> • <i>Business Rules Catalog</i> – Provides complete inventory of all business rules with unique identifiers, rule descriptions, categorization, source authority references, ownership, status, and version history • <i>Business Rules Specification Document</i> – Provides detailed rule definitions necessary for implementation, including rule logic, contextual interpretation, assumptions, dependencies and interactions with other rules, and mapping to system modules and underlying data elements • <i>Business Rules Traceability Matrix</i> – Establishes end to end traceability by linking each business rule to its originating policies or regulations, associated business and system requirements, supported business processes and system functions, and corresponding test cases used for validation
Deliverable	Policy and Business Rules Implementation Plan	The Contractor must develop a Policy and Business Rules Implementation Plan that describes the approach to execute and maintain business rules and policies within key business

Type	Name	Description
		<p>domains. The Policy and Business Rules Implementation Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Rules Scope and Summary</i> – Describe which business domains and systems are in scope and outline any out-of-scope areas or transitional exceptions • <i>Current-State Rules Inventory and Gap Analysis</i> – Inventory all existing in-scope business rules regardless of current location (code, engines, configuration, spreadsheets, documents, or undocumented practice), trace each to its authoritative source or flag it as unsourced for OHIP disposition, identify conflicts, duplicates, and obsolete rules, define the target-state disposition of each (migrate, re-author, consolidate, retire), and assess migration impact on operating functionality and downstream consumers • <i>Rule Inventory and Prioritization</i> – Provide a catalog of new or modified rule groupings with unique identifiers, including priority levels (must-have versus nice-to-have) based on regulatory deadlines or business impact • <i>Timeline and Milestones</i> – Outline high-level phases (design, authoring, testing, deployment) with target dates, and list key milestones including governance checkpoints (e.g., policy sign-off, User Acceptance Testing (UAT) completion, go-live approval) • <i>Rule Implementation Team</i> – Describe the implementation teams(s) key roles and responsibilities, number of rule implementers, and their relevant qualifications to write human-readable policy rules in the Medicaid domain. • <i>Rule Testing Methods</i> – Describe how testing will be implemented to provide full coverage and meet the quality control requirements. Business rules must also reference the source/authority of the rule that is being tested for validation • <i>Rule Deployment Methods</i> – Describe how deployment (including pre-deployment checks, deployment steps, and post-deployment) validation will occur • <i>Rule Change Analysis</i> – Describe how proposed changes to rules can be tested against existing data for overall impact on the business • <i>Tools</i> – List all the tools that the Contractor wishes to use. This includes rule engines and rule management tools

Type	Name	Description
		<ul style="list-style-type: none"> • <i>Rule Reporting and Performance Metrics</i> – List runtime metrics to track (e.g., rule execution times, exception rates), associated business KPIs (e.g., percentage of claims auto adjudicated), and schedule for periodic reviews and audits • <i>Risk and Mitigation</i> – Enumerate potential technical or regulatory risks (e.g., delayed data feeds, ambiguous rule text) and outline mitigation plans (e.g., fallback behaviors, escalation paths) • <i>Assumptions, Constraints, and Dependencies</i> – Describe any assumptions, constraints, and dependencies that may impact policy and business rule implementation
Artifact	Business Rules Catalog (BRC)	The complete, current inventory of rules and rule artifacts maintained in the State-managed rules repository. The Business Rules Catalog must be exportable and reviewable by OHIP on demand; static documents and spreadsheets must not serve as the authoritative repository.

Table 35 – Code Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Patch Management Plan	<p>The Contractor must develop a Patch Management Plan that documents the approach for identifying, acquiring, installing, and verifying software updates and patches on system components. The Patch Management Plan ensures system security, stability, and compliance by addressing vulnerabilities, improving performance, and implementing enhancements. The Patch Management Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Patch Management Procedures</i> – Comprehensive documentation of the patch management procedures, durations, flowcharts and workflows that conform to ITIL best practices. This also includes procedures for emergency patches. • <i>Roles and Responsibilities Matrix</i> – Detailed matrix identifying the duties and functions of the patch management team, including system administrators, security personnel, and other relevant stakeholders • <i>Patch Identification and Acquisition</i> – Process for identifying vulnerabilities and for acquiring patches from trusted Contractors or sources • <i>Patch Evaluation and Testing</i> – Criteria for evaluating patches in terms of severity, necessity, and impact; specify the use of a segregated testing environment to verify patches prior to deployment; outline procedures for testing patches, including functionality and security checks

Type	Name	Description
		<ul style="list-style-type: none"> • <i>Patch Deployment Schedule</i> – Details of the regular schedule for deploying patches, in addition to defining how patches are prioritized based on the severity of vulnerabilities • <i>Verification and Documentation</i> – Process of verifying and documenting successful patch deployment • <i>Reporting</i> – Details of the reporting mechanisms to communicate status and effectiveness of patch management activities • <i>Rollback Strategy</i> – Describe the process for rolling back patches in case of failure, including restoration from backups and effective communication to the relevant stakeholders. • <i>Assumptions, Constraints, and Dependencies</i> – Description of any assumptions, constraints, and dependencies that may impact Patch Management

Table 36 – Test Management Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Test Management Plan	<p>For each Release, the Contractor must develop a comprehensive Test Management Plan, including a detailed test schedule, for review and approval. The Test Management Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Test Objectives and Scope</i> – Define the objectives and scope of the testing phase, specifying which parts of the system will be tested • <i>Roles and Responsibilities</i> – Details who is responsible for various aspects of the testing process, including, but not limited to, preparatory steps for each type of test, procedures for performing the testing, and the process for developing test scenarios, cases, and scripts • <i>Test Case Development</i> – Provide the approach for developing test cases and scripts, in addition to defining the process for test case reviews and approvals to ensure all test cases meet predefined standards prior to execution • <i>Entry and Exit Criteria</i> – Clearly defines the entry and exit criteria for each test to ensure tests are only executed when all predefined criteria have been met, and that they are considered complete only when all required outcomes are achieved and verified • <i>Testing Tools and Automation</i> – List the testing tools and automation frameworks to be used, if applicable, and includes schedules for tool setup, configuration, and validation to ensure readiness before testing begins (all tools must be updated with this information) • <i>Test Data Requirements</i> – Includes detailed requirements for test data, including any data

Type	Name	Description
		<p>creation or data masking efforts needed to protect sensitive information, and outlines the process for creating and refreshing test data to ensure it is clean and consistent and supports testing procedures (the Contractor is responsible for creating all test data, unless otherwise agreed to, and ensuring its accuracy)</p> <ul style="list-style-type: none"> • <i>Metrics and Reporting</i> – Defines how testing progress and outcomes will be measured, specifying the format and frequency of test status reports, and includes metrics such as pass/fail rates, defect density, and coverage levels • <i>Communication Plan</i> – Outlines how progress will be reported, including scheduling regular status meetings and including mechanisms for issue escalation to ensure clear and ongoing communication • <i>Content for All Test Types</i> – Outlines the overall management of each test type; clearly describes how each is designed to fully test various functions and features of the system; and traces each requirement to one or more test cases to ensure each requirement has been implemented and tested successfully • <i>Assumptions, Constraints and Dependencies</i> – Describes the assumptions, constraints, and dependencies that may impact test management

Type	Name	Description
Deliverable	Test Results Report	<p>The Contractor must develop a Test Results Report, which is a detailed document that provides an overview of the outcomes from all testing types executed during the testing phase. The Test Results Report must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Test Results Overview</i> – Documentation of the outcomes of executing test cases, indicating whether a test has passed or failed based on expected results, and offering insights into solution performance • <i>Defect Resolution Summary</i> – Comprehensive information on defects identified during testing, including the resolution approach and analysis for further corrective actions • <i>Testing Metrics and Analysis</i> – Compilation of detailed status and progress information, including metrics on test coverage, defect density, and resolution rates, enabling stakeholders to assess the effectiveness of tests • <i>Decision Support</i> – Insights provided by the report for informed decision-making regarding solution deployment, based on test outcomes and performance analysis • <i>Contextual Information</i> – Information necessary for understanding how the test results relate to the expected solution performance and deployment readiness

Table 37 – Release Management Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Defect Management Plan	<p>The Contractor must develop a Defect Management Plan that provides the approach, activities, and roles for handling defects, in addition to defining the activities associated with the defect management strategy, reporting, conducting the triage, and performing actions to resolve defects. The Defect Management Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Defect Management Policy and Procedures</i> – Comprehensive documentation of the defect management policy, procedures, durations, and workflows that conform to ITIL best practices • <i>Defect Management Process Flowchart</i> – Visual representation of the defect management process flow, illustrating each step from defect identification to resolution • <i>Roles and Responsibilities Matrix</i> – Detailed matrix outlining the roles and responsibilities of all stakeholders involved in the defect management process, and process for coordinating response teams for effective defect resolution • <i>Defect Identification and Classification System</i> – Process for identifying and documenting defects,

Type	Name	Description
		<p>their descriptions, and the number of defects/bugs in the backlog by non-production or production environment, including:</p> <ul style="list-style-type: none"> o <i>Prioritization and Escalation Procedures</i> – Process for assessing the impact severity level and priority of defects on each business process o <i>Defect Log</i> – Process for maintaining, tracking, and updating the defect log, including the details and resolution status with dates and times of defects • <i>Continuous Improvement Plan</i> – Steps for implementing practical activities to reduce the occurrence and impact of defects in business operation, including aligning with M&O planning effort • <i>Assumptions, Constraints, and Dependencies</i> – Description of any assumptions, constraints, and dependencies that may impact defect management
Deliverable	Post-Release Documentation	<p>The Contractor must develop Post-Release Documentation that provides an overview of updates to relevant design documentation. Post-Release Documentation must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Documentation Update Overview</i> – A high-level summary listing all updates made to design and technical documents post-release • <i>Source Information Details</i> – Identification of source information used for updates, including versions of documents utilized • <i>Functional Design Documents (FDDs) Updates</i> – Updates to functional design documents or their equivalents, ensuring alignment with the latest release • <i>Design Documentation Updates</i> – Updates to design documents or their equivalents • <i>Requirements Traceability Matrix (RTM) Updates</i> – Updates to the RTM to reflect changes or additions to requirements made during and after the release process, ensuring continued traceability between requirements, design, and test artifacts • <i>Storyboards for All Screens Updates</i> – Updates to the storyboard documentation for user interfaces, ensuring all screen designs are current • <i>Updates to other System Artifacts</i> – Updates to the lessons learned register, failure/fix analysis, estimate/actual budget analysis, and other artifacts as requested • <i>Summary Walkthrough</i> – A provision for conducting a walkthrough meeting of the documentation updates, as needed or upon request
Deliverable	Release Implementation Completion Report	The Contractor must develop a Release Implementation Completion Report that serves as a formal record of a

Type	Name	Description
		<p>Release's deployment into production. The Release Implementation Completion Report must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Release Overview</i> – A high-level summary of the release, including the objectives, scope, and key features addressed • <i>Design and Development Compliance</i> – Details demonstrating that the release has met specific design and development requirements, ensuring all features are implemented as intended • <i>Testing and Validation</i> – Summary of testing activities conducted, including unit, integration, and user acceptance testing, along with outcomes that confirm the release's functionality and stability • <i>Approval for Production Deployment</i> – Confirmation that the release was reviewed and approved for production deployment by relevant stakeholder • <i>Impact Assessment</i> – Comprehensive overview of the release's impact on existing systems, including any enhancements or changes to workflows and processes • <i>Post-Implementation Challenges</i> – Identification and analysis of any issues or challenges encountered after implementation, including steps taken for resolution and prevention • <i>Operational Readiness</i> – Details on the readiness of support teams and end-users, including any training or resources provided to ensure successful adoption • <i>Recommendations and Improvements</i> – List of suggested improvements and recommendations for future releases, based on lessons learned during the implementation process • <i>Contact Information</i> – Details of who to contact for more information or to discuss any issues related to the release implementation
Deliverable	Release Management Plan	<p>The Contractor must develop a Release Management Plan that includes the approach and procedures for managing and tracking Releases, deployment of Releases to system environments, and the process for obtaining approval for production deployments of Releases. The Release Management Plan must include, but is not limited to:</p> <ul style="list-style-type: none"> • <i>Release Procedures</i> – The framework for overseeing the release management process, including the roles and responsibilities of the governance bodies and stakeholders involved in authorizing and reviewing releases • <i>Key Roles and Responsibilities</i> – A detailed assignment of roles and responsibilities related to release management, including those responsible for planning, building, testing, and deploying releases.

Type	Name	Description
		<p>This ensures accountability and coordination among team members</p> <ul style="list-style-type: none"> • <i>Scope Prioritization</i> – The process for prioritizing features and functionalities to be included in each release. This involves engaging stakeholders to assess the value, impact, and resource requirements of proposed changes • <i>Planning Releases</i> – Procedures for developing release schedules and plans, including timelines, milestones, and dependencies. This ensures that releases are strategically aligned with project and organizational objectives • <i>Building Releases</i> – The creation and assembly of release components, including configuration, integration, and version control. This subsection emphasizes the importance of consistency and accuracy in building release packages • <i>Testing Releases</i> – The process for testing releases to ensure they meet quality standards and function as expected. This includes various testing activities such as unit testing, integration testing, system testing, and user acceptance testing • <i>Preparing Releases for Deployment</i> – Steps for preparing releases for deployment, including final quality checks, documentation, and stakeholder sign-off. This ensures readiness for production. • <i>Deployment Process (including Rollback)</i> – The procedures for deploying releases to the production environment, including steps, tools, and roles involved. It also includes Rollback procedures to restore previous versions if necessary, minimizing disruption • <i>Validating Releases</i> – Methods for validating that the deployed release meets all requirements and performs as intended in the production environment. This includes post-deployment testing and stakeholder feedback collection • <i>Closing Releases</i> – The process for closing out releases, including documenting lessons learned, conducting reviews, and archiving release artifacts. This supports continuous improvement and captures valuable insights for future releases • <i>Assumptions, Constraints, and Dependencies</i> – Description of any assumptions, constraints, and dependencies that may impact release management

Type	Name	Description
Deliverable	Release Plan	<p>The Contractor must develop a Release Plan that strategically outlines how a product or feature is introduced to end-users and ensures all aspects of the specific Release are planned, well-coordinated, and effectively executed. The Release Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Deployment Items</i> – Describes the list of infrastructure, application, and software components to be deployed and their associated features • <i>Release Schedule</i> – A holistic view of the Release Schedule with a timeline, deployment dates, dependencies, pre-deployment activities, and key milestones • <i>Change Control</i> – Describes the process for release-level change and escalation procedures • <i>Contingency Plan</i> – Describes the procedures and protocols for addressing potential collisions, delays, or other unforeseen circumstance, in the event of a Release failure or other issue • <i>Assumptions, Constraints and Dependencies</i> – Describe any assumptions, constraints, and dependencies that may impact release planning
Artifact	Bill of Materials	<p>A complete inventory of the hardware, software, COTS products, libraries, and configuration components that make up the solution release, with versions and dependencies, enabling OHIP to understand and reproduce the deployed environment.</p>

Table 38 – SDLC Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Alternative Solutions Matrix	<p>The Contractor must develop an Alternative Solutions Matrix that provides a structured comparison of the proposed design alternatives proposed during the exploration step of the Iterative Design cycle. The Alternative Solutions Matrix must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>List of Alternatives</i> – Each option identified during exploration • <i>Evaluation Criteria</i> – Key factors examined such as cost, feasibility, functional and non-functional requirements alignment, etc. • <i>Scoring or Ranking</i> – The scoring or ranking methods used to quantitatively and/or qualitatively assess each criterion, and the scores assigned to each design alternative • <i>Pros and Cons</i> – A summary list of strengths/weaknesses allocated to each alternative • <i>Recommendation Notes</i> – Initial observations on which options seem most promising

Type	Name	Description
Deliverable	Daily Production Status Report	<p>The Contractor must develop a Daily Production Status Report, which provides a snapshot of the daily solution operational status. The Daily Production Status Report must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>System Status Overview</i> – High-level summary of the operational status, including any downtime or performance issues experienced during the reporting period • <i>Maintenance Activities</i> – Detailed information on maintenance work conducted, including both scheduled and unscheduled maintenance tasks • <i>Incident and Problem Management</i> – Summary of any incidents or problems that have arisen, including a brief description, the impact on operations, how the problem was resolved, and steps taken to prevent recurrence • <i>Change Control</i> – Details on any changes made to the systems, including software updates, infrastructure upgrades, or configuration changes, as well as descriptions of the changes, the reason(s) for the changes, and any anticipated impact on system operations • <i>Security Updates</i> – Summary of any security-related patches applied, vulnerabilities addressed, or incidents that occurred • <i>Compliance and Audit Information</i> – Any compliance or audit activities that have taken place or are planned, if applicable • <i>Upcoming Scheduled Maintenance</i> – List of any scheduled maintenance or system changes for stakeholder awareness • <i>Action Items and Recommendations</i> – List of recommended actions or decisions that need to be addressed by the State • <i>Contact Information</i> – Details of who to contact for more information or to discuss any issues raised within the report

Type	Name	Description
Deliverable	Data Archival Verification Report	<p>The Contractor must submit a Data Archival Verification Report, which provides formal confirmation that legacy or superseded data has been successfully archived in accordance with approved retention, security, and regulatory requirements before system decommissioning. The Data Archival Verification Report must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Archival Scope and Datasets</i> – Identification of systems, tables, files, and records subject to archival and verification • <i>Source to Archive Traceability</i> – Verification, with evidence, that archived data is complete and reconcilable to source systems • <i>Retention and Compliance</i> – Verification, with evidence, that archival methods meet Department, State, and CMS retention requirements. The archived data must also be protected according to security and privacy data classification standards. • <i>Access and Retrieval Readiness</i> – Verification, with evidence that archived data is searchable, retrievable, and accessible by authorized users
Deliverable	Deployment Plan	<p>The Contractor must develop a Deployment Plan that guides the transition of tested changes or new solutions into the production environment. It aims to ensure smooth deployments, minimize risks, and align with business goals. The Deployment Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Strategy and Objectives</i> – Outline of the deployment approach, such as phased or pilot methods, aligned with business goals • <i>Task Breakdown and Schedule</i> – Detailed list and timeline of tasks, integrated into the overall project schedule for seamless coordination • <i>Cutover Activities</i> – Identification of final data migration, system configuration validation, user access setup, backup preparation, technical and business process verification, training and support readiness, communication updates, system lockdown, a go/no-go decision, and continuous post-transition monitoring to ensure a seamless and controlled transition to the new solution • <i>Resource Management</i> – Plan for the availability and assignment of resources, including infrastructure, software, personnel, and tools • <i>Roles and Responsibilities</i> – Clear documentation of the duties of Contractor and State teams to ensure accountability • <i>Risk and Issue Management</i> – Identification and management of potential risks and issues, with strategies to mitigate and resolve them to prevent disruptions and minimize impact on users

Type	Name	Description
		<ul style="list-style-type: none"> • <i>Blackout Periods</i> – Specification of times when solutions or services will be temporarily unavailable • <i>Rollback Procedures</i> – Steps to revert changes if problems occur, ensuring solution stability • <i>Pre-Deployment Activities</i> – Checklist of tasks and Deployment Procedure Reviews for validation and readiness before deployment begins. • <i>Post-Deployment Verification</i> – Procedures to confirm the deployment was successful through testing • <i>Impact Analysis</i> – Assessment of how deployment affects current solutions and processes, guiding necessary adjustments • <i>Documentation</i> – All documents related to the deployment, including plans, approvals, and records of issues, for transparency and future reference • <i>Assumptions, Constraints, and Dependencies</i> – Description of any assumptions, constraints, and dependencies that may impact deployment
Deliverable	Deployment Procedure Review Results	<p>The Contractor must develop Deployment Procedure Review Results, a detailed summary documenting the performance and outcomes of each Deployment Procedure Review conducted during the Cutover process. Deployment Procedure Reviews include testing interfaces with the State and other third parties, as well as ensuring data synchronization between solutions. The Deployment Procedure Review Results must include a list of successes and failures encountered, along with a description of any corrective actions needed before the subsequent Deployment Procedure Review.</p>

Type	Name	Description
Deliverable	Hypercare Completion Report	<p>The Contractor must develop a Hypercare Completion Report that includes, but is not limited to:</p> <ul style="list-style-type: none"> • <i>Overview of the Hypercare Period</i> – Description of the release deployed, including the duration of the Hypercare window and the scope of applications, modules, or functionalities monitored • <i>Summary of Hypercare Activities</i> – Describe the support, monitoring, and stabilization activities performed during the Hypercare period, including coordination with technical and business teams • <i>Issue Management Summary</i> – Summarize all issues logged, categorized by severity, and detail how each was resolved, deferred, or transitioned to the backlog • <i>Performance and Stability Metrics</i> – Provide performance data such as system availability, interface stability, batch job outcomes, and any performance improvements or degradations observed during Hypercare • <i>User Feedback and Adoption</i> – Document user feedback, including reported issues, trends, usability challenges, and any training or support provided to end-users • <i>Risks, Mitigations, and Outstanding Items</i> – Identify residual risks, technical debt, and unresolved items, including recommended mitigation actions and items requiring transition to steady state support • <i>Go Forward Recommendation</i> – Provide a final assessment of system stability, confirm whether Hypercare exit criteria have been met, and document approvals or sign offs required from stakeholders
Deliverable	Hypercare Plan	<p>The Contractor must develop a Hypercare Plan that outlines the post-Deployment activities implemented to support the stabilization of new functionality. The Hypercare Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Hypercare Processes Description</i> – Overview of the processes and management strategies for Hypercare functions, highlighting the responsibilities of the Contractor • <i>Duration and Exit Criteria</i> – Specification of the Hypercare period's timeline (e.g., 30, 60, or 90 days post-deployment) and the specific, measurable conditions (e.g., zero open Severity 1 or 2 incidents, system uptime metrics met, knowledge transfer completed) required to formally conclude Hypercare and transition to standard operations • <i>Incident Reporting and Resolution</i> – Detailed process for documenting, tracking, and resolving all incidents with severity level one (1) and two (2) during the Hypercare period • <i>Outstanding Incidents Management</i> – Approach to addressing any remaining incidents with severity

Type	Name	Description
		<p>level three (3) and four (4) that may persist beyond the Hypercare period's conclusion</p> <ul style="list-style-type: none"> • <i>Staffing Strategy for Support</i> – Identification and roles of staff assisting with modifications and issue resolution during the Hypercare period • <i>Production Defect Remediation Plan</i> – Process for handling, documenting, tracking, and resolving defects identified in testing or operations, categorized by business criticality and resolution timelines • <i>Rollback Preparedness</i> – Guidelines for rolling back deployed functionality if necessary, including correcting critical defects and obtaining approval for the remediation of medium and low defects • <i>Knowledge Transfer</i> – Ensures availability of Contractor staff who provided configuration and customization services to resolve incidents and transfer knowledge during the Hypercare period • <i>Assumptions, Constraints, and Dependencies</i> – Description of any assumptions, constraints, and dependencies that may impact Hypercare
Deliverable	Iteration Summary Document	<p>The Contractor must develop an Iteration Summary Document that provides the summary outcome of a design/develop iteration. The Iteration Summary Document must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Iteration Objectives</i> – A summary of the what the design iteration aims to achieve • <i>Activities Completed</i> – Documents the activities completed during the iteration: exploration, option selection, demos, and assessments • <i>Key Findings</i> – Provides the key insights gained from evaluations and feedback • <i>Decision Outcome</i> – Summarizes the selected design alternative or provides the rationale for an upcoming re-iteration • <i>Next Steps</i> – Indicates the next steps to either proceed to development or prepare for a new design iteration
Deliverable	Key Performance Indicators (KPIs)	<p>The Contractor must develop KPIs in collaboration with the Department. KPIs are a set of quantifiable measures used to evaluate Contractor performance against projected business outcomes and contractual obligations. Required KPIs include, but are not limited to, measures of usability and the effectiveness of system features.</p>

Type	Name	Description
Deliverable	Legacy Sunsetting Plan	<p>The Contractor must submit a Legacy Sunsetting Plan, which defines the controlled approach for transitioning legacy systems to read only mode and decommissioning them in a manner that preserves continuity, compliance, and operational stability. The Legacy Sunsetting Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Systems in Scope</i> – Identification of legacy applications, interfaces, reports, and infrastructure components to be decommissioned • <i>Transition and Cutover Strategy</i> – Description of how operations move from the legacy system to the replacement solution • <i>Data Migration and Archival Alignment</i> – Verification, with evidence that required data migration and archival activities are completed prior to shutdown • <i>Read Only and Shutdown Timeline</i> – Milestones for read only enforcement, interface severance, and final decommissioning • <i>User and Stakeholder Communication</i> – Planned notifications, training considerations, and support impacts for affected users • <i>Risk and Contingency Management</i> – Identification of retirement risks and fallback or rollback strategies • <i>Acceptance Criteria and Final Authorization</i> – Conditions required to be met before irreversible shutdown activities are executed and approved by the Department • <i>Assumptions, Constraints, and Dependencies</i> – Describe the assumptions, constraints, and dependencies that may impact sunsetting a legacy system
Deliverable	Operational Readiness Review	<p>The Contractor must support the development of any materials required for the Operational Readiness Review (ORR) and any materials required by CMS for certification now or in the future. ORR is a comprehensive evaluation process that assesses the preparedness of a solution for official CMS certification and operational deployment. ORR documentation must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Official Certification Request Letter</i> – A formal request submitted to begin the certification review process. It must include the solution's date of record acceptance, the effective date for certification approval, and a declaration • <i>System Acceptance Letter</i> – A document addressed to the Contractor or development team, confirming that the system/module was accepted as fully operational at least six months before the certification review date • <i>Project Management Documentation</i> – Includes Monthly Project Status Reports that highlight

Type	Name	Description
		<p>indicators of project health, such as progress tracking and user feedback integration.</p> <ul style="list-style-type: none"> • <i>Technical Documentation</i> – Comprises essential artifacts such as the Test Management Plan and Testing Results, Deployment Plan, Defect and Risk List, and Independent Security Audit to ensure comprehensive validation of technical readiness • <i>Compliance and Data Quality Assurance</i> – Demonstrates adherence to all state and federal requirements, including maintaining complete and accurate data submissions and meeting data quality targets • <i>Security and Privacy Controls Assessment</i> – An independent audit confirming compliance with security standards such as NIST and HIPAA. It includes tasks such as vulnerability assessments and penetration testing • <i>Risk Management and Mitigation Plans</i> – Detailed risk assessments and plans for mitigating identified risks and resolving defects, ensuring minimal operational impact • <i>Training and Knowledge Transfer Plans</i> – Documentation outlining the processes for transferring operational knowledge to end-users and staff to effectively support and maintain the system • <i>Incident and Breach Reporting Procedures</i> – Established protocols for immediate and effective reporting of security incidents and breaches to ensure continual compliance and system integrity • <i>Go-Live Checklist</i> – A comprehensive list that verifies all necessary pre-launch activities have been completed and confirmed. This includes system testing, verification of configurations, backup procedures, stakeholder approvals, and communication plans for different scenarios that may occur post-deployment.
Deliverable	Source Code	<p>The Contractor must develop Source Code in the Department's Repository. Source Code consists of human-readable programming instructions crafted to meet functional and regulatory needs, derived from detailed design documentation. Source Code transforms design specifications into executable code that supports and operates solutions.</p>

Type	Name	Description
Deliverable	Transition-In Execution	The Contractor must provide Transition-In Execution to systematically implement all outlined operations, services, and functions are addressed systematically. This deliverable facilitates project transparency and accountability by orchestrating the execution of tasks that align with strategic objectives defined within the Transition-In Plans. It must include comprehensive coordination and management of resources and activities to achieve a seamless transition, adhering to established timelines and ensuring that all contractual requirements are met. The execution efforts must be maintained in accordance with the preferred methods and practices to ensure successful implementation and integration.
Deliverable	Transition-In Management Plan	<p>The Contractor must develop a Transition-In Management Plan (TIMP) that details the solution-level execution of the master Transition-In Plan governed by the OHIP Program and Project Management Domain Document, Section 2.18 (Transition Management), addressing management of the transition into new operations. The Transition-In Management Plan must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Transition Approach</i> – High-level summary of the approach and activities to be conducted by the Contractor to ensure a smooth transition • <i>Assumptions and Constraints</i> – Identification of key assumptions and constraints that influence the transition planning and execution • <i>Planned Transition Activities</i> – Detailed information on planned transition activities and tasks necessary for a successful transition • <i>Roles and Responsibilities</i> – Clarification of the roles and responsibilities of the State and the Incumbent Contractor, ensuring coordinated efforts • <i>Contractor Collaboration</i> – Strategies for effective collaboration with the Incumbent Contractor to facilitate knowledge transfer and operational continuity • <i>Risk Management</i> – Identification of potential risks and corresponding mitigation measures to address challenges during the transition • <i>Transition Training and Knowledge Transfer</i> – Activities aimed at training and transferring knowledge to new teams involved in the transition • <i>Readiness Reviews</i> – Process for conducting readiness reviews to gain approval for the Transition Cutover, ensuring all preparatory tasks are complete • <i>Assumptions, Constraints, and Dependencies</i> – Description of any assumptions, constraints, and dependencies that may impact Transition-In Management
Deliverable	Transition-In Weekly Status Reports	The Contractor must develop Transition-In Weekly Status Reports that provide a regular update on the progress of the

Type	Name	Description
		<p>Transition-In period. These reports must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Transition-In Schedule Review</i> – High-level summary of the status of the Transition-In Schedule, including alignment with planned activities and timelines • <i>Status Report on Transition-In Activities</i> – Detailed information on the activities conducted during the Transition-In period, highlighting key accomplishments and ongoing tasks • <i>Risk and Issue Management</i> – Summary of any risks and issues that have arisen, including a brief description, the impact on the transition process, how the issues are being addressed, and steps taken to prevent recurrence • <i>Overall Progress Discussion</i> – Insights into the overall progress of the Transition-In effort, assessing achievements against planned objectives and identifying any deviations • <i>Meeting Agenda Development</i> – Preparation of the agenda prior to each meeting to ensure focused and productive discussions • <i>Meeting Notes Distribution</i> – Recording and sharing of the meeting notes following each meeting to maintain transparency and keep all stakeholders informed
Deliverable	Usage and Performance Analytics Report	The Contractor must develop a Usage and Performance Analytics Report, which details insights and metrics about the performance efficiency of the solution and how it is being utilized by end-users. It must include, at a minimum, the following data on user engagement patterns, such as peak usage times, common user actions, and pathways through the application, alongside metrics that capture operational health, such as response times, error rates, and server uptime.
Deliverable	User Engagement Surveys and Analysis	The Contractor must develop User Engagement Survey(s) and Analysis, which include a comprehensive array of questions aimed at gauging the satisfaction and engagement levels of solution users, including ease of use by various system user groups, responsiveness, analytics support, and reliability. The Contractor must develop the surveys and coordinate all associated logistics. The Contractor must also collect and assess the responses and provide survey results, key observations, and recommendations for improvement.
Artifact	Cycle Evaluation Report	The Contractor must develop a Cycle Evaluation Report when executing a Spiral Solution Development Life Cycle methodology. The Report is a formal assessment produced at the conclusion of a spiral cycle that documents results achieved against planned objectives, evaluated residual risks, deliverables produced, feedback, and provides a recommendation on whether to proceed, adjust, or terminate

Type	Name	Description
		<p>the next spiral. The Cycle Evaluation Report must include, at a minimum, the following:</p> <ul style="list-style-type: none"> • <i>Spiral Overview</i> – Summary of the spiral scope, objectives, and planned outcomes as defined for the approved spiral cycle • <i>Spiral Evaluation</i> – Evaluation of actual results against planned scope, schedule, quality, resource and risks (including any residual risks or exposure) • <i>Changes and Deviations</i> – Description of approved changes or deviations from the Sprint Project Plan, including rationale and impact • <i>Issues, Defects, and Lessons Learned</i> – Summary of significant issues, defects, and lessons learned affecting current or future spirals • <i>Deliverables</i> – List and status of deliverables produced during the spiral, including acceptance, conditional acceptance, or outstanding items • <i>Stakeholder Feedback and Review Outcomes</i> – Summary of feedback from demonstrations, reviews, or formal evaluations conducted during the spiral • <i>Readiness Assessment for Next Spiral</i> – Assessment of technical, operational, and organizational readiness to proceed, including the Contractor's recommendation to proceed, adjust, pause, or terminate the next spiral • <i>Action Items and Next Steps</i> – Identified actions, owners, and timelines required to support the next spiral
Artifact	Definition of Done	The Contractor must develop a Definition of Done, which is a clear and concise list of requirements that user stories must meet for them to be considered complete. The definition ensures quality and consistency across the project deliverables.
Artifact	Epic	The Contractor must develop an Epic, which is a large user story that can be broken down into smaller, more manageable stories. Epics help organize the product backlog in terms of broad features or functionalities.
Artifact	Post Go-Live Adoption Measurement Dashboard	A dashboard presenting post-go-live adoption and usage metrics (active users, feature adoption, transaction volumes, and support trends) to measure realized value and inform stabilization.
Artifact	Product Backlog	The Contractor must develop a Product Backlog, which is an evolving list of features, enhancements, and fixes required for the product, prioritized by importance to guide development activities. It is used to plan and manage Agile sprints.

Type	Name	Description
Artifact	Product Backlog Report	The Contractor must develop a Product Backlog Report, which is a dynamic and prioritized list of all the features, enhancements, fixes, and requirements needed to deliver a successful product. It is continuously updated throughout the project life cycle, with items added, removed, prioritized, or refined as necessary.
Artifact	Release Notes	The Contractor must develop Release Notes, which are a summary of the enhancements, new features, bug fixes, and any other changes included in a particular release of a solution.
Artifact	Spiral Risk Register	The Contractor must develop a Spiral Risk Register, which is an inventory of identified risks at the beginning of each spiral cycle. The register should include, but not be limited to: capturing likelihood, impact, the mitigation strategy, ownership, and the status.
Artifact	Sprint Backlog	The Contractor must develop Sprint Backlog, which is a subset of the product backlog that a team commits to completing in a sprint, detailing tasks and activities needed to deliver the sprint goals.
Artifact	Sprint Backlog Report	The Contractor must develop a Sprint Backlog Report, which is a subset of the product backlog. The report contains the task the team committed to deliver in a sprint, who is working on them, and how they are progressing.
Artifact	Sprint Burndown Chart	The Contractor must develop a Sprint Burndown Chart, which is a graphical representation of work left to do versus time in a sprint or release. It is used to track progress and ensure that the team is on track to complete the planned work within the sprint duration.
Artifact	Sprint Velocity Chart	The Contractor must develop a Sprint Velocity Chart, which is a chart visualizing the number of story points completed per sprint. This data is then plotted on the chart with sprints on the x-axis and completed story points on the y-axis.
Artifact	Story Points	The Contractor must develop Story Points, which are a unit of measure used to estimate the complexity and effort required to implement a user story. Story Points help in sprint planning and measuring team capacity.
Artifact	User Documentation	The Contractor must develop User Documentation, which helps end-users understand and use the project deliverables effectively. User Documentation may include user manuals, system documentation, and training materials.
Artifact	User Story	The Contractor must develop User Stories, which are short, simple descriptions of a feature from an end-user perspective to articulate Agile requirements and guide development activities, forming the basis for task breakdown and prioritization in sprints. A User Story typically follows the format: "As a [type of user], I want [action] so that [benefit]." User stories help align the development with user needs.

Table 39 – Business and Technical Architecture Standard Deliverables and Artifacts

Type	Name	Description
Deliverable	Functional Design Document	<p>The Contractor must develop Functional Design Documents that describe the functional requirements and specifications for new or updated system components. These documents must include comprehensive descriptions of functionalities, user interfaces, workflows, data inputs and outputs, and business rules requirements when supporting Medicaid operations like eligibility determination, claims processing, and reporting. Functional Design Documents may include a Business Rules Specification Document. Functional Design Documents must include, but not be limited to:</p> <ul style="list-style-type: none"> • <i>Functional Requirements</i> – Describe the dependencies and acceptance criteria • <i>Non-functional Requirements</i> – Describe the performance and usability requirements • <i>User Interface Specifications</i> – Describe the user interface layout and navigation, focusing on usability and user experience • <i>Workflow Descriptions</i> – Describe the workflow in a sequential order • <i>Inputs and Outputs</i> – Describe any related business logic, integration points with other systems, APIs, data sources, etc. • <i>Security, Performance, and Acceptance Criteria</i> – Describe the specifications for security measures, expected performance, and the criteria for functionality acceptance
Deliverable	Data Dictionary	The Contractor must develop a Data Dictionary defining and documenting all data elements in the solution, including the description, type, format, and permissible values for each active and obsolete data element.
Deliverable	Integration Plan	The Contractor must develop an Integration Plan that outlines a comprehensive strategy for interfacing and integrating disparate environments and components with and within the solution to ensure seamless communication and data exchange.
Artifact	Architecture Decision Record/Solution Design Document (ADR/SDD) Intake Form	The Contractor must submit the ADR/SDD Intake Form for OHIP review of solution architecture decisions, design alternatives, impacts, and exceptions as part of the architecture governance process.

Artifact	Business Architecture Models	The Contractor must develop Business Architecture Models, which provide structured representations of the organization's capabilities, processes, information flows, and stakeholder relationships to align strategy and execution, independent of specific organizational structures, technologies, or implementation choices. Business Architecture Models may include, but are not limited to: a Business Capability Model, which identifies the high-level abilities the organization must possess; a Business Process Model, which depicts end-to-end business activities and workflows; and a Use Case Model, which illustrates business interactions and functional scenarios from a stakeholder perspective.
Artifact	Data Model(s)	The Contractor must develop Data Models that provide structured representations of how data is structured, related, and managed within the system, which may include conceptual, logical, and physical data models.
Artifact	State Self-Assessment	The Contractor must collaborate with OHIP to update the State Self-Assessments Results, which are the artifacts created when NYS performs a self-assessment required by CMS as part of the Medicaid Information Technology Architecture (MITA) requirements. The Contractor's software implementations and upgrades will increase the MITA maturity levels for business, information and/or technology capabilities. As such increases occur, the Contractor will assess the current maturity levels as detailed in the State Self-Assessment and update the results as appropriate.
Artifact	Technical Architecture Models	The Contractor must develop Technical Architecture Models, which provide structured representations of the technical components, relationships, and environments necessary to support business and solution requirements. These models describe how technology elements are conceptually organized, integrated, and deployed to enable business capabilities and system functionality. Technical Architecture Models may include, but are not limited to: a Deployment Model, which illustrates runtime environments and component distribution; a Network Topology Model, which depicts connectivity and communication paths; a Conceptual Integration Architecture Model, which defines logical data and system interactions; an Application Component Model, which identifies major application modules, their logical scope, and the relationships and dependencies among them; and a Solution Architecture Model, which provides an end-to-end technical view of a solution, including major systems, interactions, and alignment to business capabilities.

Appendix B: Master Requirements Summary

The following table consolidates all 651 requirements from this domain standard with individual RFC 2119.

Req. ID	Requirement Statement	Tier	Standard Section
Section 2.0—Usability and Accessibility Standard UACC-0001 to UACC-0093 93 Requirements: MUST 87 SHOULD 6 MAY 0			
UACC-0001	Ensure consumers, providers, and administrative staff can complete tasks without confusion or error.	MUST	2.5.1—Effectiveness
UACC-0002	Design solutions to support a high level of task effectiveness.	MUST	2.5.1—Effectiveness
UACC-0003	Demonstrate that at least 90–95% of representative users can successfully complete core workflows in controlled testing. Exact targets SHOULD be defined per workflow and user segment in an OHIP-approved measurement plan; report results using percentile targets (e.g., P90) by segment.	SHOULD	2.5.1—Effectiveness
UACC-0004	Demonstrate task efficiency by targeting time-on-task benchmarks based on user role and context.	SHOULD	2.5.2—Efficiency
UACC-0005	Evaluate solution designs against CMS usability performance metrics.	SHOULD	2.5.2—Efficiency
UACC-0006	Establish workflow-specific time-on-task targets by role and context in the measurement plan; report using percentiles (e.g., P75, P90). Example: for a trained user in a controlled test scenario, a new eligibility application SHOULD meet an OHIP-approved P75/P90 benchmark; targets SHOULD be calibrated per release as baseline data becomes available.	SHOULD	2.5.2—Efficiency
UACC-0007	Design solution interfaces with straightforward navigation, responsive design, and user-centered content.	SHOULD	2.5.3—Satisfaction
UACC-0008	Test solutions using the System Usability Scale (SUS) or equivalent. Target SUS ≥85 as a minimum; preferred SUS ≥90. If SUS is below 85, the Contractor SHOULD provide a remediation plan and demonstrate release-over-release improvement. Results SHOULD be reported by user segment.	SHOULD	2.5.3—Satisfaction

Req. ID	Requirement Statement	Tier	Standard Section
UACC-0009	Provide intuitive workflows and clear guidance that minimize the learning curve.	SHOULD	2.5.4—Learnability
UACC-0010	Facilitate usability evaluations that include novice user testing.	SHOULD	2.5.4—Learnability
UACC-0011	Ensure solutions support task success after no more than 15 minutes for routine functions.	SHOULD	2.5.4—Learnability
UACC-0012	Comply with Section 508, ADA Title II, NYS ITS accessibility requirements, and WCAG 2.2 AA.	MUST	2.4—Accessibility Standards
UACC-0013	Submit and maintain a current Voluntary Product Accessibility Template (VPAT)—Revised 508 Edition—using the latest ITI template that supports WCAG 2.2 and EN 301 549, and provide an Accessibility Conformance Report (ACR) documenting conformance status for each WCAG success criterion.	MUST	2.4—Accessibility Standards
UACC-0014	Facilitate both automated and manual accessibility testing.	MUST	2.4—Accessibility Standards
UACC-0015	Obtain documented proof of accessibility compliance before releasing any OHIP MES solution.	MUST	2.4—Accessibility Standards
UACC-0016	Ground user interfaces in user-centered design practices.	MUST	2.5.5—User Experience (UX)
UACC-0017	Test user interfaces in real-world scenarios.	MUST	2.5.5—User Experience (UX)
UACC-0018	Demonstrate end-user involvement and iterative refinement of interfaces.	MUST	2.5.5—User Experience (UX)
UACC-0019	Document UX research, personas, and feedback loops as part of solution design artifacts.	MUST	2.5.5—User Experience (UX)
UACC-0020	Ensure solution designs include fail-safes, field validation, and error prevention patterns.	MUST	2.5.6—Avoidance of Harm
UACC-0021	Track, report, and mitigate harmful error events during usability testing with stress testing, to align with applicable federal certification and regulatory requirements, including CMS SMC and, for Exchange-related functions, ACA and QHP certification requirements.	MUST	2.5.6—Avoidance of Harm

Req. ID	Requirement Statement	Tier	Standard Section
UACC-0022	Define harmful error categories (e.g., misapplied eligibility determinations, PHI exposure, erroneous notices) with severity levels and remediation SLAs; instrument analytics to detect, monitor, and prevent recurrence.	MUST	2.5.6—Avoidance of Harm
UACC-0023	Complete usability regression testing for all changes to MES user-facing components.	MUST	2.5.7—Maintainability
UACC-0024	Use design systems and component libraries to facilitate visual and interaction continuity.	SHOULD	2.5.7—Maintainability
UACC-0025	Demonstrate maintainability through version control and documentation of usability impact post-deployment.	SHOULD	2.5.7—Maintainability
UACC-0026	Provide release notes highlighting user-facing changes and accessibility impacts; perform targeted accessibility regression testing for modified components and document results.	MUST	2.5.7—Maintainability
UACC-0027	Ensure multilingual parity for alt text, labels, error messages, transcripts/captions, and downloadable content; support locale-specific number/date formats and allow for text expansion in translated strings.	MUST	2.5.8—Localization
UACC-0028	Adhere to WCAG 2.2 AA perceivability success criteria.	MUST	2.4.1—Perceivability
UACC-0029	Provide text alternatives for non-text content.	MUST	2.4.1—Perceivability
UACC-0030	Ensure sufficient color contrast and resizable text.	MUST	2.4.1—Perceivability
UACC-0031	Caption all video content and provide audio descriptions when applicable.	MUST	2.4.1—Perceivability
UACC-0032	Support reflow at a width of 320 CSS pixels without horizontal scrolling and zoom to 400% without loss of content or functionality.	MUST	2.4.1—Perceivability
UACC-0033	Verify compliance through automated and manual accessibility testing documented in CMS certification packets.	MUST	2.4.1—Perceivability
UACC-0034	Avoid content that flashes more than three times per second; honor user motion preferences (prefers-reduced-motion) by providing non-animated alternatives.	MUST	2.4.1—Perceivability

Req. ID	Requirement Statement	Tier	Standard Section
UACC-0035	Verify that the solution supports complete keyboard operability, proper focus management, and input method alternatives.	MUST	2.4.2—Operability
UACC-0036	Confirm interactive elements are accessible without mouse or gestures.	MUST	2.4.2—Operability
UACC-0037	Develop solutions with extensions or adjustable durations for time-based tasks.	MUST	2.4.2—Operability
UACC-0038	Document proof of keyboard and input operability in usability validation and certification artifacts.	MUST	2.4.2—Operability
UACC-0039	Ensure that focus indicators meet WCAG 2.2 Focus Appearance and Focus Not Obscured criteria; maintain logical and visible focus order throughout all interactions.	MUST	2.4.2—Operability
UACC-0040	Ensure that all functionality is available without gestures requiring path-based input; provide alternatives to dragging movements.	MUST	2.4.2—Operability
UACC-0041	Provide accessible alternatives to CAPTCHA; avoid reliance on visual puzzles or audio-only challenges.	MUST	2.4.2—Operability
UACC-0042	Provide consistent navigation patterns for user interfaces.	MUST	2.4.3—Understandability
UACC-0043	Use clearly labeled and readable form fields.	MUST	2.4.3—Understandability
UACC-0044	Develop descriptive error messages with guidance for resolution.	MUST	2.4.3—Understandability
UACC-0045	Use plain language targeting approximately a 6th–8th grade reading level for consumer-facing content; avoid jargon and idioms.	MUST	2.4.3—Understandability
UACC-0046	Validate Understandability through cognitive usability testing and accessibility compliance scans.	MUST	2.4.3—Understandability
UACC-0047	Demonstrate user guidance and error prevention features to enable CMS certification.	MUST	2.4.3—Understandability
UACC-0048	Associate error messages programmatically to their fields; use ARIA live regions to announce status changes and validation results to assistive technologies.	MUST	2.4.3—Understandability

Req. ID	Requirement Statement	Tier	Standard Section
UACC-0049	Provide semantically correct HTML, ARIA roles, labels, and status for AT compatibility.	MUST	2.4.4—Robustness
UACC-0050	Ensure interactive widgets, biometric interfaces, and modal dialogs expose accessible roles and behaviors to AT.	MUST	2.4.4—Robustness
UACC-0051	Perform manual AT testing using an OHIP-approved matrix that includes, at minimum: JAWS with Chrome, NVDA with Firefox, VoiceOver with Safari (macOS and iOS), and TalkBack with Chrome (Android). Include speech input and switch control operability checks. Document AT test coverage, findings mapped to specific WCAG criteria, remediation actions, and re-test results.	SHOULD	2.4.4—Robustness
UACC-0052	Provide non-biometric alternatives such as PIN entry, voice prompts, or knowledge-based questions. Biometrics MUST NOT be the sole authentication factor. Non-biometric fallbacks MUST be consistent with NIST SP 800-63B and MUST NOT increase cognitive load relative to the biometric option.	SHOULD	2.4.5—Biometric Interfaces
UACC-0053	Ensure all feedback is perceivable to screen readers, keyboard-only users, and low-vision users.	SHOULD	2.4.5—Biometric Interfaces
UACC-0054	Avoid motion-only inputs and allow manual override for gesture-based or timed actions.	MUST	2.4.5—Biometric Interfaces
UACC-0055	Include clear, accessible instructions, retry prompts, and error recovery options.	MUST	2.4.5—Biometric Interfaces
UACC-0056	Verify biometric accessibility through inclusive usability testing and document in CMS assessments.	MUST	2.4.5—Biometric Interfaces
UACC-0057	Design dashboards with a consistent layout supporting efficient information consumption.	MUST	2.6.1—Dashboard Layout
UACC-0058	Place navigation elements, filters, and data currency indicators in consistent locations.	MUST	2.6.1—Dashboard Layout
UACC-0059	Establish a clear visual hierarchy directing user attention to critical information first.	MUST	2.6.1—Dashboard Layout
UACC-0060	Present summary-level metrics prominently with drill-down capability.	MUST	2.6.1—Dashboard Layout

Req. ID	Requirement Statement	Tier	Standard Section
UACC-0061	Ensure dashboard layouts are responsive across desktop, tablet, and mobile resolutions.	MUST	2.6.1—Dashboard Layout
UACC-0062	Use color consistently so the same color represents the same category throughout.	MUST	2.6.2—Color Usage
UACC-0063	Not use color as the sole means of conveying information.	MUST	2.6.2—Color Usage
UACC-0064	Verify color-encoded information is also available through text labels, patterns, or non-color indicators.	MUST	2.6.2—Color Usage
UACC-0065	Comply with WCAG 2.2 Level AA contrast ratios for all color combinations.	MUST	2.6.2—Color Usage
UACC-0066	Ensure data visualizations remain interpretable in grayscale.	MUST	2.6.2—Color Usage
UACC-0067	Use plain language labels for all data elements, axes, and chart components.	SHOULD	2.6.3—Labeling and Tooltips
UACC-0068	Not display technical field names or database column names to end users.	SHOULD	2.6.3—Labeling and Tooltips
UACC-0069	Tooltips SHOULD be concise and scannable, generally not exceeding approximately 200 characters, while allowing multi-line formatting when needed for accessibility.	SHOULD	2.6.3—Labeling and Tooltips
UACC-0070	Tooltips SHOULD be available on focus and on activation on both desktop and touch interfaces, and SHOULD be announced by assistive technologies.	MUST	2.6.3—Labeling and Tooltips
UACC-0071	Design tooltips that are keyboard-accessible and conform to Section 2.4 accessibility requirements.	MUST	2.6.3—Labeling and Tooltips
UACC-0072	Include descriptive titles, labeled axes with units, and legends for all charts and graphs.	MUST	2.6.3—Labeling and Tooltips
UACC-0073	Provide accessible tabular equivalents for complex charts to support screen reader users and data export.	MUST	2.6.3—Labeling and Tooltips
UACC-0074	Design dashboards to meet performance targets under defined test loads: P75 load times of 3–5 seconds and P95 ≤ 8 seconds under normal operating conditions.	SHOULD	2.6.4—Dashboard Performance

Req. ID	Requirement Statement	Tier	Standard Section
UACC-0075	Display an accessible loading indicator with progress messaging; announce loading and completion states via ARIA-live status updates.	SHOULD	2.6.4—Dashboard Performance
UACC-0076	Implement drill-down and filter capabilities where technically feasible.	SHOULD	2.6.4—Dashboard Performance
UACC-0077	Optimize queries and data structures supporting dashboards to meet performance targets.	SHOULD	2.6.4—Dashboard Performance
UACC-0078	Display the data refresh date and time on every dashboard page.	SHOULD	2.6.5—Data Currency
UACC-0079	Display a visible notification when data is known to be stale or refresh has not completed.	SHOULD	2.6.5—Data Currency
UACC-0080	Clearly communicate the update frequency for each data source.	SHOULD	2.6.5—Data Currency
UACC-0081	Identify the source system(s) feeding the dashboard at the data-source/dataset level (e.g., NYSOH, eMedNY, WMS); where a displayed value blends multiple sources, indicate the contributing sources. Field-by-field source identification is not required.	SHOULD	2.6.5—Data Currency
UACC-0082	Select chart and graph types appropriate to the data being presented, and do not use visualization types that distort or misrepresent the underlying data, including truncated axes without clear indication, 3-D effects on quantitative charts, and dual axes without distinct labeling.	SHOULD	2.6.6 Chart Selection and Branding Standards
UACC-0083	Ensure all dashboards, visualizations, and report interfaces conform to OHIP/DOH branding requirements, including approved logos, color palettes, and typography, as provided by OHIP.	SHOULD	2.6.6 Chart Selection and Branding Standards
UACC-0084	Utilize the SUS Questionnaire and Scoring Guide to measure overall usability.	SHOULD	2.7—Usability Assessment Tools
UACC-0085	Utilize the VPAT (Revised 508 Edition) to verify WCAG 2.2 AA and Section 508 compliance.	SHOULD	2.7—Usability Assessment Tools
UACC-0086	Comply with this standard upon publication and with any amendments issued by OHIP.	MUST	1.8—Compliance and Enforcement

Req. ID	Requirement Statement	Tier	Standard Section
UACC-0087	Where compliance is not feasible, formally document the exception with OHIP, develop a mitigation plan, and define a compliance timeline.	MUST	1.8—Compliance and Enforcement
UACC-0088	Produce and submit all Deliverables in Appendix A in the format, structure, and cadence specified by OHIP.	MUST	Appendix A—Deliverables
UACC-0089	Upon notice of modifications to Deliverables and Artifacts, comply with all updated requirements.	MUST	Appendix A—Deliverables
UACC-0090	Design all consumer-facing interfaces using a responsive approach, ensuring core workflows can be completed on standard mobile devices without loss of functionality.	MUST	2.5.5 – User Experience (UX)
UACC-0091	Ensure all public-facing MES interfaces support multi-language presentation in compliance with New York State Language Access requirements (Executive Order 26.1).	MUST	2.5.8 - Localization and Language Access
UACC-0092	Support English and Spanish at a minimum, with architecture designed to scale to additional languages without requiring core application changes.	MUST	2.5.8 - Localization and Language Access
UACC-0093	Not rely solely on automated browser translation tools for critical Medicaid eligibility, enrollment, or benefit terminology.	MUST	2.5.8 - Localization and Language Access
Section 3.0—Requirements Management Standard REQM-0001 to REQM-0084 84 Requirements: MUST 84 SHOULD 0 MAY 0			
REQM-0001	Ensure all three requirement types (business, functional, non-functional) are identified, documented, and traced for every MES project.	SHOULD	3.3—Requirements Definition
REQM-0002	Every requirement SHOULD be atomic—each requirement addresses a single capability or constraint.	SHOULD	3.3.1 - Requirement Quality and Readiness
REQM-0003	Every requirement SHOULD be unambiguous—one clear interpretation, consistent with other requirements.	SHOULD	3.3.1 - Requirement Quality and Readiness
REQM-0004	Every requirement SHOULD be necessary—justified by a documented business need or regulatory driver.	SHOULD	3.3.1 - Requirement Quality and Readiness

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0005	Every requirement SHOULD be feasible—achievable within known constraints.	SHOULD	3.3.1 - Requirement Quality and Readiness
REQM-0006	Every requirement MUST be testable—includes defined acceptance criteria and an identified test method.	MUST	3.3.1 - Requirement Quality and Readiness
REQM-0007	Every requirement SHOULD be prioritized using the OHIP-approved prioritization method.	SHOULD	3.3.1 - Requirement Quality and Readiness
REQM-0008	Every requirement MUST be traceable—linked forward to design, code, and tests, and backward to its source and rationale.	MUST	3.3.1 - Requirement Quality and Readiness
REQM-0009	Conduct structured requirements elicitation activities including stakeholder interviews, workshops, document analysis, and observation.	MUST	3.4—Requirements Gathering
REQM-0010	Engage OHIP business owners, subject matter experts, and end users throughout the gathering process.	MUST	3.4—Requirements Gathering
REQM-0011	Identify and document requirements from all applicable regulatory and compliance sources including CMS certification conditions.	MUST	3.4—Requirements Gathering
REQM-0012	Review and incorporate requirements from existing system documentation.	MUST	3.4—Requirements Gathering
REQM-0013	Load all gathered requirements into the OHIP-approved requirements repository.	MUST	3.4—Requirements Gathering
REQM-0014	Review and update requirement attributes for Business Area, Business Process Titles, and CMS Certification Outcomes, Conditions, and Metrics.	MUST	3.4—Requirements Gathering
REQM-0015	To support multi-vendor coordination, the Contractor MUST participate in cross-vendor dependency sessions and backlog refinement cadences established by OHIP to identify interface requirements, sequencing needs, and shared service dependencies early in the gathering process.	MUST	3.4—Requirements Gathering

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0016	Draft the Requirements Management Plan and submit to OHIP for review and approval before requirements elaboration begins.	MUST	3.5.1—Requirements Management Plan
REQM-0017	Define roles and responsibilities (RACI) for requirements gathering, analysis, approval, and maintenance, including cross-vendor coordination responsibilities.	MUST	3.5.1—Requirements Management Plan
REQM-0018	Specify the OHIP-approved prioritization method and backlog refinement cadence.	MUST	3.5.1—Requirements Management Plan
REQM-0019	Define the governance for requirements change control, including decision authority and escalation pathways.	MUST	3.5.1—Requirements Management Plan
REQM-0020	Develop and maintain the RTM as a controlled, version-managed deliverable.	MUST	3.5.2—Requirements Traceability Matrix
REQM-0021	Ensure each RTM entry identifies the requirement, its origin, and its relationships to design elements, test cases, and project objectives. Include in the RTM direct references to design artifacts (e.g., process flows, use cases) and Architecture Decision Records (ADRs) where applicable.	MUST	3.5.2—Requirements Traceability Matrix
REQM-0022	Provide an initial RTM version during the Design Stage and update it throughout all SDLC phases for every Release.	MUST	3.5.2—Requirements Traceability Matrix
REQM-0023	Baseline each iteration of the RTM upon OHIP approval. Update previous data in each iteration to maintain current traceability.	MUST	3.5.2—Requirements Traceability Matrix
REQM-0024	Submit each RTM iteration to OHIP for review and approval.	MUST	3.5.2—Requirements Traceability Matrix
REQM-0025	Include in the RTM references to interface specifications and API specifications for cross-system requirements.	MUST	3.5.2—Requirements Traceability Matrix
REQM-0026	Demonstrate 100% forward and backward traceability for all baselined requirements prior to production deployment.	MUST	3.5.2—Requirements Traceability Matrix
REQM-0027	Zero open Severity 1 or Severity 2 defects tied to release-scoped requirements unless explicitly risk-accepted in writing by OHIP.	MUST	3.5.2—Requirements Traceability Matrix

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0028	Use the Department-designated Requirements Management tooling for all requirements management activities. Propose and obtain OHIP approval for any additional or alternate tools.	MUST	3.5.3—Requirements Management Tools
REQM-0029	Ensure the Department-designated Requirements Management tooling provides unique, immutable requirement IDs.	MUST	3.5.3—Requirements Management Tools
REQM-0030	Ensure the Department-designated Requirements Management tooling provides versioning with full change history and audit trail.	MUST	3.5.3—Requirements Management Tools
REQM-0031	Ensure the Department-designated Requirements Management tooling provides role-based access controls.	MUST	3.5.3—Requirements Management Tools
REQM-0032	Ensure the Department-designated Requirements Management tooling provides export and reporting features sufficient to produce audit-ready evidence for CMS certification.	MUST	3.5.3—Requirements Management Tools
REQM-0033	Elaborate all Solution Requirements to a testable and verifiable level, or identify acceptance criteria for non-testable requirements.	MUST	3.5.5—Requirements Analysis
REQM-0034	Prioritize Solution Requirements in collaboration with OHIP Business Owners.	MUST	3.5.5—Requirements Analysis
REQM-0035	Assess requirements for completeness, consistency, feasibility, and potential conflicts.	MUST	3.5.5—Requirements Analysis
REQM-0036	Identify dependencies between requirements, including cross-Contractor and cross-module dependencies.	SHOULD	3.5.5—Requirements Analysis
REQM-0037	Collaborate with the OHIP Certification team to trace CMS Outcomes and Metrics IDs to applicable requirements.	SHOULD	3.5.5—Requirements Analysis
REQM-0038	Resolve requirements conflicts, ambiguities, or gaps before development begins. Escalate unresolved issues to OHIP.	MUST	3.5.5—Requirements Analysis
REQM-0039	Non-Functional Requirements MUST be quantified with measurable targets, defined test methods, and pass/fail criteria.	MUST	3.5.5—Requirements Analysis

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0040	NFRs MUST address, at minimum: performance and scalability (e.g., P75/P90/P95 response times, throughput, concurrent user loads).	MUST	3.5.5—Requirements Analysis
REQM-0041	NFRs MUST address availability and reliability (e.g., service level objectives, RTO/RPO).	MUST	3.5.5—Requirements Analysis
REQM-0042	NFRs MUST address security and privacy (e.g., authentication and authorization requirements, encryption, audit logging, HIPAA minimum necessary).	MUST	3.5.5—Requirements Analysis
REQM-0043	NFRs MUST address accessibility and usability (e.g., WCAG 2.2 AA conformance, SUS targets per Section 2.0).	MUST	3.5.5—Requirements Analysis
REQM-0044	NFRs MUST address interoperability (e.g., X12 5010, FHIR, HL7 as applicable to the module).	MUST	3.5.5—Requirements Analysis
REQM-0045	NFRs MUST address data quality (accuracy, completeness, timeliness).	MUST	3.5.5—Requirements Analysis
REQM-0046	The Contractor MUST define and trace requirements for external interfaces (ICDs and data contracts), data transformations, and reporting dependencies.	MUST	3.5.5—Requirements Analysis
REQM-0047	Data migration and conversion requirements MUST specify reconciliation criteria and acceptance thresholds.	MUST	3.5.5—Requirements Analysis
REQM-0048	Submit all proposed changes to baselined requirements through a formal change request process approved by OHIP.	MUST	3.6—Requirements Change Management
REQM-0049	Document the business justification, impact analysis, and affected traceability links for each proposed change. Impact analysis for proposed changes MUST include non-functional requirement impacts, security and privacy impact assessment, cross-vendor and cross-module dependencies, and required test updates.	MUST	3.6—Requirements Change Management
REQM-0050	Obtain OHIP approval before implementing any change to a baselined requirement.	MUST	3.6—Requirements Change Management
REQM-0051	Update the RTM, design artifacts, test cases, and all downstream traceability links to reflect approved changes.	SHOULD	3.6—Requirements Change Management

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0052	Maintain a change log recording all requirement changes including original requirement, nature of change, approval, and date.	SHOULD	3.6—Requirements Change Management
REQM-0053	Changes to baselined requirements are governed by a Requirements Change Control Board (RCCB) or equivalent governance body with defined membership, decision authority, and escalation pathways.	MUST	3.6—Requirements Change Management
REQM-0054	Emergency changes to baselined requirements MUST follow a documented expedited path and be ratified at the next scheduled RCCB meeting.	MUST	3.6—Requirements Change Management
REQM-0055	Maintain continuous forward and backward traceability for all requirements throughout the development life cycle.	MUST	3.7—Requirements Traceability
REQM-0056	Link requirements to business areas, data models, source code, test cases, test results, and implementation documentation.	MUST	3.7—Requirements Traceability
REQM-0057	Maintain traceability when a single requirement corresponds to multiple deliverables or artifacts.	MUST	3.7—Requirements Traceability
REQM-0058	Develop design artifacts that include traceability to Solution Requirements.	MUST	3.7—Requirements Traceability
REQM-0059	Trace test cases to requirements; each test case MUST reference the specific requirement(s) it validates.	MUST	3.7—Requirements Traceability
REQM-0060	Ensure data lineage is documented from requirement through logical and physical data models to ETL/API implementations to reports and dashboards, including validation and reconciliation evidence where applicable.	MUST	3.7—Requirements Traceability
REQM-0061	Update the RTM at each life cycle stage (Design, Development, Implementation) with stage-specific traceability attributes.	MUST	3.7.1—RTM Life Cycle
REQM-0062	Facilitate an RTM walkthrough meeting demonstrating full traceability, coverage, accuracy, and completeness before implementation.	MUST	3.7.1—RTM Life Cycle
REQM-0063	Verify that each requirement has been implemented as specified by reviewing design artifacts, code, and test results.	MUST	3.8—Verification and Validation

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0064	Validate that the implemented solution meets OHIP's business needs through structured acceptance testing with business owners.	MUST	3.8—Verification and Validation
REQM-0065	Demonstrate adequate testing coverage for all requirement types including functional, non-functional, integration, regression, and UAT.	MUST	3.8—Verification and Validation
REQM-0066	Provide a resolution plan for any outstanding requirements concerns identified during verification or validation.	SHOULD	3.8—Verification and Validation
REQM-0067	Not proceed to production deployment with requirements in an unverified or unvalidated state unless an exception is approved by OHIP in writing.	SHOULD	3.8—Verification and Validation
REQM-0068	Prior to production deployment, the Contractor MUST demonstrate requirements coverage: 100% of release-scoped requirements traced forward and backward with passing verification evidence.	MUST	3.8—Verification and Validation
REQM-0069	Prior to production deployment, the Contractor MUST demonstrate that acceptance criteria are met for each requirement with objective evidence.	MUST	3.8—Verification and Validation
REQM-0070	Prior to production deployment, the Contractor MUST demonstrate that security and privacy verification passed per applicable controls with no critical findings outstanding.	MUST	3.8—Verification and Validation
REQM-0071	Prior to production deployment, the Contractor MUST demonstrate accessibility and usability conformance per Section 2.0.	MUST	3.8—Verification and Validation
REQM-0072	Prior to production deployment, the Contractor MUST demonstrate that performance and scalability NFR targets are met under defined loads and environments.	MUST	3.8—Verification and Validation
REQM-0073	Prior to production deployment, the Contractor MUST obtain OHIP business owner UAT sign-off.	MUST	3.8—Verification and Validation
REQM-0074	No open Severity 1 or Severity 2 defects in release scope unless explicitly risk-accepted in writing by OHIP with a mitigation plan and timelines.	MUST	3.8—Verification and Validation

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0075	The Contractor MUST NOT proceed to production deployment with requirements in an unverified or unvalidated state unless a specific exception is approved by OHIP in writing.	MUST	3.8—Verification and Validation
REQM-0076	Establish and operate a Business Analysis Quality Assurance (BAQA) program that delivers consistent, traceable, and implementation-ready requirements and maintains RTM readiness across all delivery increments, defining the methods, artifacts, review gates, and metrics used to validate business needs.	SHOULD	3.5.4 – Business Analysis Quality Assurance (BAQA)
REQM-0077	Include a BAQA Plan within the Requirements Management Plan that describes the standards, templates, tools, review cadences, metrics, and escalation paths governing BAQA; apply the plan during Scope Prioritization and maintain it throughout all phases of the SDLC.	SHOULD	3.5.4 – Business Analysis Quality Assurance (BAQA)
REQM-0078	Establish standards, consistent with those defined by the Department, for use cases, acceptance criteria, and nonfunctional requirements, and maintain end-to-end traceability from business objectives through requirements, test cases, and Releases.	SHOULD	3.5.4 – Business Analysis Quality Assurance (BAQA)
REQM-0079	Use structured elicitation, rapid prototyping, and usability testing for high-risk items, and use asynchronous validation when subject matter expert (SME) availability is limited.	SHOULD	3.5.4 – Business Analysis Quality Assurance (BAQA)
REQM-0080	Implement peer reviews of requirements and scenarios prior to RTM commitment, permitting a work item to enter an Iteration only when it meets the entry criteria (linkage to business objectives and policy/process references; BDD acceptance criteria; identified dependencies and impacts; prototype or annotated artifacts for high-risk items; and Product Owner and BAQA lead approval) and passes BAQA peer review; document any exception in a mitigation plan tracked to closure.	SHOULD	3.5.4 – Business Analysis Quality Assurance (BAQA)
REQM-0081	Track and report RTM readiness rate, requirements defects that escape to delivery, rework due to requirements gaps, peer review findings, and corrective actions; provide regular BAQA status, including audit results and continuous improvement activities, to project leadership and to the Department upon request.	SHOULD	3.5.4 – Business Analysis Quality Assurance (BAQA)

Req. ID	Requirement Statement	Tier	Standard Section
REQM-0082	Conduct periodic audits of requirements quality and traceability, capture lessons learned, and update standards, checklists, and templates accordingly; maintain tools and repositories that support version control, comment history, and end-to-end traceability across objectives, use cases, tests, and Releases, aligned with the OHIP Infrastructure and Operations Domain Document, Preferred Operational Tools Standard.	SHOULD	3.5.4 – Business Analysis Quality Assurance (BAQA)
REQM-0083	Categorize each requirement in the OHIP-approved requirements repository using OHIP-defined attributes, including at a minimum requirement category (e.g., interface, reporting, data retention, security, accessibility), regulatory or compliance source where applicable, and the originating business area.	SHOULD	3.3—Requirements Definition
REQM-0084	Perform and document a production impact assessment before approving any requirement change to a solution component in production operation, identifying affected operating functions, interfaces, batch processes, reports, and business rules using RTM traceability, and defining the regression test scope that demonstrates existing functionality is not degraded prior to deployment.	SHOULD	3.7—Requirements Change Management
Section 4.0—Policy and Business Rules Standard RULE-0001 to RULE-0061 61 Requirements: MUST 59 SHOULD 2 MAY 0			
RULE-0001	Externalize all policy and business rules from application code. Rules MUST NOT be embedded within the application’s programming code base.	MUST	4.3—MITA Alignment and Business Rules Architecture
RULE-0002	Implement a business rules management approach that separates rule logic from application logic, approved by OHIP.	MUST	4.3—MITA Alignment and Business Rules Architecture
RULE-0003	Expose all business rules through standardized APIs to allow multiple applications to consume the same rules without duplication. API contracts for business rules MUST define request/response schemas, error codes, versioning strategy, and backward compatibility approach.	MUST	4.3—MITA Alignment and Business Rules Architecture
RULE-0004	Business rule API requests and responses MUST be deterministic and auditable.	MUST	4.3—MITA Alignment and Business Rules Architecture

Req. ID	Requirement Statement	Tier	Standard Section
RULE-0005	Support rule changes on both a regularized schedule and on an emergency basis.	MUST	4.3—MITA Alignment and Business Rules Architecture
RULE-0006	Ensure the rules management approach empowers OHIP business users to review, understand, and validate rule logic without requiring programming expertise.	MUST	4.3—MITA Alignment and Business Rules Architecture
RULE-0007	Use tools listed in the OHIP Preferred Operational Tools Standard. Propose and obtain OHIP approval for any alternative tools.	SHOULD	4.3—MITA Alignment and Business Rules Architecture
RULE-0008	Comply with the OHIP Security, Privacy, and Compliance Domain Document when developing, implementing, or enhancing business rules.	SHOULD	4.4—Security and Privacy Requirements
RULE-0009	Provide and obtain OHIP approval for a Policy and Business Rules Implementation Plan before rule development begins.	MUST	4.5.1—Implementation Plan
RULE-0010	The Implementation Plan MUST include rule API standards, including versioning, request/response schema, error handling, and performance targets.	MUST	4.5.1—Implementation Plan
RULE-0011	The Implementation Plan MUST describe the approach for effective dating, including support for retroactive application and parallel operation of multiple rule versions when policy requires grandfathering or staggered effective dates.	MUST	4.5.1—Implementation Plan
RULE-0012	The Implementation Plan MUST describe the testing strategy, including unit and decision-table coverage targets, regression testing, backtesting against representative historical data to validate expected impact before release, and performance testing for rule APIs.	MUST	4.5.1—Implementation Plan
RULE-0013	The Contractor MUST use de-identified or masked data for all rule testing, adhering to OHIP security and privacy standards.	MUST	4.5.1—Implementation Plan
RULE-0014	Produce and maintain a consolidated Business Rules Documentation deliverable combining traceability matrix, catalog, and specification.	MUST	4.5.2—Business Rules Documentation

Req. ID	Requirement Statement	Tier	Standard Section
RULE-0015	Obtain OHIP acceptance of the Business Rules Documentation before rule creation begins and update throughout the rule life cycle.	MUST	4.5.2—Business Rules Documentation
RULE-0016	Provide all rules and rule artifacts in an OHIP-approved location in a State-managed repository. Business rules MUST NOT be proprietary.	MUST	4.6.1—Rule Transparency
RULE-0017	Demonstrate all rule activities, artifacts, and executions upon OHIP request.	MUST	4.6.1—Rule Transparency
RULE-0018	Author rules in clear, business-friendly language that promotes understanding by non-technical stakeholders.	MUST	4.6.2—Human Readability
RULE-0019	Obtain OHIP agreement on rule readability and understanding before implementation.	SHOULD	4.6.2—Human Readability
RULE-0020	Use consistent, descriptive identifiers for rules and decision tables.	SHOULD	4.6.2—Human Readability
RULE-0021	Generate both human-readable and machine-readable formats for all rules.	MUST	4.6.2—Human Readability
RULE-0022	Break complex logic into reusable sub-rules or functions.	SHOULD	4.6.2—Human Readability
RULE-0023	Embed explanatory notes for assumptions, business context, and known limitations.	SHOULD	4.6.2—Human Readability
RULE-0024	Maintain a clear distinction between substantive rules and procedural rules. Outcome-determination rules MUST NOT be mixed with procedural, screen flow, or visibility logic.	MUST	4.6.3—Substantive and Procedural Separation
RULE-0025	Business rules MUST produce deterministic, explainable outcomes—the same inputs and effective date MUST always produce the same result.	MUST	4.6.4 Determinism
RULE-0026	The Contractor MUST define default behavior for missing or ambiguous inputs.	MUST	4.6.4 Determinism
RULE-0027	Rules MUST comply with applicable civil rights and nondiscrimination requirements; protected class attributes MUST NOT be used in rule logic in ways that violate law or policy.	MUST	4.6.4 Determinism

Req. ID	Requirement Statement	Tier	Standard Section
RULE-0028	Maintain end-to-end traceability from each policy rule to its originating legislation or agency policy.	MUST	4.7—Rule Traceability
RULE-0029	Perform rule-to-statute mapping referencing the specific section, subsection, and paragraph in the legal text.	MUST	4.7—Rule Traceability
RULE-0030	Reference official policy manuals or guidance documents with version and publication date.	MUST	4.7—Rule Traceability
RULE-0031	Maintain forward traceability from rules through test cases and test results to implementation evidence. The Contractor MUST link all test cases and results to rule IDs in the Business Rules Documentation and include objective evidence (screenshots, logs, or reports).	MUST	4.7—Rule Traceability
RULE-0032	Ensure rule deployment is independent of application deployment. Rules MUST be deployable without application redeployment.	MUST	4.8.1—Independent Rule Deployment
RULE-0033	Implement robust rule versioning allowing applications to invoke the correct validated ruleset.	MUST	4.8.1—Independent Rule Deployment
RULE-0034	Enable rules that do not impact the interface to be changed and deployed without application redeployment.	MUST	4.8.1—Independent Rule Deployment
RULE-0035	Enable regularly changing rules (e.g., annual poverty levels, rate tables) to be updated without modifications to calling applications.	MUST	4.8.1—Independent Rule Deployment
RULE-0036	Ensure all rules are application-agnostic. Rules MUST NOT be part of the code base of any specific application.	MUST	4.8.2—Rule Decoupling and Reuse
RULE-0037	Expose all rules through a standardized API for consumption by multiple applications.	MUST	4.8.2—Rule Decoupling and Reuse
RULE-0038	Not recreate the same rules for multiple applications. A single rule definition MUST be consumed by all applications through the standard API.	MUST	4.8.2—Rule Decoupling and Reuse
RULE-0039	Submit all proposed changes to production business rules through a formal change request process approved by OHIP.	MUST	4.8.3—Rule Change Management

Req. ID	Requirement Statement	Tier	Standard Section
RULE-0040	Document the regulatory or policy basis, affected rules, downstream impact, and proposed effective date for each rule change.	MUST	4.8.3—Rule Change Management
RULE-0041	Follow the designated OHIP approval process for releasing any rule change to production.	MUST	4.8.3—Rule Change Management
RULE-0042	Support both scheduled rule changes and emergency rule changes.	MUST	4.8.3—Rule Change Management
RULE-0043	Update Business Rules Documentation, traceability records, and test cases to reflect all approved changes.	MUST	4.8.3—Rule Change Management
RULE-0044	For emergency changes, the Contractor MUST follow the Department's established emergency change management process.	MUST	4.8.3—Rule Change Management
RULE-0045	Emergency rule changes MUST be ratified at the next scheduled change control meeting and include backout criteria and a communications plan.	MUST	4.8.3—Rule Change Management
RULE-0046	Maintain version control for all business rules including version number, effective date, expiration date, and regulatory basis.	SHOULD	4.8.4—Rule Versioning
RULE-0047	Preserve all prior rule versions in the repository to support audit and point-in-time analysis.	MUST	4.8.4—Rule Versioning
RULE-0048	Ensure current and historical rule versions are accessible to OHIP at all times.	MUST	4.8.4—Rule Versioning
RULE-0049	The Contractor MUST support parallel operation of multiple rule versions when policy requires grandfathering, phase-in periods, or staggered effective dates.	MUST	4.8.5 - Effective Dating
RULE-0050	The system MUST enable time-travel queries that reproduce past determinations using the rule version and effective date that was in effect at the time of the original decision.	MUST	4.8.5 - Effective Dating
RULE-0051	The Contractor MUST instrument rule executions with metrics (volume, latency, error rates) and structured logs that include the rule version and effective date used for each execution.	MUST	4.8.6 - Monitoring

Req. ID	Requirement Statement	Tier	Standard Section
RULE-0052	The Contractor MUST implement alerts for anomalous patterns (e.g., a sudden spike in denial rates or processing errors after a rule change) and provide dashboards and reports for OHIP oversight, including post-deployment impact reviews.	MUST	4.8.6 - Monitoring
RULE-0053	Immutable, time-stamped audit logs of all rule changes and deployments MUST be maintained, including approver identity and release artifacts.	MUST	4.8.6 - Monitoring
RULE-0054	Archive production rules only after OHIP approval and only when the data created by the rules has also been archived.	SHOULD	4.8.7—Rule Archival
RULE-0055	Maintain rules in the rules management system as long as data derived from the rules remains in use.	SHOULD	4.8.7—Rule Archival
RULE-0056	Identify the applicable Federal and/or State regulation, statute, or official policy that each rule implements, where applicable.	MUST	4.8.7—Rule Archival
RULE-0057	Ensure rules are independently testable and deployable, decoupled from application release cycles, so that a rule change can be validated and promoted without requiring an application-tier release.	MUST	4.8.7—Rule Archival
RULE-0058	Remediate, at no additional cost to the Department, any rules not authored in the prescribed application-agnostic, streamlined manner.	MUST	4.8.7—Rule Archival
RULE-0059	Implement Department-initiated rule changes through the standard OHIP change and approval process.	MUST	4.8.7—Rule Archival
RULE-0060	Static documents and spreadsheets MUST NOT serve as the authoritative repository or system of record for business rules; any rule content in such artifacts MUST originate from, and be reconcilable to, the rules maintained in the OHIP-approved repository.	MUST	4.1 – Introduction
RULE-0061	Perform and document a current-state rules inventory and gap analysis before rule development begins, identifying all existing in-scope rules regardless of location, tracing each to its authoritative source or flagging it as unsourced for OHIP disposition, and defining each rule’s target-state disposition (migrate, re-author, consolidate, retire) and migration impact.	SHOULD	4.5.1 – Implementation Plan

Req. ID	Requirement Statement	Tier	Standard Section
Section 5.0—Code Standard CODE-0001 to CODE-0060 60 Requirements: MUST 59 SHOULD 0 MAY 1			
CODE-0001	Comply with 45 CFR 95.617 ensuring the State obtains unrestricted rights to all custom code.	MUST	5.3.1—Ownership Rights
CODE-0002	Ensure OHIP retains code ownership rights from project initiation and all ownership rights to every stage of developed software.	MUST	5.3.1—Ownership Rights
CODE-0003	Store all source code, scripts, and low-code artifacts in a State-owned version control repository at all times.	MUST	5.3.2—Repository Control
CODE-0004	Restrict repository access to authorized personnel according to the principle of least privilege.	MUST	5.3.2—Repository Control
CODE-0005	Support the Lead Maintainer and Backup Maintainer structure for all codebases.	MUST	5.4—Code Leadership and Accountability
CODE-0006	Ensure all changes are reviewed and approved by the Lead Maintainer or Backup Maintainer before merging.	MUST	5.4—Code Leadership and Accountability
CODE-0007	Adopt the secure coding standard prescribed by the Department (e.g., OWASP Secure Coding Practices, NIST 800-218, or SEI CERT Coding Standards, as OHIP designates), and obtain OHIP approval.	MUST	5.5—Secure Coding Standards
CODE-0008	Apply the approved coding standard consistently. Compliance MUST be enforced through automated Static Application Security Testing (SAST) tooling integrated into CI/CD pipelines.	MUST	5.5—Secure Coding Standards
CODE-0009	Ensure all code adheres to best practices for validation, authentication, error handling, and other secure coding requirements.	MUST	5.5—Secure Coding Standards
CODE-0010	Follow the OHIP Artificial Intelligence Standard for any use of AI in coding.	MUST	5.5—Secure Coding Standards
CODE-0011	Repositories MUST implement automated secret detection to prevent committed credentials, tokens, certificates, or cryptographic material from entering the codebase.	MUST	5.5—Secure Coding Standards

Req. ID	Requirement Statement	Tier	Standard Section
CODE-0012	Applications MUST satisfy security verification requirements aligned with an OHIP-approved application security verification standard (e.g., OWASP ASVS). The applicable verification level MUST be agreed with OHIP before development begins.	MUST	5.5—Secure Coding Standards
CODE-0013	Obtain OHIP approval for all programming, scripting, and markup language selections before development.	MUST	5.6.1—Language Approval Criteria
CODE-0014	Ensure approved languages have active community releases within 12 months and a roadmap extending 24 months.	SHOULD	5.6.1—Language Approval Criteria
CODE-0015	Languages, frameworks, and runtimes MUST be selected from the OHIP Approved Technology Stack. The Contractor MUST submit a Standards Deviation Request (SDR) for any language, framework, or runtime not currently on the approved list.	MUST	5.6.1—Language Approval Criteria
CODE-0016	Languages, frameworks, and runtimes MUST remain within vendor-supported life cycle periods for all production deployments. Technologies approaching end-of-life MUST be identified and a migration plan submitted to OHIP.	MUST	5.6.1—Language Approval Criteria
CODE-0017	Approved technologies MUST support automated dependency scanning, patch management, and CI/CD integration.	MUST	5.6.1—Language Approval Criteria
CODE-0018	Consider other languages not on the preferred list for OHIP approval with documented justification.	MAY	5.6.3—Utility Languages
CODE-0019	Establish a Patch Management Plan to upgrade platforms and dependencies proactively.	MUST	5.7—Dependency Management
CODE-0020	Obtain OHIP approval for the Patch Management Plan for all third-party libraries, frameworks, and tools prior to development.	MUST	5.7—Dependency Management
CODE-0021	Deploy all third-party libraries with the latest supported version and keep them up to date.	MUST	5.7—Dependency Management
CODE-0022	The Contractor MUST generate and maintain a machine-readable Software Bill of Materials (SBOM) in SPDX or CycloneDX format for all production releases. SBOMs MUST be updated for every release and provided to OHIP upon request.	MUST	5.7—Dependency Management

Req. ID	Requirement Statement	Tier	Standard Section
CODE-0023	The Contractor MUST perform automated dependency vulnerability scanning as part of the CI/CD pipeline. Builds with known Critical or High vulnerabilities in dependencies MUST NOT proceed to production without documented risk acceptance from OHIP.	MUST	5.7—Dependency Management
CODE-0024	The Contractor MUST remediate dependency vulnerabilities within the following timelines: Critical—7 calendar days; High—30 calendar days; Medium—90 calendar days; or as otherwise defined by OHIP. Exceptions MUST be documented with a mitigation plan and OHIP approval.	MUST	5.7—Dependency Management
CODE-0025	Coverage thresholds MUST be automatically enforced through CI/CD pipeline quality gates. Builds that fail to meet coverage thresholds MUST NOT proceed to merge or deployment.	MUST	5.8—Unit Testing
CODE-0026	Achieve unit test coverage of 100% for auth modules, 80% of statements, and 75% of branches. Builds MUST fail if coverage drops below thresholds.	MUST	5.8—Unit Testing
CODE-0027	Coverage reports MUST be generated with every build and accessible to OHIP within the State-owned CI/CD tooling.	MUST	5.8—Unit Testing
CODE-0028	Develop and obtain OHIP approval for a code security design demonstrating Zero Trust Principles.	MUST	5.9.1—Secure by Design
CODE-0029	Design all solutions with modular architecture separating presentation, business logic, and data layers.	MUST	5.9.2—Modular, Layered Design
CODE-0030	Services MUST be loosely coupled and follow modular architecture principles with explicit separation of concerns (e.g., microservices, SOA, modular monolith, or domain-oriented services) to prevent tight coupling and support independent deployment, testing, and maintenance.	MUST	5.9.2—Modular, Layered Design
CODE-0031	Expose well-defined, documented, and secured open APIs for all business modules and integrations.	MUST	5.9.3—Open Interfaces
CODE-0032	APIs MUST follow a versioning strategy (e.g., semantic versioning or URL-based versioning) with a documented backward compatibility and deprecation policy approved by OHIP.	MUST	5.9.3—Open Interfaces

Req. ID	Requirement Statement	Tier	Standard Section
CODE-0033	API contracts MUST be defined using a machine-readable specification format (e.g., OpenAPI/Swagger) and maintained in the State-owned repository alongside the code they describe.	MUST	5.9.3—Open Interfaces
CODE-0034	The Contractor MUST evaluate existing OHIP Medicaid technologies, shared services, and enterprise components before developing new capabilities. Where existing enterprise services are available (e.g., IAM, integration platform, monitoring), the Contractor MUST consume them rather than build alternatives.	MUST	5.9.4—Reuse and Interoperability
CODE-0035	Include structured documentation comments for all public functions, classes, and packages sufficient for reference generation.	MUST	5.9.5—Code Documentation
CODE-0036	Maintain README.md files explaining how to build and test the code.	MUST	5.9.5—Code Documentation
CODE-0037	Repositories MUST maintain architecture and integration documentation sufficient to support independent operation, maintenance, and transition by a successor Contractor. This includes, at minimum: system context diagrams, dependency maps, deployment architecture, and integration flows.	MUST	5.9.5—Code Documentation
CODE-0038	Designate a repository administrator for every project.	SHOULD	5.10.1—Version Control
CODE-0039	Manage all code changes in a version-controlled repository and track every modification to an individual user account.	MUST	5.10.1—Version Control
CODE-0040	Link each code change to a work item or requirement for full traceability.	MUST	5.10.1—Version Control
CODE-0041	Production and release branches MUST enforce branch protection controls preventing direct commits, force pushes, and bypass of required reviews or pipeline checks.	MUST	5.10.1—Version Control
CODE-0042	Release artifacts MUST be traceable to source commits and generated through controlled CI/CD pipelines. Manual builds for production deployment are prohibited.	MUST	5.10.1—Version Control

Req. ID	Requirement Statement	Tier	Standard Section
CODE-0043	Not merge code with the main branch without at least one (1) qualified reviewer's approval. Configure repository controls (branch protection) to enforce this review and approval before any merge to protected branch(es).	SHOULD	5.10.2—Peer Review
CODE-0044	Enable cryptographic signing of commits and use signed tags or checksums for release artifacts.	MUST	5.10.3—Integrity of Changes
CODE-0045	Ensure all deployments originate from version-controlled source. Direct changes to production are prohibited.	MUST	5.10.3—Integrity of Changes
CODE-0046	Develop code in an environment isolated from test and production environments.	MUST	5.10.4—Developer Environments
CODE-0047	Coordinate code changes with associated database changes in the same Release per DIM Section 4.5.3.	MUST	5.10.4—Developer Environments
CODE-0048	Production PHI/PII MUST NOT be used in development or test environments unless explicitly authorized by OHIP and protected in accordance with OHIP security and privacy standards. The Contractor MUST use de-identified, masked, or synthetic data for development and testing.	MUST	5.10.4—Developer Environments
CODE-0049	Provide code with contextual logs for all errors and security-relevant events per the defined minimum fields.	MUST	5.11.1—Comprehensive Logging
CODE-0050	Follow OWASP's Logging Cheat Sheet for event taxonomy and format guidance.	SHOULD	5.11.1—Comprehensive Logging
CODE-0051	Logs MUST NOT contain unmasked Protected Health Information (PHI), Personally Identifiable Information (PII), credentials, secrets, tokens, or cryptographic material. All sensitive data MUST be masked or redacted prior to logging.	MUST	5.11.1—Comprehensive Logging
CODE-0052	Ensure all logs are centralized, uniform, and retained per the OHIP Log Management Standard.	MUST	5.11.2—Log Management
CODE-0053	Implement end-to-end distributed tracing to capture request flows across services.	MUST	5.11.3—Performance Observability
CODE-0054	Remove unused or obsolete code, libraries, features, and documentation.	MUST	5.11.4—Continuous Code Maturity

Req. ID	Requirement Statement	Tier	Standard Section
CODE-0055	Use the Department-designated automated tools to scan for code smells, complexity, and structural weaknesses.	MUST	5.11.4—Continuous Code Maturity
CODE-0056	Set targets for key quality factors (Security, Reliability, Performance Efficiency, Maintainability) per ISO/IEC 5055:2021.	MUST	5.11.4—Continuous Code Maturity
CODE-0057	Evaluate quality metrics continuously and act to improve code quality to comply with defined goals.	MUST	5.11.4—Continuous Code Maturity
CODE-0058	Provide OHIP with access to all code and artifacts within a State-owned central repository from project initiation.	MUST	5.11.4—Continuous Code Maturity
CODE-0059	The Contractor MUST track, prioritize, and remediate identified technical debt items according to severity classifications and remediation timelines defined by OHIP. Technical debt items MUST be visible in the OHIP-approved project management tool.	MUST	5.11.4—Continuous Code Maturity
CODE-0060	The Lead Maintainer MUST use the Department-designated code-quality and security (SAST) scanning tools and retain the resulting reports as audit evidence for CMS certification.	MUST	5.11.4—Continuous Code Maturity
Section 6.0—Test Management Standard TMS-0001 to TMS-0076 76 Requirements: MUST 74 SHOULD 1 MAY 1			
TMS-0001	Ensure all testing activities adhere to the CMS Medicaid Enterprise System Testing Guidance Framework.	MUST	6.3.1 – MES Testing Guidance Framework
TMS-0002	For Streamlined Modular Certification modules, also meet CMS Medicaid Streamlined Certification Guidance requirements.	MUST	6.3.1 – MES Testing Guidance Framework
TMS-0003	Testing scope, rigor, and control selection MUST be proportionate to business, security, operational, and integration risk. The Contractor MUST document the risk-based rationale for test depth and coverage in the Test Management Plan.	MUST	6.3.1 – MES Testing Guidance Framework
TMS-0004	Develop and provide OHIP with a Test Management Plan Deliverable satisfying quality control and CMS certification requirements.	MUST	6.4 – Test Planning
TMS-0005	Obtain OHIP approval of the Test Management Plan as part of contract execution.	MUST	6.4 – Test Planning

Req. ID	Requirement Statement	Tier	Standard Section
TMS-0006	Obtain OHIP approval for any subsequent changes to the Test Management Plan before affected testing begins.	MUST	6.4 – Test Planning
TMS-0007	The Contractor MUST provide complete, traceable, and reproducible testing evidence demonstrating comprehensive and passing tests for all quality control types required by this standard.	MUST	6.4 – Test Planning
TMS-0008	The Contractor MUST maintain bidirectional traceability between requirements, test cases, test results, defects, and release artifacts. Traceability MUST be demonstrable on demand for CMS certification review.	MUST	6.4 – Test Planning
TMS-0009	Develop and maintain comprehensive test artifacts throughout Test Execution: test cases, scenarios, scripts, results, and reports.	MUST	6.5 – Test Execution
TMS-0010	Submit a Test Results Report documenting the outcomes of all testing activities.	MUST	6.5 – Test Execution
TMS-0011	Use OHIP-approved test management tools for test case management, execution tracking, defect management, reporting, and collaboration.	MUST	6.5.1 – Test Management Tools
TMS-0012	Ensure test management tools provide traceability between test cases, requirements, and defects.	MUST	6.5.1 – Test Management Tools
TMS-0013	Test management platforms MUST maintain immutable audit history for test execution, defect modification, status changes, approvals, and evidence updates.	MUST	6.5.1 – Test Management Tools
TMS-0014	Testing tools MUST implement role-based access control aligned with least privilege principles and segregation of duties between test execution, defect management, and test approval.	MUST	6.5.1 – Test Management Tools
TMS-0015	Provide one or more UAT environments for OHIP's exclusive use, capable of testing all functionality planned for production.	MUST	6.5.2 – UAT Environment
TMS-0016	Provide at least one UAT environment dedicated exclusively to OHIP user testing, isolated from Contractor access.	MUST	6.5.2 – UAT Environment

Req. ID	Requirement Statement	Tier	Standard Section
TMS-0017	The Contractor MUST establish and maintain a test data management approach that addresses: provisioning of test data sets, use of synthetic or masked data for all non-production environments, refresh cadence, retention policies, and traceability of test data sets to the test cases they support.	MUST	6.5.3 – Test Data Management
TMS-0018	Test data sets used for verification MUST be documented and reproducible to support re-testing and CMS audit review.	MUST	6.5.3 – Test Data Management
TMS-0019	Provide or demonstrate all test activities and artifacts to OHIP upon request, prior to production release signoff.	MUST	6.5.4 – Test Transparency
TMS-0020	Production PHI or PII MUST NOT be used in UAT or test environments unless explicitly authorized by OHIP and protected in accordance with OHIP security and privacy standards. The Contractor MUST use de-identified, masked, or synthetic test data.	MUST	6.5.4 – Test Transparency
TMS-0021	Allow OHIP to review the test process and procedures.	MUST	6.5.4 – Test Transparency
TMS-0022	The Contractor MUST maximize the use of automation and automated reporting to reduce cost and SDLC time. Where manual testing is required, the Contractor MUST document the justification and obtain OHIP approval.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0023	Enforce adherence to adopted coding standards using automated tools.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0024	Implement structured peer code review processes before merging code changes.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0025	Implement comprehensive static code analysis as part of the development workflow.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0026	Perform sanity/smoke testing to verify critical functions before proceeding to more comprehensive testing.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0027	Develop comprehensive unit tests validating functionality of individual components in isolation.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0028	Execute system integration tests assessing interoperability between integrated modules and services.	MUST	6.5.5 – Mandatory Quality Controls

Req. ID	Requirement Statement	Tier	Standard Section
TMS-0029	Implement system/end-to-end testing validating complete workflows in a production-like environment.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0030	Undertake solutions exploration testing during early feature development to assess initial designs.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0031	Conduct detailed regression testing to ensure new changes do not introduce regressions.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0032	Perform interface testing to verify data exchange integrity across system interfaces.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0033	Execute data conversion and migration testing to verify accurate transformation and migration.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0034	Perform performance, stress, and load testing to determine benchmarks and maximum operating capacity.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0035	Conduct comprehensive security reviews and testing throughout any solution implementation.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0036	Implement ongoing security and vulnerability testing covering OWASP Top 10, CVE scanning, and penetration testing.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0037	Perform rigorous accessibility testing to verify compliance with Section 508 and WCAG 2.1 AA.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0038	Conduct usability testing with real users to identify usability issues and improve user experience.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0039	Support OHIP in conducting UAT in which OHIP validates solution functionality before production deployment.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0040	Perform parallel testing when transitioning from legacy systems.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0041	Perform post-go-live testing throughout ongoing operations and support.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0042	Regularly validate business continuity and disaster recovery plans.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0043	Prepare for and participate in the CMS Operational Readiness Review (ORR).	MUST	6.5.5 – Mandatory Quality Controls

Req. ID	Requirement Statement	Tier	Standard Section
TMS-0044	Engage in the CMS certification review process.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0045	Utilize synthetic monitoring for continuous automated production checks.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0046	Implement synthetic alerting mechanisms providing real-time notifications when anomalies are detected.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0047	Automate SLA monitoring and reporting for key indicators including uptime and response times.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0048	Establish real-time SLA alerting and defined escalation paths for threshold breaches.	SHOULD	6.5.5 – Mandatory Quality Controls
TMS-0049	Apply mandatory quality controls tailored for specific release types, subject to documented risk assessment and OHIP approval.	MAY	6.5.5 – Mandatory Quality Controls
TMS-0050	Track and maintain the number of issues and defects by type and per Release discovered during any testing activity.	MUST	6.5.6 – Defect Rate
TMS-0051	Provide root cause analysis after each Release on all major or critical defects.	MUST	6.5.6 – Defect Rate
TMS-0052	The Contractor MUST track and report escaped defects—defects identified in production that were not detected during pre-production testing. Escaped defect rates MUST be reported per release and analyzed for root cause.	MUST	6.5.6 – Defect Rate
TMS-0053	The Contractor MUST provide recommendations on quality control activities that may be added or enhanced to prevent similar defects in future Releases, especially following UAT or production-escaped defects.	MUST	6.5.6 – Defect Rate
TMS-0054	Submit an OHIP-approved Defect Management Plan, updated to reflect the testing type.	MUST	6.5.7 – Defect Management Plan
TMS-0055	Ensure the Defect Management Plan includes processes to identify, assess, manage, coordinate, and align defect resolution.	MUST	6.5.7 – Defect Management Plan

Req. ID	Requirement Statement	Tier	Standard Section
TMS-0056	Root cause analysis MUST classify defects using a standardized taxonomy that includes, at minimum: requirements deficiency, design deficiency, coding error, configuration error, integration failure, infrastructure issue, data quality issue, or operational/process failure.	SHOULD	6.5.7 – Defect Management Plan
TMS-0057	Ensure every Defect Report captures at minimum: unique ID, title, description, steps to reproduce, environment, severity/priority, attachments, expected vs actual results, status, assignee, dates, and resolution notes.	SHOULD	6.5.8 – Defect Report
TMS-0058	Make testing and quality metrics available for CMS, including requirement coverage, code coverage, defect list, severity charts, age, and reopening statistics.	SHOULD	6.5.9 – Testing and Quality Metrics for CMS
TMS-0059	Testing metrics MUST include release-over-release trend analysis for defect density, defect escape rate, recurring root cause categories, and test coverage progression.	SHOULD	6.5.9 – Testing and Quality Metrics for CMS
TMS-0060	Testing metrics MUST include release-over-release trend analysis for defect density, defect escape rate, recurring root cause categories, and test coverage progression.	MUST	6.5.10 – Artifact Retention
TMS-0061	Testing evidence submitted for CMS certification or OHIP approval MUST be timestamped, immutable once submitted, and reproducible. Evidence MUST include sufficient context (environment, data set, date, tester identity) to independently validate the test outcome.	MUST	6.5.11 – Evidence Integrity
TMS-0062	Establish and maintain operational quality checks after each Release.	MUST	6.6 – Operational Quality Checks
TMS-0063	Ensure operational quality checks provide visibility into production issues, system reliability, and overall health.	MUST	6.6 – Operational Quality Checks
TMS-0064	Monitor and track user-reported issues from all end users (members/consumers, providers, State and county staff, and Contractor operational staff): categorize, maintain, and analyze severity, frequency, and trends.	MUST	6.6 – Operational Quality Checks
TMS-0065	The Contractor MUST provide recommendations for quality control enhancements—such as expanded test scenarios, strengthened validation steps, or additional regression coverage—to prevent recurrence of production issues. Recommendations MUST be documented and tracked in the OHIP-approved project management tool.	MUST	6.6 – Operational Quality Checks

Req. ID	Requirement Statement	Tier	Standard Section
TMS-0066	Develop, implement, and maintain system health metrics; establish automated alerts for threshold breaches.	MUST	6.6 – Operational Quality Checks
TMS-0067	The Contractor MUST evaluate system health trends and provide actionable recommendations to mitigate risks or performance degradation, reported at a cadence defined by OHIP.	MUST	6.6 – Operational Quality Checks
TMS-0068	Produce and submit all Deliverables in Appendix A in the format and cadence specified by OHIP.	MUST	Appendix A
TMS-0069	Upon notice of modifications to Deliverables and Artifacts, comply with all modifications promptly.	MUST	Appendix A
TMS-0070	Develop a comprehensive Test Management Plan and detailed test cases for each Release.	MUST	Appendix A
TMS-0071	Develop a Test Results Report providing an overview of testing activities, outcomes, and quality metrics.	MUST	Appendix A
TMS-0072	Document the planned timeline, phases, and milestones for all testing activities within the Test Management Plan.	MUST	Appendix A
TMS-0073	Implement automated API contract tests for all inter-module and inter-vendor integrations, integrated into the CI/CD pipeline, to detect breaking API changes before they reach SIT.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0074	Run Software Composition Analysis (SCA) / Dependency Scanning tools on every build; enforce zero tolerance for critical CVEs in production-bound artifacts; remediate high CVEs within the timeframe defined in the OHIP Security, Privacy, and Compliance Domain Document, Security and Privacy Compliance Standard.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0075	Scan all Infrastructure as Code (IaC) artifacts for security misconfigurations as a mandatory gate in the CI/CD pipeline before infrastructure is provisioned in any environment.	MUST	6.5.5 – Mandatory Quality Controls
TMS-0076	Define and maintain within the Test Management Plan a test environment strategy covering every required test level, including environment-to-test-level mapping, production configuration parity for pre-deployment verification environments, provisioning and refresh controls, and test data management per Section 6.5.3.	SHOULD	6.5.2 – UAT Environment

Req. ID	Requirement Statement	Tier	Standard Section
Section 7.0—Release Management Standard RLM-0001 to RLM-0084 84 Requirements: MUST 76 SHOULD 6 MAY 2			
RLM-0001	Create an initial release schedule in collaboration with OHIP State Business Leaders.	MUST	7.3 – Release Types
RLM-0002	Report all Expedited and HotFix releases as Standalone releases rather than part of the Enterprise suite.	MUST	7.3 – Release Types
RLM-0003	Plan and develop all Major releases in collaboration with OHIP.	MUST	7.3.2 – Major Release
RLM-0004	Subject Major releases to thorough review by technical leads, project managers, and stakeholders.	MUST	7.3.2 – Major Release
RLM-0005	Deploy Major releases on an OHIP-approved cadence.	MUST	7.3.2 – Major Release
RLM-0006	Test all Major releases in lower environments before receiving OHIP approval for production deployment.	MUST	7.3.2 – Major Release
RLM-0007	Adhere to all SDLC requirements for Major releases.	MUST	7.3.2 – Major Release
RLM-0008	Complete SIT, UAT, Regression Testing, and all other test types required by OHIP before Major release deployment approval.	MUST	7.3.2 – Major Release
RLM-0009	Test all Minor releases in lower environments before preparing for production deployment.	MUST	7.3.3 – Minor Release
RLM-0010	Adhere to all SDLC requirements for Minor releases.	MUST	7.3.3 – Minor Release
RLM-0011	Conduct appropriate reviews for targeted changes in Minor releases ensuring enhancements meet specified objectives.	MUST	7.3.3 – Minor Release
RLM-0012	Complete a full SIT, UAT, and Regression cycle before receiving OHIP approval for Minor release deployment.	MUST	7.3.3 – Minor Release
RLM-0013	Obtain OHIP approval before including enhancements beyond bug fixes in a Minor release.	MAY	7.3.3 – Minor Release
RLM-0014	Obtain OHIP approval for all Expedited releases.	MUST	7.3.4 – Expedited Release
RLM-0015	Test all Expedited releases in lower environments before production deployment.	MUST	7.3.4 – Expedited Release

Req. ID	Requirement Statement	Tier	Standard Section
RLM-0016	Complete an agreed-upon set of testing and validation before deploying an Expedited release; may bypass full SDLC with OHIP approval.	MAY	7.3.4 – Expedited Release
RLM-0017	Document and communicate all enhancements and fixes applied in each Expedited release.	MUST	7.3.4 – Expedited Release
RLM-0018	Obtain OHIP approval for all HotFix releases.	MUST	7.3.5 – HotFix Release
RLM-0019	Complete an agreed-upon set of tests adequate for the HotFix timeline without significantly delaying deployment.	MUST	7.3.5 – HotFix Release
RLM-0020	Complete additional testing and validation for HotFix content in the next numbered software release.	MUST	7.3.5 – HotFix Release
RLM-0021	Limit HotFix releases primarily to Severity 1 and 2 items; Severity 3 items require additional OHIP and Business Leadership review.	MUST	7.3.5 – HotFix Release
RLM-0022	Track and report dates, status, and risks for all release milestones.	MUST	7.4 – Release Milestones
RLM-0023	Work with OHIP PMO to ensure initial release scope is within the capacity of testing and supporting teams.	MUST	7.4.1 – Scope Phase
RLM-0024	Provide planned release freeze dates prior to the start of any testing phase.	MUST	7.4.1 – Scope Phase
RLM-0025	Use only de-identified, masked, or synthetic data in non-production environments. Production PHI or PII MUST be used in a non-production environment only when explicitly authorized by OHIP and protected in accordance with the OHIP Security and Privacy Compliance Standard.	MUST	7.4.1 – Scope Phase
RLM-0026	Promote releases through the defined environment sequence (e.g., Development → SIT → UAT → Staging → Production); no environment may be bypassed without documented OHIP approval.	MUST	7.4.1 – Scope Phase
RLM-0027	Track and report Deployment/Go-Live milestones: PRR date and result, Go/No-Go result, Final Deployment Approval, Deployment report, Go-Live date.	MUST	7.4.2 – Deployment/Go-Live

Req. ID	Requirement Statement	Tier	Standard Section
RLM-0028	Conduct post-Go-Live solution testing for updates, defect fixes, changes affecting the solution, and infrastructure patches.	MUST	7.4.3 – Post-Release Cycle
RLM-0029	Retain control of release schedule creation and reporting, adhering to OHIP constraints on type, milestones, and naming.	MUST	7.5.2 – Contractor Schedule Creation
RLM-0030	Strive to align the Contractor release schedule with the OHIP Enterprise Schedule.	SHOULD	7.5.2 – Contractor Schedule Creation
RLM-0031	Bring any potential schedule dependency or conflict to OHIP's attention promptly.	MUST	7.5.2 – Contractor Schedule Creation
RLM-0032	Schedule and obtain advance approval for all deployments during moratorium or blackout window periods.	MUST	7.5.4 – Moratoriums and Blackout Windows
RLM-0033	Submit at least one change request for every software release for tracking and approval.	SHOULD	7.6 – Change Control
RLM-0034	Submit all post-freeze scope change proposals through a documented risk-based review meeting.	MUST	7.5 – Post-Release Freeze
RLM-0035	Ensure only Severity 1 and 2 items are added after scope freeze; Severity 3 items require additional review.	MUST	7.7.1 – Approval Steps Required
RLM-0036	When adding/removing items after scope freeze: update analysis, archive it, present justification to OHIP Leadership, confirm scope in a meeting, document reason, and send informational email to UAT Manager.	MUST	7.7.1 – Approval Steps Required
RLM-0037	If additional resources are required due to a post-freeze addition, note this clearly in the email with quantity and names.	MUST	7.7.1 – Approval Steps Required
RLM-0038	Document approved post-scope-freeze changes and send to OHIP PMO including all required fields.	MUST	7.7.2 – Post-Scope Freeze Template
RLM-0039	Populate the trade-off incident field when a replacement item is being swapped.	SHOULD	7.7.2 – Post-Scope Freeze Template
RLM-0040	Include in all release communications: issue description, release impact, planned schedule, and root cause analysis.	MUST	7.8 – Release Communications

Req. ID	Requirement Statement	Tier	Standard Section
RLM-0041	Propose content for release communications not specified in the RFP.	SHOULD	7.8 – Release Communications
RLM-0042	Submit the Release Implementation Completion Report following each deployment.	SHOULD	7.8.1 – Release Completion Report
RLM-0043	Initiate deployment activity planning as soon as a release is identified as having dependencies.	SHOULD	7.9 – Deployment Coordination
RLM-0044	Develop and maintain Deployment Plans from Integrating Modules into a comprehensive Release Management Plan.	SHOULD	7.9 – Deployment Coordination
RLM-0045	Lead, facilitate, and coordinate all deployment activities including integrating module inputs.	SHOULD	7.9 – Deployment Coordination
RLM-0046	Follow OHIP-provided processes defining deployment indicators, status reporting, communication expectations, and update cadences.	SHOULD	7.9 – Deployment Coordination
RLM-0047	Provide stakeholder distribution list updates consistently throughout the deployment cycle.	SHOULD	7.9.1 – Activities
RLM-0048	Provide deployment status updates at minimum intervals: Go-Live approval, deployment start, milestone completions, EOD/BOD if multi-day, deployment completion, and post-deployment validation.	SHOULD	7.9.1 – Activities
RLM-0049	Trigger a status callout when: a delay risk is identified, a milestone is missed, staffing concerns arise, moratoriums apply, or deployment is significantly ahead of schedule.	SHOULD	7.9.1 – Activities
RLM-0050	Establish a recurring status readout meeting after initial integration planning.	SHOULD	7.9.2 – Status Readout
RLM-0051	Employ a Go/No-Go process during all release cycles with multiple projects deploying with dependencies.	MUST	7.9.2 – Status Readout
RLM-0052	Submit a Release Plan to OHIP for approval for each release.	SHOULD	7.9.3 – Release Plan
RLM-0053	Submit a Hypercare Plan for OHIP approval and schedule a Hypercare Coordinator meeting before commencing post-deployment operations.	MUST	7.9.4 – Post-Deployment Hypercare
RLM-0054	Attempt internal resolution of coordination conflicts before escalating to OHIP.	MUST	7.9.5 – Escalation Process

Req. ID	Requirement Statement	Tier	Standard Section
RLM-0055	Follow the OHIP escalation path if a Requesting Module issue cannot be resolved by OHIP.	MUST	7.9.5 – Escalation Process
RLM-0056	Notify OHIP Project Managers immediately of any at-risk or missed milestone and provide impact, recovery date, and RCA within one week.	MUST	7.9.5.1 – Initial Reporter Tasks
RLM-0057	Propose the format for missed-milestone notifications unless specified in procurement documents.	SHOULD	7.9.5.1 – Initial Reporter Tasks
RLM-0058	Distribute Release Notes to identified stakeholders before each software deployment is complete.	MUST	7.10 – Release Notes
RLM-0059	Create and distribute Release Notes with all versioned software deployments.	MUST	7.10 – Release Notes
RLM-0060	Create best-effort Release Notes for HotFix and Expedited releases with as much standard content as possible.	SHOULD	7.10 – Release Notes
RLM-0061	Include the release contents portion in the change request for deployment.	MUST	7.10 – Release Notes
RLM-0062	Use a multi-tab MS Excel email attachment as the recommended Release Notes delivery format.	SHOULD	7.10 – Release Notes
RLM-0063	Utilize a Go/No-Go process in collaboration with the State prior to deploying new code in production.	MUST	7.11 – Go/No-Go Process
RLM-0064	Immediately after any HotFix or Expedited release, communicate to the distribution group with a key stakeholder identified for RCA completion.	SHOULD	7.12 – RCA Process
RLM-0065	Propose RCA contents unless specified in the RFP.	SHOULD	7.12 – RCA Process
RLM-0066	Conduct a defect triage process including severity review, business urgency assessment, impact analysis, and scheduling decisions.	SHOULD	7.14.1 – Defect Severity Assignment
RLM-0067	Fully analyze DEV/SIT/UAT impact for defects proposed post-freeze and present effort estimates to the business.	SHOULD	7.14.1 – Defect Severity Assignment
RLM-0068	Confirm the release label for each defect and present testing effort details to the UAT Manager.	MUST	7.14.1 – Defect Severity Assignment

Req. ID	Requirement Statement	Tier	Standard Section
RLM-0069	Participate in the Post-Release Review Meeting with Contractor Release Management Leadership representation.	MUST	7.14 – Post-Release Review Meeting
RLM-0070	Hold the Post-Release Review Meeting within two (2) weeks of each Major/Minor release Go-Live.	SHOULD	7.14 – Post-Release Review Meeting
RLM-0071	Create and update required KPI metrics monthly, documenting and providing creation methodology to OHIP before acceptance.	SHOULD	7.15 – KPIs
RLM-0072	Provide time/deployment-based statistics broken out by severity for overall scope per KPI table.	SHOULD	7.15.1 – Time/Deployment-Based Statistics
RLM-0073	Provide impact-based statistics broken out by severity per KPI table.	SHOULD	7.15 – KPIs
RLM-0074	Provide usable queries for KPI generation where possible.	SHOULD	7.15 – KPIs
RLM-0075	Provide regular communication when releases impact other OHIP information systems.	SHOULD	7.16 – Release Oversight
RLM-0076	Coordinate with OHIP and other Contractors' release schedules when implementing cross-system releases.	SHOULD	7.16 – Release Oversight
RLM-0077	Produce and submit all Deliverables in Appendix A in the format and cadence specified by OHIP.	MUST	Appendix A
RLM-0078	Upon notice of modifications to Deliverables and Artifacts, comply with all modifications promptly.	MUST	Appendix A
RLM-0079	Develop a Defect Management Plan providing the approach, activities, and roles for handling defects.	MUST	Appendix A
RLM-0080	Develop Post-Release Documentation providing an overview of updates to design, configuration, and operational documentation.	MUST	Appendix A
RLM-0081	Develop a Release Implementation Completion Report serving as the formal record of each release's deployment activities.	MUST	Appendix A
RLM-0082	Develop a Release Management Plan documenting the approach and procedures for managing all aspects of a release.	MUST	Appendix A

Req. ID	Requirement Statement	Tier	Standard Section
RLM-0083	Develop a Release Plan for each release outlining how a product or feature will be introduced to end users.	MUST	Appendix A
RLM-0084	Schedule all routine maintenance on the OHIP Enterprise Release Calendar; manage maintenance that modifies deployed software or configuration as a Minor, Expedited, or HotFix Release per its criteria; and perform no maintenance outside a scheduled and communicated window except under HotFix provisions.	MUST	7.3 – Release Types
Section 8.0—SDLC Standard SDLC-0001 to SDLC-0170 170 Requirements: MUST 160 SHOULD 7 MAY 3			
SDLC-0001	Review and receive OHIP approval on the SDLC methodology prior to initiating any SDLC activities.	MUST	8.4 – Methodology Requirements
SDLC-0002	Manage and execute all information system projects through established project management practices with stakeholder input.	MUST	8.1 – Introduction
SDLC-0003	Integrate CMS-approved outcomes and metrics into all deliverables from the outset.	MUST	8.3 – CMS SMC Requirements
SDLC-0004	Ensure all MES solutions generate required data and uphold compliance standards set by federal and State requirements.	MUST	8.3 – CMS SMC Requirements
SDLC-0005	Maintain close collaboration with OHIP, Contractors, and the State CMS Officer via the State's liaison.	MUST	8.3 – CMS SMC Requirements
SDLC-0006	Submit timely documentation and actively participate in all CMS review sessions.	MUST	8.3 – CMS SMC Requirements
SDLC-0007	Deliver regular structured reports and demonstrations to highlight project progress.	MUST	8.3 – CMS SMC Requirements
SDLC-0008	Assist in developing and submitting evidence supporting CMS certification readiness.	MUST	8.3 – CMS SMC Requirements
SDLC-0009	Support OHIP in submitting post-certification operational reports validating sustained outcome achievements.	MUST	8.3 – CMS SMC Requirements
SDLC-0010	Identify and address performance issues promptly as specified by OHIP.	MUST	8.3 – CMS SMC Requirements

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0011	Adhere to any new CMS SMC deliverables or processes issued after this standard's publication date.	MUST	8.3 – CMS SMC Requirements
SDLC-0012	Thoroughly document the SDLC methodology including all key processes and deliverables.	MUST	8.4.1 – Overarching Requirements
SDLC-0013	Obtain OHIP approval for all SDLC methodologies including tools utilized.	MUST	8.4.1 – Overarching Requirements
SDLC-0014	Conduct proof of concept or prototype demonstrations monthly, or on an OHIP-determined cadence.	SHOULD	8.4.1 – Overarching Requirements
SDLC-0015	Clearly identify and obtain approval of all OHIP staff commitment and resource requirements before project initiation.	MUST	8.4.1 – Overarching Requirements
SDLC-0016	Ensure OHIP staff have access to all SDLC-produced artifacts.	MUST	8.4.1 – Overarching Requirements
SDLC-0017	Ensure any proposed changes to existing solution architecture follow SDLC change control processes and receive OHIP approval.	MUST	8.4.1 – Overarching Requirements
SDLC-0018	Gather and define the complete set of detailed requirements for the solution in the Requirements Analysis phase.	MUST	8.4.2.1 – Requirements Analysis Planning
SDLC-0019	Conduct Work Sessions with end-users to validate and document detailed functional and non-functional requirements.	MUST	8.4.2.1 – Requirements Analysis Planning
SDLC-0020	Document validated requirements within the RTM.	MUST	8.4.2.1 – Requirements Analysis Planning
SDLC-0021	Obtain OHIP approval on all validated requirements in the RTM.	MUST	8.4.2.1 – Requirements Analysis Planning
SDLC-0022	Use approved requirements as the baseline reference for all subsequent phases; changes may be made only through the approved requirements change control process (Section 3.7).	MUST	8.4.2.1 – Requirements Analysis Planning

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0023	Define the overall solution architecture including technology stack, components, and interactions with supporting diagrams.	MUST	8.4.2.2 – Solution Design
SDLC-0024	Design user interfaces and system interfaces, including communication protocol specifications.	MUST	8.4.2.2 – Solution Design
SDLC-0025	Design the database schema and incorporate it into the Data Management Plan.	MUST	8.4.2.2 – Solution Design
SDLC-0026	Prepare all design documents, diagrams, and specifications for the Development phase.	MUST	8.4.2.2 – Solution Design
SDLC-0027	Conduct design reviews with all stakeholders to verify compliance, feasibility, consistency, and completeness.	MUST	8.4.2.2 – Solution Design
SDLC-0028	Write all code in OHIP's designated repository per approved design specifications and the OHIP Code Standard.	MUST	8.4.2.3 – Development
SDLC-0029	Conduct peer code reviews to verify quality, standards compliance, and proper requirements implementation.	MUST	8.4.2.3 – Development
SDLC-0030	Conduct oversight reviews to ensure code complies with OHIP MES Standards.	SHOULD	8.4.2.3 – Development
SDLC-0031	Adhere to OHIP's version control requirements to manage and track all code changes.	SHOULD	8.4.2.3 – Development
SDLC-0032	Document all code including inline comments and developer guide documentation.	SHOULD	8.4.2.3 – Development
SDLC-0033	Integrate individual modules using integration tests to validate interoperability.	SHOULD	8.4.2.3 – Development
SDLC-0034	Plan and execute data migration from legacy systems using appropriate tools.	MUST	8.4.2.3 – Development
SDLC-0035	Follow all requirements in the OHIP Test Management Standard for the Waterfall testing phase.	MUST	8.4.2.4 – Testing
SDLC-0036	Verify and validate the solution against design specifications and business requirements.	MUST	8.4.2.4 – Testing
SDLC-0037	Conduct systematic QA testing to identify and resolve all defects to DOH's approval before go-live.	MUST	8.4.2.4 – Testing

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0038	Deliver a fully validated, deployable product ready for the Deployment phase.	MUST	8.4.2.4 – Testing
SDLC-0039	Follow all requirements in the OHIP Release Management Standard for production deployments.	MUST	8.4.2.5 – Deployment
SDLC-0040	Develop a detailed Deployment Plan including approval steps, installation, configuration, roles, risks, and contingency plans.	MUST	8.4.2.5 – Deployment
SDLC-0041	Practice deployment in a controlled Pre-Production or Staging environment (Deployment Procedure Reviews).	MUST	8.4.2.5 – Deployment
SDLC-0042	Complete the installation and configuration of the solution in the Production environment.	MUST	8.4.2.5 – Deployment
SDLC-0043	Manage all configuration settings and environment variables.	MUST	8.4.2.5 – Deployment
SDLC-0044	Perform final verification and validation checks upon production deployment.	MUST	8.4.2.5 – Deployment
SDLC-0045	Prepare deployment guides, installation manuals, and user documentation throughout the Deployment phase.	MUST	8.4.2.5 – Deployment
SDLC-0046	Provide Hypercare support following deployment, transitioning to the M&O team.	MUST	8.4.2.5 – Deployment
SDLC-0047	Adhere to the OHIP Requirements Management Standard in the Agile Requirements Definition phase.	MUST	8.4.3.1 – Requirements Definition
SDLC-0048	Identify key stakeholders and validate engagement throughout the project.	MUST	8.4.3.1 – Requirements Definition
SDLC-0049	Capture high-level requirements broken down into user stories or features for the Product Backlog.	MUST	8.4.3.1 – Requirements Definition
SDLC-0050	Continually refine the Product Backlog to align with business goals.	MUST	8.4.3.1 – Requirements Definition

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0051	Maintain a dynamic, prioritized Product Backlog containing Epics, Features, and User Stories.	MUST	8.4.3.1 – Requirements Definition
SDLC-0052	Create effective user stories meeting SMART, INVEST, and Definition of Done criteria.	MUST	8.4.3.1 – Requirements Definition
SDLC-0053	Implement backlog refinement as a continuous process including clarification, prioritization, estimation, decomposition, and routine sessions.	MUST	8.4.3.2 – Backlog Refinement
SDLC-0054	Submit a Release Plan for OHIP approval prior to initiating any Release.	MUST	8.4.3.3 – Release Planning
SDLC-0055	Commence each Release with Sprint 0 for planning, environment preparation, and stakeholder alignment.	MUST	8.4.3.3 – Release Planning
SDLC-0056	Maintain a continuously refined Product Backlog ensuring readiness for upcoming Sprints.	SHOULD	8.4.3.4 – Sprint Planning
SDLC-0057	Provide a Sprint Backlog with selected PBIs, Sprint Goal, detailed tasks, and progress tracking.	SHOULD	8.4.3.4 – Sprint Planning
SDLC-0058	Implement continuous backlog refinement maintaining accuracy, stakeholder engagement, decomposed tasks, and Sprint Goal alignment.	SHOULD	8.4.3.4 – Sprint Planning
SDLC-0059	Conduct daily standup meetings with all team members to monitor progress and identify impediments.	SHOULD	8.4.3.5 – Sprints
SDLC-0060	Adhere to coding and test management standards, conducting unit testing and CI/CD during Sprints.	SHOULD	8.4.3.5 – Sprints
SDLC-0061	Follow the OHIP Test Management Standard and implement continuous testing and QA throughout development.	SHOULD	8.4.3.6 – Continuous Testing and QA
SDLC-0062	Execute automated tests (security, unit, integration, end-to-end) and manual/exploratory testing.	MUST	8.4.3.6 – Continuous Testing and QA
SDLC-0063	Develop and maintain automated unit tests for application components, meeting the coverage thresholds defined in Section 5.0; apply TDD where appropriate.	MUST	8.4.3.6 – Continuous Testing and QA

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0064	Apply BDD when appropriate to ensure developer, QA, and stakeholder alignment on expected behaviors.	SHOULD	8.4.3.6 – Continuous Testing and QA
SDLC-0065	Perform automated regression testing to confirm new changes do not negatively affect existing functionality.	MUST	8.4.3.6 – Continuous Testing and QA
SDLC-0066	Validate each user story against acceptance criteria to ensure quality and readiness.	MUST	8.4.3.6 – Continuous Testing and QA
SDLC-0067	Conduct a collaborative review session with stakeholders after each Sprint to demonstrate completed functionality.	SHOULD	8.4.3.7 – Review
SDLC-0068	Ensure Sprint Reviews confirm ongoing alignment and demonstrate agreed-upon acceptance criteria.	SHOULD	8.4.3.7 – Review
SDLC-0069	Close each Sprint with a Retrospective identifying what went well, what did not, and areas for improvement.	SHOULD	8.4.3.8 – Retrospective
SDLC-0070	Develop action items and strategies for improving future Sprints from each Retrospective.	SHOULD	8.4.3.8 – Retrospective
SDLC-0071	Follow all requirements within the OHIP Release Management Standard for Agile deployments.	SHOULD	8.4.3.9 – Deployment
SDLC-0072	Prepare the deployment package ensuring all tests are passed, acceptance criteria met, and defects resolved.	MUST	8.4.3.9 – Deployment
SDLC-0073	Provide all deployment documentation prior to production deployment.	MUST	8.4.3.9 – Deployment
SDLC-0074	For Waterfall + Scrum: perform Sprint 0 Release Planning, then execute a full SDLC per Release.	SHOULD	8.4.4.1 – Waterfall + Scrum
SDLC-0075	Conduct Refinement Sessions to identify end-user needs during Sprints.	SHOULD	8.4.4.1 – Waterfall + Scrum
SDLC-0076	Conduct Sprint-end demos to showcase new functionality to the OHIP Product Owner.	SHOULD	8.4.4.1 – Waterfall + Scrum
SDLC-0077	Provide OHIP with developed prototypes and testing scripts for Solution Exploration; address all OHIP feedback.	MUST	8.4.4.1 – Waterfall + Scrum
SDLC-0078	Execute all required production readiness and pre-deployment testing before production promotion.	MUST	8.4.4.1 – Waterfall + Scrum

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0079	For Waterfall + Iterative Design: follow all Waterfall requirements, plus iterative design steps until OHIP approves.	MUST	8.4.4.2 – Waterfall + Iterative Design
SDLC-0080	During Iterative Exploration: identify and document multiple design alternatives aligned with OHIP business needs.	MUST	8.4.4.2 – Waterfall + Iterative Design
SDLC-0081	During Iterative Option Selection: recommend a best-fit option with clearly documented rationale.	MUST	8.4.4.2 – Waterfall + Iterative Design
SDLC-0082	During Iterative Demonstration: present the recommended option using workflows, models, or a demonstration of a configured COTS solution.	MUST	8.4.4.2 – Waterfall + Iterative Design
SDLC-0083	During Iterative Assessment: obtain OHIP design approval before proceeding; if not received, iterate from Exploration.	MUST	8.4.4.2 – Waterfall + Iterative Design
SDLC-0084	Agree upon and document any methodology not specified in this standard with OHIP before starting work.	MUST	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0085	Develop Training Plans addressing distinct needs of both technical resources and end-users.	MUST	8.5.1 – Training
SDLC-0086	Ensure technical training covers architecture, integration, security, and maintenance.	MUST	8.5.1 – Training
SDLC-0087	Ensure end-user training is role-based, scenario-driven, accessible, and includes hands-on practice and job aids.	MUST	8.5.1 – Training
SDLC-0088	Plan for and provide Hypercare services at OHIP-designated locations for a mutually agreed period after Go-Live.	MUST	8.5.2 – Hypercare
SDLC-0089	Submit a Hypercare Plan for OHIP approval defining scope, duration, roles, coverage, escalation paths, and exit criteria.	MUST	8.5.2.1 – Hypercare Planning
SDLC-0090	Provide reliable and knowledgeable staff to support configuration and customization during Hypercare.	MUST	8.5.2.1 – Hypercare Planning
SDLC-0091	Resolve incidents and provide knowledge transfer to OHIP during Hypercare.	MUST	8.5.2.1 – Hypercare Planning
SDLC-0092	Submit a Hypercare Completion Report for OHIP review containing all incidents, defects, and resolutions.	MUST	8.5.2.1 – Hypercare Planning

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0093	Monitor performance in real time during Hypercare: availability, latency, throughput, error rates, and key transactions.	MUST	8.5.2.2 – Execution and Support Activities
SDLC-0094	Operate continuous incident management during Hypercare with immediate Sev1/Sev2 response.	MUST	8.5.2.2 – Execution and Support Activities
SDLC-0095	Integrate automated and manual testing into CI/CD during Hypercare to detect and prevent regressions.	MUST	8.5.2.2 – Execution and Support Activities
SDLC-0096	Correct data errors from the conversion process; perform data backfills at no additional cost during warranty.	MUST	8.5.2.2 – Execution and Support Activities
SDLC-0097	Maintain daily stakeholder standups during Hypercare including issue lists, SLA compliance, risks, and next-day plan.	MUST	8.5.2.2 – Execution and Support Activities
SDLC-0098	Complete Knowledge Transfer including runbooks and training prior to production Go-Live.	MUST	8.5.2.2 – Execution and Support Activities
SDLC-0099	Conduct refresher sessions as needed to educate stakeholders and address gaps observed at Go-Live.	SHOULD	8.5.2.2 – Execution and Support Activities
SDLC-0100	If rollback is required, correct critical and high defects immediately and obtain OHIP approval.	MUST	8.5.2.2 – Execution and Support Activities
SDLC-0101	Track and report KPIs including Contract-specific SLAs by severity, reporting compliance daily during Hypercare.	MUST	8.5.2.3 – Success Criteria and Exit/Extension
SDLC-0102	Meet all SLAs as defined in the Contract during Hypercare.	MUST	8.5.2.3 – Success Criteria and Exit/Extension
SDLC-0103	Develop Hypercare Exit Criteria for OHIP review and approval.	MUST	8.5.2.3 – Success Criteria and Exit/Extension
SDLC-0104	Exit Hypercare only when OHIP-approved exit criteria are met.	MUST	8.5.2.3 – Success Criteria and Exit/Extension
SDLC-0105	Request Hypercare extension only if OHIP-approved criteria are met.	MUST	8.5.2.3 – Success Criteria and Exit/Extension

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0106	Submit a Transition-In Management Plan (TIMP) for OHIP review and approval.	MUST	8.5.3 – Transition-In Phase
SDLC-0107	Coordinate with the Incumbent Contractor and OHIP through weekly joint planning sessions.	MUST	8.5.3 – Transition-In Phase
SDLC-0108	Develop, maintain, and review the transition-in schedule of all tasks, milestones, and dependencies within the Transition-In Management Plan.	MUST	8.5.3 – Transition-In Phase
SDLC-0109	Execute the Transition-In Management Plan (TIMP) while maintaining operational stability and SLA compliance.	MUST	8.5.3 – Transition-In Phase
SDLC-0110	Conduct and document knowledge transfer activities including walkthroughs, shadowing, and confirmation of understanding.	MUST	8.5.3 – Transition-In Phase
SDLC-0111	Assess solution architecture and assume full M&O responsibilities culminating in a final transition-in readiness review with OHIP.	MUST	8.5.3 – Transition-In Phase
SDLC-0112	Define and document the DDI responsibilities approach during Transition-In within the Transition-In Management Plan.	MUST	8.5.3 – Transition-In Phase
SDLC-0113	Produce and submit all Deliverables in Appendix A in the format and cadence specified by OHIP.	MUST	Appendix A
SDLC-0114	Upon notice of modifications to Deliverables and Artifacts, comply with all modifications promptly.	MUST	Appendix A
SDLC-0115	Apply SMART, INVEST, and Definition of Done criteria to all user stories.	MUST	Appendix F – SMART/INVEST/DoD
SDLC-0116	Ensure each user story meets SMART criteria: Specific, Measurable, Achievable, Relevant, Time-bound.	MUST	Appendix F – SMART/INVEST/DoD
SDLC-0117	Ensure each user story meets INVEST criteria: Independent, Negotiable, Valuable, Estimable, Small, Testable.	MUST	Appendix F – SMART/INVEST/DoD
SDLC-0118	Meet all elements of the Definition of Done before a user story can be accepted.	MUST	Appendix E – Definition of Done

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0119	Develop and maintain an SDLC Governance Plan, approved by OHIP, documenting: the selected methodology; phase gate criteria; SDLC change control procedures; and the CMS SMC evidence collection approach. Review and update the plan at each major phase transition.	MUST	8.4.1 – Overarching Requirements
SDLC-0120	Update the RTM at the close of each Sprint to reflect completed, in-progress, and deferred user stories and their test case linkages, in accordance with Section 3.0 (Requirements Management Standards) of this document.	MUST	8.4.3.5 – Sprints
SDLC-0121	Apply Behavior Driven Development (BDD) when appropriate to ensure alignment among developers, QA, and stakeholders on expected system behaviors, using a shared language (e.g., Gherkin) to define acceptance criteria prior to development.	SHOULD	8.4.3.6 – Continuous Testing and QA
SDLC-0122	Where the solution is pursuing CMS Streamlined Modular Certification, retain Sprint Review demonstration materials—recordings, screenshots, and documented outcomes—as required CMS certification evidence, organized and accessible for CMS review sessions.	MUST	8.4.3.7 – Review
SDLC-0123	Establish an automated or semi-automated evidence collection approach; index and link Sprint Review artifacts directly within the work item record at Sprint close; document the evidence indexing approach in the Test Management Plan prior to the first Sprint Review.	MUST	8.4.3.7 – Review
SDLC-0124	Where data migration scope is significant, structure data engineering and conversion activities as a dedicated parallel workstream with its own sprint cadence; document this approach in the Data Migration and Conversion Plan, approved by OHIP prior to execution.	SHOULD	8.4.3.9 – Deployment
SDLC-0125	Commence each Release under the Waterfall + Scrum Hybrid with Sprint (0) as defined in Section 8.4.3.3 (Release Planning), establishing the Release planning framework, high-level user stories, environment preparation, and stakeholder alignment before iterative development begins.	SHOULD	8.4.4.1 – Waterfall + Scrum

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0126	Limit Solution Exploration feedback to usability and design observations or defects against approved acceptance criteria only; route all new functional requests to the Product Backlog; not begin development on any Solution Exploration item without formal Product Backlog acceptance and OHIP Product Owner approval.	SHOULD	8.4.4.3 – Hybrid Iterative Development
SDLC-0127	Designate an OHIP Solution Exploration Coordinator responsible for enforcing Solution Exploration scope boundaries and ensuring all out-of-scope requests are formally routed to the Product Backlog during each Bundle's exploration period.	SHOULD	8.4.4.3 – Hybrid Iterative Development
SDLC-0128	Conduct a formal Code Quality Assessment of all incumbent Contractor codebases before assuming M&O responsibility; document all gaps (≥80% statement coverage, ≥75% branch coverage, 100% auth/authz coverage, zero Sev 1/2 static analysis findings) in a Transition-In Code Quality Gap Report submitted to OHIP within 30 days of assessment.	MUST	8.5.3 – Transition-In Phase
SDLC-0129	Obtain OHIP's written determination of responsibility for all identified code quality gaps before assuming contractual responsibility for code quality deficiencies attributed to the incumbent Contractor.	MUST	8.5.3 – Transition-In Phase
SDLC-0130	Conduct a CMS Certification Readiness Review as a mandatory gate prior to Go-Live approval, demonstrating that all functionality maps to defined CMS MES Outcomes as documented in the approved SMC Intake Form.	MUST	8.3.1 – CMS SMC Readiness Review Gate
SDLC-0131	Provide a consolidated CMS artifact package—including RTMs, VPAT documentation, security assessment results, and test results reports—to OHIP at least ten (10) business days prior to the scheduled Go-Live date for review and formal approval.	MUST	8.3.1 – CMS SMC Readiness Review Gate
SDLC-0132	Track and remediate all gaps identified during the CMS Readiness Review before Go-Live gate approval; document any unresolved gaps in a Risk Register with OHIP-approved mitigation plans and obtain OHIP written authorization before deployment proceeds.	MUST	8.3.1 – CMS SMC Readiness Review Gate

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0133	UAT Gate: All relevant training materials, user manuals, and job aids MUST be drafted and approved by OHIP before UAT commences, ensuring UAT findings reflect operational gaps rather than documentation deficiencies.	MUST	8.5.1 – Training (OCM Alignment)
SDLC-0134	Go-Live Gate: Provide documented evidence that all required end-user and help desk training has been executed, with participant sign-off sheets or training completion records collected and accessible for OHIP review before the Go-Live gate is approved.	MUST	8.5.1 – Training (OCM Alignment)
SDLC-0135	Develop and submit a Legacy Sunsetting Plan for OHIP approval defining the timeline for legacy read-only transition, interface severance, and final shutdown. Submit no later than the UAT gate of the Release replacing the legacy functionality.	MUST	8.6 – Decommissioning and Data Archival
SDLC-0136	Ensure all historical legacy data required for Medicaid compliance, federal audit trails, and legal holds is migrated to an OHIP-approved archival system before legacy decommissioning. Validate and document completeness and integrity in a Data Archival Verification Report.	MUST	8.6 – Decommissioning and Data Archival
SDLC-0137	Obtain formal written sign-off from the OHIP Product Owner and OHIP Security Officer before executing any destructive actions against legacy systems (interface severance, data deletion, or infrastructure shutdown).	MUST	8.6 – Decommissioning and Data Archival
SDLC-0138	When the V-Shaped methodology is selected, design and document the full test strategy, test cases, and acceptance criteria for each development phase before that phase's development work commences; submit test artifacts per phase to OHIP for approval as a phase entry gate; and maintain bilateral RTM traceability linking each requirement forward to its development artifact and backward to its corresponding test result.	MUST	8.4.2.7 – V-Shaped Methodology
SDLC-0139	The Contractor SHOULD evaluate the project against the methodology decision framework (Table 21) and document the alignment rationale in the SDLC Governance Plan.	SHOULD	8.4 - Methodology Requirements
SDLC-0140	Define and submit to OHIP for approval the anticipated spiral cycles, scope, risk analysis approach, and completion criteria.	SHOULD	8.4.4.5 Spiral SDLC

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0141	Conduct a formal Risk Analysis phase at the beginning of each spiral cycle and document findings in a Spiral Risk Register.	SHOULD	8.4.4.5 Spiral SDLC
SDLC-0142	Submit the Spiral Risk Register to OHIP for review and obtain approval of the risk mitigation approach before the Engineering phase begins.	SHOULD	8.4.4.5 Spiral SDLC
SDLC-0143	Conduct a formal Evaluation review with OHIP at the conclusion of each spiral cycle and obtain concurrence before proceeding.	SHOULD	8.4.4.5 Spiral SDLC
SDLC-0144	Update the RTM per the OHIP Requirements Management Standard at the conclusion of each spiral cycle.	SHOULD	8.4.4.5 Spiral SDLC
SDLC-0145	Follow the OHIP Release Management Standard and Test Management Standard for all deployment and testing activities.	SHOULD	8.4.4.5 Spiral SDLC
SDLC-0146	DevOps practices proposed for adoption MUST be documented in the SDLC Governance Plan, including CI/CD configuration, SDLC methodology integration, tool compliance, and CMS evidence generation approach	MUST	8.4.1.1 DevOps Practices Within SDLC Methodologies
SDLC-0147	Automated deployment pipelines MUST satisfy the release-level change control and Go/No-Go requirements defined in Section 7.0 before promoting any code to production.	MUST	8.4.1.1 DevOps Practices Within SDLC Methodologies
SDLC-0148	The Contractor MUST use one of the structured methodologies defined in this Standard as the primary delivery methodology for any OHIP MES project. RAD elements MAY be incorporated within an approved SDLC solely for designated prototyping or UI/UX exploration activities, via the Alternate Methodology Proposal process.	MUST	8.4.4.4 Proposing an Alternative Methodology
SDLC-0149	The Contractor MAY propose the adoption of DevOps practices (e.g., Continuous Integration/Continuous Delivery, infrastructure as code, automated testing and deployment) as part of the SDLC Governance Plan. DevOps is a delivery layer applied within an approved SDLC methodology and does not replace the phase gates, governance reviews, or OHIP approval requirements defined in this Standard.	MAY	8.4.1.1 – DevOps Practices Within SDLC Methodologies

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0150	All code developed, tested, and deployed through DevOps pipelines MUST comply with Section 5.0 (Code Standard) and Section 6.0 (Test Management Standard).	SHOULD	8.4.1.1 – DevOps Practices Within SDLC Methodologies
SDLC-0151	CI/CD pipelines MUST enforce segregation of duties so that no developer approves their own code for production deployment; production deployment approval MUST be performed by an independent qualified reviewer who did not author the change; and CI/CD administrative privileges MUST be restricted to authorized personnel and be auditable.	MUST	8.4.1.1 – DevOps Practices Within SDLC Methodologies
SDLC-0152	Submit a written Alternate Methodology Proposal to OHIP that includes a description of the proposed methodology, justification for why none of the methodologies defined in Sections 8.4.2 through 8.4.4 are suitable, a mapping of the proposed methodology's phases to the deliverables in Appendix A, and an assessment of any CMS SMC certification implications.	MUST	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0153	Obtain written OHIP approval of the Alternate Methodology Proposal before commencing any project activities under the alternate methodology.	MUST	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0154	OHIP SHOULD consider the proposed methodology's alignment with CMS MES Testing Guidance Framework expectations and SMC certification requirements when evaluating an Alternate Methodology Proposal.	SHOULD	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0155	The Contractor MAY propose Kanban as the primary workflow methodology for Operations and Maintenance (M&O) and steady-state support activities that fall outside of active DDI Releases.	MAY	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0156	A Kanban proposal MUST be submitted as an Alternate Methodology Proposal and MUST include the Kanban board structure and work-in-progress limits; the process for triaging, prioritizing, and escalating incoming work items; how SLA compliance will be tracked and reported to OHIP; and how Kanban activities will interface with the Release Management Standard (Section 7.0) when work items require formal release packaging for production deployment.	MUST	8.4.4.4 – Proposing an Alternative Methodology

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0157	Kanban is limited to M&O and steady-state support activities; new DDI efforts, major feature development, and projects receiving CMS SMC MUST use a structured methodology with defined phase gates as specified in Sections 8.4.2 through 8.4.4.3.	MUST	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0158	A RAD Alternate Methodology Proposal MUST describe which project activities will use RAD techniques, how RAD prototyping outputs will transition into the project's formal design and development process, and how regulatory compliance, security requirements, and CMS certification evidence will be maintained for deliverables originating from RAD activities.	MUST	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0159	Contractors seeking an incremental delivery approach SHOULD evaluate the Hybrid Iterative Development methodology or the Hybrid Waterfall + Scrum approach before proposing a standalone Incremental methodology.	SHOULD	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0160	If no approved hybrid approach meets the project's needs, the Contractor MAY propose an Incremental methodology as an Alternate Methodology Proposal per the requirements of Section 8.4.4.4.	MAY	8.4.4.4 – Proposing an Alternative Methodology
SDLC-0161	Conduct Joint Application Design (JAD) sessions with OHIP and designated stakeholders during the Requirements and Design activities to collaboratively define and validate requirements and solution design before development begins. JAD sessions are distinct from Solution Exploration, which OHIP conducts after development to verify the delivered solution.	SHOULD	8.4.4.3 – Hybrid Iterative Development
SDLC-0162	Obtain OHIP acceptance of the Iteration's functionality at the conclusion of Solution Exploration before that functionality advances to the End-to-End SIT phase.	SHOULD	8.4.4.3 – Hybrid Iterative Development
SDLC-0163	Maintain a log of outstanding questions and decisions raised during each Iteration (e.g., regarding flows, processes, and legacy documentation), record OHIP-provided answers, and close out open items prior to advancing to the next phase, as needed.	MUST	8.4.4.3 – Hybrid Iterative Development
SDLC-0164	Obtain OHIP Enterprise Architecture approval of the solution architecture artifacts defined in Section 9.0 prior to commencing development activities.	MUST	8.4.1 – Overarching Requirements

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0165	Document, assess for impact, and submit significant architecture changes occurring after initial approval through OHIP's established enterprise architecture governance and change control processes prior to implementation.	MUST	8.4.1 – Overarching Requirements
SDLC-0166	Support the State-led Cost Allocation Plan by providing the required cost data (typically on a quarterly basis), including the data needed to distinguish DDI from M&O costs, mailing from postage, and training costs, and to allocate costs across funding sources (e.g., MECM versus AMS) and facilities in accordance with applicable Federal Financial Participation (FFP) rules; document any cost-allocation assumptions that require State agreement.	MUST	8.3 – CMS Streamlined Modular Certification (SMC) Requirements
SDLC-0167	Comply with CMS Medicaid Enterprise Systems (MES) guidance, including providing the documentation and evidence required to support OHIP's Federal Financial Participation (FFP)—for example, the evidence required for Advance Planning Documents (APDs), Operational Readiness Reviews (ORR), Certification Reviews (CR), and Streamlined Modular Certification (SMC). Where CMS-required artifacts are not separately identified as RFP deliverables, the Contractor must provide the supporting documentation and evidence upon OHIP request, consistent with CMS Appendix A and other applicable CMS guidance.	MUST	8.3 – CMS Streamlined Modular Certification (SMC) Requirements
SDLC-0168	Document, within the Integration Plan, the design of all interfaces and processes involved in integrations during the Design phase, serving as the consolidated single point of reference for all integrations, including: an overview of integration points and interfaces; an interface catalog with summary characteristics; the detailed interface specification for each interface; batch control processes triggered by an interface (purpose, scope, roles, and responsibilities), with full batch job detail maintained in the Functional Design Document; and assumptions, constraints, and dependencies that may impact the integration design.	SHOULD	8.4.2.2 – Solution Design

Req. ID	Requirement Statement	Tier	Standard Section
SDLC-0169	Provide, within the Integration Plan, the detailed record and technical specification of each interface/integration, including: a description of the interface/integration (e.g., federal, state, county, private entities); detailed requirements; the approach and processing involved; communication protocols; endpoint configurations to connect to partner systems; orchestration details; the test schedule and cadence; and error handling.	SHOULD	8.4.2.2 – Solution Design
SDLC-0170	Document all batch jobs and batch processes in the Functional Design Document, regardless of triggering mechanism (interface-triggered, scheduled, or event-driven), including each job's purpose and scope; execution schedule or triggering mechanism; dependencies and job sequencing; error handling, restart, and recovery procedures; and the roles and responsibilities of all stakeholders.	SHOULD	8.4.2.2 – Solution Design
Section 9.0—Business and Technical Architecture Standard BTA-0001 to BTA-0025 25 Requirements: MUST 22 SHOULD 0 MAY 1			
BTA-0001	Reference a shared definitions catalog maintained in OHIP's Enterprise Architecture Repository for all business architecture artifacts.	MUST	9.3—Business Architecture Artifacts
BTA-0002	Apply standardized vocabulary for terms, capability names, and component descriptions across all domains to ensure architectural consistency.	MUST	9.3—Business Architecture Artifacts
BTA-0003	Develop a Business Capability Model that identifies and organizes high-level business functions into structured domains and sub-capabilities, leveraging recognized frameworks (e.g., APQC Process Classification Framework, BIZBOK), and structured into a logical hierarchy of core, supporting, and strategic capabilities.	MUST	9.3.1—Business Capability Model
BTA-0004	Develop a Business Capability Matrix that maps each capability to current and target maturity levels using an industry or customized maturity model, groups capabilities into business domains, and prioritizes them by strategic importance, performance gaps, and consumer impact.	MUST	9.3.2—Business Capability Matrix

Req. ID	Requirement Statement	Tier	Standard Section
BTA-0005	Develop Business Process Models using BPMN 2.0 notation that define process start and end points and boundaries, sequence activities, assign role responsibilities, include decision branches and conditions, define inputs and outputs, and are validated with stakeholders for logical consistency and completeness.	SHOULD	9.3.3—Business Process Model
BTA-0006	Develop a Use Case Model (UML behavioral) that defines the system boundary, identifies primary and secondary actors and their goals, and represents the use cases the system must perform.	MUST	9.3.4—Use Case Model
BTA-0007	Phrase use case names as actions (e.g., “Log In,” “Place Order,” “Generate Report”).	SHOULD	9.3.4—Use Case Model
BTA-0008	Document detailed use case descriptions that include, at minimum: Title, Actors, Preconditions, Main Flow, Alternate Flows, Postconditions, and Special Requirements.	MUST	9.3.4—Use Case Model
BTA-0009	Develop a Conceptual Architecture Diagram that provides a high-level, end-to-end view of product components, service layers, data integrations, and hosting or platform layers, consolidating content from the Application Component Model, Solution Architecture Model, Conceptual Integration Architecture Model, and Deployment Model.	MUST	9.4—Application Architecture Artifacts
BTA-0010	Create a Conceptual Application Integration Architecture Model (Contractor-neutral) and a Solution Architecture Model (the Contractor’s specification of products) to show the integration of applications.	MUST	9.4—Application Architecture Artifacts
BTA-0011	Analyze the individual systems or components to be integrated, identify their commonalities and differences, and define the requirements and relationships needed to integrate them into a cohesive system.	MUST	9.4.1—Conceptual Application Integration Architecture
BTA-0012	Ensure the Application Component Model diagram and/or accompanying chart includes each layer, module, or component name (using Contractor-neutral business names), the functions performed, application status, the implementation phase, and other descriptive or strategic information as needed.	MUST	9.4.2—Application Component Model

Req. ID	Requirement Statement	Tier	Standard Section
BTA-0013	Provide additional diagrams or charts with further detail where the Solution Architecture Model does not include sufficient detail.	MUST	9.4.3—Solution Architecture Model
BTA-0014	Ensure the integration overview artifacts (Conceptual Application Integration Architecture, Solution Architecture Model, Deployment Model, and Network Topology) are aligned.	MUST	9.5.1—Integration Overviews
BTA-0015	Ensure all integrations and interfaces shown in architectural models include, at minimum: interface name, unique identifier, source and target systems, data format(s), protocol/transport mechanism, direction of data flow, and triggering event or frequency.	SHOULD	9.5.1—Integration Overviews
BTA-0016	Develop a Deployment Model (ArchiMate 3.1) depicting hardware and software nodes arranged into security zones and linked by communication paths, showing how applications are integrated securely.	SHOULD	9.5.2—Deployment Model
BTA-0017	Ensure the physical Network Topology Model, once solution design is complete, specifies actual (rather than representative) clouds and network equipment, including IP addresses and other details.	MUST	9.5.3—Network Topology Model
BTA-0018	Use commonly recognized symbols in the Network Topology Model to represent secured-area boundaries (clouds, VPCs, subnets, data centers), infrastructure nodes, network hardware and software nodes, network connections, and the Internet.	SHOULD	9.5.3—Network Topology Model
BTA-0019	Technology and security details such as IP addresses, subnets, or port numbers MAY be shown on the Network Topology diagram.	MAY	9.5.3—Network Topology Model
BTA-0020	Establish and maintain Architecture Decision Records (ADRs) that document significant solution and technical architecture decisions, including the decision, its context and rationale, the alternatives considered, and the resulting consequences; reference the applicable ADRs from the RTM and design artifacts.	MUST	9.7 Architecture Governance and Compliance

Req. ID	Requirement Statement	Tier	Standard Section
BTA-0021	Conduct architecture compliance reviews at OHIP-defined milestones to verify that the implemented solution conforms to the architecture artifacts approved under Section 9.0, and document and remediate any deviations through OHIP's enterprise architecture governance and change control processes.	MUST	9.7 Architecture Governance and Compliance
BTA-0022	Demonstrate alignment of the solution architecture with the OHIP Business Reference Architecture and the MITA framework, documenting and obtaining OHIP approval for any justified departures.	MUST	9.7 Architecture Governance and Compliance
BTA-0023	<p>Maintain traceability from solution architecture components to the requirements they satisfy (through the RTM) and to the deployed solution, so that each significant architecture element can be traced to its driving requirement and its implementation.</p> <ul style="list-style-type: none"> • Submit for EARB review, using the ADR/SDD Intake Form: (a) the solution architecture artifacts required for the approval gate of Section 8.4.1, as an SDD submission; and (b) significant architecture decisions arising after that approval, as ADR submissions. A decision is significant if it introduces, replaces, or retires a major technology or platform component; changes integration patterns, data exchanges, or data contracts with other systems; affects security posture, data classification handling, or the processing of member data; materially changes cost or total cost of ownership; impacts enterprise shared services; or deviates from an approved enterprise standard. Decisions below this threshold are recorded in the project's ADR log without EARB submission. • Record the EARB disposition, any conditions, and the decision rationale in the corresponding ADR, and track conditions to closure. Where a proposed design deviates from a MUST requirement of any OHIP Domain Document, an exception noted on the intake form does not substitute for the Standards Deviation Request process: the SDR MUST be submitted under Section 1.7.1 and its identifier cross-referenced on the intake form. 	MUST	9.7 Architecture Governance and Compliance

Req. ID	Requirement Statement	Tier	Standard Section
BTA-0024	Submit for EARB review, using the ADR/SDD Intake Form: (a) the solution architecture artifacts required for the Section 8.4.1 approval gate, as an SDD submission; and (b) significant architecture decisions arising after that approval, as ADR submissions, per the significance criteria of Section 9.7. Decisions below the threshold are recorded in the project ADR log without EARB submission.	MUST	9.7 Architecture Governance and Compliance
BTA-0025	Record the EARB disposition, conditions, and rationale in the corresponding ADR and track conditions to closure. Where a proposed design deviates from a MUST requirement of any OHIP Domain Document, an exception on the intake form does not substitute for the SDR process: the SDR MUST be submitted under Section 1.7.1 and cross-referenced on the intake form.	MUST	9.7 Architecture Governance and Compliance

Appendix C: Release Naming Guidelines

The following guidelines define the naming convention for Enterprise and Contractor releases. These conventions apply to all versioned software deployments under Section 7.0.

Enterprise Release Naming

To provide an enterprise-level release tracking/reporting experience, OHIP will be utilizing the following top-level naming convention for Project and/or Application software releases, with individual Projects and/or Applications adhering to the naming convention set out in the section below.

Enterprise Release Version utilizes the following structure, with each portion separated by an underscore and Release number using a “.” to designate type and version within-type.

- **Enterprise Identifier:** A static identifier. Anything starting with “OHIP_” is an Enterprise Release version.
- **Year:** Four (4)-digit year, 2025
- **Release #:** Release number specific to the Application. For example, OHIP is currently preparing release twenty-two (22) (Major, Minor, Minor)
- **Numeric Type Identifier** broken out as follows
 - o **Major:** ##.0
 - o **Minor:** ##.1 and .2

Examples:

- **Example One (1):** OHIP_2025_1.0 (MAJOR) being the first of four (4) Major Enterprise releases in 2025
- **Example Two (2):** OHIP_2025_1.1 (MINOR) being the first of the Minor Enterprise releases in 2025
- **Example Three (3):** OHIP_2026_4.0 (MAJOR) being the fourth Major Enterprise release for 2026

Contractor Release Naming

To help ensure consistent, easily readable release versioning, OHIP requires the following convention for project and/or application software releases.

Software versions **MUST** utilize the following structure, with each portion separated by an underscore and Release number and Type using a “.” to designate type and version within-type.

- **Year:** Four (4)-digit year, 2025
- **Application:** Application abbreviation i.e. MECM, APCD, etc.
- **Release #:** Internal release number specific to the Application. These numbers are sequential.
- **Type identifier:** (Major, Minor, Expedited/HotFix) broken out as follows
 - o **Major:** ##.0

- o **Minor:** ##.1 and .2
- o **Expedited Release:** ##.#.1
- o **HotFix:** ##.##.1

Examples:

- **Example One (1):** 2025_MECM_21.0.0.0 being MECM release 21, in 2025 with no Minor, Expedited or HotFix releases attached
- **Example Two (2):** 2025_MECM_21.1.0.0 being Minor release 1 under MECM release 21, in 2025
- **Example Three (3):** 2025_MECM_21.1.1.0 being Expedited release 1 under MECM Minor release 1 under release 21, in 2025
- **Example Four (4):** 2025_MECM_21.0.0.1 being the first HotFix under MECM release 21, in 2025 with no Minor or Expedited releases attached

Appendix D: Requirements Management Guidance

Table 40 – Design Stage Activities (Part 1)

Activities	Responsibilities	Project Documents/Results
Review and approve initial requirements	<ul style="list-style-type: none"> OHIP staff review, edit, and approve the list of initial requirements. 	<ul style="list-style-type: none"> Initial requirements
Set-up requirements repository	<ul style="list-style-type: none"> OHIP EA sets up the repository and provides access to the necessary Contractor and OHIP EA support resources. OHIP EA, with the business owners, loads the requirements. OHIP EA confirms that the project Requirements Dashboard(s) are set up. 	<ul style="list-style-type: none"> OHIP EA contacts identified Initial requirements loaded in the requirements tool Update to the Project Requirements Dashboard
Training for Requirements Management Tools and documentation	<ul style="list-style-type: none"> OHIP EA provides a holistic overview and guidance of requirements management and associated activities, deliverables, tools, and artifacts. OHIP coordinates tool overview. 	<ul style="list-style-type: none"> The Contractor confirmation of requirements overview and training.
CMS requirements alignment	<ul style="list-style-type: none"> The Contractor reviews and updates the requirement attributes for the Business Area/Functional Area and the Business Process Titles (as applicable). The Contractor reviews and updates the requirement attributes with the applicable CMS Certification Outcome, Condition, and Metrics. Business Owner(s) and OHIP EA confirms updates to the RTM. 	<ul style="list-style-type: none"> The following updated requirements attributes in the requirements repository: Business Area/Business Process Title, CMS Required, Certification Conditions/Outcomes ID, Certification Metrics ID Requirements are updated with the CMS attributes

Activities	Responsibilities	Project Documents/Results
Develop Requirements Management Plan	<ul style="list-style-type: none"> The Contractor drafts the Requirements Management Plan deliverable, explaining their plan to comply with this standard. The Contractor coordinates with OHIP to submit the Requirements Management Plan deliverable for review and approval. 	<ul style="list-style-type: none"> Approved Requirements Management Plan Deliverable

Table 41 – Design Stage Activities (Part 2)

Activities	Responsibilities	Project Documents/Results
Requirements elaboration	<ul style="list-style-type: none"> The Contractor's Project Management Office and OHIP's Project Management Office coordinates requirements elaboration appointments with OHIP. The Contractor MUST elaborate the detailed Solution Requirements to a testable and verifiable level or identifies the acceptance criteria. 	<ul style="list-style-type: none"> Elaborated Solution Requirements: functional and non-functional requirements are in the repository. Applicable fields are updated in the repository. Acceptance criteria for non-testable requirements and traced verification methods for applicable requirements.
Requirements prioritization	<ul style="list-style-type: none"> The Contractor MUST prioritize the Solution Requirements in collaboration with the Business Owner(s). 	<ul style="list-style-type: none"> Updated status field for applicable requirements in the repository.
Validate certification requirements	<ul style="list-style-type: none"> The Contractor MUST collaborate with the OHIP Certification team to ensure that the correct CMS Outcomes and Metrics IDs are traced to applicable requirements. 	<ul style="list-style-type: none"> Updated CMS Outcomes and Metric ID fields for applicable requirements in the repository.
Develop and finalize the RTM deliverable	<ul style="list-style-type: none"> The Contractor MUST collaborate with OHIP EA to export the requirements to the repository in an RTM format. The Contractor MUST coordinate with OHIP to submit the first iteration of the RTM deliverable for review and approval. 	<ul style="list-style-type: none"> OHIP Approved RTM deliverable. Solution Requirements Baseline. Repository Requirements Dashboard updated.

Table 42 – Design Stage Activities (Part 3)

Activities	Responsibilities	Project Documents/Results
Design artifact development and tracing	<ul style="list-style-type: none"> The Contractor MUST develop the deliverables and design artifacts to include traceability to Solution Requirements, where applicable. The Contractor MUST input and manage updates to the repository requirements repository with traceability of requirements to applicable deliverables and artifacts (e.g., Epics, Use Cases, User Stories, Business Processes, and Reports, etc.). 	<ul style="list-style-type: none"> Applicable fields are updated in the repository. Deliverables and Artifacts with traced Solution Requirements.
Detailed design review	<ul style="list-style-type: none"> The Contractor MUST review the design's requirements coverage with OHIP and OHIP EA. Business Stakeholders and OHIP EA validates requirement coverage. The Contractor MUST update requirements in the repository where applicable from the walkthrough feedback. 	<ul style="list-style-type: none"> Applicable fields are updated in the repository. Updated design deliverables and artifacts.
Validate certification requirements	<ul style="list-style-type: none"> The Contractor MUST collaborate with the OHIP Certification team to ensure that the correct CMS Outcomes and Metrics IDs are traced to the applicable requirements. 	<ul style="list-style-type: none"> Updated CMS Outcomes and Metric ID fields for applicable Solution Requirements in the repository.
RTM Update	<ul style="list-style-type: none"> The Contractor MUST collaborate with the OHIP EA to update the RTM. The Contractor MUST coordinate with OHIP EA to submit the updated RTM for review and approval. 	<ul style="list-style-type: none"> Approved RTM Update with design updates. Repository Requirements Dashboard updated.

Table 43 – Development Phase Activities (Part 1)

Activities	Responsibilities	Project Documents / Results
Develop artifact traceability	<ul style="list-style-type: none"> The Contractor MUST develop phase deliverables and artifacts and includes the traceability of the solution requirements where applicable. The Contractor MUST update the requirements repository with traceability of requirements to applicable deliverables and artifacts (e.g., Module Information and Test Case ID, etc.). 	<ul style="list-style-type: none"> Applicable fields are updated in the repository. Deliverables and Artifacts with traced Solution Requirements. Test plan updated.
Validate certification requirements	<ul style="list-style-type: none"> The Contractor MUST collaborate with the OHIP Certification Management team to ensure correct CMS Outcomes and Metrics IDs are traced to applicable requirements. 	<ul style="list-style-type: none"> Updated CMS Outcomes and Metric ID fields for applicable requirements in the repository.
RTM Update	<ul style="list-style-type: none"> The Contractor MUST submit the Deliverable for Review and Approval. 	<ul style="list-style-type: none"> Approved RTM with Development phase updates. *SaaS/Cloud solutions: Must provide high-level module traceability to the solution requirements. Repository Requirements Dashboard updated.

Table 44 – Development Stage Activities (Part 2)

Activities	Responsibilities	Project Documents / Results
Solution Requirements traceability	<ul style="list-style-type: none"> The Contractor MUST link each Solution Requirement to a test case or identified verification method in the RTM. The repository MUST link the defects to the Test Cases from each Test Run. 	<ul style="list-style-type: none"> Deliverables and Artifacts with traced Solution Requirements. Applicable fields are updated in the repository with Defects and Issues from the repository and repository.
Requirements Release traceability	<ul style="list-style-type: none"> The Contractor MUST provide Release information for each requirement in the RTM. 	<ul style="list-style-type: none"> Applicable fields are updated in the repository with Release information for each requirement.

Activities	Responsibilities	Project Documents / Results
Non-functional requirements (NFR) testing	<ul style="list-style-type: none"> The Contractor MUST work with the OHIP testing team to confirm non-functional requirement testability. 	<ul style="list-style-type: none"> Applicable fields are updated in the repository with non-functional requirement testing information.
Testing Types	<ul style="list-style-type: none"> The Contractor MUST work with the OHIP Testing team to demonstrate adequate Testing type. The Contractor MUST provide the resolution plan for outstanding requirements concerns. 	<ul style="list-style-type: none"> Applicable fields are updated in the repository with a 'verified' status for each applicable requirement. Resolution plan for requirements concerns.
Validate certification requirements	<ul style="list-style-type: none"> The Contractor MUST collaborate with OHIP staff responsible for CMS certification to ensure correct CMS Outcomes and Metrics IDs are traced to applicable requirements. 	<ul style="list-style-type: none"> Applicable fields are updated in the repository with updated CMS Outcomes and Metrics Identification.
RTM Update	<ul style="list-style-type: none"> The Contractor MUST submit the Deliverable for Review and Approval. 	<ul style="list-style-type: none"> Approved RTM with Test phase updates. Repository Requirements Dashboard updated.

Table 45 – Implement Phase Activities

Activities	Responsibilities	Project Documents / Results
Validate test results	<ul style="list-style-type: none"> The Contractor MUST review, update, and confirm traceability between test cases and requirements and validates the testing status for each requirement in the RTM. 	<ul style="list-style-type: none"> Applicable fields updated in the repository with 'verified,' 'passed,' or 'ready to implement' status applied to each requirement.
RTM walkthrough	<ul style="list-style-type: none"> The Contractor MUST facilitate a walkthrough meeting for the RTM to indicate full traceability, requirements coverage, accuracy, and completeness. The Contractor MUST update the RTM with each requirement's applicable implementation or baseline status. 	<ul style="list-style-type: none"> Applicable fields updated in the repository with 'implemented' or 'baselined' status applied to each requirement.

Activities	Responsibilities	Project Documents / Results
Validate certification requirements	<ul style="list-style-type: none"> The Contractor MUST collaborate with the OHIP staff involved in CMS certification to ensure that the correct CMS Outcomes and Metrics IDs are traced to applicable requirements. 	<ul style="list-style-type: none"> Applicable fields are updated in the repository with updated CMS Outcomes and Metrics Identification.
RTM Update	<ul style="list-style-type: none"> The Contractor MUST submit the Deliverable for Review and Approval. 	<ul style="list-style-type: none"> Approved RTM Update. Repository Requirements Dashboard updated.

Appendix E: MITA Service Definition Packages

MITA prescribes as part of the Business Service Definition Package and Technical Service Definition Package in Part III Chapter's 3 and 4.⁴⁹

The following is a recap of the MITA Service Definition Package that **MUST** be accounted for when documenting Business Services in OHIP's Enterprise Architecture Repository:

Business Service Parts

- **Service Name:** The title of the service.
- **Business Service Definition Package (BSDP):** Defined set of metadata describing the service.
- **Service Contract:** Describes the expected behavior of the interface as well as the security and privacy constraints on the service. The following are examples of interface behavior patterns:
 - o One-way only receives or outputs data (e.g., report generator).
 - o Involved parties negotiate a specific Service Level Agreement (SLA).
 - o Two-way receives and sends data. Two-way traffic has two (2) other attributes:
 1. Initiator defines who initiates the interface, either the service (e.g., request for information) or the outside client (e.g., inquiry).
 2. Processing characteristic defines the relationship between the input and the output:
 - **Point-of-Sale (POS) transaction:** A real-time transaction (e.g., a pharmacy POS) that features a very constrained response time and high reliability.
 - **Online transaction:** An inquiry on a provider that features a more relaxed response time while still allowing for conversation-type human interaction.
 - **Batch:** Typical batch processing constraints.
 - **Asynchronous:** No constraints on processing times but response and coordinating data are required.
- **Purpose:** Developed from the business process definition and associated BCM entry.
- **Business Logic:** Describes what is occurring within the business service. It documents the underlying logic, functionality, and capability provided by the service. Initially, the business logic is free-form text or template driven (e.g., MITA Business Process Template), that an application developer will define and code in order to incorporate into a business process orchestration step and possibly integrate to a rules engine.
- **Constraints:** List of any constraints of the service. The MITA business process triggers, results, and the MITA Logical Data Model (LDM) provide a formal interface definition. It also documents

⁴⁹ Refer to [MEDICAID CHIP Part III Zip File](#) for more guidance.

the interfaces and operations used by the service. Initially, the MITA Framework provides an informal textual description (i.e., template) of the services interface. As groups populate MITA Services, they can then define and develop interfaces using WSDL interface descriptions.

- **Use Case:** Documents the main success path and critical failure conditions of the service. Initially, the TA writes a use case in free-form text and then models it with OMG Unified Modeling Language (UML) prior to the development of the business services.
- **Solution Set:** Maps the specific detail developed to execute a service.
- **Structure Diagram:** Graphically depicts the business logic performed by the service and inter-connects to the solution sets.
- **Performance Standards:** Define the anticipated performance metrics of the service so all stakeholders measure the same criteria in the same manner.
- **Test Scenarios and Test Cases:** Documentation to validate compliance to the service contract.
- **Map to MITA Data Models:** A trace of data used by the service (e.g., data in motion and shared business data only) to the MITA Conceptual Data Model (CDM) and LDM. This is an enhancement of the mapping done for the business service. Both American National Standards Institute (ANSI) Accredited Standards Committee (ASC) X12 positional transactions and W3C XML Schema are examples of defined data in motion.

Technical Service Parts

- **Service Name:** The name of the service component invoked by the Technical Service Definition Package (TSDP) that is a MITA-defined set of metadata describing the service. The service contract information describes the expected behavior of the interface (e.g., whether the interface is real-time or an online interface), as well as the security and privacy constraints on the service (i.e., requires strong encryption). The following are examples of interface behavior patterns:
 - o A one-way interface only receives or outputs data (e.g., report generator).
 - o A two-way interface receives and sends data. Two-way traffic has two other attributes: (1) initiator defines who initiates the interface, either the service (e.g., request for information) or the outside client (e.g., inquiry), and (2) processing characteristic defines the relationship between the input and the output:
 - **Point-of-sale (POS) Transaction:** A real-time transaction (e.g., a pharmacy POS) that features very constrained response time and high reliability.
 - **Online Transaction:** An inquiry on a provider that features more relaxed response time while still allowing for conversation-type human interaction.
 - **Batch:** Typical batch processing constraints.
 - **Asynchronous:** No constraints on processing times but require response and data coordination.
- **Purpose:** The technical area's principles and TCM depict the purpose.
- **Business Logic:** Initially, the business logic is free-form text or template driven (e.g., MITA Business Process Template), application developers will eventually define and code business logic in order to incorporate into business process orchestration steps and standardized business rule definitions. States **MUST** ensure the use of business rules engines to separate business rules from core programming and **MUST** provide information about the change control process

that will manage development and implementation of business rules. States **MUST** be able to accommodate changes to business rules on a regularized schedule and on an emergency basis.

- **Constraints:** Lists constraints of the service, e.g., loose authentication control.
- **Formal Interface Definition:** Defines the services triggers and results. It also documents the interfaces and operations the service uses while using Web Service Definition Language (WSDL). Initially, the MITA Framework provides an informal textual description (i.e., template) of the services interface. As stakeholders develop the MITA services, they also develop the WSDL interface descriptions.
- **Use Case:** The architect or designer writes the use case in free-form text and supports the text with Object Management Group (OMG) Unified Modeling Language (UML) models prior to the development of the business services.
- **Solution Set:** Maps the solution set(s) currently available to execute this service. Solution sets are previously defined groupings of technical services that are operational.
- **Structure and Activity Diagrams:** Graphically depicts the business logic performed by the service and interconnects the solution sets. The architect or designer models the diagram in OMG Business Process Model and Notation (BPMN). Using a modeling tool based on UML notation produces an UML diagram as part of the set.
- **Performance Standards:** Defines the anticipated performance standards of the service so all stakeholders measure the same metrics in the same way.
- **Test Scenarios and Test Cases:** Validates documentation of compliance to the service contract.
- **Map to MITA Data Models:** Maps a trace of data used by the service (e.g., data in motion and shared business data only) to the MITA Conceptual Data Model (CDM) and Logical Data Model (LDM). This is an enhancement of the mapping done for the business service. The XML schema defines the data in motion. American National Standards Institute (ANSI) Accredited Standards Committee (ASC) X12 positional transactions and World Wide Web Consortium (W3C) XML Schemas are examples of defined data in motion.

Appendix F: SMART, INVEST, and Definition of Done (DoD) Criteria

The following criteria apply to all user story development activities under [Section 8.0](#). The Contractor **MUST** apply SMART, INVEST, and Definition of Done (DoD) criteria to ensure user stories are clear, actionable, and aligned with project goals to maintain quality, reduce ambiguity, and support predictable delivery in Agile environments. Refer to the criteria for each below:

SMART Criteria Requirements

The Contractor **MUST** ensure each user story meets the following SMART requirements:

- **Specific:** Clearly defines the intended outcome.
- **Measurable:** Includes acceptance criteria or other measurable indicators of success.
- **Achievable:** Feasible for completion within a single iteration.
- **Relevant:** Provides business or user value aligned with project needs.
- **Time-bound:** Deliverable within the established iteration or Sprint timeframe.

INVEST Criteria Requirements

The Contractor **MUST** ensure each user story meets the following INVEST requirements:

- **Independent:** Self-contained and minimally dependent on other stories.
- **Negotiable:** Flexible and open to refinement during team collaboration.
- **Valuable:** Provides clear value to users or the business.
- **Estimable:** Defined clearly enough to enable accurate effort estimation.
- **Small:** Sized appropriately for completion within a single iteration.
- **Testable:** Includes acceptance criteria sufficient for verification and validation activities.

Definition of Done (DoD) Requirements

The Contractor **MUST** meet all elements of the project's Definition of Done (DoD) before a user story can be accepted. At a minimum, the DoD **MUST** include:

- **Code complete:** All development tasks are finished, reviewed, and approved.
- **Test complete:** All required testing (e.g., unit, integration, and acceptance) is executed and passed.
- **Documentation updated:** All required user-facing and internal documentation is created or revised.
- **Deployment ready:** The feature is packaged for deployment with applicable release notes.
- **Quality assurance:** The feature meets established performance, usability, and quality standards.

Appendix G: Acronyms

The following acronyms are used throughout the OHIP Solution Delivery Domain Document.

ADR: Architecture Decision Record
API: Application Programming Interface
ARIA: Accessible Rich Internet Applications
BDD: Behavior-Driven Development
BRC: Business Rules Catalog
BC/DR: Business Continuity/Disaster Recovery
CAP: Corrective Action Plan
CI/CD: Continuous Integration/Continuous Deployment
CMS: Centers for Medicare & Medicaid Services
COTS: Commercial Off-The-Shelf
CVE: Common Vulnerabilities and Exposures
DAST: Dynamic Application Security Testing
DDI: Design, Development, and Implementation
DoD: Definition of Done (in Agile context)
DOH: Department of Health (New York State)
E&E: Eligibility and Enrollment
EARB: Enterprise Architecture Review Board
EDW: Enterprise Data Warehouse
eMedNY: Electronic Medicaid of New York
FFP: Federal Financial Participation
FHIR: Fast Healthcare Interoperability Resources
FTI: Federal Tax Information
IaC: Infrastructure as Code
IMS: Integrated Master Schedule
INVEST: Independent, Negotiable, Valuable, Estimable, Small, Testable (Agile user story criteria)
JAWS: Job Access With Speech (screen reader)
KPI: Key Performance Indicator
MECM: Medicaid Eligibility and Client Management System
MES: Medicaid Enterprise System
MITA: Medicaid Information Technology Architecture
M&O: Maintenance and Operations
NFR: Non-Functional Requirement
NVDA: NonVisual Desktop Access (screen reader)
OCM: Organizational Change Management
OHIP: Office of Health Insurance Programs
ORR: Operational Readiness Review
OWASP: Open Web Application Security Project
PBI: Product Backlog Item
PHI: Protected Health Information
PII: Personally Identifiable Information
PMO: Project Management Office
RCA: Root Cause Analysis
RFP: Request for Proposal
RTM: Requirements Traceability Matrix
SAST: Static Application Security Testing
SCA: Software Composition Analysis
SDI: System and Data Integrator
SDLC: Solution Development Life Cycle

SDD: Solution Design Document
SDR: Standards Deviation Request
SIT: System Integration Testing
SLA: Service Level Agreement
SMART: Specific, Measurable, Achievable, Relevant, Time-bound (Agile user story criteria)
SMC: Streamlined Modular Certification
SOR: System of Record
SOA: Service-Oriented Architecture
SUS: System Usability Scale
TDD: Test-Driven Development
TIMP: Transition-In Management Plan
UAT: User Acceptance Testing
UML: Unified Modeling Language
VPAT: Voluntary Product Accessibility Template
WCAG: Web Content Accessibility Guidelines