Link IT&TS Orchestration Users' Group shared folder

Software Currency Strategy for EMBL-EBI

1. Executive Summary

In the rapidly evolving technological landscape, maintaining "software currency" is critically important for the operational health, security, and overall efficiency of an organisation. Software currency specifically refers to the diligent practice of ensuring that all deployed software, encompassing operating systems, applications, and core infrastructure components, is consistently updated to the latest stable versions, inclusive of essential security patches and feature enhancements.

This comprehensive strategy outlines a proactive methodology for managing software updates and upgrades throughout EMBL-EBI. Its primary aim is to effectively mitigate various risks, bolster system performance, fortify the security posture, ensure regulatory compliance, and ultimately maximise the return on investment in software assets. Adherence to these established principles will foster a more stable, secure, and productive information technology environment within EMBL-EBI.

2. Goals and Objectives

The fundamental objectives of this Software Currency Strategy are delineated as follows:

- Enhance Security: To minimise organisational susceptibility to cyber threats through the prompt application of security patches and the systematic deprecation of unsupported software.
- Improve Performance & Stability: To leverage performance enhancements, defect resolutions, and stability improvements inherent in newer software versions, thereby reducing system outages and enhancing user experience.
- Ensure Compatibility: To maintain seamless interoperability between diverse software systems and hardware components, preventing costly integration challenges.
- Foster Innovation & Productivity: To provide personnel with access to cutting-edge features and functionalities that can significantly boost productivity and enable novel business capabilities.
- Reduce Technical Debt: To proactively address the accumulation of outdated software, thereby preventing an increase in maintenance costs and operational complexity.
- Ensure Compliance: To meet all pertinent regulatory and industry compliance requirements by utilising actively supported and appropriately patched software.
- Optimise Costs: To achieve cost efficiencies by adopting a planned upgrade approach, which will increase vendor ability to support and reduce the need for emergency fixes to be applied due to unforeseen incidents and issues.

3. Scope

This strategy is applicable to all software deployed and actively utilised within EMBL-EBI's information technology environment, both within ITS and wider. This encompasses, but is not restricted to, the following categories:

- Operating Systems: Server environments (e.g., Windows Server, various Linux distributions) and Endpoint devices (e.g., Windows, macOS, Linux).
- Productivity Applications: Comprehensive suites such as Microsoft Office, Google Workspace, Adobe Creative Cloud, and similar professional tools.
- Business-Specific Applications: Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM) platforms, Human Resources Information Systems (HRIS), and Financial Management Systems.
- Development Tools & Frameworks: Integrated Development Environments (IDEs), programming languages, software libraries, and database management systems.
- Infrastructure Software: Network device firmware, virtualisation platforms, data backup solutions, and system monitoring utilities.
- Security Software: Antivirus solutions, Endpoint Detection and Response (EDR) systems, network firewalls, and Identity and Access Management (IAM) tools.
- Custom Developed Applications: Internally developed applications and their underlying software dependencies.

4. Guiding Principles

The execution of this Software Currency Strategy will be governed by the following core principles:

- Proactive Management: Adherence to a proactive "plan-and-prevent" methodology.
- Risk-Based Prioritisation: Prioritisation of updates and in-scope software will be determined by an ongoing assessment of security vulnerabilities, the criticality of software to business operations, the architectural risk of the component, and the potential impact of falling behind.
- Automation Emphasis: Automation tools will be used to streamline update processes, minimise manual effort, reduce the potential for human error, reduce the need for direct access and increase standardisation and consistency of the estate.
- Central Control: Implementation of centralised management platforms for efficient software distribution, patch application, and comprehensive inventory management.
- Transparent Communication: Ensuring clear, concise, and timely communication with all relevant stakeholders and end-users regarding planned updates, associated timelines, and potential operational impacts.

- Rigorous Testing and Validation: All software updates and upgrades are subject to rigorous testing and validation procedures prior to their broad deployment.
- Phased Rollout: Updates will be deployed in a structured, phased manner (e.g., pilot groups, departmental rollouts, then broader deployment) to minimise disruption and facilitate early identification of issues.
- Continuous Improvement: Ongoing review and refinement of the strategy, incorporating lessons learned from previous deployments and adapting to evolving technological landscapes.

5. Strategic Pillars/Components

5.1. Software Inventory and Assessment

- Maintain Accurate Inventory: Utilise robust IT Asset Management (ITAM) tools to maintain a comprehensive and current inventory of all deployed software, including detailed version numbers, associated licenses, and deployment locations.
- Dependency Mapping: Document critical software interdependencies to accurately assess potential cascading effects of updates or changes.
- Lifecycle Tracking: Systematically track the lifecycle dates of all critical software applications, including release dates and crucial end-of-life (EoL) or end-of-support dates.
- Vulnerability Scanning: Conduct regular and automated vulnerability scans across all systems to identify known security weaknesses.

5.2. Patch Management (Security & Bug Fixes)

- Established Patch Cycles: Implement and adhere to defined monthly or quarterly patch cycles for operating systems and mission-critical applications.
- Expedited Critical Patching: Develop and maintain an expedited process for the rapid deployment of critical security patches (e.g., those addressing zero-day vulnerabilities) outside of routine cycles.
- Automated Patch Deployment: Leverage centralised tools such as WSUS, SCCM, Intune, or similar solutions to automate patch deployment wherever technically feasible.
- Endpoint Management: Implement robust endpoint management solutions to ensure consistent and timely patch application across all end-user devices.

5.3. Minor Version Upgrades (Feature & Performance)

- Scheduled Reviews: Conduct periodic (e.g., quarterly or semi-annual) reviews of minor version releases for key applications.
- Feature Evaluation: Systematically assess newly introduced features for their potential business value and their impact on existing user workflows.
- Performance Benefits: Evaluate minor versions for significant performance enhancements or the resolution of persistent software defects.

- Controlled Rollout: Deploy minor version upgrades in a controlled manner, analogous to patch management, but with augmented communication to users regarding new functionalities.

5.4. Major Version Upgrades (Significant Changes)

- Strategic Planning: Initiate planning for major upgrades well in advance, typically 6-12 months prior to the projected deployment, accounting for budgetary allocations, resource availability, and potential operational disruption.
- Business Case Development: For all substantial upgrades, develop a clear and detailed business case outlining the anticipated benefits, comprehensive costs, and inherent risks.
- Extensive Compatibility Testing: Conduct exhaustive compatibility testing with all existing systems, integrations, and custom applications to ensure seamless operation post-upgrade.
- User Acceptance Testing (UAT): Engage key business users in the User Acceptance Testing phase to validate that the upgraded software effectively meets operational requirements.
- Training & Documentation: Provide comprehensive training programs and updated documentation for all staff members impacted by major software changes.
- Phased Deployment: Employ a strategic phased deployment approach, potentially commencing with pilot user groups, followed by specific departments, before proceeding to a broader organisational rollout.

5.5. Lifecycle and End-of-Life (EoL) Management

- Proactive Identification: Formal EMBL-EBI roadmaps identifying software applications approaching their End-of-Life (EoL) status well in advance (typically 12-24 months) to allow ample time for strategic planning of migration or replacement.
- Risk Assessment: Conduct thorough risk assessments associated with operating unsupported software, encompassing security vulnerabilities, lack of vendor support, and potential compliance infractions.
- Migration/Replacement Plan: Develop detailed, actionable plans for migrating to newer software versions or replacing EoL software with suitable alternative solutions.
- Budget Allocation: Ensure that appropriate budgetary provisions are allocated in line with risk assessment and prioritisation for EoL migrations or replacement initiatives.

5.6. Licensing and Compliance

- Centralised License Management: Implement and maintain a centralised system for tracking all software licenses to ensure strict compliance with

licensing agreements and optimise procurement by avoiding both over-licensing and under-licensing.
- Audit Readiness: Maintain meticulously accurate license records and deployment information to ensure readiness for potential vendor audits.
- Compliance Checks: Integrate compliance validation into the update process to confirm that new software versions align with relevant industry standards and regulatory mandates.

5.7. Testing and Validation

- Dedicated Test Environments: Maintain dedicated test environments that accurately replicate the production environment as closely as possible.
- Regression Testing: Conduct thorough regression testing to confirm that software updates do not introduce new defects or compromise existing functionalities.
- Performance Testing: Perform comprehensive performance testing for major upgrades to ensure no degradation in system responsiveness or scalability.
- Security Testing: Incorporate robust security validation into the testing process, particularly for security patches and major version upgrades.

5.10. Essential Tools and Automation Platforms *(This section still to be tailored)*

The efficient management of software currency will be facilitated by leveraging appropriate tools and automation platforms, including but not limited to:

- Patch Management Tools: Microsoft SCCM/Intune, Tanium, Ivanti, BigFix.
- Software Distribution Tools: Group Policy, SCCM, Intune, JAMF.
- IT Asset Management (ITAM) Software: ServiceNow, Flexera, Snow Software.
- Vulnerability Scanners: Qualys, Nessus, OpenVAS.
- Configuration Management Tools: Ansible, Foreman, Puppet, Chef (primarily for server environments).
- Ticketing/Change Management Systems: ServiceNow,

5.11. Governance and Roles *(This section still to be tailored)*

Effective implementation requires clearly defined roles and responsibilities:

- IT Leadership: Responsible for overall strategic oversight, resource allocation, and policy enforcement concerning software currency.
- Software Currency Committee: A cross-functional committee comprising representatives from IT Operations, Security, Application Management, and relevant Business Units. This committee is tasked with reviewing upcoming updates, prioritising deployments, and approving schedules.
- System Administrators/Engineers: Responsible for the technical implementation, testing, and deployment of software updates.

- Application Owners: Accountable for understanding the impact of updates on their specific applications and actively participating in User Acceptance Testing.
- Security Team: Responsible for identifying vulnerabilities, prioritising the deployment of security patches, and validating the security aspects of all updates.
- Service Desk: Serves as the primary point of contact for end-user issues arising post-update.

6. Metrics and Reporting

The success of this strategy will be rigorously measured and reported through a set of key performance indicators (KPIs), which include:

- Patch Compliance Rate: The percentage of systems successfully patched within defined Service Level Agreements (SLAs).
- Unsupported Software Ratio: The percentage of software operating on End-of-Life (EoL) versions.
- Vulnerability Count (Trend Analysis): A demonstrable reduction in critical and high-severity vulnerabilities over time.
- Downtime Attributable to Software Issues: A decrease in incidents directly resulting from outdated or unpatched software.
- User Satisfaction (Post-Upgrade Surveys): Feedback gathered through surveys regarding the impact and usability of newly deployed software versions.
- Successful Major Upgrades: The number of strategic major upgrade projects successfully completed within established parameters.

7. Challenges and Mitigation Strategies ***(This section still to be tailored)***

Addressing potential challenges is integral to the strategy's success:

- Legacy Systems:
  - Challenge: The inability to update certain systems due to critical dependencies or the cessation of vendor support.
  - Mitigation: Implement robust isolation measures (e.g., network segmentation), deploy compensating security controls (e.g., enhanced monitoring), and prioritise systematic replacement or modernisation.
- Budgetary Constraints:
  - Challenge: Insufficient financial resources for software licenses, necessary tools, or comprehensive staff training.
  - Mitigation: Develop compelling business cases to justify investments, meticulously prioritise critical updates, and explore viable open-source alternatives where appropriate.
- User Resistance to Change:

    - Challenge: Reluctance among staff to adopt new interfaces or adapt to modified functionalities.
    - Mitigation: Clearly articulate and emphasise the benefits of updates, provide comprehensive training programs, offer extensive support, and involve users in the early testing phases.
- Resource Availability:
    - Challenge: A shortage of skilled IT personnel or insufficient time allocated for managing updates.
    - Mitigation: Invest in advanced automation tools, implement cross-training initiatives for staff, consider engaging managed services for specific operational aspects, and strictly prioritise efforts based on risk.
- Testing Complexity:
    - Challenge: Difficulties in replicating complex production environments for thorough testing.
    - Mitigation: Invest in robust and representative test environments, automate testing procedures wherever feasible, and develop comprehensive rollback plans.

8. Conclusion

The establishment and sustained maintenance of a comprehensive software currency strategy represents an ongoing endeavour that necessitates unwavering commitment, adequate resource allocation, and a proactive organisational mindset. By embracing the principles and rigorously implementing the practices detailed within this document, EMBL-EBI will significantly enhance its cybersecurity posture, optimise operational efficiency, and strengthen its capacity to leverage technology for a distinct competitive advantage. This approach will ensure a stable, secure, and future-ready information technology environment for its 3000 staff members.