Secure Coding Practices Policy

InterSystems Product Management and Development follows a series of policies and procedures to ensure that products designed and developed by InterSystems are stable, secure, and reliable for intended customer use. While it is not our policy to share specific details of coding practices that we use to safeguard our products and to protect against security vulnerabilities, this document includes details on the governance processes surrounding our policies regarding the development of our products.

All members of our product teams are asked to confirm understanding of key policies on an annual basis, including our Secure Coding Policy.

This Policy highlights more specifics of our product design, development and quality processes as they pertain to maintaining and delivering secure products.

Development Processes

- **Development Methodology.** Development activities shall be carried out in accordance with a documented system development methodology that incorporates both design performance evaluations and security assessments.
- **Development Environments.** System development activities shall be performed in specialized development environments, isolated from any customer environment, and protected against disruption and disclosure of information.
- **Segregation of Duties.** Segregation of duties shall be in place for system development, including ensuring that system developers do not have access to any customer environment, unless in an emergency where such access would be protected with adequate controls such as break-glass procedures. Such activities in these circumstances shall be logged and subject to independent review by the Data Protection Officer.
- **System Design.** Information security requirements for the systems under development shall be included when designing the system.
- **Specification Requirements.** All systems requirements (including functional and technical specifications and information security requirements) shall be documented and agreed before detailed design commences, as applicable. Applicable documentation requirements for bug-fixes (as opposed to enhancements or structural changes) shall be limited to code change notes.
- **Quality Assurance.** Quality assurance of key security elements shall be performed during the development lifecycle.

- **System Build.** System build activities (including coding) shall be carried out in accordance with Good Industry Practice, performed by individuals with the relevant skills and provided with the relevant tools. System build activities shall be inspected to identify unauthorised modifications or changes which may compromise security controls. Any modifications must be documented in the core change management system associated with the relevant product processes.
- **Secure Coding Practices.** Secure coding practices for the design and development of our products shall be in-place for all product management, development, and implementation teams, including any sub-contractors, to include the definition and testing of security requirements. Such practices shall be fully documented and based on Good Industry Practice. Our Guidelines are provided below.
- **Source Code Security.** Access to program source code shall be restricted and strictly controlled to prevent the introduction of unauthorised functionality and to avoid unintentional changes.
- **Testing Process.** All performance- and security-related elements of the systems shall be tested at all stages of the system development lifecycle.
- **Production Data.** Production, or 'Live,' data will not be used within non-production environments without prior written approval and agreement of the controls to be implemented by the appropriate data owner in order to safeguard and to protect that production data. When production data is maintained and used in non-production environments, that data shall be secured to the same extent as the production environment.
- **Test Data.** Test data must be selected carefully, and protected and controlled. Processes must be in place to: (1) sanitise data used for testing to remove personally identifiable information or confidential information; (2) protect the integrity of production data; and (3) remove test data from the environment after testing completes.

Development Guidelines

Our developers work under and are mentored on coding standards to ensure security vulnerabilities are not introduced inadvertently. Code reviews are done as part of our standard process to ensure coding guidelines are followed and to ensure a high quality of code.

These guidelines include, but are not limited to, the following:

- Assuming that adversaries could have our source code, never store anything directly in source code that could be useful in an attack, such as a cryptographic key or password.
- Never check prototype or demo code into main source trees or branches accessible to main source trees. Prototype code may not yet have had a full security review and

demo code may skip some (security-related) best practices in the interest of highlighting a particular feature or function.

- Never use client input directly to execute a statement or allow for generation of a dynamic SQL statement. Input provided by the client should only be used as raw data in specific elements and never as executable code.
- Avoid use of public variables and ensure all variables are scoped within appropriate procedure blocks to prevent external manipulation of variables.
- Avoid specific Cache Object Script functions that would allow for direct code execution or access of system executables.
- When working with files from an application, avoid use of parameters for filenames and check security resources before manipulating files.
- Do not construct Dynamic SQL from user-supplied strings, avoiding the potential for SQL injection.
- Do not allow external APIs to ask the server to execute specific methods by use of parameters.
- Avoid URL parameter vulnerabilities (the series of mechanisms to avoid are not outlined in this document for security reasons)
- Fail securely – Ensure that no diagnostic information could inadvertently expose secure information or information that could be used to form an attack. Also, avoid failing from a more trusted to a less trusted method or procedure.
- Keep sensitive data in memory for only as long as needed and clear caches as appropriate. Mechanisms for immediately overwriting the memory are provided to developers.
- When submitting a code change, include testing instructions for our quality team that describe how to attempt to provoke a Denial of Access if this area allows for user invokable code and state the level of user access required.
- Utilize cryptography appropriately (guidelines not included in this document for security reasons) and utilize the standard cryptographic algorithms provided in our system APIs.
- Ensure all web services are secured and encrypt the username tokens.
- Clearly document the roles that an API or feature requires in order to minimize the chance that site developers and administrators over-privilege roles due to lack of understanding of discrete resource needs.

Quality Assurance Process

As part of our product qualification process we make use of a third party penetration testing tool to verify that our software is free from vulnerabilities such as (but not limited to):

- Cross-site scripting
- Cross-site request forgery
- SQL injection
- Unicode transformation

Additionally, the tool provides reporting against the OWASP Top Ten List (2017), including the three new security risks — XML external entities (XXE), insecure deserialization, and insufficient logging and monitoring. The tool is used for every release of InterSystems IRIS, Caché, Ensemble, HealthShare, and TrakCare

All potential security concerns discovered by the tool are logged as potential InterSystems product issues, which then follow our standard processes for triage, resolution and re-verification by Product Management, Development and QD, respectively.

In addition, all potential security-related product issues are reviewed by the Director of QD and the appropriate Product Manager for security as an additional safeguard to ensure proper and timely mitigation. This risk management is part of our Global Trust program overseen by our Data Protection Officer.

Vulnerability Response Management

InterSystems continuously evaluates and obtains timely information about potential technical vulnerabilities regarding its products and evaluates the exposure to those vulnerabilities, and InterSystems takes the appropriate measures to address the associated risks with those vulnerabilities.

InterSystems takes appropriate and timely action in response to the identification of potential technical vulnerabilities. InterSystems has in place the following requirements establishing an effective management process for technical vulnerabilities:

- Identify resources to address identified potential technical vulnerability
- Assess the risk of the vulnerability, including, but not limited to, initial CVSS scoring
- Define a timeline for internal teams to react to notifications of potentially relevant technical vulnerabilities
- Identify the associated risks and the actions to be taken
- Test and evaluate patches or updates before they being made available; or if no patch or update can be done, define other controls must be considered, such as:

    - Turning off services or capabilities related to the vulnerability;
    - Adapting or adding access controls, e.g. firewalls, at network borders;
    - Increased monitoring to detect actual attacks; and
    - Raising awareness of the vulnerability;
- Auditable documentation must be kept for all procedures;

InterSystems will ensure that no kits available (by download or media) from InterSystems will contain identified and defined vulnerabilities, although exceptions may be made, for example, for flaws in obsoleted products or features that a customer must continue to operate. Updates will be made to the latest maintenance releases of all products from the then current “minimum supported version” to the present.

Intersystems maintains for the product vulnerability an internal risk rating process along with our quality process that is used to provide input to the decision. While InterSystems does not provide the specific detail to customers or the public about how the risk rating is determined, in order to prevent an attacker from reengineering the vulnerability management process, we will provide the final CVSS scoring as part of any announcement concerning a product vulnerability, even though the CVSS scoring alone is not the sole factor for determining the risk rating. For example, InterSystems may decide to provide notice for a product vulnerability that scores low on CVSS, but is determined to have or to create other risks that extend beyond our current risk acceptance.