

OpenHIE Non-Functional Requirements - Draft

The following are recommended non-functional requirements for OpenHIE reference software.

- a. Well Documented: An OpenHIE reference system should include appropriate background, design, installation, configuration, and operational documentation to ensure it is easy to understand, maintain, and debug.
 - i. Source code should have comments so that developers do not need to look anywhere else to understand the code.
 - ii. Configuration files should have embedded comments explaining the different options.
 - iii. Installation, configuration, and operational activities should be described.
- b. Easy to implement for common use cases
- c. Built using open source tools and technology: The OpenHIE component should be built using widely available open-source technology (including development environments and languages).
- d. Open, easy access to source code: A standard version control system (e.g., GitHub) should be used to ensure that source code access is fast, easy to download, compile, and execute code.
- e. Standards-based: The software should use broadly adopted standards that enable interoperability among systems.
- f. Reliable and easy-to-use User Interface: Common identity management workflows must be supported by the user interface, including initial system configuration, and routine workflows.
- g. Minimal software library dependencies: An OpenHIE component should minimize dependencies on 3rd-party libraries.
- h. Minimal abstraction: A OHIE component should not have more layers of abstraction than necessary, and seek to minimize abstraction that confounds design.
 - i. Easy Initiation: When properly installed and configured, administrators should be able to initiate the OpenHIE and any associated supporting processes with a single step.
- j. Build on commonly used technology:
 - i. In order to make it easy to run/configure/debug, the software should be built on popular technologies that developers like to use.
 - ii. Any 3rd party libraries used by the software should be easy for a typical developer to use.
 - iii. Any external software/systems (like the database) should also be easy to use.
 - iv. It should be easy to view the contents of the database.
 - v. If a traditional SQL database is used, then multiple databases should be supported (MySQL/PostgreSQL/Oracle).
- k. Unit Tests:
 - i. The source code should include unit tests that are based on the specific requirements of OpenHIE.
- a. License: The component would ideally be distributed under an open-source license that minimizes complexity and enables an implementer community to leverage the software in a broad variety of sustainability contexts.
 - i. The component should have a clear and standard license so that it is easy to understand what kinds of usage are allowed.
- a. Accessible Code:
 - i. The code should be hosted somewhere that developers like to use.
- a. GUI
 - i. The component should have an easy to use, well thought out and well implemented front end