SOFTWARE ENGINEERING STANDARDS & PRACTICES

Best Coding Practices

9AAD135446

Department	GF-IS ADM Applications Performance Excellence (APE)
Approver	Giulio Bitella, Global Department Manager
Owner	Tomasz Jastrzębski, Global Leader for Software Engineering (SE)

For the latest distributable version of this and other Software Engineering standards please visit this link to ABB Library.

WHAT IS THIS?

This document presents set of the coding (programming) standards and practices.

WHAT IS THE PURPOSE OF THIS DOCUMENT?

The purpose of the document is to establish complete programming standards and practices reference, containing both references to applicable Group-IS standards like InfoSec and Enterprise Architecture (EA) publications as well as practical standard implementation details/guidelines and the recommended software development practices.

TABLE OF CONTENTS

PLATFORMS

1. Azure PaaS is the most preferred hosting model. (ref. 9AAD133993)
2. Whenever PaaS hosting is not possible Azure IaaS should be used and IIS as Web application server. (ref. 9AAD133992)
3. Docker containers and microservices architecture should be considered for complex systems/apps.
4. Microsoft SQL Server is the most preferred transactional data storage. (ref. 9AAD135312)

DESIGN PRINCIPLES

1. Test Driven Development (TDD) approach is recommended.
2. Use design principles, like SOLID, DRY, KISS and YAGNI.
3. Use design patterns wisely. Do not over-engineer the solution and introduce abstractions if not justified.
4. Be aware of software anti-patterns.

WEB APPLICATIONS

1. Microsoft Azure PaaS (Platform as a Service) App Service is the most preferred hosting platform for web applications. (ref. 9AAD133993)
2. Currently we prefer not to solely rely on Microsoft Azure specific services. It is recommended to use App Services, SQL Server and Web Jobs services only. Storage service should be used via generic interfaces/base classes as a provider configured in application configuration file. (ref. 9AAD133993)
3. Refer to EA Architecture website for typical application building blocks and configuration. https://abb.sharepoint.com/sites/EnterpriseArchitecture/SitePages/Reference%20Architecture.aspx
4. Docker containers and microservices architecture should be considered for complex systems/apps and/or CI/CD infrastructure.
5. Applications should at least support Internet Explorer 11, Edge, Safari and Chrome on Windows 7 & 10, OSX, iOS and Android (ref. 9AAD133386, 9AAD131789)
6. Preferred front-end frameworks are: React and Angular (ref. 9AAD133991)
7. Use Google Material design for front end, contact UX & Design Team for more details. (ref. 9AAD134800, https://brand.abb/ux)

WEB API/SERVICES

1. Web services should be stateless, based on HTTP, use RESTful principles.
2. Web services available externally must use HTTPS protocol with TLS 1.2 or newer.

3. If required use Swagger/OpenAPI for service description. Do not expose service description externally.
4. Web APIs available for external services should rely on MuleSoft technology. This does not apply to web application backends. For more details contact C4E MuleSoft Operations or visit http://inside.abb.com/cloudintegration.
5. Always use UTF-8, both for input and output encoding.

MOBILE APPLICATIONS

1. Use Microsoft Xamarin for Windows 10 tablets, Android, IOS or Universal Windows Platform (UWP) for Windows 10 tablets (ref. 9AAD133994).
2. For more details on Mobile Applications development in ABB contact IS Innovation – Mobility team at mobileapps@abb.com.

CLOUD SERVICES

1. To ensure proper governance and cost management and security management every application environment should use a separate management container (e.g. Azure Resource Group).
2. Every cloud resource and management container name should end with environment name prefix starting with dash. E.g. “-prod”, “-dev”, “-stage”. Example: **myapplication-test** Resource Group, **myapplication-prod** SQL Server Database
3. Publicly visible resource URLs (registered in public DNS servers) should start with “abb-“ whenever dashes are allowed. Otherwise dash can be omitted. Example: http://abb-myapplication-backend-test.azurewebsites.net

AUTHENTICATION AND AUTHORIZATION

1. Applications authentication should rely on:
   a. Open ID Connect (OIDC) protocol and Azure Active Directory (AAD) or
   b. SAML 2.0 protocol and ABB managed ADFS/Open IdM OFIS servers (stsint.abb.com).
2. Use of active authentication is NOT allowed. Use passive authentication flows/schemas only.
3. Other authentication services/protocols are not allowed. It is NOT allowed to use own authentication services.
4. Refer to 9AAD133318, 9AAD133317 for Federation Services and Multi factor Authentication (MFA).
5. ABB applications typically use own, custom authorization. Use of AAD groups for authorization usually is not mandatory.

6. Building access control lists use user GUID rather than email. It happens users change email address or rejoin ABB but in a different role, with different permission set.
7. Mechanism/process that removes user from access group when user leaves ABB is required.
8. Whenever application uses custom authorization, make sure efficient caching mechanisms are in place and database calls are limited to necessary minimum.
9. User identity should be carried all the way to the backend allowing for proper authorization/logging/tracing.
10. JWT tokens are preferred and must be sent with every Web API call in “Authorization” HTTP Header using “Bearer” scheme.
    Refer to RFC6750-2.1 (https://tools.ietf.org/html/rfc6750#section-2.1)
11. Always validate token lifetime, audience, issuer and signature.
12. Web API methods should return authorization/authentication error codes according to RFC6750-3.1
    (https://tools.ietf.org/html/rfc6750#section-3.1)
13. OWASP guidelines should be followed:
    http://www.owasp.org/index.php/Guide_Table_of_Contents

DATA ACCESS

1. Whenever parametrized dynamic SQL must be used always pass parameter values using parameter objects, rather than SQL string concatenation/interpolation.
2. Storing passwords or any other credentials in source code repository is strictly forbidden.
3. Avoid using passwords and usernames: use Managed Service Identity/Principal and Azure AD (for Azure PaaS) or certificates and Azure AD (for IaaS).
4. If password-based authentication is required store it only on target machine.
5. Use encryption for data traveling through public networks, e.g. SQL Server Transparent Data Encryption (TDE).

UNIT TESTS

1. The name of test method should consist of three parts:
   a. the name of the method being tested,
   b. the scenario under which it's being tested,
   c. the expected behavior when the scenario is invoked.
2. Pay special attention to:
   a. properly arranging your test steps: arrange, act, assert,
   b. writing minimally passing tests,

c. avoiding multiple asserts in a single test method.

LOGGING AND TRACING

1. Use of Azure App Insights services is allowed under the condition that no strictly confidential or GDPR protected data is logged. (ref 9AAD125419)
2. Logging strategy:
   a. provide enough information to spot a problem,
   b. be able to target a specific section of code,
   c. not affect performance of the entire system,
   d. logs from one request should be connected,
   e. should be easy to add/modify,
   f. should be easy to configure.
3. Minimal Attributes set:
   a. **when**: event timestamp, interaction identifier etc.
   b. **where**: component name, server identifier, address (if webpage or API) etc.
   c. **who**: source address (IP, machine name), user identity
   d. **what**: type of an event, description and any relevant data. (ref. OWASP logging cheat sheet)
4. Minimal event set:
   a. log-on and log-off events,
   b. log-on failures,
   c. privileged users activities,
   d. attempts for unauthorized access,
   e. enabling, disabling and altering log configuration,
   f. start and stop of the system/process/component,
   g. initiation and completion of transaction that involve business critical and confidential information.

OPEN SOURCE SOFTWARE (OSS) COMPONENTS AND LIBRARIES

1. Use of open source software (OSS) in ABB is subject to strict approval process. Refer to group instruction GF/LI-35.04 Open Source Software (http://groupcharter.abb.com/Pages/LegalandIntegrity.aspx).
2. Open source software licensed under “copyleft” type of license, e.g. GPL (GNU General Public License) must NOT be used. Use of any “viral license” or “copyleft” type of license that allows derivative works only when permissions are preserved in modified versions of the work is prohibited.
3. Consider using only mature and actively maintained software packages.
4. All ABB-owned source code must be reviewed by the Open Source Competence Center (OCC) prior to its release. This includes external and internal

ABB applications, as well as source code provided by contractors. An application is defined as a versioned release of a software, no matter whether this is a stand-alone software package, a hosted web-service, a mobile app or firmware embedded in hardware.

5. Approvals are needed for all components that are part of the application prior to its release. The OCC is not involved in approvals possibly needed for software not included in the released artefact (e.g., development tools, standard ABB software).
6. When licensing commercial software, follow the indemnity guidelines. Obtain a list of OSS components to evaluate the risk to ABB and to monitor security vulnerabilities.

For more information and current list of approved software contact OCC Team at occ@in.abb.com. Source: https://abb.sharepoint.com/sites/SDIP/Portal/SitePages/OCC.aspx

ABB GROUP STANDARDS & POLICIES

1. Management of Information Systems (GD/IS-01) - 7ABA101160
2. Management of IS Applications (GI/IS-01.04) - 7ABA101116
3. Application Security Best Practices - 9AAD131734
4. Cryptographic Controls Policy - 9AAD125413
5. Identity and Access Control Policy - 9AAD125416
6. Backup and Archiving Policy - 9AAD125418
7. ABB Group Backup Standard - 9AAD114119
8. Monitoring and Logging Policy - 9AAD125419
9. Information Classification and Handling Standard - 9AAD126846
10. Open Source Software - GF/LI-35.04

ABB ENTERPRISE ARCHITECTURE (EA) STANDARDS

https://abb.sharepoint.com/sites/EnterpriseArchitecture/Lists/Architecture%20Standard%20List/AllItems.aspx

1. IS Standard for Application Development Frameworks - 9AAD135404
2. IS Standard for Managing Transactional Data - 9AAD135312
3. IS Standard for Managing Analytical Data - 9AAD135311
4. IS Standard for Data Integration - 9AAD135310
5. IS Standard for Web Java Script Frameworks - 9AAD133991
6. IS Standard for Application Platform as a Service - 9AAD133993
7. IS Standard for Mobile Development Frameworks - 9AAD133994
8. IS Standard for Virtual and Augmented Reality Frameworks - 9AAD134010
9. IS Standard for Performance and API Testing Tools - 9AAD134131
10. IS Standard for Code Repository Platforms - 9AAD133181

11. IS Standard for Continuous Integration and Delivery Solutions - 9AAD133303
12. IS Standard for ABB Web Browsers - 9AAD133386
13. IS Standard for Application Servers - 9AAD133992
14. IS Standard for Virtual and Augmented Reality Frameworks - 9AAD134010
15. IS Standard for Application Performance Monitoring - 9AAD134218
16. IS Standard for Automated Test Solutions - 9AAD133475
17. IS Standard for Desktop Automation Tools - 9AAD133326
18. IS Standard for Archiving Solutions - 9AAD133805
19. IAM - Federation Services - 9AAD133318
20. IAM - Multi-factor Authentication - 9AAD133317
21. Basic Computing Environment (BCE) - 9AAD131789

ABB APPLICATION PERFORMANCE EXCELLENCE (APE) STANDARDS

1. C# Coding Standards - 9AAD134036
2. C# Coding Standards - Field Guide - 9AAD134039
3. C# Best Practices - 9AAD134037
4. SQL Server Coding Standards - 9AAD134842
5. Java Coding Standards - 9AAD135383
6. Test Strategy - 9AAD134969
7. Unit Testing Field Guide - 9AAD135249
8. Source Code Management Standards - 9AAD134843
9. User Experience & Design Standards and Practices - 9AAD134800

The latest versions of the above standards are available in ABB Library (http://library.abb.com)

REFERENCES

1. OWASP Logging Cheat Sheet: https://www.owasp.org/index.php/Logging_Cheat_Sheet

RECOMMENDED READING

1. Martin, R. C. (2016). Clean code: A handbook of agile software craftsmanship. Upper Saddle River, NJ: Prentice Hall.
2. Wikipedia. Software engineering anti-patterns: https://en.wikipedia.org/wiki/Anti-pattern#Software_engineering

REVISION HISTORY

Rev.	Page	Change Description	Author(s)	Date
A	all	approved	Tomasz Jastrzębski et al.	2019-04-09

Dev

Information Systems

Applications Performance Excellence Department (APE)

Software Engineering Standards & Practices