European Maritime Safety Agency

Project Delivery

EMSA

IMPORTANT NOTE

Please note that this document is a template. In all projects/contracts, Project Delivery rules goes always along the lines described in this template but the final document will be modified and tuned according to each specific project/contract requirements, when needed.

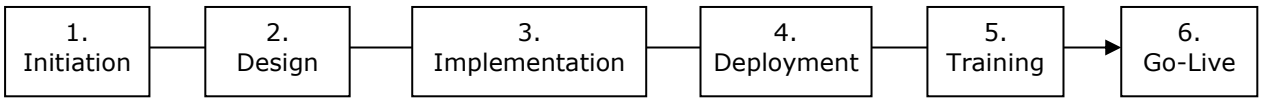
Table of contents

This document details the requirements of the project's processes for the provision of development services until final acceptance of the developed system excluding warranty.

Projects are composed by specific phases. This document presents all these phases. It describes what deliverables are requested, when they should be delivered and how they will be accepted by EMSA.

1. PROJECT PHASES

Six major steps are foreseen from signature of the contract until the final system is accepted, which are:



Deliverables are expected for each phase. They are detailed in Chapter 2 – Project Deliverables of this document.

If one deliverable is not accepted by EMSA during one phase no acceptance of any deliverable of the following phase can be done.

This should be fine tuned per each project, having always an initiation phase, followed by one or more iterations of Design, Implementation and Deployment, with training and a go-live at the end.

1.1. INITIATION

The objective of this initiation phase is to have a mutual understanding and agreement of methods and means that will be used for the completion of the project.

Immediately after the signature of the contract the contractor should prepare the kick-off meeting to cover at least the following subjects:

- objectives and organization;
- contractor team;
- project tools;
- project plan;
- methodologies and procedures;
- software development plan;
- content and level of detail of the project management documentation.

During this phase, the contractor is asked to work in close contact with EMSA in order to create a common view of the whole project.

1.2. DESIGN

The objective of the design phase is to create a complete set of functional and technical specifications describing what and how is to be implemented and the methodologies that shall be used to verify and validate the project execution.

During this phase the contractor shall deliver the documents described in 2.2 for review of the design and approval by EMSA.

1.3. IMPLEMENTATION

The objective of this phase is to develop and test the final version of the system before deployment and import of data.

The application will be developed according to the deliverables of the design phase.

Before delivery the contractor must test the developed application to verify the conformity with expected results and validate that the procedures as stated during the previous phases have been applied. The contractor should respect the "two chamber principle" which means that the team in charge of the tests should be different from the team in charge of the design and development. **Tests cannot be executed prior to the acceptance of Test documentation by EMSA** (this is, the acceptance of documentation that drives the tests execution campaigns)**.** Test report and test evidence should be transmitted to EMSA.

During this phase the contractor is responsible to:

- Prepare system documentation;
- Deliver code;
- Deliver a final version of the system;
- Prepare User documentation, including training support material;
- Prepare Test documentation;
- Test final version before its delivery and report test results to EMSA.

1.3.1. Continuous integration

Except if specified otherwise, all contractors are expected to work on a Continuous Integration/Delivery mode: project code (as well as all other deliverables) is stored in EMSA's GitLab repository during the implementation phase, as specified on paragraph 3.1. Doing so enables contractors to benefit from the continuous integration and deliver setup at EMSA, along with the possibility to identify as early as possible any situation that may hinder regular implementation process.

1.3.2. Factory Acceptance Test (FAT)

The development phase results in deliveries and their corresponding acceptance test. The purpose of these milestones is to validate that the releases are ready for delivery: each release satisfies quality & security requirements, passes unit tests and all the requested functionalities have been implemented and tested.

1.3.2.1. Tests readiness review

The contractor delivers the following documents for review at least 10 working days prior to the execution of the FAT. The test readiness review aims at checking that the FAT milestone can be held.

The delivery shall include at least:

- Test plan, describing how the delivery requirements will be validated;
- Interface Control document (final) including all updates;
- Test report and Test evidence for the tests to be run in the FAT milestone. FAT tests missing at this stage shall be listed;
- The set of application documents impacted by the release. EMSA shall provide feedback on the bundle within three working days. FAT may be rescheduled if the required coverage of the requirements is not met.

EMSA will decide if the FAT milestone can be held on the basis of these deliveries and the development progress tracking described in *2.3 - IMPLEMENTATION* and *3 - SOURCE CODE DELIVERY*.

1.3.2.2. Factory Acceptance Tests execution

The FAT is executed in a representative environment provided by the contractor, including all necessary test tools and infrastructure to validate the release requirements.

The FAT shall include all tests necessary to validate both functional and non-functional (e.g. performance, availability and security) requirements, unless otherwise agreed with EMSA.

The duration of the FAT for an EMSA application release is typically 2 or 3 days, but it might be longer or shorter, depending on the scope.

Prior to the FAT, the contractor should provide report and evidence of at least one long running test (minimum 24 hours) with representative load, in order to demonstrate the system stability.

The outcome of the milestone is based on the FAT test results demonstrating the compliance of the application release to the requirements, to the level specified.

1.4. DEPLOYMENT

Immediately after the validation delivery of a release, EMSA will start its deployment.

The objective of the deployment phase is to configure and make the final version available and fully running on its environments:

- Test;
- Pre-Production/Quality;
- Training (for applications having a training environment);
- Business Continuity Facility (BCF);
- Production.

During this phase EMSA will perform acceptance tests to accept the final version, the system documentation, the user documentation and the training materials.

Installation in a subsequent environment will only be done after all relevant tests for the current environment have passed; e.g. non-functional tests in pre-production/quality environment will only be started after all functional tests were successfully completed in test environment.

1.5. TRAINING

The training phase starts after acceptance of final version, user documentation and training materials.

During this phase the contractor should organise and conduct the following training sessions:

- teach EMSA's operational unit end-users to use the application;
- teach EMSA's IT personnel to manage and administrate the system.

Training will be done on the test and/or quality environment.

In case adjustments or corrections are found necessary during training sessions the contractor will be asked to update user documentation and training materials.

1.6. GO-LIVE

The go-live phase starts after the final version is accepted by EMSA. The objective of this phase is to:

- obtain an optimum configuration of the system and maximal performance in the production environment by fine tuning the complete technical infrastructure;
- perform necessary correction and adjustments of the system while it is used by end users in real situation.

The go-live phase ends at the final acceptance of the system.

2. PROJECT DELIVERABLES

EMSA follows a Continuous Integration and Delivery workflow. Usually these processes are associated with code integration, however it does not have to be limited to coding. Continuous integration of diverse artefacts will allow continuous verification, and will improve the performance of the entire development project. As such contractors are advised to make use of tools EMSA makes available to continuously build all artefacts.

Documentation must be provided in electronic format compatible with MS Office.

2.1. INITIATION

2.1.1. Project Management documentation

Project management documentation shall reflect the project management methodology proposed by the contractor in its bid. It shall include at least the following documents:

- **Project plan**: must include the following items at least: project charter, project management approach, scope, Work Breakdown Structure (WBS), project team, Gantt chart, deliverables milestones, working locations, meetings planning and reports.
- **List of outstanding and closed Action Items**.
- **Highlight report**: weekly report on the status on the project containing (at least) ongoing tasks, resources usage, progress status, and issues foreseen.
- **Agenda of the meetings**: the contractor is responsible for providing detailed agenda and additional requests 3 days before the meetings for all relevant meetings held between EMSA and the contractor.

Minutes of the meetings: the contractor is responsible for providing the minutes of the meetings for all relevant meetings held between EMSA and the contractor. The minutes of the meetings must include at least the topics discussed, decisions taken and action items with indication of the responsible person and deadline of the actionsProject Management documentation shall be defined and agree in the Initiation Phase and shall be kept updated during the entire project lifecycle.

2.1.2. Software Development Plan

The Software Development Plan identifies the general methodologies, processes and working practices to be used on the development of the project. It serves as the basic rules and practices guideline for all the remaining technical tasks of the project. The Software Development Plan will at least address the following issues:

- Software development approach: description of the strategy of the software development life cycle (waterfall, incremental, evolutionary life cycle, etc);
- Software engineering environment: identification of any standards, methods and tools that apply to the project, whether defined by the contractor or required by EMSA. Details regarding the application of each item identified shall be addressed on the subsequent sections when referring them;
- Software quality assurance process: definition of rules, practices and conventions to be used as to obtain the desired quality of the final product and to evaluate if this plan is being properly implemented. References to standards or one or more contactor's own documents are possible;
- Software security assurance process: definition of rules, practices and conventions to embed security from design and throughout the whole software development lifecycle, and to evaluate if this plan is being properly implemented. References to standards or one or more contactor's own documents are possible;

- Software configuration management plan: definition of rules, practices, conventions and tools to be used as to maintain the software configuration. References to standards or one or more contactor's own documents are possible;
- Design standards: definition of rules, practices, conventions and tools to be used in definition of the design. References to standards or one or more contactor's own documents are possible;
- Coding standards: definition of rules, practices, conventions and tools to be used in development of the code. References to standards or one or more contactor's own documents are possible;
- Testing standards and practices: identifies the standards, practices, conventions and tools that will be used to test the developed software. References to standards or one or more contactor's own documents are possible.

2.1.3. Tools for project controlling and design

EMSA Jira/Confluence is the mandatory tool for project issue tracking and requirements tracking.

EMSA ATLAS tool is the Architecture repository for EMSA applications.

Unified Modelling Language (UML) should be used for object and system modelling. EMSA could suggest the use of a tool (e.g. Enterprise Architect UML). The contractor is free to use another UML modelling tool as long as he guarantees its compatibility with EMSA tools. Therefore prior agreement with EMSA is required.

Development progress shall be tracked using reporting based on the documents described in 2.4.

2.2. DESIGN

The following activities shall be defined during the design:

- Mandatory Clustering – All critical services must be fully clustered with no SPOFs;
- Clean Restarts – All services must restart in a clean manner at least 95% of times;
- BCF certification – it is mandatory for next major release and must be tested and accepted before go-live;
- Application configuration files - shall be externalized from the application binaries.

Design documentation should be prepared in close collaboration with EMSA's personnel.

2.2.1. Design documentation

Design Documentation should cover:

- Updated ATLAS architecture file for the application (with the support and supervision from EMSA Technical Project Manager, who will be responsible to complete and validate this action)
- Functional design specifications;
- Technical design specifications;
- A draft version of the Software Test Plan containing at least the test strategy;
- A draft version of the Software Security Assurance Plan containing at least the high level strategy and practices to enforce Security by Design, with reference to relevant standards and practices for secure coding and security testing during the software development lifecycle;
- Data and/or Processes Migration Strategy and Plan.

2.2.2. Functional specifications

Functional specifications will be used as guidelines for the implementation of the system.

They should describe in detail at least the following:

- Use Cases representing the system functionalities;
- Capabilities and processes;
- Interactions with users/systems;
- Traceability matrix between Functional Requirements/Business Rules and Use Cases.

2.2.3. Technical design specifications

Technical design specifications will be used as a blueprint for the system implementation. They describe how the system will be implemented in order to cope with functional specifications.

They should include as a minimum:

- Context diagram of the application;
- Conceptual and physical system architecture;
- Software design and layering;
- Modules and components;
- Process, workflows and algorithms design and documentation;
- Interfaces definitions: Interface Control Document specifying the application interfaces (delivered version shall highlight the changes introduced in the new version)Interface definitions shall follow the EMSA SOA guidelines and Rules.

2.2.4. Draft version of the Software Test Plan[1]

The draft version of the Software Test Plan will serve as the basis for the Software Test Plan to be implemented during "Development and Test" phase. This draft version shall include at least:

- Definition of the Software Test Plan Structure and global strategy;
- Reference to the different test phases to be implemented;
- Definition of the test detailed strategy presenting an overall perspective of testing and identifying individual test phase plans for unit, integration, functional, performance, load and stress test phases. Each test phase plan shall include at least:
    - Description of the test phase strategy,
    - Test phase standards and practices,
    - Test phase supporting guidelines,
    - Test phase selection criteria,
    - Test phase evaluation metrics,
    - Completion criteria for the test phase,
    - Test phase implementation templates,
    - Requirements Traceability Verification (ensuring that each requirement has one or more test cases associated with it);
- Test Cases Specification
- Reference to the test environment(s) to be used;

[1] JIRA and Xray are the tools use by EMSA.

- Software Test plan execution planning;
- Software Test team responsibilities and staff.

The final version of the Software Test Plan is to be provided during the Development and Test phase.

2.2.5. Draft version of the Software Security Assurance Plan

Security by Design should be addressed from the design phase of every project for ensuring confidentialiy, integrity, availability of information. The draft version of the Software Security Assurance Plan will serve as the basis for the Software Security Assurance Plan to be implemented during "Development and Test" phase. This draft version shall include at least:

- Software security design principles, with reference to the main security standards and guidelines that will be followed (please refer also to the secure code requirements set under the "EMSA System and Application landscape" i.e. related to compliance with OWASP framework);
- Risk assessment, threat analysis and enumeration;
- Secure code reviews and inspections integrated in the lifecycle;
- Software Security Testing plan;
- Software Security Testing team responsibilities and staff;
- Application security configuration;
- Application security verification i.e. how to verify application security regularly;
- Tools and techniques.

The final version of the Software Security Assurance Plan is to be provided during the Development and Test phase.

2.2.6. Data and/or Processes Migration Strategy and Plan

The Data and/or Processes Migration Strategy shall be presented describing what will be the methodology used to execute this process.

A Migration Plan shall be developed and agreed based on the following phases:

- Analysis
- Design
- Development and Unit Testing
- Validation
- Deployment

2.3. IMPLEMENTATION

During this phase it is common to identify the need to alter the contents of deliverables of previous phases. As such, if needed, the contractor and/or EMSA may suggest modifying the content of the deliverables of previous phases. These modifications should be agreed by EMSA.

Figure 1 – CONTINUOUS INTEGRATION & DELIVERY

In regard to code deliverables:

- during implementation process, EMSA will enable continuous inspection where each code delivery goes through a series of automated tests. Results of such tests will be made available to contractor during implementation.
- associated with a milestone, a formal delivery will go through a similar process before being submitted to EMSA's formal delivery acceptance. If accepted, the delivery is deployed to the defined target environment, usually test environment.

Besides

i. coding and development of the solution
ii. associated automated development phase tests
iii. development of all project defined deliverables
iv. incremental versions of the Drafted documents defined in previous stages

during the implementation phase contractor should, at least, produce the system documentation described below.

2.3.1. System documentation

2.3.1.1. Operational and Maintenance Documentation

Operational and maintenance documentation must explain how the system should be operated and maintained on a daily basis. It should include the following documentation:

- Installation manual;
- Operation and Maintenance Manual;
- Incident Handling Procedures. HOW-TO do troubleshooting and root-cause analysis.

2.3.1.2. System building procedures

System building procedures should allow EMSA to completely build the latest version of the system at any moment.

System building procedures shall be executed in EMSA building environment. The contractor has to provide all the needed information to EMSA to prepare this building environment. EMSA favours the use of virtual Linux building environments.

At the delivery of the final version the contractor shall provide an automatic build procedure with the complete source code, additional software packages and code generators.

For each code generator used during development a corresponding generator should be provided to EMSA.

2.3.1.3. Infrastructure (HW and SW) documentation

The contractor is requested to provide a complete and detailed architecture definition and sizing for the following environments:

- Test;
- Pre-Production/Quality;

- Training;
- Production.

The environments will be provided at EMSA Data Centre, BCF or Cloud.

In order to correctly size the production environment the contractor must consider the following elements: system architecture, implementation, non-functional requirements and the performance requirements specified in Annex I – Tender Specifications and/or in a RFC.

For the production environment, detailed information about requirements for servers characteristics, network, bandwidth, base software, databases, security and accessibility shall be provided to EMSA. For the others environments, the same level of information must be provided with an indication of expected performance.

2.3.1.4. System monitoring documentation

This documentation must explain how the system should be monitored both for availability and performance. EMSA current real-time monitoring system is implemented on Nagios. The documentation should describe the main real-time monitoring checks for the system, with related scripts and queries to perform them (at high level, if not possible to define them in details). The "incident handling procedures" to be documented under point 2.3.1.1 should be linked to the relevant monitoring checks in the following way: the main failure modes of the system should be documented, providing both a monitoring check to detect the failure, and an associated incident handling procedure to recover the system.

2.3.1.5. System security plan

This documentation must explain how to:

- Perform appropriate security hardening and configuration of the environment where the system is installed (referring to EMSA System and Application landscape, where available security technologies like OS, firewall, IDS/IPS, WAF is provided);
- Identify relevant security logs produced by the software, supporting middleware and infrastructure that must be collected in the SIEM (Security Information and Event Management) for real-time security monitoring;
- Define the most important security events produced at run-time, and define how to monitor those in real-time as collected in the SIEM;
- Propose appropriate security incident handling procedures to respond to the most important security events identified above.

EMSA current SIEM system is implemented in Splunk.

2.3.2. A version of the system[2]

In formal milestones a version of the system is delivered to EMSA and should contain:

- System implementation;
- Related source codes, build procedures and supporting documentation;
- A complete system documentation;

[2] Also known as "Functional Prototype". Please note that on a Specific Contract for implementation, EMSA can request one or more "Functional Prototypes" as formal milestones.

- Test documentation including automated tests for at least all non GUI tests and performance tests using EMSA approved test tools (e.g. JMeter, SoapUI, SoapUI Pro, ReadyAPI, Robot Framework, Cumcumber)[3];
- Mocks of the services implemented by the system for integrating client applications;
- Automated deployment scripts;
- Release notes.

2.3.3. Data and/or Processes Migration

The approved Migration Plan shall be implemented applying an incremental and controlled transition based in the following phases:

- Analysis
- Design
- Development and Unit Testing
- Validation
- Deployment

2.3.4. Unit tests and Memory Profiling tools

The contractors shall deliver unit tests alongside the source code to be run after the daily build. Unit test coverage requirements are specified inEMSA Quality Gate (see below).

Unit tests can be run every day by the CI tools based on the daily push. A report containing the results can be sent to all project managers and heads of units.

For every requirement concerning analysis of data, a realistic set of data has to be provided in such a way that it can be played back at any time.

Memory profiling tools shall also be used in the development to avoid possible memory leaks. A memory checker shall be used by the contractor to verify that no memory leaks exist.

2.3.5. Deployment automation

EMSA uses Puppet as a tool for deployment automation. The contractor shall deliver Puppet modules for installation of each release, complemented with other scripting tools (e.g. WLST for Weblogic), if applicable and where needed. Installation based on unpacking archive files or RPM installation packages are not allowed. The deployment scripts shall not include the installation of middleware and infrastructure components. It is recommended to use Flyway DB to manage Data Base changes.

Applications shall always be installed from scratch, with the exception of databases. The contractor shall deliver alongside each version, the scripts to create the database from scratch and to update to the new version from the preceding one.

Environment dependent configuration shall be provided to Puppet as parameters defined in the Puppet module and therefore will only be specified in the Puppet manifest applied during 'puppet apply' or by 'puppet agent'. To avoid unnecessary releases of new Puppet modules, version dependencies should also be defined in the same manner.

Administration accounts (root, sysdba,…) usage is not allowed (exceptions can be evaluated).

The application components shall not require root access to be executed/running.

For tests see chapter "2.3.8 - Test Automation" below.

[3] JMeter, SoapUI, SoapUI Pro are not needed if ReadyAPI is used, as ReadyAPI already includes SoapUI, SoapUI Pro and a tool similar to JMeter for performance testing.

Once formally validated, the delivery is deployed on test environment and submitted to acceptance test.

2.3.6. Integration, regression and performance tests

Integration, regression and performance tests must be automated by Contractor(s) and are executed in the corresponding stage of the continuous integration pipeline.

Contractor is expect to deliver during the development life-cycle the automated test scripts along with solution source code and unit tests.

Integration, Regression and performance test execution should produce an output format compatible with Jenkins so it can be included in the build and build pipeline reports.

2.3.7. User documentation

2.3.7.1. User documentation

The user documentation will explain the different components of the system to EMSA's users.

User documentation should include:

- A quick start guide to explain how to access the system and use main functionalities;
- A complete user manual to describe how to use all functionalities of the system;
- On line help content. This will contain a contextual help explaining content and functionalities of every screen of the system and generic help which will provide the same content of the user manual.

Text should be supported by illustrations and screen copies all through the user documentation. User manual and on line help should include a table of content, a glossary and an index.

2.3.7.2. Training support material

As a minimum training support material should contain:

- training presentations to be displayed during the sessions;
- practical examples;
- training data sets – if applicable.

2.3.8. Test documentation

Tests to be performed must cover the two following objectives:

- Verification tests: verify that the product is in line with the functional and technical requirements and design specifications and that implementation best practices were applied (to be done by EMSA and contractor);
- Validation tests: ensure that business requirements are met by applying change management procedures, following the activities described in the project plan and in the software development plan (to be done by EMSA).

The test documentation and test results should provide evidence that these objectives are met.

Test documentation should detail all necessary documents to plan, design, execute and report tests. This should include as a minimum:

- The Software Test Plan with all details regarding the test process:
    - Definition of the Software Test Plan Structure and global strategy;
    - Reference to the different test phases to be implemented;
    - Definition of the test detailed strategy presenting an overall perspective of testing and identifying individual test phase plans for unit, integration,

functional, performance, load and stress test phases. Each test phase plan should include at least:

- Description of the test phase strategy,
- Test phase standards and practices,
- Test phase supporting guidelines,
- Test phase selection criteria,
- Test phase evaluation metrics,
- Completion criteria for the test phase,
- Test phase implementation templates;

- Results achieved with the test phase implementation including at least[4]:
  - Test cases,
  - Test scripts,
  - Data sets,
  - Test results,
  - Test phase report;
- Reference to the test environment(s) to be used;
- Software Test plan execution planning;
- Software Test team responsibilities and staff.

Test results should be added to each test plan once the corresponding tests have been executed.

The contractor will be responsible for preparing all documentation including test cases, test data to be used and test environment.

Before the Factory Acceptance Tests (FAT) the developer contractor must deliver the test cases and test results in EMSA's Xray[5] (a JIRA add-on), which should be updated after FAT if needed.

2.3.9. Test Automation

Test automation scripts shall be provided (and maintained) within the scope of the project as well a consistent and meaningful Test Dataset.

- Unit Test (See Code quality & Security – EMSA Quality Gate on Section 3.3);
- Test Scripts
  - Initially, must cover meaningful Test Cases for a set of critical business functions (to be agreed) to always be executed as regression tests
  - At each delivery, Test Scripts must be enriched with:
    - New Test Cases for critical business functions either new or not yet covered,
    - New Test Cases to allow the specific validation and verification (V&V) of critical and blocking error found in the previous versions;

[4] Some of these itens could be delivered in TestLink.

[5] https://www.getxray.app/

- Consistent Test Dataset to be provided (and maintained);
    - Test Dataset must be automatically loaded into the TEST environment (using a data loading mechanism),
    - Initial dataset must be aligned with the Test Scripts and Test Cases agreed,
    - Test Dataset must be enriched with:
        - Data for new Test Cases,
        - New data to allow V&V of critical and blocking error found in the previous versions;

- SOAPUI, JUnit and JMeter shall be used for test automation to the possible extent[6].

2.4. DEPLOYMENT

Deliverables of the deployment phase are:

- A final version of the system deployed and fully working in Test, Pre-Production/Quality, Prodution and if applicable in training and BCF;
- Updates of the system documentation, user documentation and training materials, if needed;
- Updates of the deliverables of the design phase, if needed.

2.5. TRAINING

Training sessions are foreseen for:

- EMSA's operational unit end-users;
- EMSA's IT personnel.

All sessions should mix theoretical and practical parts. In the practical parts, the users actually use the system with "hands on examples".

The contractor will be responsible to conduct training sessions. The contractor should provide all supporting material and prepare "hands on examples". It will also be responsible for the preparation of the technical environment (software and data).

Training sessions will take place at EMSA's premises in Lisbon. EMSA will provide infrastructure (rooms, IT equipment, video equipment, etc.) and will be in charge of administrative organisation of the courses (planning, notifications, and evaluation) and duplication and distribution of course documentation. All information and documentation required for the training should be delivered to EMSA well in advance by the Contractor.

[6] [6] JMeter and SoapUI are not needed if ReadyAPI is used, as ReadyAPI already includes SoapUI and LoadUI (a tool similar to JMeter) for performance testing.

2.6. GO-LIVE

Deliverables of the Go-live phase are:

- Updates of the system documentation if needed;
- Report on the tasks undertaken by the contractor and their results;
- Final system.

3. SOURCE CODE DELIVERY

3.1. SOURCE CODE MANAGEMENT (EMSA QUALITY & SECURITY GATE)

During development, the contractor shall use EMSA's GitLab repository as their main source code repository.
Normally EMSA will perform daily builds of the code in the development branches, run automated tests and perform automated quality checks of the code as described below (EMSA Quality Gate), in a similar way than the one that occurs upon Formal Release.
To ensure the successful activity described above the contractor will need to set up in their environment the checks as explained in section 4.4.4.

3.1.1. Access to resources - Repository

As indicated in 1.3.1, except if specified otherwise, the contractor is expected to work on *Continuous Integration* mode, using EMSA's Gitlab repository to store the code and all other associated deliverables being developed.

As owner of source code, EMSA makes it accessible to its contractors. *Annex A - How to deliver Source Code via GitLab* provides the details on how to use GitLab for EMSA projects.

3.1.2. Delivery acknowledgement Checkpoint - Deliverables Inspection and Code Tests

During regular development contractor can benefit from the continuous integration and corresponding inspection procedures, however to maintain the formalism associated with adopted project framework, contractor has to, at agreed milestones, formally submit a version of the system deliverables. In practice this means that at the event of formally execute a delivery, contractors will submit a Gitlab Merge Request to a target Branch, as described in the *Annex A - How to deliver Source Code via GitLab*. This Merge Request will be processed by EMSA's, contents will be validated and subject to a series of automated and manual controls and checks as illustrated in following *Continuous Integration* workflow:

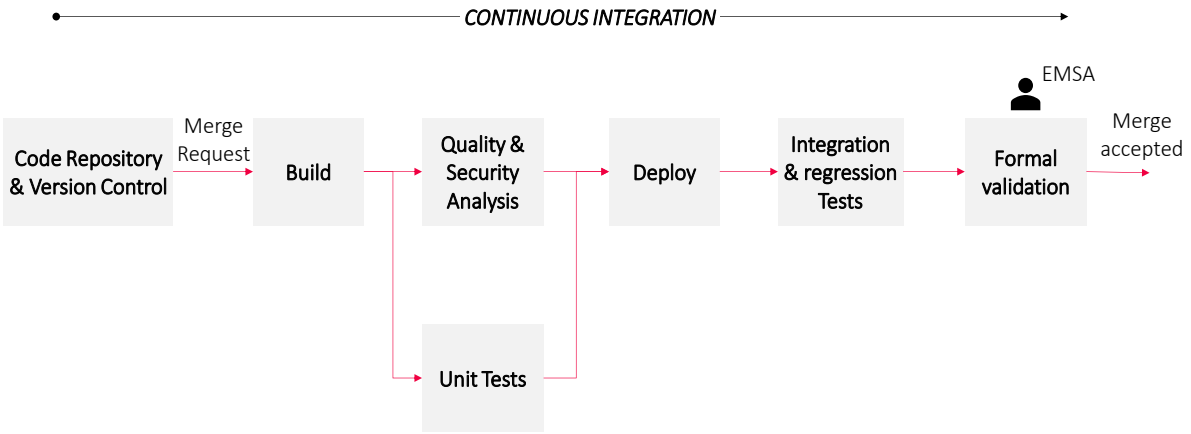


Figure 2 – CONTINUOUS INTEGRATION TESTS – *MERGE REQUEST* VALIDATION

Upon the execution of these verification and validation procedures, EMSA can accept the merge request or refuse the merge request.

By accepting the Merge Request EMSA is not bound to accept the Delivery. Accepting the merge request only means that the intention of delivery a set or sub-set of artifacts from

contractor has been acknowledge. Only after this acknowledge of deliver, the remaining activities associated could be initiated.

3.1.3. Submitting code for formal delivery acceptance

As previously stated, formal deliveries are performed through GitLab merge request action, which will trigger the delivery reception acknowledgement procedure.

Standard contents expected which each delivery are:

- Source code;
- Unit test code (and dataset when applicable);
- Build scripts;
- Database scripts;
- Deployment scripts
- Integration & regression tests;

Note: each project may have additional requirements.

All deliveries are expected to be compliant with EMSA tools and processes:

Continuous integration pipeline is defined enabling gated control of the diferent stages of the build management as described in 3.2, and addind additional flow controls to simplify the acceptance process.

Contractor must comply with the imposed standard and, upon justifiable need, explain and instruct EMSA team of such changes.

The automated continuous integration and deployment pipeline assumes:

- **GitLab as the code repository and version control tool**:
- **Build with Maven**
- Execute **Unit Tests via Maven** plugins;
- **Quality & Security audits via SonarQube and Kiuwan**;
- **Deploy through Puppet scripts** and Flyway DB
- Execute **integration and regression** tests via agreed automated tool
- **Final (formal) validation** is a human activity that formally acknowledges a given delivery

More details can be found in Annex A – How to deliver Source Code via GitLab.

3.2. BUILD MANAGEMENT

EMSA currently uses the following applications in building and testing the source code delivered by contractors: Jenkins, Maven, Sonar and Nexus.

Jenkins checks out the source code from GitLab (either latest version or a specific branch or tag) and builds it locally. The build definition is contained in a Maven POM. After a successful build, the CI server will trigger all kinds of automatic tests, build the test coverage report and trigger Sonar to run the code quality measurements. It will also upload the generated build artifacts to Nexus. The CI server also provides a dashboard with the overview of the status of all projects. Contractor shall deliver Maven POMs for the application.

During development a daily build cycle in EMSA will be put in place. Jenkins will be exposed outside, allowing contractors to verify results of the build process and upload new components, i.e. the contractor will have read access only to his own projects. EMSA may

trigger an alert/veto on upload of new components. Builds will automatically run every night; and results will be made available to EMSA through a dashboard for progress tracking purposes.

3.3. CODE QUALITY & SECURITY – EMSA QUALITY GATE

SonarQube is used for quality checking. EMSA uses a special Java Quality Gate based on "Sonar way" quality profile. **Failure to pass this Quality Gate will imply the rejections of the version being delivered.**

All projects (new and existing ones) will be submitted to the Quality Gate. For existing projects an adoption plan may be established and agreed with EMSA. However, "**Blocking issues**" shall not be accepted.

Quality Gate shall be mandatory for all projects. As for any other project task, a Quality Gate will consume effort and time. Contractors are encouraged to adopt continuous and rigorous quality checking measures during the development process and submit each version to EMSA Quality Gate before delivery.

New versions of the EMSA Quality Gate may be released during the project implementation. In such case, contractors will be informed about it.

Quality validation shall be executed using SONAR QUBE 7.9.1 and quality gate and quality profile are the defaults ones also known as SONAR way.

4. ACCEPTANCE PROCEDURES

For each deliverable, EMSA provides a formal indication of the acceptance, conditional acceptance or rejection of the deliverable to the contractor.

4.1. CLASSIFICATION OF SOFTWARE ISSUES

EMSA will classify issues found on software into three different categories according to their impact and severity as follows:

- Blocking issues: structural problems or serious issues (functional or technical) considered as limitations of the implementation with very high probability of interfering with the expected result. The contractor will be obliged to correct/execute all issues considered in the category. Blocking issues stop any kind of acceptance procedure until the correction is provided;
- Critical issues: problems or issues that do not conform to the requirements or specifications or best practices or considered to be the wrong approach to obtain the result, but for each one of them a workaround is available. Correction of Critical issues is mandatory for the next delivery;
- Minor issues: changes considered to be a better solution but without a deep impact in the quality of the system. The correction/execution of the issues of this category will be decided case by case.

The outcome of the acceptance procedure is positive if no issue is found by EMSA. If issues are found by EMSA during the acceptance procedure, the contractor is requested to immediately correct them and the acceptance procedure restarts from the date of the delivery of the corrected deliverable.

EMSA can decide to conditionally accept the deliverable when some issues remain uncorrected if those issues are not blocking the usage of system. In order to accept such remaining issues the contractor shall propose a deadline for the correction and EMSA to accept it. The EMSA will take the decision on conditionally acceptance of the product after evaluation of each remaining issue.

EMSA reserves the right to perform code reviews, either during the project execution or before final acceptance. During a code review, attention will be given to the topics described in Chapter 17 "Code smells and Heuristics" of the "Clean code" book[7]. Based on this review, if a large number of issues are discovered, the delivery may not be accepted until the issues have been solved.

No acceptance shall be made by EMSA without a successful execution of the automatic build procedure nor without a successfull installation of all deliveries in EMSA quality environment.

4.2. DOCUMENTATION

In the case of Project Management documents, EMSA will provide comment and/or reservations which will be transmitted to the contractor within **one week** of the date of delivery. Based on this comment and/or reservations EMSA will either accept or reject the deliverables. In the case of rejection the contractor will be requested to provide a new appropriate revision.

In the case System documentation and User Documentation, EMSA will provide comment and/or reservations which will be transmitted to the contractor within **two weeks** of the date of delivery. Based on this comment and/or reservations EMSA will either accept or reject the deliverables. In the case of rejection the contractor will be requested to provide a new appropriate revision.

In the case of Design Documentation, EMSA will provide comment and/or reservations which will be transmitted to the contractor within **two weeks** of the date of delivery. Based on this comment and/or reservations EMSA will either accept or reject the deliverables. In the case of rejection the contractor will be requested to provide a new appropriate revision.

4.3. VERSION OF THE SYSTEM

A version of the system will be evaluated by EMSA when available and running on the test, quality, production and BCF environments.

Before the final version is accepted EMSA will verify if:

- all issues detected in the previous acceptance tests have been corrected;
- it conforms with the functional specifications;
- it conforms with the technical specifications;
- non-functional requirements are met;
- it works correctly in its environments according to all requirements and specifications.

[7] Clean Code by Robert C. Martin

Publisher: Prentice Hall

Release Date: August 2008

ISBN: 9780136083238

EMSA will provide issues which will be transmitted to the contractor within the agreed time for delivery. Based on these issues EMSA will either accept or reject the version. In the case of rejection the contractor will be requested to provide a new appropriate version.

As a principle, no acceptance shall be made by EMSA without a successful execution of the automatic build procedure nor without a successfull installation of all deliveries in EMSA quality environment (pre-production).

4.4. SOFTWARE QUALITY & SECURITY ANALYSIS

EMSA has integrated Security testing in development lifecycles and it starts at the coding phase. Delivered software's Quality & Security is systematically and automatically analysed to guarantee that deliveries do not contain important quality flaws and security breaches and lead to a progressive improvement of software assets or at least does not increase technical debt.

4.4.1. Types of monitoring

- Quality of individual deliveries (Delta/Incremental);
- Quality of current assets (full picture that includes all pre-existing problems);
- Quality of service (lifecycle monitoring of Quality & Security scores evolution).

Note: to isolate pre-existing Quality & Security defects from new developments, EMSA's analyses both:

- Initial Baseline assessment – snapshot of existing situation to assess global software quality, to identify any existing major problem requiring urgent decision and action and to monitor long-term evolution of software quality;
- Delta/Incremental analysis – focus exclusively on changes (new/change/delete) to prevent the introduction of any new (significant) defect and the increase of existing technical debt.
-

4.4.2. Measures definitions

Software Quality & Security analysis are based on international standards (ISO 25000 for Quality and OWASP, MISRA, CWE, PCI, programming languages' manuals and other major security standards for Security).

- **Efficiency**: represents performance relative to the amount of resources used under specific conditions (CPU, memory, network bandwidth…). Response times perceived by software users are adequate, it makes an efficient use of the platform that hosts it and it is able to support increasing volumes of use without degradation in performance (with changes in the execution platform).
- **Maintainability**: software is easy to change or extend (for a future maintainer) effectively and efficiently. This emergent property is distributed between software architecture, applied engineering design, coupling degree, code normalization and the absence of defects like code duplication or the use of anti-design patterns.
- **Portability** (ability to be migrated): software can be effectively and efficiently migrated to other platforms (technological environment, operating system or database, middleware, etc.) and operational or use environments.
- **Reliability**: the ability of a system or component to perform specified functions when it is used under specific conditions and period of time.
- **Security**: the ability to protect the information and data, so they cannot be read or modified by unauthorized people or systems.

4.4.3. Acceptance criteria

Acceptance criteria are divided into two major groups:

- **Mandatory compliance criteria** – these are essential criteria that must be followed by a development to be released. Failure to comply with one of these criteria leads automatically to a failure in Quality & Security acceptance tests.
- **Optional fulfilment criteria** – these are desirable criteria for developments to have and should be reached as a goal. Failure to meet one or more of these criteria does not automatically imply a acceptance tests failure.

4.4.4. Results

EMSA provides the SonarQube profile for the contractor to be able to run on their side a pre-check and verify if the software passes the quality gate criteria.

4.4.5. Pre-testing code prior to formal delivery

Contractors are invited to test their deliverable prior to submitting it for formal *merge request*. This testing engine uses exactly the same rules as automated delivery tests. It enables contractors to check if their code meets the SLAs, perform any necessary corrections and only request formal *merge request* when the deliverable meets SLAs, thus avoiding *merge request* refusals.

4.4.6. Service level requirements

Default service level requirements shall be defined on a case by case basis.

4.5. FINAL SYSTEM

The final system will be evaluated by EMSA when the accepted final version will be available in the production environment. EMSA will verify if the system operates correctly while being used by end users in real situation.

The Final system is accepted within the agreed acceptance period at the condition that no urgent issues as described in chapter 4.1 are found.

In the case a blocking issue is found, the acceptance period is reset until a corrected version is made available on the production environment by the contractor.

5. MEETINGS

5.1. PROJECT MANAGEMENT MEETINGS

Action list, risk registry and planning will be reviewed during project management meetings.

At each project management meeting, the contractor should present an updated project status report.

In addition to the project status reports, between the project management meetings, the contractor delivers to EMSA a flash report.

The contractor is responsible for providing detailed agenda and supporting documents for the meetings, support the discussions during the meeting, and providing the minutes of the meetings. The detailed agenda and supporting documents must be provided by the contractor 3 days before each meetings. The minutes of the meetings must include at least the topics

discussed, decisions taken and action items with indication of the responsible person and deadline of the actions.

6. ENVIRONMENTS

Development Environment is the Contractor's responsibility

- The Contractor is responsible for setting up and maintaining the development environments and their own integration test environments if needed.

TEST Environment

- Managed by EMSA but in a best effort basis;
- Access to TEST Environment can be granted to the contractor if justified and approved by EMSA.

PRE-PRODUCTION Environment

- Managed by EMSA;
- Open to internet through IP filtering;
- Mimics PRODUCTION as much as possible: same architecture, same versions;
- The contractor does not have access to PRE-PROD Environment. Access can be granted for a limited period of time if justified and approved by EMSA.

TRAINING Environment

- Managed by EMSA;
- Open to Internet;
- Smaller infrastructure with reduced architecture with the same versions;
- The Contractor does not have access to Training Environment. Access can be granted for a limited period of time if justified and approved by EMSA.

PRODUCTION and BCF Environments

- Fully managed and controlled by EMSA;
- Monitored 24x7;
- The Contractor will not have access to PRODUCTION (exceptions might be considered on a case by case basis).

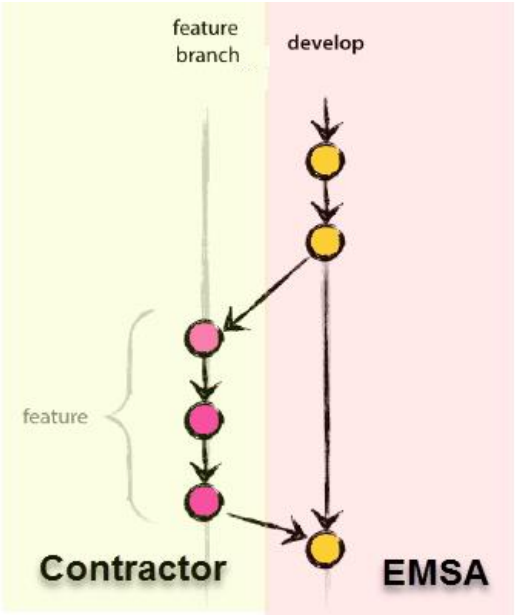
Each environment shall have its own set of application configuration files fully externalized from the application binaries and concentrated in a single place

Annex A
to Project Development Practices

How to deliver Source Code via GitLab (Branching Strategy)

For software releases deliveries:

1. EMSA will make available to the Contractor, a branch named "**develop**". Contractors will be prevented from pushing code changes directly to this branch.
2. The Contractor will be authorized to create additional branches ("**feature**" branches) from the "**develop**" branch. Alternatively, if the contractor wants more control over the repository (managing access etc …), we recommend for the contractor to fork the "**develop**" branch.
3. The contractor will work/deliver the source code on the created "**feature**" branches.
4. The contractor will create a "merge request" to the "**develop**" branch when each feature is ready
5. EMSA will accept the "merge request" in order to merge the finished "**feature**" branch into the "**develop**" branch and will periodically validate the source code by building it from the "**develop**" branch.
6. Once all features agreed are completed, the contractor will request EMSA to create a new "**release**" branch.
7. The contractor will have permissions to push fixes directly to this "**release**" branch. However, no new features should be added after the "**release**" branch was created.
    a. If new features need to be added, a new "**release**" branch is needed. The contractor will need to request this to EMSA. Before a new "**release**" branch can be created:
        i. All fixes on the existing "**release**" branch need to be merged to "**develop**" branch. Contractor will create a merge request;
        ii. The new "**release**" branch will be created by EMSA from "**develop**" branch.
8. Once the release candidate has passed all tests (SAT) and EMSA schedules it for release to production, EMSA will merge the "**release**" branch to "**develop**" and "**master**" branches.



For software hotFixes deliveries:

1. EMSA will make available to the Contractor, a branch named "**hotfix-<version>**" (i.e. **hotfix-1.2.1** - created from the "**master**" branch). Contractors will be able to push code directly to this branch.
2. The contractor will work/deliver the source code on the created "**hotfix-<version>**" branch.
3. The contractor will create a "Merge Request" when ready to deliver.
4. EMSA will accept the Merge Request in order to merge the source code into the "**develop**" and "**master**" branches.