

[Template] Release Scope Overview

****Draft****

- [Planned deadlines](#)
- [Planned scope](#)
 - [Requirements](#)

Planned deadlines

Milestone	Planned deadline	Actual deadline	Description of the Milestone	Comments
M0	<i>date as initially planned</i>	<i>date as really committed</i>	<ul style="list-style-type: none"> • Blueprint proposal represented as a Jira EPIC ticket • Feature lead assigned to Jira Epic ticket - declaration of participation in the simultaneous release • The initial version of Increment Blueprint stored on github/xf-spec/Release folder, • The blueprint presented, discussed and approved by related committers • Release Scope Overview updated accordingly • The initial version of the Release Plan for the project presented • Dependencies between projects/components/areas identified 	<i>problems, decisions, reasons for the delay, etc</i>
M1			<ul style="list-style-type: none"> • All dependencies between existing functionality and requested change/feature discussed and agreed by PTL and related approvers (based on Modules/Committers) • Jira Epic ticket broken down into stories in related Jira project (backlog, plan) - presented and discussed on TWS meeting • The final version of Blueprints, Release Scope and Release Plans for participating projects discussed and approved by approvers 	
M2			<ul style="list-style-type: none"> • Technical design fully provided, agreed (documentation on the Confluence/github/xf-spec available) • Jira Epic ticket updated with link to technical design • Documentation started • Feature tests started • Feature/Functionality Freeze - Technical Design approved by relevant approvers (Modules/Committers) 	
M3			<ul style="list-style-type: none"> • External API available for beta-tests • Jira Epic ticket updated with information about API availability (documentation link - xf repo) • Documentation in progress • Tests in progress • API Freeze confirmed by Feature Lead (comment on Jira Epic ticket with repo /commit ID link) 	
M4			<ul style="list-style-type: none"> • all functionality and APIs available for testing • Jira Epic ticket updated with information about final API documentation, executed UI tests documentation • Documentation provided - confirmed by Documentation project PTL • Code freeze confirmed by Feature Lead (comment on Jira Epic ticket) - only bug fixing allowed 	
RC0			<ul style="list-style-type: none"> • Release candidates agreed, approved, tested • System tests conducted, quality confirmed by CI/CD PTL (should we have some QA? Security?) • Final documentation provided, reviewed, approved by Documentation PTL • Marketing information provided to Marketing Advisory Council - confirmed by Brandon Wick? • release branch cut off - branch stabilization from now on • Release Candidates freeze, confirmed by PTL and TSC(?) 	
Deployment			official deployment executed	

Planned scope

Project	Scope	Dependencies
<i>Project Name (link)</i>	<i>Overall description of the project's increment in the particular release</i>	<ul style="list-style-type: none"> • Dependent on: projects, on which this increment relay on • Affect: projects which depend on this increment

Requirements

1. Releases **MUST** have a web/wiki page that explains exactly what is in the release
 - a. The page **SHOULD** be automatically generated if at all possible
2. Releases **MUST** be in the form of docker images stored in DockerHub
 - a. Releases **MUST** be architected as microservices and be Kubernetes compatible
3. Releases **MUST** have a regular cadence
 - a. Ideally releases **WOULD** happen every month
 - b. The minimum cadence velocity **MUST** be 3 months, not 6 months
4. Releases **MUST** have some form of labels, tags, or branching scheme that allows for mapping included features and bug fixes into a CHANGELOG and RELEASE NOTES
 - a. See #1 above
 - b. The scheme **MUST** be documented here in the Wiki
5. There **MUST** be a variety of release types for various purposes to be determined; a starting point might be:
 - a. Experimental releases for nightly or high interval testing looking for build breakages
 - i. Lightly tested builds
 - b. Stable regular builds for doing downstream integration testing into commercial distributions
 - i. Standard testing
 - c. Production builds that can be used for official releases
 - i. Extensive scalability and performance testing (as possible)
6. **SHOULD** develop a scheme for LTS builds based on production builds (above)
 - a. QUESTION: what does an LTS build mean for an open source project that has no official support?