ETSI TS 103 673 V1.1.1 (2020-08)

SmartM2M;
SAREF Development Framework and Workflow,
Streamlining the Development of SAREF and its Extensions

Reference

DTS/SmartM2M-103673

Keywords

data, IoT, M2M, oneM2M, ontology, open source software, SAREF, semantic, software

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document defines the development framework for the SAREF ontology [1] and its extensions (referred to in a general way in the present document as SAREF projects) based on the ETSI forge. The development framework defines the different workflows to be followed for new SAREF project versions, SAREF project version development, and SAREF project release.

The present document also defines how SAREF project versions are specified and documented in the SAREF public forge. The accompanying SAREF pipeline software enables to automatically check the conformance of SAREF project versions with respect to the present document. The accompanying SAREF public portal enables to browse the documentation of SAREF project versions and engage the community of users.

The present document, the SAREF pipeline, and the SAREF public portal, will enable the SAREF developers to speed up the development of SAREF and its extensions as well as the SAREF community of users to actively contribute in the development. The present document is based on the requirements and guidelines defined in the associated ETSI TR 103 608 [i.1].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 103 264: "SmartM2M; Smart Applications; Reference Ontology and oneM2M Mapping".

[2] W3C Recommendation 11 December 2012: "OWL 2 Web Ontology Language, Document Overview (Second Edition)".

NOTE: Available online at https://www.w3.org/TR/owl2-overview/.

[3] W3C Recommendation 25 February 2014: "RDF 1.1 Turtle, Terse RDF Triple Language", Beckett David, Berners-Lee Tim, Prud'hommeaux Eric, and Carothers Gavin.

NOTE: Available online at https://www.w3.org/TR/turtle/.

[4] W3C Recommendation 11 December 2012: "OWL 2 Web Ontology Language, Direct Semantics (Second Edition)", Motik Boris, Patel-Schneider Peter F. and Cuenca Grau Bernardo.

NOTE: Available online at https://www.w3.org/TR/owl2-direct-semantics/.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 103 608: "SmartM2M; SAREF publication framework reinforcing the engagement of its community of users".

[i.2] ETSI TS 103 410-1: "SmartM2M; Extension to SAREF; Part 1: Energy Domain".

[i.3] ETSI TS 103 410-2: "SmartM2M; Extension to SAREF; Part 2: Environment Domain".

[i.4] ETSI TS 103 410-3: "SmartM2M; Extension to SAREF; Part 3: Building Domain".

[i.5] ETSI TS 103 410-4: "SmartM2M; Extension to SAREF; Part 4: Smart Cities Domain".

[i.6] ETSI TS 103 410-5: "SmartM2M; Extension to SAREF; Part 5: Industry and Manufacturing Domains".

[i.7] ETSI TS 103 410-6: "SmartM2M; Extension to SAREF; Part 6: Smart Agriculture and Food Chain Domain".

[i.8] ETSI TS 103 548: "SmartM2M; SAREF consolidation with new reference ontology patterns, based on the experience from the SEAS project".

[i.9] ETSI TS 103 410-7: "SmartM2M; Extension to SAREF; Part 7: Automotive Domain".

[i.10] ETSI TS 103 410-8: "SmartM2M; Extension to SAREF; Part 8: eHealth/Ageing-well Domain".

[i.11] ETSI TS 103 410-9: "SmartM2M; Extension to SAREF; Part 9: Wearables Domain".

[i.12] ETSI TS 103 410-10: "SmartM2M; Extension to SAREF; Part 10: Water Domain".

[i.13] ETSI TR 103 411 (V1.1.1): "SmartM2M; Smart Appliances; SAREF extension investigation".

[i.14] Poveda-Villalón, María, and Gómez-Pérez, Asunción, and Suárez-Figueroa, Mari Carmen. "OOPS! (Ontology Pitfall Scanner!): An on-line tool for ontology evaluation". International Journal on Semantic Web and Information Systems (IJSWIS) 10.2 (2014): 7-34.

[i.15] Garijo, Daniel, and Poveda-Villalón, María: "Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web". arXiv preprint arXiv:2003.13084 (2020).

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

ontology: formal specification of a conceptualization, used to explicitly capture the semantics of a certain reality

SAREF actor: role that a person can play when using or contributing to SAREF

SAREF core: versioned reference ontology for the IoT developed by ETSI SmartM2M and specified in ETSI TS 103 264 [1]

SAREF development framework: actors, software, and infrastructure that support the SAREF development workflows

SAREF development workflows: specification of a lifecycle of SAREF project versions, where SAREF actors interact in a codified manner. For example their creation, development, and release

SAREF extension: versioned ontology extending SAREF core for a certain domain, and specified in an ETSI TS

SAREF extension acronym: A SAREF extension is named **SAREF4ABCD**, where ABCD is the **SAREF extension acronym**, and is any sequence of four letters.

SAREF pipeline: software that can check the conformance of one or more SAREF project versions with respect to the present document, and generate part of the SAREF public portal

NOTE: The SAREF pipeline may be run manually by a SAREF actor, or automatically by a continuous integration and continuous deployment service.

SAREF project: SAREF core, or any SAREF extension

SAREF project version: A SAREF project has several **versions**, each being numbered by a **version number** v*x*.*y*.*z*. The first number x is the **major version**. The second number y is the **minor version**. The third number z is the **patch version**.

NOTE: The version numbering system for SAREF projects is different from the ETSI version numbering system.

SAREF project release: SAREF project version whose documentation is exposed on the SAREF public portal

SAREF project repository: git repository that consists of **git branches**, which consist of sequences of **git commits**

NOTE: Git commits have a unique identifier. There are four types of branches in a SAREF project repository: **issue branches**, **develop branches**, **pre-release branches**, and **release branches**:

- issue branches are named `issue-w`, where w is an issue number of the SAREF project;
- develop branches are named `develop-vx.y.z`, where v*x*.*y*.*z* is a SAREF project version number;
- pre-release branches are named `prerelease-vx.y.z`, where v*x*.*y*.*z* is a SAREF project version number;
- release branches are named `release-vx.y.z`, where v*x*.*y*.*z* is a SAREF project version number.

SAREF project sources: git repository called **the SAREF project repository**, an associated **public issue tracker**, and a **continuous integration and continuous deployment service**

SAREF public forge: software development and git-based source code web platform managed by the ETSI Secretariat

NOTE: The SAREF public forge contains the **SAREF projects sources**. The entry point to the SAREF public forge is https://saref.etsi.org/sources/.

SAREF public portal: web server hosted on an **ETSI server** and managed by the ETSI Secretariat

NOTE: It exposes the documentation of SAREF and the SAREF projects to the public. The entry point to the SAREF public portal is https://saref.etsi.org/. The SAREF public portal contains the documentation of all the SAREF projects for different **SAREF project releases**.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

CD	Continuous Deployment
CI	Continuous Integration
CSV	Comma Separated Values
DL	Description Logics
EUREKA	European Research Coordination Agency
HTML	Hyper Text Markup Language
IoT	Internet of Things
IPR	Intellectual Property Right
IRI	Internationalized Resource Identifier
ITEA	Information Technology for European Advancement
OWL	Web Ontology Language
RDF	Resource Description Framework
SAREF	Smart Applications REFerence ontology
SEAS	Smart Energy Aware Systems
STF	Specialist Task Force
TR	Technical Report
TS	Technical Specification
URL	Universal Resource Locator
UTF-8	Unicode Transformation Format (the 8-bit form)
W3C®	World Wide Web Consortium

4 Introduction

SAREF consists of **SAREF projects**, including **SAREF core**, and many **SAREF extensions**. SAREF core [1] is a versioned reference ontology for the IoT developed by ETSI SmartM2M in close interaction with the industry. SAREF contains core concepts that are common to several IoT domains and, to be able to handle specific data elements for a certain domain, dedicated extensions of SAREF have been created. At the date of publication of the present document, the set of SAREF extensions include SAREF4ENER [i.2], SAREF4ENVI [i.3], SAREF4BLDG [i.4] and SAREF4CITY [i.5], SAREF4INMA [i.6], SAREF4AGRI [i.7], SAREF4SYST [i.8], SAREF4AUTO [i.9], SAREF4EHAW [i.10], SAREF4WEAR [i.11], SAREF4WATR [i.12]. Each domain can have one or more extensions, depending on the complexity of the domain. As a core ontology, SAREF core serves as the means to connect the SAREF extensions in different domains. The earlier document ETSI TR 103 411 [i.13] specifies the rationale and methodology used to create, publish and maintain the SAREF extensions. ETSI TR 103 608 [i.1] specifies requirements and guidelines for the SAREF development and publication framework.

The value of SAREF is strongly correlated with the size of its community of users; therefore the SAREF ontologies are available and documented on the Web. As such, SAREF users and the industry actors can be attracted to SAREF with clear documentation and a clear indication about how to provide their input and the kind of input that they can provide.

The ETSI members that contribute to SAREF are able to get benefit from feedback coming from its open community of industrial users, to better plan new evolution of the current and future extensions, and to reduce the costs of developing these extensions. That being said, the development and monitoring of SAREF lies in ETSI's hands to ensure that high quality standards are met, and users that provide feedback have to understand the implication in terms of IPR. The publication and/or use of such feedback has to therefore be controlled by ETSI, but the possibility to provide feedback will be open to the world.

The present document is the technical specification of the SAREF development framework, as described below and in the following clauses. The SAREF development framework consists of the actors, software, and infrastructure that support the SAREF development workflows.

The SAREF actors and their use cases are specified in clause 5 of the present document.

The SAREF development workflows are specified in clause 6, clause 7 and clause 8 of the present document.

The SAREF project version specification and documentation in the SAREF project sources are specified in clause 9 of the present document.

The present document has been developed in the context of the STF 578 (https://portal.etsi.org/STF/STFs/STFHomePages/STF578.aspx), which followed the STF 556 (https://portal.etsi.org/STF/STFs/STFHomePages/STF556.aspx). STF 578 was established with the goal to consolidate SAREF and its community of industrial users based on the experience of the EUREKA ITEA 12004 SEAS (Smart Energy Aware Systems) project. The present document specifies the SAREF publication framework to reinforce the engagement of its community of users and to enable them to implement solutions with SAREF faster.

5 Actors of the SAREF development framework

5.1 Introduction

The following list shows the different SAREF actors and the use cases that each actor may carry out in the SAREF public forge and the SAREF public portal. The SAREF actors are organized into the following categories: Steering actors, Development actors, and Community actors:

- Steering actors include Steering Board members and Technical Board members.
- Development actors include Project leaders and Ontology developers.
- Community actors include Contributors and Ontology users.

This list has been inspired by the list on ETSI TR 103 608 [i.1], clause 6, but has been simplified and grounded on the permission rights that can be implemented on the actual ontology development platform in the SAREF public forge.

One person may play the role of many actors at a time, and potentially different roles for different SAREF projects.

An ontology user may start contributing to some SAREF project, and become a Contributor.

Experts may be nominated by SmartM2M to become a development actor for some SAREF project.

The SAREF public forge allows defining users with the following roles: Guest, Reporter, Developer, Maintainer, and Owner; each with its own permissions.

The content in the SAREF public portal shall be open to anyone, so no specific roles are defined for it.

5.2 Steering actors

Steering Board member: A Steering Board member belongs to the group of persons in charge of steering the SAREF development, including SAREF core and SAREF extensions. The community involvement and the underlying infrastructure.

Steering Board members may have at least the role of Reporter in the SAREF public forge.

The Steering Board is composed by the SmartM2M Chairman, Vice-Chairman, and Technical Officer, and experts nominated by SmartM2M.

The use cases specific to the Steering Board member are the following:

- Manage the workflow of SAREF project proposals, as specified in clause 6.1.
- Review the SAREF project proposals, as specified in clause 6.3.
- Manage the workflow of SAREF project releases, as specified in clause 8.1.
- Review the SAREF project releases, as specified in clause 8.3.
- Identify and resolve overlaps between SAREF projects.
- Decide SAREF project release creators and SAREF project release contributors list, as specified in clauses 9.4.3.3 and 9.6.1.

Technical Board member: A Technical Board member belongs to the group of persons in charge of maintaining the SAREF public forge and the SAREF public portal.

Technical Board members may have at least the role of Maintainer in the SAREF public forge.

The Technical Board is composed by ETSI Secretariat and experts nominated by SmartM2M.

The use cases specific to the Technical Board member are the following:

- Setting up a SAREF project, as specified in clause 6.4.
- Setting up continuous integration tests, as specified in clause 7.6.
- Publish SAREF project releases, as specified in clause 8.3.

5.3 Development actors

Project leader: A project leader is the person in charge of the SAREF project who carries out the project management tasks.

A project leader shall have experience in ontology development projects.

Project leaders may have at least the role of Maintainer in the ETSI public forge.

There should exist at least one project leader for SAREF core and at least one for each SAREF extension.

The use cases specific to the project leader are the following:

- Set up the SAREF project, as specified in clause 6.4.
- Add the SAREF project to the portal, as specified in clause 6.5.
- Manage the ontology development workflow, as specified in clause 7.1.
- Review change requests, as specified in clause 7.3.
- Review change request implementations, as specified in clause 7.5.
- Manage the project release workflow, as specified in clause 8.1.
- Prepare the SAREF project release, as specified in clause 8.2.
- Trigger the SAREF project release, as specified in clause 8.3.

Ontology developer: An ontology developer is a member of the ontology development team who has high knowledge about ontology development and rights to modify the ontology and interact in the development cycle. Ontology developers create and modify the different development artefacts, provide new requirements to the ontology and validate whether they are satisfied or not when implemented, and have decision rights about what contributions can be included in the ontology.

Ontology developers may have at least the role of Developer in the ETSI public forge.

There will exist different teams of ontology developers for SAREF core and for each SAREF extension.

The use cases of the ontology developer are the following:

- Contribute to the ontology development workflow, as specified in clause 7.1.
- Propose change requests, as specified in clause 7.2.
- Review change requests, as specified in clause 7.3.
- Implement change requests, as specified in clause 7.4.
- Review change request implementations, as specified in clause 7.5.

- Trigger the SAREF continuous integration tests, as specified in clause 7.6.

5.4 Community actors

Contributor: A contributor is a person knowledgeable about the ontology domain and proposes contributions.

Contributors have an account on the SAREF public forge.

The role of contributor is not assigned beforehand, it is obtained when submitting some contribution.

The use cases of the contributor are the following:

- Contribute to the workflow of SAREF project proposals, as specified in clause 6.1.
- Propose a new SAREF project, as specified in clause 6.2.
- Contribute to the ontology development workflow, as specified in clause 7.1.
- Propose a new change request, as specified in clause 7.2.
- Review the SAREF project proposals, as specified in clause 6.3.
- Manage the workflow of SAREF project releases, as specified in clause 8.1.
- Review the SAREF project releases, as specified in clause 8.3.
- Review SAREF project proposals.
- Propose change request, as specified in clause 6.2.
- Report change request.
- Register as user of the ontology.
- Report usage of the ontology.
- Subscribe to notifications and news about a SAREF project.

Ontology user: An ontology user is someone interested in any of the SAREF projects or in proposing a new SAREF project.

Ontology users do not have an account on the SAREF public forge.

Ontology users include potential end users of the ontology, software developers that will make use of the ontology within their applications, industry stakeholders, researchers, domain experts, etc.

The use cases of the ontology user are the following:

- Access ontology development artefacts (ontology code, documentation, tests, etc.).
- Access user oriented documentation (tutorials, guidelines, etc.).
- Access available conformance results for the standards.
- Ontology suggestion based on ontological requirements.
- Search ontology terms in SAREF core and SAREF extensions.
- Access ontology users list.
- Access SAREF project metrics (includes contributors, requirements, pipeline report, users, analytics, etc.).

6 New SAREF project version workflow

6.1 Workflow

6.1.0 Introduction

The **new SAREF project version workflow** supports the creation of **new SAREF project versions proposals** from the SAREF community of users. New SAREF project versions may be new versions of SAREF core, new versions of existing SAREF extensions, or initial versions (V1.1.1) of new SAREF extensions.

Any contributors may propose new SAREF project versions. Steering Board members should review new SAREF project version proposals, and interact with contributors to clarify the proposals. The Steering Board is responsible for approving or dismissing proposals. In case a proposal is accepted, the Steering Board is responsible for the selection of the project leader and the team of ontology developers. The Technical Board and the project leader are responsible for setting up the project version development infrastructure and publishing the project to the portal.

New SAREF project versions proposals may go through different states, which can be one of the following:

- **Submitted**. This state is used for new SAREF project version proposals that have been submitted but the decision has not been taken yet on whether to approve them or to dismiss them.
- **Needs Clarification**. This state is used for new SAREF project version proposals that do not clearly describe the project itself.
- **Approved**. This state is used for new SAREF project version proposals that have been approved for creation.
- **Infrastructure Ready**. This state is used for new SAREF project version proposals whose sources on the SAREF public forge have been set up.
- **Closed**. This state is used for new SAREF project version proposals that have been dismissed, or that have been approved, set up, and added to the portal.

Figure 1 depicts the different states of a new SAREF project version proposal and the transitions among them. For all the transitions, it is defined: the conditions that are required for the transition to occur (CONDITION); the actions that initiated the transition (ACTION); and the output of the transition (OUTPUT). The next clauses describe each of these transitions.

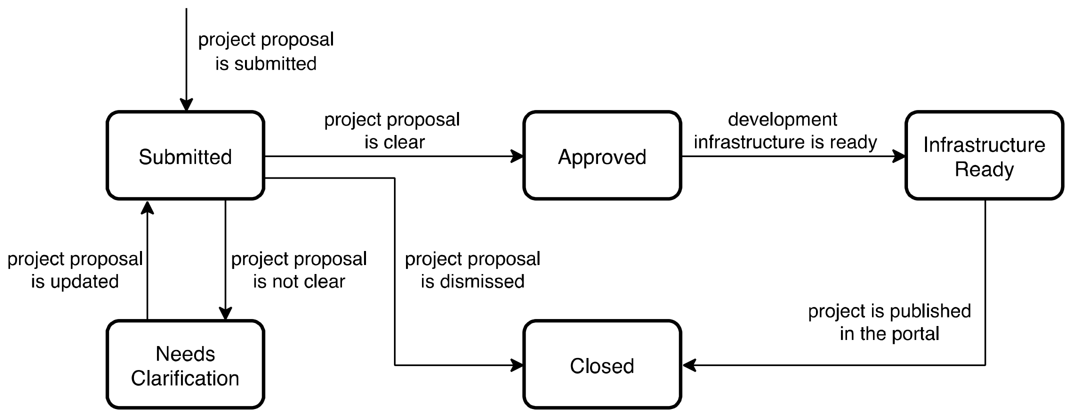
project proposal is submitted
Submitted
project proposal is clear
Approved
development infrastructure is ready
Infrastructure Ready
project proposal is updated
project proposal is not clear
project proposal is dismissed
project is published in the portal
Needs Clarification
Closed

Figure 1: Different states of a new SAREF project version proposal and the transitions among them

6.1.1 New SAREF project version proposal is submitted

CONDITION: A contributor wants to request a new SAREF project version.

ACTION: The contributor creates an issue in the `saref-portal-static` issue tracker describing the new SAREF project version. This issue is called the **proposal issue**.
Clause 6.2 specifies how to submit new SAREF project version proposals.

OUTPUT: The proposal issue is created in the `saref-portal-static` project with label "Submitted".

6.1.2 New SAREF project version proposal is not clear

CONDITION: Some of the details of the new SAREF project version proposal are not clear from its description.

ACTION: The Steering Board reviews the proposal issue and asks for clarifications.
Clause 6.3 specifies the reviewing process for new SAREF project version proposals.

OUTPUT: One or more comments in the proposal issue indicating what needs to be clarified.
The label of the proposal issue should be updated to "Needs clarification".

6.1.3 New SAREF project version proposal is updated

CONDITION: The new SAREF project version proposal is updated addressing the petitions raised by the Steering Board.

ACTION: A contributor updates the proposal issue.

OUTPUT: A new comment in the proposal issue with clarifications.
The label of the proposal issue should be updated to "Submitted".

6.1.4 New SAREF project version proposal is dismissed

CONDITION: The Steering Board decides that the new SAREF project version proposal will not be accepted.

ACTION: The Steering Board reviews the proposal issue and takes the decision of not accepting the proposal.
Clause 6.3 specifies the reviewing process for new SAREF project version proposals.

OUTPUT: The proposal issue is closed.

6.1.5 New SAREF project version proposal is clear

CONDITION: The Steering Board decides that the new SAREF project version proposal will be accepted.

ACTION: The Steering Board reviews the "Submitted" proposal issue and takes the decision of accepting the proposal.
Clause 6.3 specifies the reviewing process for new SAREF project version proposals.

OUTPUT: The label of the project proposal issue should be updated to "Approved".

6.1.6 Development infrastructure is ready

CONDITION: The ontology development infrastructure required for the new SAREF project proposal is deployed and configured.

ACTION: If it is a new SAREF project, the Technical Board creates the sources of the SAREF project on the SAREF public forge. Clause 6.4 specifies how to set up the sources of a new SAREF project.
The Steering Board elects a project leader, who is added as a Maintainer to the sources of the SAREF project.
The Technical Board and the project leader set up the sources of the new SAREF project version.
Clause 6.4 specifies how to set up the sources of a new SAREF project version.

OUTPUT: The label of the proposal issue should be updated to "Infrastructure ready".

6.1.7 New SAREF project version is published in the portal

CONDITION: The new SAREF project is ready for development.

ACTION: The project leader and steering board add the new SAREF project version to the portal.
Clause 6.5 specifies how to add the new SAREF project version to the portal.

OUTPUT: A new release of the ontology portal is published; the project proposal issue is closed.

6.2 Proposing a new SAREF project version

In order to propose a new SAREF project version, a contributor shall create a new issue in the issue tracker of the `saref-portal-static` project. The issue shall include a description of the new SAREF project version proposal. The issue should be labelled as "Submitted".

If the SAREF project version concerns an existing SAREF project, then the description:

- Shall contain information about the proposer and its organization.
- Shall contain the name and acronym of the SAREF project.
- Shall describe the purpose and justification.
- Shall contain a proposed target SAREF project version number.
- Should contain a timeline for the SAREF project version.
- Should contain a list of the members of the ontology development team (including the project leader).
- May contain other relevant information.

If the SAREF project version concerns a new SAREF project, then the description:

- Shall contain information about the proposer and its organization.
- Shall contain the name and acronym of the SAREF project.
- Shall describe the purpose and justification.
- Shall contain the scope of the SAREF project.
- Should describe the relation with existing extensions.
- Should contain a timeline for the SAREF project.
- Should contain a list of the members of the ontology development team (including the project leader).
- May contain other relevant information.

6.3 Reviewing new SAREF project version proposals

The Steering Board should review periodically the suitability of every SAREF project version proposals.

The board may ask for clarifications to the user proposing the new SAREF project version.

The proposals may be presented to SmartM2M.

The decision to dismiss or accept a new SAREF project version proposal is taken by the Steering Board.

6.4 Setting up an SAREF project

6.4.0 Introduction

The Technical Board and the Project Leader should set up the ontology development infrastructure for a new SAREF project version.

If the SAREF project version concerns a new SAREF project, then:

- The SAREF project shall be created in the SAREF public forge, as specified in clause 6.4.1.
- The SAREF project development team shall be set up, as specified in clause 6.4.2.
- The SAREF project version development branch shall be created, as specified in clause 6.4.3.
- The CI/CD service shall be configured, as specified in clause 6.4.4.

If the SAREF project version concerns an existing SAREF project, then:

- The SAREF project development team may be updated, as specified in clause 6.4.2.
- The SAREF project version development branch shall be created, as specified in clause 6.4.3.
- The CI/CD service shall be configured, as specified in clause 6.4.4.

6.4.1 Creating a new SAREF project in the SAREF public forge

The source of a SAREF project consists of a git repository called the **SAREF project repository**, an associated **public issue tracker**, and a **continuous integration and continuous deployment service**.

The sources of SAREF core shall be in a project named `saref-core` on the SAREF public forge, accessible at URL https://saref.etsi.org/sources/saref-core/.

The sources of a SAREF extension SAREF4ABCD, with acronym ABCD, shall be in a project named `saref4abcd` on the SAREF public forge, accessible at URL https://saref.etsi.org/sources/saref4abcd/.

The SAREF project shall include a description similar to that of others and the visibility level shall be "*Public*".

Branches whose name matches the pattern `develop-*` shall be protected such that only Maintainers and Developers are allowed to merge, and no one is allowed to push.

Branches whose name matches the pattern `prerelease-*` shall be protected such that no one is allowed to merge, and only Maintainers are allowed to push.

Branches whose name matches the pattern `release-*` shall be protected such that no one is allowed to merge and push.

There shall be no master branch. The "default branch" of the repository is specified by clause 6.4.2.

6.4.2 Setting up the ontology development team

All the users that are involved in the SAREF project shall have a user account in the SAREF public forge.

The following roles should be granted to such users in the project:

- Members of the Technical Board should have the role of at least Maintainer.
- Members of the Steering Board should have the role of at least Reporter.
- The Project leader should have the role of at least Maintainer.
- Ontology developers should have the role of at least Developer.

6.4.3 Preparing the project version development branch

Accepted SAREF project version proposals shall be assigned a target project version number v*x.y.z* by SmartM2M.

If the SAREF project version concerns a new SAREF project, then a new branch named `develop-vx.y.z` shall be created after the target project version number v*x.y.z*, and it shall be defined as the "default branch" of the repository. The initial structure of the repository shall be set as specified in clause 9.2.

If the SAREF project version concerns an existing SAREF project, then the branch `release-vx'.y'.z'` corresponding to the greatest version number v*x'.y'.z'* lower than v*x.y.z* shall be forked and given the name `develop-vx.y.z`. The initial structure of the repository shall be set as specified in clause 9.2.

6.4.4 Configuring continuous integration and continuous deployment

The SAREF project should be configured in order to enable the continuous integration and continuous deployment activities.
When a commit is pushed to any branch, a pipeline should be triggered to check the contribution.
This pipeline should run the **SAREF pipeline program** whose sources are located at the URL https://forge.etsi.org/rep/SAREF/saref-pipeline.

The SAREF pipeline program may be run in **develop mode** or **release mode**.
When run in develop mode, the SAREF pipeline program should check that the repository satisfies the specifications defined in clause 9 of the present document.
When run in release mode, the SAREF pipeline program shall check that the repository satisfies the specifications defined in clause 9 of the present document.

The SAREF pipeline program shall be run in develop mode on branches whose name match patterns `issue-*` and `develop-*`.

The SAREF pipeline program shall be run in release mode on branches whose name match patterns `prerelease-*` and `release-*`.

6.5 Adding a SAREF project version to the portal

The `saref-portal-static` project on the ETSI public forge contains the static content of the ontology portal. It contains different files that shall be created or updated to include the new extension:

- The file named `extensions.html` contains an overview of all the SAREF project versions. The new SAREF project version shall be added to this file in a way similar than for the other SAREF project versions.
- A directory named after the SAREF project version shall be created for the new project version. A file named `index.html` shall be created inside this directory stating that the new extension is currently under development and including a summary of the new ontology proposal.

EXAMPLE 1: For SAREF core version v4.5.6, there should be a directory `core/v4.5.6` that contains a document named `index.html`.

EXAMPLE 2: For SAREF extension SAREF4ABCD version v1.2.3, there should be a directory `saref4abcd/v1.2.3` that contains a document named `index.html`.

7 SAREF project version development workflow

7.1 Workflow

7.1.0 Introduction

The **SAREF project version development workflow** supports the development of **SAREF project versions** from the SAREF community of users. SAREF project versions may be new versions of SAREF core, new versions of existing SAREF extensions, or initial versions (V1.1.1) of new SAREF extensions.

The SAREF project version development workflow is articulated around the use of issues in the corresponding SAREF project issue tracker on the ETSI public forge. This enables not only to have a single point of interaction for development but also to keep track of the development activity and discussions.

Any update in a SAREF project version should be made through a **change request**, which is posted as an issue in the corresponding repository of the ETSI public forge, and is assigned an **issue number**. This includes change requests related to new ontology requirements, to defects or improvements in the ontology specification, in the ontology tests, ontology examples, or ontology documentation.

Any contributor may create a new change request, or review and discuss existing change requests. Ontology developers, should review change requests, propose and review implementations of accepted change requests. The Steering Board should review change requests. The Project leader is responsible for ensuring that change requests are approved by SmartM2M, and that change request implementations do implement the requested change.

Change requests that are related to the project version development should be tagged with their corresponding state, which can be one of the following:

- **Submitted:** This state is used for change requests that have been submitted but the decision has not been taken yet on whether to implement their changes or to dismiss them.
- **Needs Clarification:** This state is used for change requests that do not clearly describe what should be implemented.
- **Propose Closing:** This state is used for change requests that will not be dealt with, either because the change request has been dismissed or because their implementation has been dismissed.
- **Approved:** This state is used for change requests that have been approved for implementation.
- **Needs Implementation:** This state is used for change requests that have been approved for implementation and it is clear how to implement them.
- **Needs Discussion:** This state is used for change requests that have been approved for implementation but it is not clear how to implement them.
- **Work in Progress:** This state is used for change requests that are being implemented.
- **Implementation Available:** This state is used for change requests that have a full implementation available.
- **Closed:** This state is used for change requests that have been closed.

Figure 2 depicts the different states of a change request and the transitions among them. For all the transitions, the following clauses define the conditions that should be met for the transition to occur (CONDITION); the actions that should initiate the transition (ACTION); and what should be the output of the transition (OUTPUT).

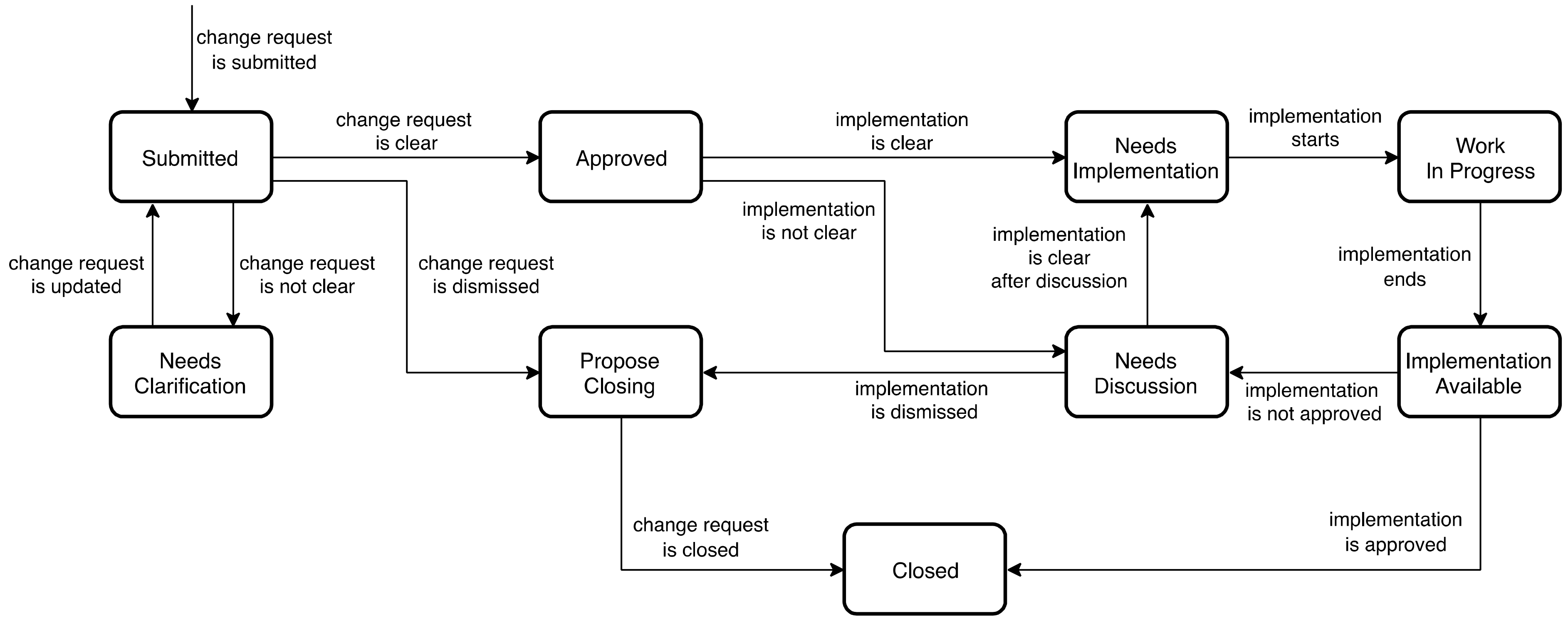
change request is submitted
Submitted
change request is clear
Approved
implementation is clear
Needs Implementation
implementation starts
Work In Progress
change request is updated
change request is not clear
change request is dismissed
implementation is not clear
implementation is clear after discussion
implementation ends
Needs Clarification
Propose Closing
Needs Discussion
Implementation Available
implementation is dismissed
implementation is not approved
change request is closed
Closed
implementation is approved

Figure 2: Different states of a change request and the transitions among them

7.1.1 Change request is submitted

CONDITION: A contributor wants to request a new change to the SAREF project version.

ACTION: The contributor creates an issue in the SAREF project issue tracker describing the change request. This issue is called the **request issue**, and is identified by its **issue number**.
The change request may concern the ontology requirements as specified in clause 9.3.
The change request may concern the ontology specification as specified in clause 9.4.
The change request may concern the ontology tests as specified in clause 9.5.
The change request may concern the ontology examples as specified in clause 9.6.
The change request may concern the ontology documentation as specified in clause 9.7.
Clause 7.2 specifies how to submit new change requests.

OUTPUT: The change request issue is created in the SAREF project and has label "Submitted".

7.1.2 Change request is not clear

CONDITION: Some of the details of the change request are not clear from its description.

ACTION: The project members review the request issue and ask for clarification.
Clause 7.3 specifies the reviewing process for change requests.

OUTPUT: One or more comments in the request issue indicating what needs to be clarified.
The label of the request issue should be updated to "Needs clarification".

7.1.3 Change request is updated

CONDITION: The change request is updated addressing the petitions raised by the project members.

ACTION: A contributor updates the request issue.

OUTPUT: A new comment in the request issue with clarifications.
The label of the request issue should be updated to "Submitted".

7.1.4 Change request is dismissed

CONDITION: There is consensus in dismissing the change request.

ACTION: The project leader reviews the request issue, and takes the decision of proposing to close the request.
Clause 7.3 specifies the reviewing process for change requests.

OUTPUT: The label of the request issue should be updated to "Propose closing".

7.1.5 Change request is clear

CONDITION: There is consensus in accepting the change request.

ACTION: The project leader reviews the request issue, ensures that the change request is approved by SmartM2M, and takes the decision of accepting the request.
Clause 7.3 specifies the reviewing process for change requests.

OUTPUT: The label of the request issue should be updated to "Approved".
An additional label should be added to the issue indicating whether it is an "Enhancement" or a "Bug".

7.1.6 Change request is closed

CONDITION: There is consensus in closing the change request.

ACTION: The project leader ensures that the request closure is approved by SmartM2M.
Clause 7.3 specifies the reviewing process for change requests.

OUTPUT: The request issue shall be closed.

7.1.7 Implementation is clear

CONDITION: There is consensus on how to implement the change request.

ACTION: Project members should review the request issue.
The implementation plan may include updates to the ontology requirements as specified in clause 9.3.
The implementation plan may include updates to the ontology specification as specified in clause 9.4.
The implementation plan may include updates to the ontology tests as specified in clause 9.5.
The implementation plan may include updates to the ontology examples as specified in clause 9.6.
The implementation plan may include updates to the ontology documentation as specified in clause 9.7.

OUTPUT: The label of the issue should be updated to "Needs implementation".
The request issue should be updated with a description of how to make the implementation.

7.1.8 Implementation is not clear

CONDITION: There is no consensus on how to implement the change request.

ACTION: Project members should review the request issue.

OUTPUT: The label of the issue should be updated to "Needs discussion".
The request issue should be updated with a description of what needs to be updated.

7.1.9 Implementation is dismissed

CONDITION: There is consensus that the implementation of the change request will not be performed.

ACTION: Project members should review the request issue.

OUTPUT: The label of the issue should be updated to "Propose closing".
The request issue should be updated with the reasons for dismissing the implementation.

7.1.10 Implementation is clear after discussion

CONDITION: There is consensus on how to implement the change request.

ACTION: Project members should review the request issue.
If the change request comes from a contributor, he or she may be included in the discussion.

OUTPUT: The label of the issue should be updated to "Needs implementation".
The request issue should be updated with a description of how to make the implementation.

7.1.11 Implementation starts

CONDITION: An ontology developer or contributor starts working on the implementation of the change request.

ACTION: The ontology developer selects a "Needs implementation" change request to implement; alternatively, the contributor selects a "Needs implementation" change request raised by him/her to implement.
The implementation shall happen in an **issue branch** named after the request issue number, as specified in clause 7.4.

OUTPUT: The label of the issue should be updated to "Work in progress".

7.1.12 Implementation ends

CONDITION: An ontology developer or contributor finishes working on the implementation of the change request.

ACTION: The ontology developer or contributor should implement what is expected in the plan for implementation.
A new merge request should be created from the issue branch to the development branch, as specified in clause 7.4.

OUTPUT: The label of the issue should be updated to "Implementation available".
A new merge request should be created as specified in clause 7.4.

7.1.13 Implementation is not approved

CONDITION: There is no consensus that the implementation is approved.

ACTION: Project members should review the available implementation against the implementation plan in the request issue. They should review the results of the continuous integration tests.

OUTPUT: The label of the issue should be updated to "Needs discussion".
The issue should be updated with a description of what problems the implementation has.

7.1.14 Implementation is approved

CONDITION: There is a consensus to approve the implementation.

ACTION: Project members should review the available implementation against the implementation plan in the request issue. They should review the results of the continuous integration tests.

OUTPUT: The merge request shall be merged. The issue shall be closed. The issue branch should be deleted.

7.2 Proposing a new change request

Any contributor may create a new change request to a SAREF project. The contributor shall do so by creating a new issue describing the change request in the corresponding repository of the ETSI public forge.

This issue is called the **request issue**, and is identified by its **issue number**.
The request issue should be labelled as "Submitted" and should include a description of the new change request.

The change request may concern the ontology requirements as specified in clause 9.3.
The change request may concern the ontology specification as specified in clause 9.4.
The change request may concern the ontology tests as specified in clause 9.5.
The change request may concern the ontology examples as specified in clause 9.6.
The change request may concern the ontology documentation as specified in clause 9.7.

The description of a new change request:

- Shall describe the current situation that requires implementing the change request.
- Should contain a proposal on how to implement the change request.
- May contain other relevant information.

7.3 Reviewing new change requests

Any contributor may review and discuss new request issues.

Periodically, the project members should review the submitted request issues.
If needed, the project members may ask for clarifications to the user proposing the new request issue.

The project leader should review the request issue and should make a proposal for either closing the request issue, or approving the request issue.
The project leader is responsible for bringing these proposals to SmartM2M and ensuring that there is sufficient consensus in SmartM2M before closing or approving the request issue.

The request issue should be updated with a description of how to make the implementation.

Any contributor may propose, review, and discuss, implementation plans for change requests.

The implementation plan may include updates to the ontology requirements as specified in clause 9.3.
The implementation plan may include updates to the ontology specification as specified in clause 9.4.

The implementation plan may include updates to the ontology tests as specified in clause 9.5.
The implementation plan may include updates to the ontology examples as specified in clause 9.6.
The implementation plan may include updates to the ontology documentation as specified in clause 9.7.

Periodically, the project members should review the implementation plans for change requests.
If needed, the project members may ask for clarifications to the user proposing the implementation plan.

The project leader should review the request issue, and should take the decision of proposing to close the request issue, or approving the implementation plan.
The project leader is responsible for ensuring that there is sufficient consensus in SmartM2M before closing the request issue, or accepting an implementation plan for a change request.

7.4 Implementing change requests

Any ontology developer or contributor may start working on the implementation of a change request.

A change request shall be implemented in its **issue branch**.
The issue branch shall be created by forking the development branch of the SAREF project version.
The name of the issue branch should begin with `issue-`, and end with the request issue number.

EXAMPLE: The issue branch for a change request with request issue number 43 in SAREF project version SAREF4ABCD version v1.2.3 shall be created by forking the development branch `develop-v1.2.3`, and should be named `issue-43`.

A change request implementation should contain what is expected in the implementation plan.

When a change request implementation is ready to be reviewed, a request to merge into the development branch of the SAREF project version shall be issued.
This **merge request** shall reference the original request issue that is aimed to be solved.
The request issue should be labelled with "Implementation available".

The description of the merge request should include a link to the original request issue. It should not have assignees.

7.5 Reviewing change request implementations

Any contributor may review and discuss change request implementation in their corresponding merge request.

Periodically, the project members should review the available implementation against the implementation plan in the request issue. They should review the results of the continuous integration tests, which should complete successfully.

If needed, the project members may request some modification in the issue branch to the user proposing the change request implementation.

The project leader should review the merge request, and should take the decision of approving or rejecting the merge request.
The project leader is responsible for ensuring that there is sufficient consensus in SmartM2M before approving or rejecting the merge request.

7.6 Continuous integration tests

When a commit is pushed to a develop branch or to an issue branch of the SAREF extension repository, the SAREF pipeline program should be run in develop model to check the contribution, as specified in clause 6.4.4.

8 SAREF project release workflow

8.1 Workflow

8.1.0 Introduction

The **SAREF project release workflow** is about SmartM2M to accept and trigger the generation of **SAREF project releases**. SAREF project versions may be new versions of SAREF core, new versions of existing SAREF extensions, or initial versions (V1.1.1) of new SAREF extensions.

The project leader is responsible for preparing the SAREF project release.
The Steering Board is responsible for accepting to trigger the SAREF project release.
The project leader and the Technical Board are responsible for triggering the SAREF project release.

Ontology releases will go through different states, which can be one of the following:

- **Prepared:** This state is used for ontology releases that have been prepared to be published.
- **Needs Improvement:** This state is used for ontology releases that need to be improved before being published.
- **Closed:** This state is used for ontology releases that have been published in the ontology portal.

Figure 3 depicts the different states of an ontology release and the transitions among them. For all the transitions, it is defined: the conditions that are required for the transition to occur (CONDITION); the actions that initiated the transition (ACTION); and the output of the transition (OUTPUT).

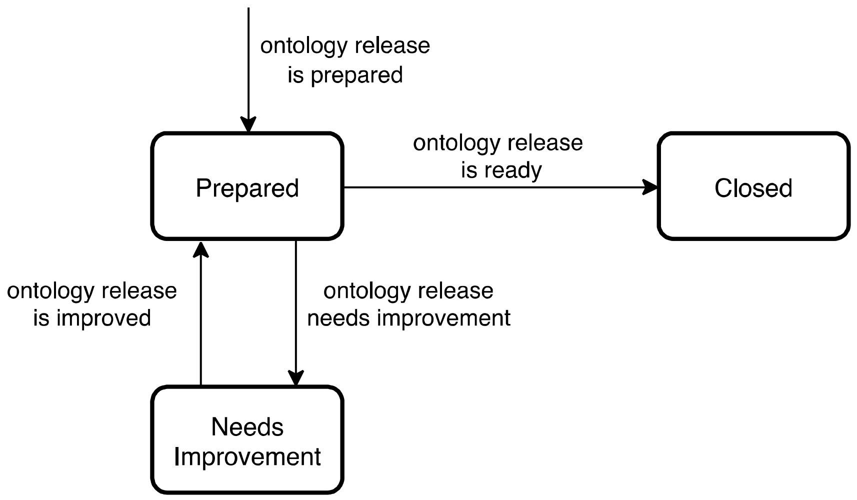
ontology release is prepared
Prepared
ontology release is ready
Closed
ontology release is improved
ontology release needs improvement
Needs Improvement

Figure 3: Different states of a SAREF project release and the transitions among them

8.1.1 SAREF Project release is prepared

CONDITION: The project leader wants to prepare a new SAREF project release.

ACTION: The project leader creates the pre-release branch for the SAREF project version by forking the development branch.
Clause 8.2 specifies how to prepare a SAREF project release.

OUTPUT: The pre-release branch is created for the SAREF project version. The SAREF pipeline software is run in release mode during the CI/CD pipeline.

8.1.2 SAREF Project release needs improvement

CONDITION: A new change request is accepted for this SAREF project version, or the SAREF pipeline software triggers errors when run in release mode.

ACTION: If a new change request is accepted, the project leader waits until the change request is closed, and merges the updated develop branch into the pre-release branch.
If the SAREF pipeline software triggers errors when run in release mode, the project leader solves the errors in the pre-release branch. He should merge the pre-release branch into the develop branch.
Clause 8.2 specifies how to prepare a SAREF project release.

OUTPUT: The SAREF pipeline software is run in release mode during the CI/CD pipeline.

8.1.3 Project release is improved

CONDITION: The pre-release branch is synced with the development branch, no change request is open for this SAREF project version.

ACTION: The project leader requests the Steering Board to approve the triggering of the release.
Clause 8.3 specifies how to review a SAREF project release.

OUTPUT: The SAREF pipeline software is run in release mode during the CI/CD pipeline.
The project leader requests the Steering Board to approve the triggering of the release.

8.1.4 Project release is ready

CONDITION: The SAREF pipeline software triggers no error when run in release mode. The Steering Board accepts to trigger the SAREF project release.

ACTION: The project leader and the Technical Board are responsible for triggering the release.
Clause 8.4 specifies how to publish SAREF project releases.

OUTPUT: The release branch shall be created. The SAREF public portal shall be updated.

8.2 Preparing a project release

8.2.0 Introduction

The project leader is responsible for preparing the SAREF project release. This requires to perform the following actions:

- To prepare the project release documentation.
- To prepare the pre-release branch of the SAREF project version.

The next clauses describe these steps in detail.

8.2.1 Preparing the project release documentation

The project leader should check that the SAREF extension is well described in the `saref-portal-static` project of the ETSI public forge, which hosts the static pages of the ontology portal.

The file named `extensions.html` shall contain a description of the SAREF project version in a way similar than the other SAREF project versions.

The directory named after the SAREF project version shall exist, and contain only a document named `README` with the link to the sources of the SAREF extension repository on the forge.

EXAMPLE 1: For SAREF core version v4.5.6, there should be a directory `core/v4.5.6` that contains a document named `README`.

EXAMPLE 2: For SAREF extension SAREF4ABCD version v1.2.3, there should be a directory `saref4abcd/v1.2.3` that contains a document named `README`.

8.2.2 Preparing the project version pre-release branch

The pre-release branch shall be named after the project version number of the SAREF project version.

The development branch `develop-vx.y.z` corresponding to the target project version number v*x.y.z* shall be forked and given the name `prerelease-vx.y.z`. To achieve this step, the branch protections may need to be relaxed temporarily.

EXAMPLE: For a SAREF project with target project version number v1.2.3, the branch `develop-v1.2.3` shall be forked and given the name `prerelease-v1.2.3`.

The project leader should ensure that:

- The development branch is synced with the pre-release branch.
- No change request is open for this SAREF project version.
- The SAREF repository conforms to the specifications in clause 9 of the present document.
- The SAREF pipeline software does not trigger errors when run in release mode.

8.3 Reviewing ontology releases

The Technical Board will review all the prepared ontology releases. If needed, the Technical Board will ask for clarifications to the project leader in charge of the ontology release.

The decision will be taken of publishing the ontology release when at least two members of the Technical Board have reviewed and approved the artefacts of the ontology release, notifying the user that created it.

8.4 Publishing an ontology release

SmartM2M is responsible for accepting the SAREF project release.
The Project leader and the Technical Board are responsible for triggering the release.

The content of the SAREF public portal is generated from the **release branches** of SAREF projects, and from the `saref-portal-static` project.

The release branch of a SAREF project version shall be named after the project version number.

The pre-release branch `prerelease-vx.y.z` corresponding to the target project version number v*x.y.z* shall be forked and given the name `release-vx.y.z`. To achieve this step, the branch protections may need to be relaxed temporarily.

EXAMPLE: For a SAREF project with target project version number v1.2.3, the branch `prerelease-v1.2.3` shall be forked and given the name `release-v1.2.3`.

The SAREF public portal may be updated manually or using a CD pipeline.

A CD pipeline may be triggered when and only when:

- when a change is made to a `release-*` branch of a SAREF extension repository;
- when a change is made to the `master` branch of the `saref-portal-static` repository.

This CD pipeline shall:

- run the SAREF pipeline project on the ETSI public forge in **portal mode** to generate the content of the SAREF public portal;

- push the SAREF public portal to the SAREF portal server;
- refresh or re-start the SAREF portal server.

9 SAREF project version specification and documentation

9.1 Introduction

The present clause specifies how SAREF project versions should be specified and documented in the sources of the SAREF project on the SAREF public forge. SAREF project version should be documented in an ETSI Technical Specification document, and potentially an associated ETSI Technical Report document. It is recommended to use existing documents as templates. Deviation from these templates needs to be strongly motivated.

9.2 Structure of the SAREF project version directory

The directory of the different SAREF project versions shall have a homogeneous structure.

There shall exist a file named `README.md`. This file shall contain a description of the SAREF project version in a similar way as other SAREF project versions.

There shall exist a file named `.gitignore`. This file shall contain each of the following strings on a separate line: `target`, `*~`,`.DS_Store`, `catalog-v001.xml`, `saref-pipeline.jar`.

There shall exist a file named `LICENSE`. This file shall contain the license of the SAREF project version. The SAREF project version is licensed under the terms of the BSD-3-clause. The license file shall contain the terms of such license following the ETSI licensing guidelines (https://forge.etsi.org/index.php/legal-matters).

There shall exist a directory named `requirements`. This directory contains requirement specifications for the SAREF project version. Clause 9.3 specifies the content of this directory.

There shall exist a directory named `ontology`. This directory contains the source ontology of the SAREF project version. Clause 9.4 specifies the content of this directory.

There shall exist a directory named `tests`. This directory contains tests for the SAREF project version. Clause 9.5 specifies the content of this directory.

There shall exist a directory named `examples`. This directory contains examples for the SAREF project version. Clause 9.6 specifies the content of this directory.

There shall exist a directory named `documentation`. This directory contains the sources for the human-readable documentation of the SAREF project version, with diagrams. Clause 9.7 specifies the content of this directory.

9.3 Ontology requirements

9.3.1 The `requirements` directory

The `requirements` directory should contain one file named `requirements.csv`.

This file shall conform to the requirements specification as defined in clause 9.3.2.

9.3.2 Requirements specification

The file with the requirements specification shall be a CSV (Comma-Separated Values) file encoded in UTF-8. The delimiter character used in the file shall be ';', the quote character (if needed) '"' and the end of line one "\n".

The first line of the file shall be the following:

```
Id;Category;Requirement
```

The rest of the lines shall contain a requirement per line, indicating for each requirement:

- The identifier of the requirement (`Id`).
- The category of the requirement (`Category`).
- The requirement itself, in form of statement or competency question (`Requirement`).

9.4 Ontology specification

9.4.1 The `ontology` directory

The `ontology` directory of the SAREF project version shall contain a single file called the **ontology document** of the SAREF project version.

If the SAREF project is SAREF core, then the ontology document shall be named `saref.ttl`.

If the SAREF project is a SAREF extension with acronym ABCD, then the ontology document shall be named `saref4abcd.ttl`.

The ontology document shall contain the sources of a consistent OWL 2 DL ontology [2], in the Turtle 1.1 [3] format.

The following clauses specify the Turtle 1.1 prefixes declaration, ontology definition, and term definition, sections of the ontology document.

9.4.2 Prefixes declaration

The ontology document may contain a list of prefixes declarations. If a prefix declaration has a prefix in the first column of Table 1, then the namespace shall be equal to the content of the second column.

NOTE: In the last line of Table 1, `efgh` is any SAREF extension acronym as specified in clause 3.1.

Table 1: List of standard namespaces and their corresponding prefixes

Prefix	Namespace
`rdf`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`
`owl`	`http://www.w3.org/2002/07/owl#`
`xsd`	`http://www.w3.org/2001/XMLSchema#`
`dcterms`	`http://purl.org/dc/terms/`
`vann`	`http://purl.org/vocab/vann/`
`voaf`	`http://purl.org/vocommons/voaf#`
`schema`	`http://schema.org/`
`saref`	`https://saref.etsi.org/core/`
`s4efgh`	`https://saref.etsi.org/saref4efgh/`

If the SAREF project is a SAREF extension with acronym ABCD, then the ontology document shall contain namespace https://saref.etsi.org/saref4abcd/ with corresponding prefix `s4abcd`.

The ontology document may contain a base declaration.

If the ontology document contains a base declaration and the SAREF project is SAREF core, then the base declaration shall be equal to `https://saref.etsi.org/core/`.

If the ontology document contains a base declaration and the SAREF project is a SAREF extension with acronym ABCD, then the base declaration shall be equal to https://saref.etsi.org/saref4abcd/.

EXAMPLE: The SAREF extension SAREF4CITY version v1.1.2 contains prefix declarations:

```
@prefix : <https://saref.etsi.org/saref4city/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix cpsv: <http://purl.org/vocab/cpsv#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix vann: <http://purl.org/vocab/vann/> .
@prefix geosp: <http://www.opengis.net/ont/geosparql#> .
@prefix saref: <https://saref.etsi.org/core/> .
@prefix s4city: <https://saref.etsi.org/saref4city/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
```

9.4.3 The ontology declaration

9.4.3.1 Ontology IRI and Ontology Version IRI

The **Ontology IRI** of the ontology document of SAREF core shall be:

`https://saref.etsi.org/core/`

The **Ontology IRI** of the ontology document of a SAREF extension with acronym ABCD shall be:

`https://saref.etsi.org/saref4abcd/`

EXAMPLE 1: The ontology IRI of the ontology document of SAREF4CITY is `https://saref.etsi.org/saref4city/`.

The **Ontology Version IRI** of the ontology document of SAREF core version v*x.y.z* shall be:

`https://saref.etsi.org/core/vx.y.z/`

EXAMPLE 2: The ontology IRI of the ontology document of SAREF core v1.2.3 is `https://saref.etsi.org/core/v1.2.3/`.

The **Ontology Version IRI** of the ontology document of a SAREF extension with acronym ABCD version v*x.y.z* shall be:

`https://saref.etsi.org/saref4abcd/vx.y.z/`

EXAMPLE 3: The ontology version IRI of the ontology document of SAREF4CITY v1.3.2 is `https://saref.etsi.org/saref4city/v1.3.2/`.

The ontology document shall contain exactly one ontology declaration. The IRI of the declared ontology shall be the Ontology IRI.

The ontology document shall contain exactly one metadata `owl:versionIRI`.
The value for this metadata shall be the **Ontology Version IRI**.

The ontology document shall contain exactly one metadata `owl:versionInfo`.
The value for this metadata shall be an `xsd:string` literal with lexical form equal to the target version number.

EXAMPLE 4: The value for the `owl:versionInfo` metadata for the ontology document of SAREF4CITY v1.3.2 is `"v1.3.2"^^xsd:string`.

If a release prior to v*x.y.z* is published on the SAREF public portal, then the ontology document shall contain exactly one metadata `owl:priorVersion`. The value for this metadata shall be the **Ontology Version IRI** of the most recent prior version that is published on the SAREF public portal.

The ontology document may contain some `owl:imports` declaration. If it does, then each value of the declaration shall be an Ontology Version IRI of a SAREF project version.

The ontology document shall contain exactly one metadata `vann:preferredNamespacePrefix`.
If the SAREF project version is SAREF core with version v*x.y.z*, then the value for this metadata shall be an `xsd:string` literal with lexical form `saref`.
If the SAREF project version is a SAREF extension with acronym ABCD and version v*x.y.z*, then the value for this metadata shall be an `xsd:string` literal with lexical form `s4abcd`.

The ontology document shall contain exactly one metadata `vann:preferredNamespaceUri`.
If the SAREF project version is SAREF core with version v*x.y.z*, then the value for this metadata shall be an `xsd:anyUri` literal with lexical form `https://saref.etsi.org/core/`.
If the SAREF project version is a SAREF extension with acronym ABCD and version v*x.y.z*, then the value for this metadata shall be an `xsd:string` literal with lexical form `https://saref.etsi.org/saref4abcd/`.

EXAMPLE 5: The ontology document of SAREF4CITY version v1.3.2 may contain ontology metadata:

```
<https://saref.etsi.org/saref4city/> a owl:Ontology ;
  owl:versionIRI <https://saref.etsi.org/saref4city/v1.3.2/> ;
  owl:versionInfo "v1.3.2" ;
  owl:priorVersion <https://saref.etsi.org/saref4city/v1.2.2/> ;
  owl:imports <https://saref.etsi.org/core/v3.1.1/> ;
  owl:imports <https://saref.etsi.org/saref4ener/v1.1.2/> ;
  vann:preferredNamespacePrefix "s4city" ;
  vann:preferredNamespaceUri "https://saref.etsi.org/saref4city/" .
```

9.4.3.2 Documentation metadata

The ontology document shall contain at least one metadata `dcterms:title`.
Each value for this metadata shall be a literal with datatype IRI `xsd:string` or `rdf:langString`
There should be at least one value with the `rdf:langString` datatype and the `en` language.

EXAMPLE: The ontology document of SAREF4CITY version v1.1.2 contains ontology metadata:
`<https://saref.etsi.org/saref4city/> dcterms:title "SAREF extension for Smart City"@en.`

The ontology document may contain at least one metadata `dcterms:abstract`.
Each value for this metadata should be a literal with datatype IRI `xsd:string`, `rdf:langString`.

The ontology document shall contain at least one metadata `dcterms:description`.
Each value for this metadata should be a literal with datatype IRI `xsd:string`, `rdf:langString`.

The ontology document shall contain exactly one metadata `dcterms:issued`.
The value for this metadata shall be a valid `xsd:date` literal, encoding the release date of the SAREF project version on the SAREF public portal.

The ontology document shall contain exactly one metadata `dcterms:modified`.
The value for this metadata shall be a valid `xsd:date` literal, encoding the last modification date of the ontology document.

The ontology document shall contain exactly one metadata `dcterms:source`.
The value for this metadata shall be the IRI of the repository on the SAREF public forge.

If the SAREF project version has an associated ETSI Technical Specification document published on the ETSI Portal, then the ontology document shall contain exactly one metadata `rdfs:seeAlso`.
The value for this metadata shall be the IRI to the corresponding ETSI Technical Specification.

The ontology document shall contain exactly one metadata `dcterms:license`.
The value for this metadata shall be the IRI `<https://forge.etsi.org/etsi-software-license>.`

The ontology document shall contain exactly one metadata `dcterms:publisher`.
The value for this metadata shall be the IRI `<https://www.etsi.org/>.`

9.4.3.3 Creators and contributors

The ontology document shall contain at least one metadata `dcterms:creator`, encoding the identity of any creator. The Steering Board and Project leader are responsible for approving the set of creators.

The ontology document may contain at least one metadata `dcterms:contributor`, encoding the identity of any contributor. The Steering Board and Project leader are responsible for approving the set of contributors.

The value for the `dcterms:creator` and `dcterms:contributor` annotation properties shall describe a **person**.

A person may be described either by a blank node, or by an IRI.
If a person is described by a blank node, then it shall be declared of type `schema:Person`.
If a person is described by an IRI, then it may be declared of type `schema:Person`.

Every instance of `schema:Person` shall have exactly one metadata `schema:givenName`.
The value for this metadata shall be a literal with datatype IRI `xsd:string` or `rdf:langString`.

Every instance of `schema:Person` shall have exactly one metadata `schema:familyName`.
The value for this metadata shall be a literal with datatype IRI `xsd:string` or `rdf:langString`.

Every instance of `schema:Person` may have one, and shall have at most one, metadata `schema:affiliation`.
The value for the `schema:affiliation` shall describe an **organization**.

An organization may be described either by a blank node, or by an IRI.
If an organization is described by a blank node, then it shall be declared of type `schema:Affiliation`.
If an affiliation is described by an IRI, then it may be declared of type `schema:Affiliation`.

Every instance of `schema:Affiliation` shall have exactly one metadata `schema:name`.
The value for this metadata shall be a literal with datatype IRI `xsd:string` or `rdf:langString`.

9.4.4 Term declarations

9.4.4.1 Term IRI

If the SAREF project version is SAREF core with version v*x.y.z*, then the ontology document **declares** all and only those **Terms** whose **Term IRI** have the form:

```
https://saref.etsi.org/core/localName
```

If the SAREF project version is a SAREF extension with acronym ABCD and version v*x.y.z*, then the ontology document **declares** all and only those **Terms** whose **Term IRI** have the form:

```
https://saref.etsi.org/saref4abcd/localName
```

The `localName` shall contain only letters and digits.
The `localName` of a class should be in Camel Case.
The `localName` of an object property or datatype property should be in Mixed Case.

9.4.4.2 Documentation metadata

Every Term declared in the ontology document shall have at least one metadata `rdfs:label`.
The value for this metadata shall be a literal with datatype IRI `xsd:string`, `rdf:langString`.
There should be at least one value with the `rdf:langString` datatype and the `en` language.

Every Term declared in the ontology document shall have at least one metadata `rdfs:comment`.
The value for this metadata should be a literal with datatype IRI `xsd:string` or `rdf:langString`.

9.4.5 OWL profile, consistency, lack of pitfalls and class satisfiability

The ontology in the ontology document shall satisfy the OWL2 DL profile [2], with the exception that unknown datatypes may be used.

The ontology in the ontology document shall be Consistent [4].

The ontology in the ontology document should not present ontology development pitfalls [i.14].

Every Term declared in the ontology document that is an OWL Class should be satisfiable [4].

9.5 Ontology tests

9.5.1 The `tests` directory

The `tests` directory should contain one file named `tests.csv`.

This file shall conform to the test specification as defined in clause 9.5.2.

9.5.2 Tests specification

The file with tests specification shall be a CSV (Comma-Separated Values) file encoded in UTF-8. The delimiter character used in the file shall be ';', the quote character (if needed) '"' and the end of line one "\n".

The first line of the file shall be the following:

```
Id;Requirement;Category;Test
```

The rest of the lines shall contain a test per line, indicating for each test:

- The identifier of the test (`Id`).
- The requirement associated to the test (`Requirement`) if it exists; if not, the value shall be empty.
- The category of the test (`Category`).
- The test itself, describing the expected behaviour of the test (`Test`).

9.6 Ontology examples

9.6.1 The `examples` directory

The `examples` directory of the SAREF project version directory should contain at least one file.

Files contained in the `examples` directory are called the **example documents** of the SAREF project version. Their goal is to illustrate how the ontology can be used in practice.

Every example document shall have the extension `.ttl`. It shall contain the sources of a consistent OWL 2 DL ontology [2], in the Turtle 1.1 [3] format.

Main classes and properties defined by the SAREF project version ontology should be illustrated with at least one example document.

Named individuals defined by the SAREF project version ontology may be illustrated with some example document.

The following clauses specify every example document, taking as example an example document `example12.ttl`.

9.6.2 Prefixes declaration

The present clauses specifies every example document, taking as example an example document `example12.ttl`.

The example document `example12.ttl` may contain a list of prefix declarations. If a prefix declaration has a prefix in the first column of Table 2, then the namespace shall be equal to the content of the second column.

NOTE: In the last line of Table 2, `efgh` is any ETSI SAREF project version acronym as specified in clause 6.2.

Table 2: List of standard namespaces and their corresponding prefixes

Prefix	Namespace
`rdf`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
`rdfs`	`http://www.w3.org/2000/01/rdf-schema#`
`owl`	`http://www.w3.org/2002/07/owl#`
`xsd`	`http://www.w3.org/2001/XMLSchema#`
`dcterms`	`http://purl.org/dc/terms/`
`dctype`	`http://purl.org/dc/dcmitype/`
`vann`	`http://purl.org/vocab/vann/`
`voaf`	`http://purl.org/vocommons/voaf#`
`schema`	`http://schema.org/`
`saref`	`https://saref.etsi.org/core/`
`s4efgh`	`https://saref.etsi.org/saref4efgh/`
`ex`	`https://saref.etsi.org/saref4abcd/v1.2.3/example/example12/`

If the SAREF project is SAREF core version v*x*.*y*.*z*, then:

- The example document shall contain namespace `https://saref.etsi.org/core/` with corresponding prefix `saref`.
- If the example document contains a prefix declaration with prefix `ex`, then the namespace shall be equal to `https://saref.etsi.org/core/vx.y.z/example/example12/`.
- The example document may contain a base declaration; if it does, then the base declaration shall be equal to `https://saref.etsi.org/core/vx.y.z/example/example12/`.

If the SAREF project is a SAREF extension with acronym ABCD and version v*x*.*y*.*z*, then:

- The example document shall contain namespace `https://saref.etsi.org/saref4abcd/` with corresponding prefix `s4abcd`.
- If the example document contains a prefix declaration with prefix `ex`, then the namespace shall be equal to `https://saref.etsi.org/saref4abcd/vx.y.z/example/example12/`.
- The example document may contain a base declaration; if it does, then the base declaration shall be equal to `https://saref.etsi.org/saref4abcd/vx.y.z/example/example12/`.

9.6.3 The example declaration

If the SAREF project is SAREF core version v*x*.*y*.*z*, then the **Dataset IRI** of the example document `example12.ttl` shall be:

`https://saref.etsi.org/core/vx.y.z/example/example12#`

If the SAREF project is a SAREF extension with acronym ABCD and version v*x*.*y*.*z*, then the **Dataset IRI** of the example document `example12.ttl` shall be:

`https://saref.etsi.org/saref4abcd/vx.y.z/example/example12#`

The example document shall contain exactly one instance of `dctype:Dataset`. The IRI of the declared Dataset shall be the Dataset IRI.

The example document shall contain at least one `dcterms:conformsTo` declaration.
If the SAREF project is SAREF core version v*x*.*y*.*z*, then one value for this declaration shall be `https://saref.etsi.org/core/vx.y.z/`.
If the SAREF project is a SAREF extension with acronym ABCD and version v*x*.*y*.*z*, then one value for this declaration shall be `https://saref.etsi.org/saref4abcd/vx.y.z/`.
Values for this declaration shall not be Ontology IRI of SAREF project versions.
Values for this declaration may be Ontology Version IRI of SAREF project versions.
Values for this declaration may be standard ontology IRI published by an international Standard Development Organization.

The example document should contain at least one metadata `dcterms:title`.
Each value for this metadata shall be a literal with datatype IRI `xsd:string` or `rdf:langString`.
There should be at least one value with the `rdf:langString` datatype and the `en` language.

The example document may contain at least one metadata `dcterms:abstract`.
Each value for this metadata shall be a literal with datatype IRI `xsd:string` or `rdf:langString`.

The example document should contain at least one metadata `dcterms:description`.
Each value for this metadata shall be a literal with datatype IRI `xsd:string` or `rdf:langString`.

The example document shall contain exactly one metadata `dcterms:license`.
The value for this metadata shall be the IRI `<https://forge.etsi.org/etsi-software-license>`.

9.6.4 OWL Profile, Consistency, and Satisfiability of Classes

The dataset in the example document, when augmented with an ontology declaration that imports all the ontologies the example conforms to, shall satisfy the OWL2 DL profile [2], with the exception that unknown datatypes may be used, and shall be consistent [4].

9.7 Ontology documentation

9.7.1 The `documentation` directory

The `documentation` directory of the SAREF project version directory should contain at least one file.

Files contained in the `documentation` directory are called the **documentation sources** of the SAREF project version.
Their goal is to provide human-readable documentation for the ontology and how it can be used in practice.
The documentation pages for the SAREF project version on the SAREF Ontology portal use the documentation sources.

Every documentation source shall be a HTML snippet and have the extension `.html` or be a markdown snippet and have extension `.md`.

The optional documentation source `creators.html` or `creators.md` shall describe the list of any creator.
The Steering Board and project leader are responsible for approving the list of creators.
The set of creators in the ontology document shall correspond to the list of creators in this documentation source.

The optional documentation source `contributors.html` or `contributors.md` shall describe the list of any contributor. The Steering Board and project leader are responsible for approving the list of contributors.
The set of contributors in the ontology document shall correspond to the list of contributors in this documentation source.

The optional documentation source `abstract.html` or `abstract.md` shall contain a short description of the SAREF project version ontology. It shall not include diagrams.

The optional documentation source `description.html` or `description.md` shall contain a description of the SAREF project version ontology and how it is intended to be used. It should include diagrams as defined in clause 9.6.2.

The optional documentation source `examples.html` or `examples.md` shall contain complete examples. It should include diagrams as defined in clause 9.6.2.

The optional documentation source `references.html` or `references.md` shall contain a list of references. It shall not include diagrams.

The optional documentation source `acknowledgement.html` or `acknowledgement.md` shall contain a list of acknowledgements. It shall not include diagrams.

9.7.2 The documentation/diagrams directory

The `documentation` directory of the ETSI SAREF project version directory should contain exactly one directory named `diagrams`, with the ontology diagrams to help the users understand the ontology conceptualization at a glance, or at least, in an easier and more comprehensive way than navigating through the individual ontology elements.

The sources of these diagrams shall be included in an open format and able to be processed with open software.

This clause provides guidelines to represent the most common ontology elements. The guidelines provided in this clause are based on the UML-Ont profile proposed in [i.15].

The diagrams suggestion for classes are depicted in Figure 4, for object properties in Figure 5, for datatype properties in Figure 6 and for individuals in Figure 7. Each diagram shows how to declare the existence of the type of element (class, object property, datatype property and individual) and how to attach the corresponding characteristics, if any.

If there are alternatives for stating element characteristics, the different options are separated in the figures by a dashed (vertical or horizontal) line. For example, in Figure 4, two alternatives are provided to represent class intersections, unions, subclasses and equivalences.

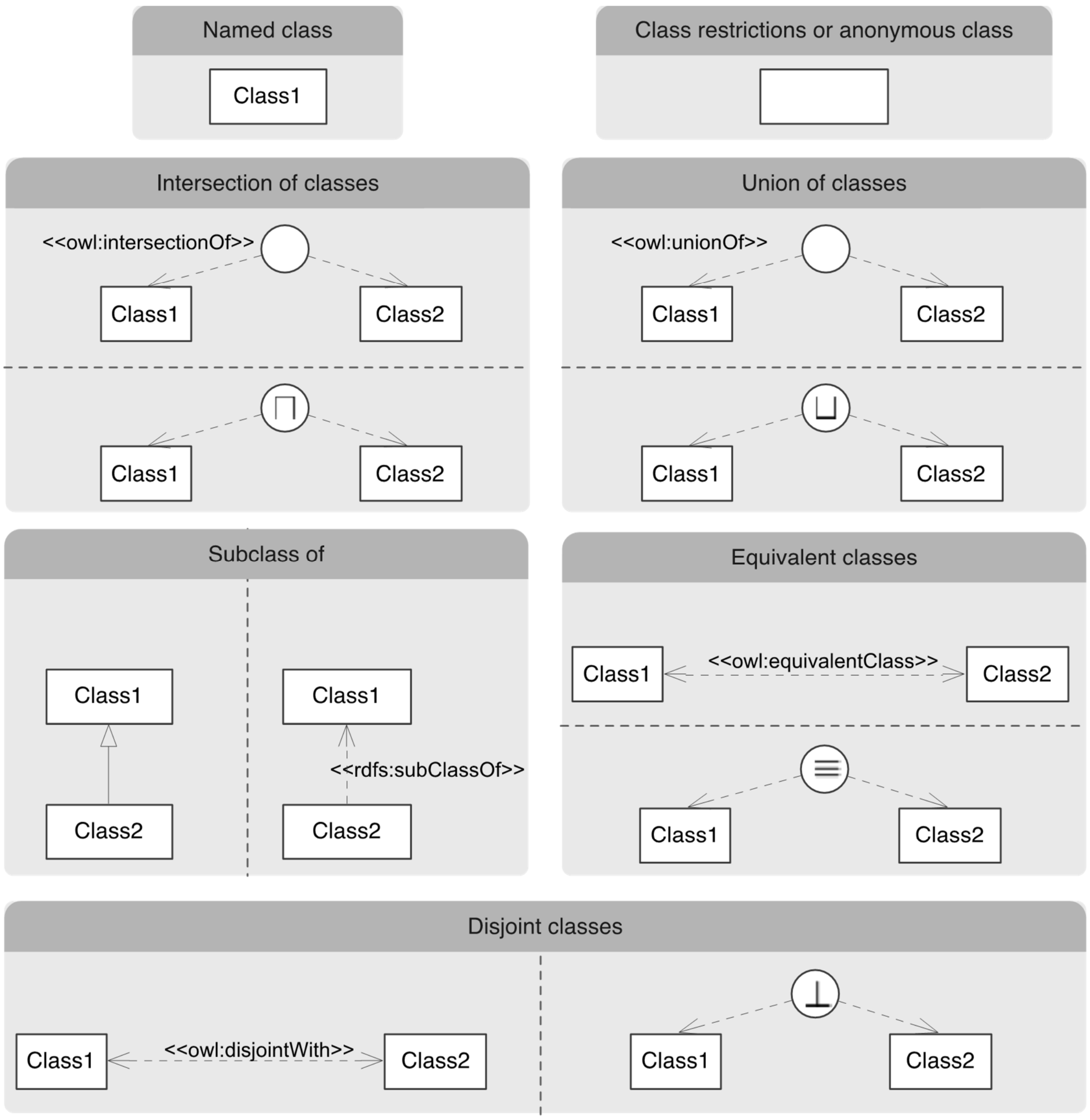

Figure 4: Diagrams guidelines for classes and classes characteristics

It should be mentioned that in Figure 5, in the "Object property" box the arrow to represent the object property in the first option is depicted with a dashed line to represent the existence of a property that is expected to be used among the classes in the origin and the target, however it does not implies declaration of domain and range axiom. Indeed, this case is used to represent expected use of properties for which domain and rage is not defined. The case in which the domain and range are declare is represented in the box "Domain and Range for object properties" in Figure 5. Same applies to the datatype property declaration in the box "Datatype property" in Figure 6. In that case the datatype property is defined within a dashed box as no domain and range are defined. The case in which the domain and range are declare is represented in the box "Domain and Range for datatype properties" in Figure 6.

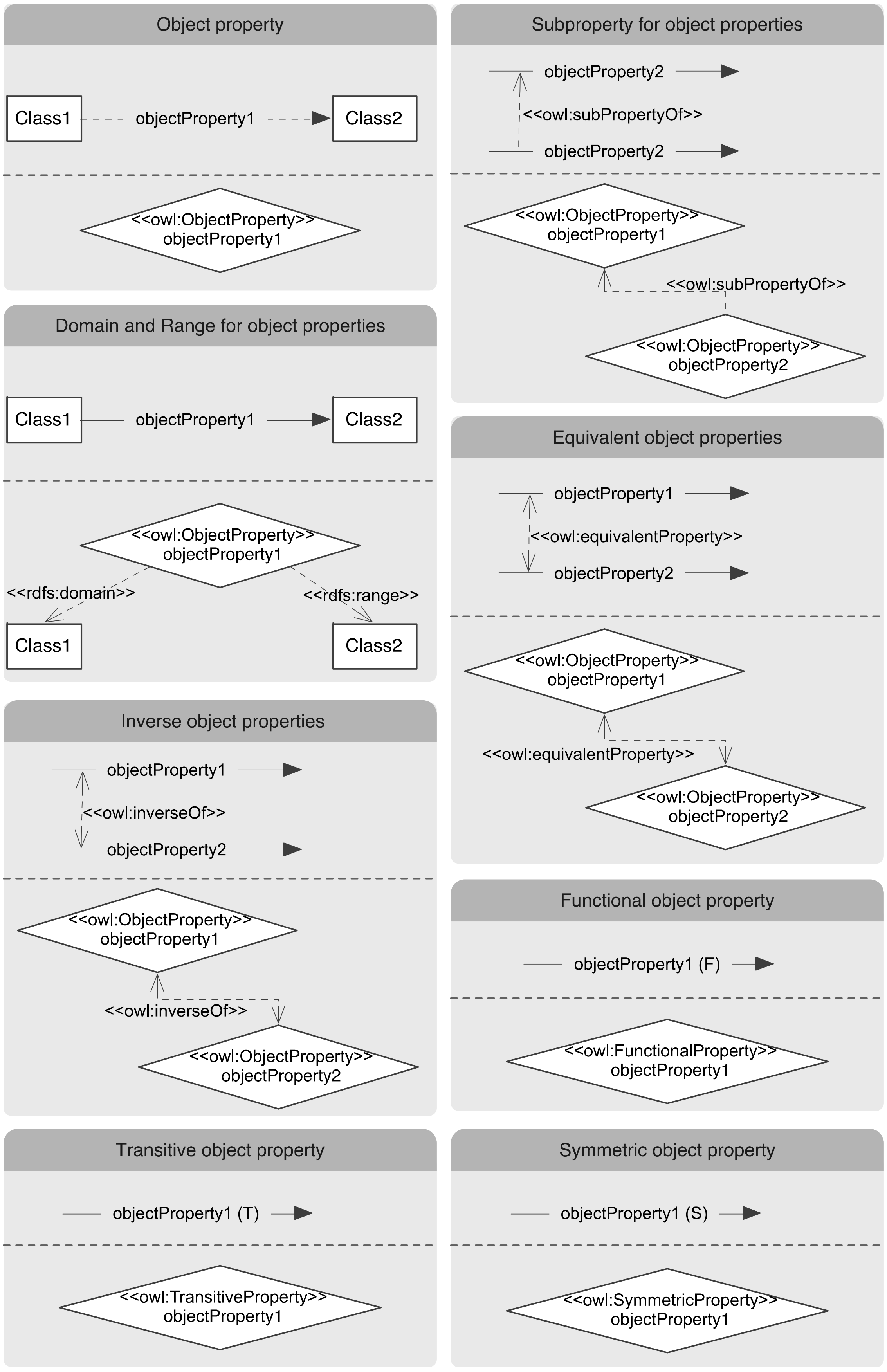

Figure 5: Diagrams guidelines for object properties and object properties characteristics

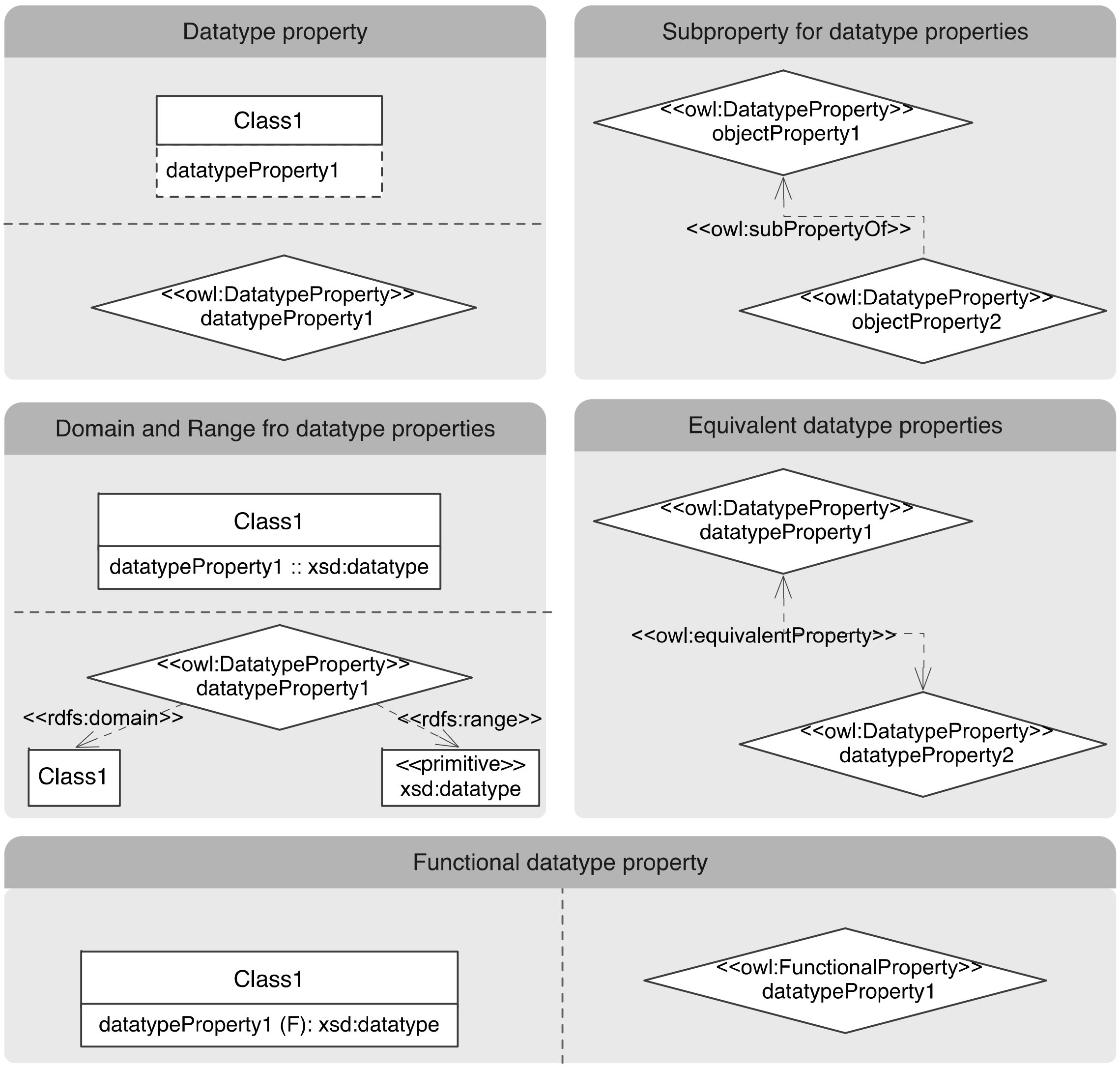

Figure 6: Diagrams guidelines for datatype properties and datatype properties characteristics

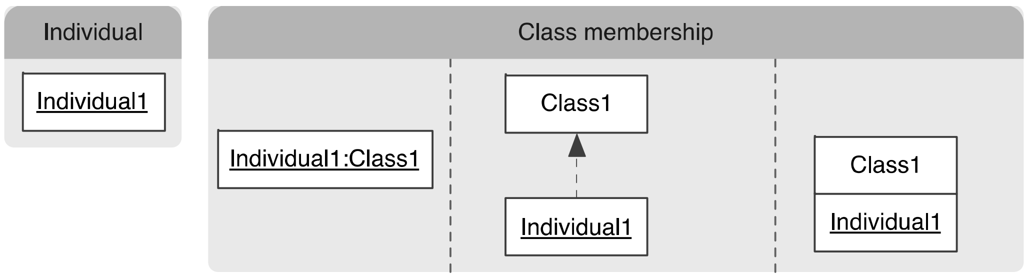

Figure 7: Diagrams guidelines for individuals

History

Document history		
V1.1.1	August 2020	Publication