OSLC PROMCODE Version 1.0. Part 1: Specification

Committee Specification Draft 01

28 April 2021

This stage:
https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-spec.html (Authoritative)
https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-spec.pdf

Previous stage:
N/A

Latest stage:
https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/promcode-spec.html (Authoritative)
https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/promcode-spec.pdf

Latest editor's draft:
http://tools.oasis-open.org/version-control/browse/wsvn/oslc-promcode/shape/trunk/spec.html

Technical Committee:
OASIS OSLC Lifecycle Integration for Project Management of Contracted Delivery (OSLC PROMCODE) TC

Chair:
Tom Kamimura (tomk@nanzan.jp), Nanzan University

Editors:
Mikio Aoyama (amikio@nanzan.jp), Nanzan University
Yoshio Horiuchi (hoy@jp.ibm.com), IBM
Tom Kamimura (tomk@nanzan.jp), Nanzan University
Shinji Matsuoka (matuoka.sinji@jp.fujitsu.com), Fujitsu Limited
Shigeaki Matsumoto (shigeaki.m@nec.com), NEC Corporation
Masaki Wakao (wakao@jp.ibm.com), IBM
Kazuo Yabuta (yabuta@nanzan.jp), Nanzan University
Hiroyuki Yoshida (yhryuki@nanzan-u.ac.jp), Nanzan University

Additional artifacts:

This specification is one component of a Work Product that also includes:

- *OSLC PROMCODE Version 1.0. Part 1: Specification* (this document). https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-spec.html.
- *OSLC PROMCODE Version 1.0. Part 2: Vocabulary*. https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-vocab.html.

- *OSLC PROMCODE Version 1.0. Part 3: Constraints*. https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-shapes.html.
- Machine-readable vocabulary terms: https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-vocab.ttl.
- Machine-readable constraints: https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-shapes.ttl.

Related work:

This document is related to:

- *OSLC PROMCODE Use Cases.* Work in progress. Latest editor's draft accessible at http://tools.oasis-open.org/version-control/browse/wsvn/oslc-promcode/shape/trunk/usecase.html. To be published at: https://docs.oasis-open.org/oslc-promcode/usecase/v1.0/usecase-v1.0.html.

Abstract:

This document defines the overall approach to PROMCODE (PROject Management of COntracted DElivery) based on the Open Services for Lifecycle Collaboration (OSLC) Core 3.0 [OSLCCore3] Specification. PROMCODE specification constitutes the approach outlined in this document and interface definitions referenced in other documents.

Status:

This document was last revised or approved by the OASIS OSLC Lifecycle Integration for Project Management of Contracted Delivery (OSLC PROMCODE) TC on the above date. The level of approval is also listed above. Check the "Latest stage" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=oslc-promcode#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list oslc-promcode-comment@lists.oasis-open.org, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/oslc-promcode/.

This specification is provided under the RF on Limited Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/oslc-promcode/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification, the following citation format should be used:

[OSLC-PROMCODE-v1.0-Spec]
OSLC PROMCODE Version 1.0. Part 1: Specification. Edited by Mikio Aoyama, Yoshio Horiuchi, Tom Kamimura, Shinji Matsuoka, Shigeaki Matsumoto, Masaki Wakao, Kazuo Yabuta, and Hiroyuki Yoshida. 28 April 2021. OASIS Committee Specification Draft 01. https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/csd01/promcode-spec.html. Latest stage: https://docs.oasis-open.org/oslc-promcode/promcode/v1.0/promcode-spec.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1. Introduction

This section is non-normative.

1.1 Background and Motivation

Global software delivery is commonplace today. With ever increasing pressure on the delivery of software projects for faster delivery, lower cost, and improved quality, it is becoming common for software delivery to be performed with collaboration of multiple organizations. Effective collaboration between multiple organizations requires activities to be managed and data to be shared across organizational boundaries. The management of software delivery can be highly challenging due to the diversity of the development processes, methods, tools and platforms used by organizations participating in a project [SPMC]. Management data and management practice are usually unique to each organization. Typically, manual operations are performed in exchanging proprietary management data and in coordinating activities, resulting in inefficient, error-prone and inflexible operations. As the number of organizations involved increases in a software delivery, the need for more systematic and standards-based data sharing and coordination becomes critical.

To address the need of systematic sharing of project management information, the PROMCODE consortium was created in Japan with seven member organizations consisting of Nanzan University and six IT service companies which are IBM, Hitachi, Fujitsu, NEC, NRI, and NTT Data. The consortium worked on defining standard information for management of contracted software delivery and published a specification document in 2013 [PROMCODE13][PROMCODE14]. The work was built on the OSLC framework and the technical content was based on the OSLC Core specification at that time. As a center of OSLC activities moved to OASIS, some of the core members of the consortium decided to bring its content to OASIS for global standard creation, and created the OASIS PROMCODE project. The main difference of the OASIS project from the work done by the consortium is an extended scope in OASIS and synchronization with the evolution of the OSLC Core specification [OSLCCore3]. The consortium work focused on minimal set of resources considered at that time. When each company implemented the specification, it was recognized that more resources were needed to reflect the general practice used in the industry. In particular, when two collaborating organizations do not share the management environment, one organization sends to another organization a status of all relevant project information as a report. This is a common practice and the consortium specification did not address that. The OASIS project decided to address that with addition of further resources needed.

1.2 Goal and Solution

The PROMCODE specification is intended to provide a common interface to exchange project management (PM) data across organizational boundaries. Figure 1 illustrates a generic model of collaborative software development. A set of organizations, including *A, B_1, …, B_n, $C_{1,1}$, …, $C_{1,m}$, …,$C_{n,m}$*, are working together to deliver a software system. Each organization employs own management process, tools and a model of management data. Therefore, we assume a model of management data and management tools used by each organization are different from those of other organizations. As a generic collaboration model, the two roles are assumed between any two organizations working together, that is, acquirer and supplier. The goal of the PROMCODE specification is to provide an open standard interface to exchange the management data between an acquirer and a supplier. In many real situations, each organization uses its own management data schema and management tools.

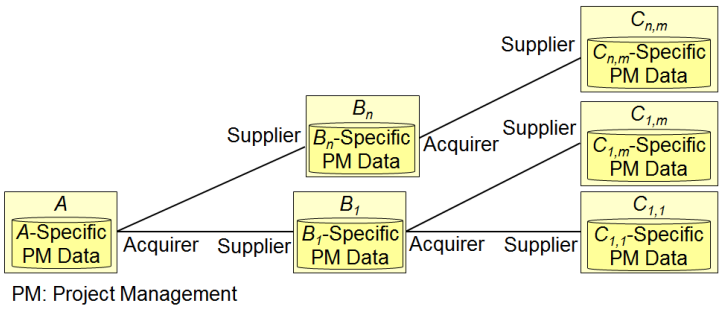

Fig. 1 Collaborative Software Development

To meet the goal, the PROMCODE specification defines a solution of two layers as illustrated in Fig. 2. The left-hand column represents the project management space and right-hand column does the corresponding OSLC Resource Definition in RDF [rdf11-concepts]. The upper layer is the PROMCODE specification and the lower layer is the Project-Specific Definitions. The PROMCODE Domain Model is an abstract definition of the structure of the project management data to be exchanged. The PROMCODE Domain Model is formulated for contracted delivery, derived from the project management knowledge [PMBOK5] and practices of contracted delivery of software [PROMCODE13], [PROMCODE14]. The PROMCODE Resource Definition is a representation of the PROMCODE Domain Model in terms of OSLC Resource Definition in RDF. For a specific project, the PROMCODE Domain Model is specialized to the Project-Specific Model, which extends the PROMCODE Domain Model. Similarly, the PROMCODE Resource Definition is specialized to the Project-Specific Resource Definition for a specific project. By this framework, different data in the different project management tools can be exchanged through the Project-Specific Resource Definition.

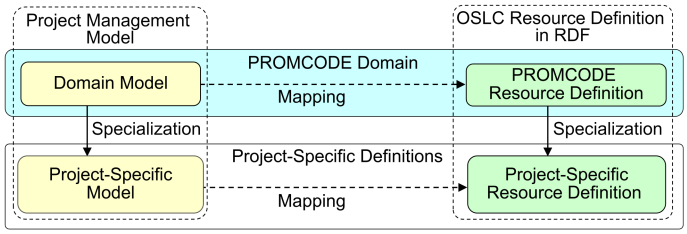

Fig. 2 PROMCODE Solution

1.3 PROMCODE Software Architecture

The PROMCODE software architecture is a typical software architecture that implements the specification. Since the PROMCODE specification is based on the OSLC Core 3.0, the architecture assumes an implementation of OSLC Core 3.0 [OSLCCore3] as an underlying platform. The goal of the PROMCODE software architecture is to enable exchanging project management data through the PROMCODE specification between multiple organizations that may use different data models of project management with different tools and methods.

The architecture is illustrated in Figure 3. As the figure shows, the PROMCODE clients and servers coordinate mapping of the organization-specific data instances from and to the Project-Specific Resources that are the instances of the Project-Specific PROMCODE Resource Definition. The interaction between the clients and servers takes place in the Resource-Oriented manner [ROA] with HTTP operations[HTTP11]. For example, supplier B_1 and supplier B_2 transform their respective B_1-specific PM (Project Management) data and B_2-Specific PM data, to the project-specific PROMCODE resources through their PROMCODE servers. Then, acquirer A can translate project-specific PROMCODE resources to A-specific PM data through either its PROMCODE client or its PROMCODE server. Note that the project management data of suppliers B_1 and B_2 may be defined as data of two separate data models, but it is possible to transform them to the data of a single model used by A. In this way, the PROMCODE specification can avoid the combinatorial explosion of transforming data across different data models.

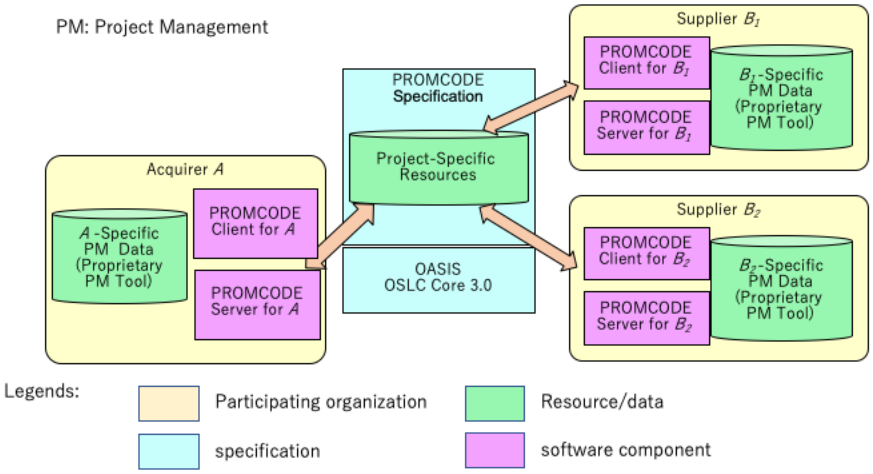

Fig. 3 PROMCODE Software Architecture for Project Management

1.4 RDF Namespaces

PROMCODE defines the namespace URI of `http://open-services.net/ns/promcode#` with a namespace prefix of `oslc_promcode`.

In addition, PROMCODE uses the following prefixes:

Prefix	Namespace
`dcterms`	http://purl.org/dc/terms/
`foaf`	http://xmlns.com/foaf/0.1/
`ldp`	http://www.w3.org/ns/ldp#
`oslc`	http://open-services.net/ns/core#
`rdf`	http://www.w3.org/1999/02/22-rdf-syntax-ns#
`rdfs`	http://www.w3.org/2000/01/rdf-schema#
`xsd`	http://www.w3.org/2001/XMLSchema#

1.5 Typographical Conventions and Use of RFC Terms

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this specification are to be interpreted as described in [RFC2119].

2. Normative and Non-normative

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative. Specifically the following list indicates normative and non-normative sections in this specification.

- 1. Introduction: **non-normative**
- 2. Normative and Non-normative: **normative**
- 3. Terminology: **normative**
- 4. PROMCODE Domain Model Specification: **non-normative**
- 5. PROMCODE Service Specification: **normative**
- 6. PROMCODE Resource Definitions: **normative**
- 7. Conformance: **normative**
- Appendix A. Change History: **non-normative**
- Appendix B. Acknowledgments: **non-normative**
- Appendix C. References: **normative and non-normative**

3. Terminology

Terminology is based on OSLC Core 3.0 [OSLCCore3], W3C Linked Data Platform [LDP], W3C's Architecture of the World Wide Web [WEBARCH] and Hyper-text Transfer Protocol [HTTP11].

Acquirer
: Acquirer is an entity to acquire software.

Supplier
: Supplier is an entity to supply software to be delivered to an Acquirer.

Project
: Project is a temporary endeavor undertaken to create unique software [PMBOK5].

Project Management
: Project Management is an application of knowledge, skills, tools, techniques, to project activities to meet project requirements [PMBOK5].

PROMCODE Resource
: PROMCODE Resource is an OSLC Resource [OSLCCore3] to represent project management data.

PROMCODE Service
: PROMCODE service is a service to exchange project management data defined by PROMCODE resources.

PROMCODE Server
: PROMCODE Server is an OSLC Server [OSLCCore3] that also supports the capabilities defined by the PROMCODE specification. See Server [OSLCCore3].

PROMCODE Client
: PROMCODE Client is an OSLC Client [OSLCCore3]that also supports the capabilities defined by the PROMCODE specification. See Client [OSLCCore3].

4. PROMCODE Domain Model Specification

This section is non-normative.

4.1 Domain Model

Fig. 4 PROMCODE Domain Model illustrates PROMCODE Domain Model denoted by a UML class diagram.

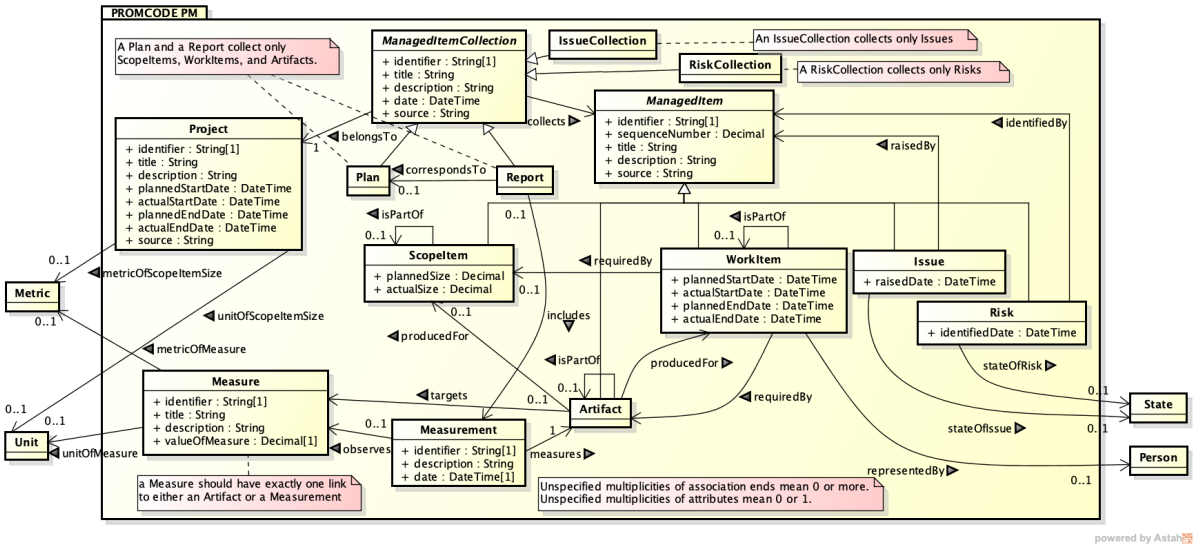

Fig. 4 PROMCODE Domain Model

In this figure, if no multiplicity is shown on an association end, it implies a multiplicity of zero to unlimited (0 .. *). The multiplicity of an attribute is also omitted for simplification. While its default is a multiplicity of zero to one (0 .. 1), there are exceptional cases where the multiplicity is exactly 1 in which it will be explicitly described as such in the subsequent part of this section.

4.1.1 ScopeItem

A scope item represents a scope of work from the acquirer's view of a software development contract. It represents a unit of value to be accomplished by the software supplier in the contract. For example, it may represent a function required, a use case in which the software will be used, a requirement which the acquirer expects, or a screen which will provide some concrete functions to the user of the software. A scope item is not an activity and therefore, cannot be started nor ended.

A ScopeItem entity has the size property that is used to determine the size of a contract between an acquirer and a supplier. This property is not used to track the progress of work. In this sense, a ScopeItem entity is different from an Artifact entity or a WorkItem entity.

An acquirer can use a set of ScopeItem entities as managed units to manage a whole scope of development. Both an acquirer and a supplier also use a set of ScopeItem entities to track the size of development. Attributes plannedSize and actualSize of a ScopeItem are used for the purpose. There should be agreement between an acquirer and a supplier on what kind of ScopeItem entities should be used and on how large each ScopeItem entity is. Change of a ScopeItem entity or its estimated size needs a change of the agreement. A ScopeItem entity can be decomposed into finer grained ScopeItem entities to be used in detailed management. In that case, a coarse grained ScopeItem entity may be used to aggregate a set of finer grained ScopeItem entities.

- Super Class
  - ManagedItem
- Attributes

    - plannedSize: Decimal [0 .. 1]
        - An estimated size agreed by both an acquirer and a supplier. The metric and the unit of size should be preliminarily agreed on between an acquirer and a supplier. They can be specified as metricOfScopeItemSize and unitOfScopeItemSize of a Project entity if necessary.
    - actualSize: Decimal [0 .. 1]
        - An actual size agreed by both an acquirer and a supplier.
- Links
    - isPartOf: ScopeItem [0 .. 1]
        - An ancestor of this ScopeItem entity.

4.1.2 WorkItem

A work item describes an activity to be performed in a software development contract. It adds details to the description of the work that is described by a scope item. These details typically include cost, schedule, and resource requirements. The set of all work items in a project forms a work breakdown structure.

A WorkItem entity represents the supplier's internal activity. For example, it may represent a development phase such as analysis, design, implementation, or test. It may also represent a finer grained activity such as document writing, reviewing, or coding. A WorkItem entity is a managed unit of activity required to implement a ScopeItem entity or to produce an Artifact entity.

Progress of a WorkItem entity is managed by comparing planned and actual dates on which it is started and ended.

A WorkItem entity can be decomposed into finer grained WorkItem entities to be used in detailed management. A coarse grained WorkItem entity is used to aggregate a set of fine grained WorkItem entities.

- Super Class
    - ManagedItem
- Attributes
    - plannedStartDate: DateTime [0 .. 1]
        - A planned date to start this WorkItem entity.
    - actualStartDate: DateTime [0 .. 1]
        - An actual date to start this WorkItem entity.
    - plannedEndDate: DateTime [0 .. 1]
        - A planned date to end this WorkItem entity.
    - actualEndDate: DateTime [0 .. 1]
        - An actual date to end this WorkItem entity.
- Links
    - representedBy: Person [0 .. 1]
        - A person responsible for the progress of this WorkItem entity who may or may not actually do this WorkItem entity.
    - isPartOf: WorkItem [0 .. 1]
        - An ancestor of this WorkItem entity.
    - requiredBy: ScopeItem [0 .. 1]
        - A ScopeItem entity which this WorkItem entity is required to implement.
    - requiredBy: Artifact [*]
        - Artifact entities which this WorkItem entity is required to produce. For example, producing specification documents requires interviewing users, writing the initial version, reviewing, reflecting the review comments, approving, and so on.

4.1.3 Artifact

An artifact is a work product that is produced in a project such as design documents, source code, test report, and so on. An artifact may be physical or digital. An artifact is produced for a work item or a scope item.

An Artifact entity can be measured using Measure entities, and their measured values may vary at each point of time on a project. The quality of an Artifact entity is managed by comparing targeted and actual Measure entities.

An Artifact entity can be decomposed into finer grained Artifact entities to be used in detailed management. A coarse

grained Artifact entity may be used to aggregate a set of finer grained Artifact entities.

- Super Class
  - ManagedItem
- Links
  - isPartOf: Artifact [0 ..1]
    - An ancestor of this Artifact entity.
  - producedFor: WorkItem [*]
    - WorkItem entities which require to produce this Artifact entity.
  - producedFor: ScopeItem [*]
    - ScopeItem entities which require to produce this Artifact entity.
  - targets: Measure [*]
    - Measure entities planned for this Artifact entity. Before project execution, the acquirer and the supplier determine which measure should be used. Once determined, the measures are used as targets of the Artifact entity until project completion. An Artifact entity can have zero or more Measure entities as targets.

4.1.4 Risk

A risk is a potential problem that must be controlled before it occurs in order to meet the objectives of a project. Failure to control the risk may result in a situation that can negatively impact the project, such as a schedule delay and quality problems. If a risk actually causes the situation to occur, it typically becomes an issue.

- Super Class
  - ManagedItem
- Attributes
  - identifiedDate: DateTime [0 .. 1]
    - An identified date of this Risk entity.
- Links
  - identifiedBy: ManagedItem [*]
    - ManagedItem entities which identify this Risk entity. A Risk entity may be identified by one or more ManagedItem entities.
  - stateOfRisk: State [0 .. 1]
    - A state of this Risk entity.

4.1.5 Issue

An issue is a situation that must be resolved to avoid negative impact to the project. Failure to resolve the issue may result in negative consequences to the project, such as a schedule delay and not meeting quality goals.

- Super Class
  - ManagedItem
- Attributes
  - raisedDate: DateTime [0 .. 1]
    - A raised date of this Issue entity.
- Links
  - raisedBy: ManagedItem [*]
    - ManagedItem entities which raise this Issue entity. An Issue entity may be raised by one or more ManagedItem entities.
  - stateOfIssue: State [0 .. 1]
    - A state of this Issue entity.

4.1.6 ManagedItem

A managed item is one of a scope item, a work item, an artifact, a risk, or an issue.

ManagedItem is a super class which abstracts five classes, that is, ScopeItem, WorkItem, Artifact, Risk, and Issue.

- Attributes
  - identifier: String [1]
    - A unique identifier of this ManagedItem entity.
  - sequenceNumber: Decimal [0 .. 1]
    - A unique number which represents an order of entities in each subclass of this ManagedItem entity.
  - title: String [0 .. 1]
    - A name of this ManagedItem entity.
  - description: String [1]
    - A text which describes of this ManagedItem entity.
  - source: String[0 .. 1]
    - A ManagedItem entity may be created as derived entity of another ManagedItem entity as a source entity. In that case, this property links from the derived entity to the source entity. The value is a string such as a URI that uniquely identifies the source entity.

4.1.7 Plan

A plan is a collection, or a snapshot, of managed entities which is agreed on between an acquirer and a supplier at the project initiation and on the timing when a plan is changed.

A Plan entity is a ManagedItemCollection entity which is a collection of planned ScopeItem, WorkItem, and Artifact entities together with targeting Measure entities.

- Super Class
  - ManagedItemCollection

4.1.8 Report

A report is a collection, or a snapshot, of managed items which is created by a supplier for project monitoring.

A Report entity is a ManagedItemCollection entity which is a collection of ScopeItem, WorkItem, and Artifact entities together with associated Measurement entities and Measure entities. It represents the situation of the project on a specific date.

- Super Class
  - ManagedItemCollection
- Links
  - correspondsTo: Plan [0 .. 1]
    - A Plan entity of this Report entity. A Report entity is created to the corresponding Plan entity. If the Plan entity is changed, the Report entity is created to link to the new Plan entity.
  - includes: Measurement [*]
    - Measurement entities that are included in the Report entity.

4.1.9 RiskCollection

A risk collection is a collection, or a snapshot, of risks.

Only Risk entities can be collected in a RiskCollection entity. It may collect all Risk entities of the project on a specific date, all Risk entities of some categories such as those in certain state or those with priority above some value, and all Risk entities related to the specific ScopeItem entity.

- Super Class
  - ManagedItemCollection

4.1.10 IssueCollection

An issue collection is a collection, or a snapshot, of issues.

Only Issue entities can be collected in an IssueCollection entity. It may collect all Issue entities of the project on a specific

date, all Issue entities of some categories such as those in certain state or those with priority above some value, and all Issue entities related to the specific ScopeItem entity.

- Super Class
  - ManagedItemCollection

4.1.11 ManagedItemCollection

A managed item collection is a collection of managed items. Four kinds of such a collection are used: a plan, a report, a risk collection, and an issue collection.

ManagedItemCollection is a super class which abstracts four kinds of collections of managed items, that is, Plan, Report, RiskCollection, and IssueCollection. A ManagedItemCollection entity must collect managed items belonging to only one project.

- Attributes
  - identifier: String [1]
    - A unique identifier of this ManagedItemCollection entity.
  - title: String [0 .. 1]
    - A name of this ManagedItemCollection entity.
  - description: String [0 .. 1]
    - A text which describes this ManagedItemCollection entity.
  - date: DateTime [0 .. 1]
    - A date on which this ManagedItemCollection entity collects ManagedItem entities. More specific definition of date is left to a project.
  - source: String[0 .. 1]
    - A ManagedItemCollection entity may be created as derived entity of another ManagedItemCollection entity as a source entity. In that case, this property links from the derived entity to the source entity. The value is a string such as a URI that uniquely identifies the source entity.
- Links
  - collects: ManagedItem [*]
    - ManagedItem entities included in this ManagedItemCollection entity. A ManagedItemCollection entity collects any type of ManagedItem entities such as ScopeItem, WorkItem, Artifact, Risk, and Issue entities.
  - belongsTo: Project [1]
    - A Project entity which this ManagedItemCollection entity belongs to. A Project entity has multiple snapshots until completion. Sometimes a supplier or an acquirer may operate multiple Project entities in parallel. The supplier or the acquirer needs to identify which Project entity each snapshot belongs to.

4.1.12 Measure

A measure is an observation of measurable attribute of an artifact.

A Measure entity has a metric such as "number of lines of codes (LOC)", "number of found bugs", and so on.

It also has a unit which defines the unit of the numbered value, for example, "1" means one line or one kilo lines. A Measure entity must be either targeted by one Artifact entity or observed by one Measurement entity.

- Attributes
  - identifier: String [1]
    - A unique identifier of this Measure entity.
  - title: String [0 .. 1]
    - A name of this Measure entity.
  - description: String [0 .. 1]
    - A text which describes this Measure entity.
  - valueOfMeasure: Decimal [1]
    - A value of this Measure entity.
- Links
  - metricOfMeasure: Metric [0 .. 1]

        - A metric of this Measure entity.
    - unitOfMeasure: Unit [0 .. 1]
        - A unit of this Measure entity.

4.1.13 Measurement

A Measurement entity links between an Artifact entity and its actual Measure entities. It also represents the date on which the Measure entities are taken.

- Attributes
    - identifier: String [1]
        - A unique identifier of this Measurement entity.
    - title: String [0 .. 1]
        - A name of this Measurement entity.
    - description: String [0 .. 1]
        - A text which describes this Measurement entity.
    - date: DateTime [1]
        - A date on which an Artifact entity is measured.
- Links
    - observes: Measure [*]
        - Measure entities measured by this Measurement entity. One Measurement entity can measure zero or more Measure entities.
    - measures: Artifact [1]
        - An Artifact entity measured by this Measurement entity.

4.1.14 Project

A software development project is a collaborative activity to be executed in a fixed time period to produce software systems and/or products.

A Project entity represents the information of the whole project such as the name of the project, the descriptions of the project, and progress of the whole project. A Project entity also specifies a Metric entity and the unit of size of ScopeItem entities because they should be unique in a project.

- Attributes
    - identifier: String [1]
        - A unique identifier of this Project entity.
    - title: String [0 .. 1]
        - A name of this Project entity.
    - description: String [0 .. 1]
        - A text which describes of this Project entity.
    - plannedStartDate: DateTime [0 .. 1]
        - A planned date to start this Project entity.
    - actualStartDate: DateTime [0 .. 1]
        - An actual date to start this Project entity.
    - plannedEndDate: DateTime [0 .. 1]
        - A planned date to end this Project entity.
    - actualEndDate: DateTime [0 .. 1]
        - An actual date to end this Project entity.
    - source: String[0 .. 1]
        - A Project entity may be created as derived entity of another Project entity as a source entity. In that case, this property links from the derived entity to the source entity. The value is a string such as a URI that uniquely identifies the source entity.
- Links
    - metricOfScopeItemSize: Metric [0 .. 1]
        - A metric of size of ScopeItem entities in this Project entity.
    - unitOfScopeItemSize: Unit[0 .. 1]
        - A unit of size of ScopeItem entities in this Project entity.

4.1.15 String

A String entity represents a character string.

4.1.16 DateTime

A DateTime entity represents a date and time.

4.1.17 Decimal

A Decimal entity represents a decimal number.

4.2 Samples of Two Project-Specific Models

4.2.1 Applying to Progress Management

Table 1 shows a typical progress management table. The table describes the status of implementing functions defined in the first column. Each function is divided into a collection of sub-functions. Each sub-function has phases of analysis, design, and coding activities. Note that the real management tables are more complex than shown in the case. Functions and sub-functions form a tree structure with several levels. There are more activities required to implement sub-functions.

Table 1 only shows the essential structure of real management tables.

Table 1 Example of Progress Management Table

Function	Sub Function		Analysis		Design		Coding	
			Start	End	Start	End	Start	End
A	A1	Planned	6/4	6/11	6/12	6/19	6/20	6/27
		Actual	6/4	6/10	6/11	6/19	6/20	6/26
	A2	Planned	6/4	6/11	6/12	6/19	6/20	6/25
		Actual	6/4	6/12	6/13	6/20	6/21	6/26
B	B1	Planned	6/4	6/11	6/12	6/19	6/20	6/27
		Actual	6/4	6/12	6/13	6/20	6/21	6/28

Fig. 5 Example of Project-Specific Model for Progress Management illustrates the corresponding project-specific model. Function and Sub Function are subclasses of ScopeItem, and the Analysis, Design, and Coding activities are subclasses of WorkItem. A Function is decomposed into a collection of Sub Functions which have three kinds of required WorkItems for implementation. This structure indicates that all project management data in Table 1 can be represented as instances of the PROMCODE Domain Model classes of ScopeItem and WorkItem.

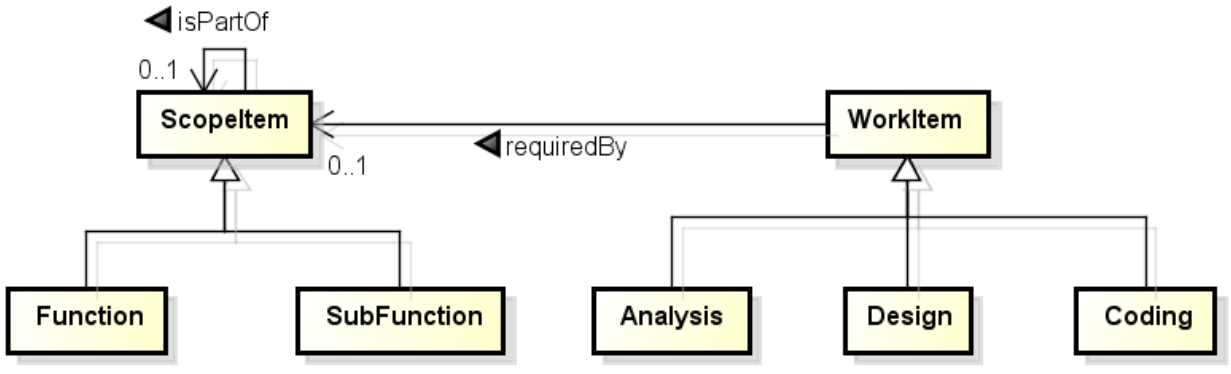

Fig. 5 Example of Project-Specific Model for Progress Management

4.2.2 Applying Quality Management

Table 2 shows a typical quality management table. Main managed items are modules which are grouped under requirements. Each module is measured using several KPIs including lines of code, number of test cases, and number of defects found.

Table 2 Example of Quality Management Table

Requirement	Module	Line of Code		#Test Case		#Defect	
		Target	Actual	Target	Actual	Target	Actual
R1	M1-1	2,000	2,130	60	62	5	5
	M1-2	1,500	1,450	45	43	3	2
R2	M2-1	2,000	1,980	60	65	5	4
	M2-2	1,000	950	30	35	2	2

Fig. 6 Example of Project-Specific Model for Quality Management illustrates the corresponding project-specific model. Requirement is a subclass of ScopeItem and Module is a subclass of Artifact. A Requirement entity requires to produce some Modules entities. The structure indicates that all project management data in Table 2 can be represented as instances of PROMCODE Domain Model classes of ScopeItem, Artifact, and Measure.

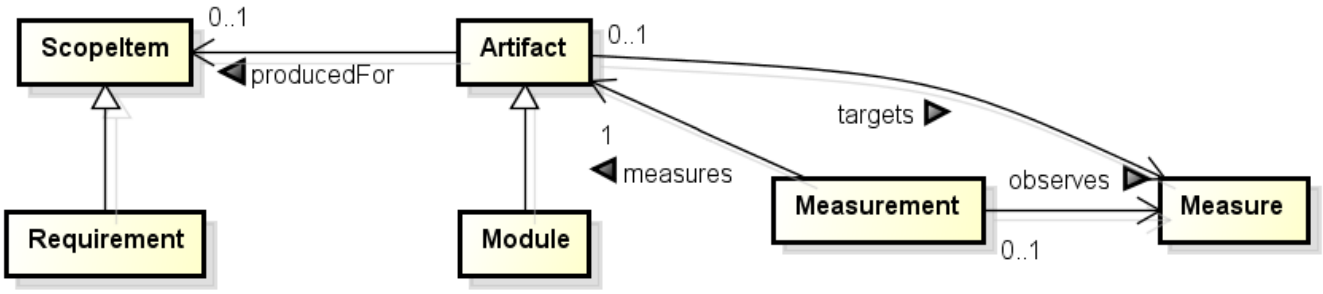

Fig. 6 Example of Project-Specific Model for Quality Management

5. PROMCODE Service Specification

5.1 Base Requirements

This sections describes the mandatory and optional requirements for an OSLC PROMCODE server.

The PROMCODE specification is based on OSLC Core 3.0 [OSLCCore3]. PROMCODE servers and clients **MUST** be compliant with both the [OSLCCore3] specification and the PROMCODE specification, and **SHOULD** follow all the guidelines and recommendations in both of these specifications. [promcode-1]

A PROMCODE server **MUST** implement the vocabulary defined in 6. Vocabulary Terms and Constraints of this specification with the exception of two abstract classes `ManagedItem` and `ManagedItemCollection`. [promcode-2]

The following table summarizes the requirements from OSLC Core Specification as well as some additional requirements specific to the PROMCODE specification. Note that this specification further restricts some of the requirements for OSLC Core Specification. See subsequent sections in this specification or the OSLC Core Specification to get further details on each of these requirements.

Requirement	Meaning
Unknown properties and content	OSLC services **MAY** ignore unknown content and OSLC clients **MUST** preserve unknown content. [promcode-3]
Resource Operations	OSLC service **MUST** support resource operations via standard HTTP operations. [promcode-4]
Resource Paging	OSLC services **MAY** provide paging for resources but only when specifically requested by client. [promcode-5]
Partial Resource Representations	OSLC services **SHOULD** support request for a subset of a resource's properties via the oslc.properties URL parameter retrieval via HTTP GET. [promcode-6]
Partial Update	OSLC services **MAY** support partial update of resources using patch semantics and **MAY** support via HTTP PUT. [promcode-7]
Discovery	OSLC servers **SHOULD** support OSLC Core 3.0 Discovery, **MAY** provide a ServiceProviderCatalog and **SHOULD** provide a ServiceProvider resource for Core v2 compatibility. [promcode-8]
Creation Factories	OSLC servers **MAY** provide LDPC creation factories to enable resource creation of PROMCODE resources via HTTP POST. [promcode-9]
Query Capabilities	OSLC servers **SHOULD** provide query capabilities to enable clients to query for resources. [promcode-10]
Query Syntax	OSLC query capabilities **SHOULD** support the OSLC Core Query Syntax and **MAY** use other query syntax. [promcode-11]
Delegated UI Dialogs	OSLC Services **MAY** offer delegated UI dialogs (creation and selections) specified via OSLC Core 3.0 Delegated Dialogs and **SHOULD** include discovery through a ServiceProvider resource for OSLC v2 compatibility. [promcode-12]
UI Preview	OSLC Services **MAY** offer UI previews for resources that may be referenced by other resources specified via OSLC Core 3.0 Preview and **SHOULD** include discovery through a server resource for OSLC v2 compatibility. [promcode-13]

Requirement	Meaning
HTTP Basic Authentication	OSLC Services **MAY** support Basic Auth and should do so only over HTTPS. [promcode-14]
OAuth Authentication	OSLC Services **MAY** support OAuth and can indicate the required OAuth URLs via the ServiceProviderCatalog or ServiceProvider resources. [promcode-15]
Error Responses	OSLC Services **MAY** provide error responses using OSLC Core 3.0 defined error formats. [promcode-16]
Turtle Representations	OSLC services **MUST** provide a Turtle representation for HTTP GET requests and **SHOULD** support Turtle representations on POST and PUT requests. [promcode-17]
RDF/XML Representations	OSLC services **SHOULD** provide an RDF/XML representation for HTTP GET requests and **SHOULD** support RDF/XML representations on POST and PUT requests. [promcode-18]
XML Representations	OSLC services **SHOULD** provide a XML representation for HTTP GET, POST and PUT requests that conform to the Core 2.0 Guidelines for XML. [promcode-19]
JSON Representations	OSLC services **MUST** provide JSON-LD representations for HTTP GET, POST and PUT requests that conform to the Core Guidelines for JSON-LD. [promcode-20]
HTML Representations	OSLC services **MAY** provide HTML representations for HTTP GET requests. [promcode-21]

The following sections describe further restrictions and guidance on the requirements of [OSLCCore3].

5.2 Namespaces

As defined in RDF Namespaces, PROMCODE defines the namespace URI of `http://open-services.net/ns/promcode#` with a preferred namespace prefix `oslc_promcode`. Also, the common URL prefixes given there are used when they are needed.

5.3 Resource Formats

In addition to the requirements for resource representations in [OSLCCore3], this section outlines further refinements and restrictions.

For HTTP GET requests on all PROMCODE and OSLC Core defined resource types,

- A PROMCODE server **MUST** provide Turtle and JSON-LD, and **SHOULD** provide RDF/XML and XML representations. The XML and JSON representations **SHOULD** follow the guidelines outlined in the OSLC Core Representations Guidance to maintain compatibility with [OSLCCore2]. [promcode-22]
- A PROMCODE client requesting RDF/XML **SHOULD** be prepared for any valid RDF/XML document. A PROMCODE client requesting XML **SHOULD** be prepared for representations that follow the guidelines outlined in the OSLC Core Representations Guidance. [promcode-23]
- A PROMCODE server **SHOULD** support an [X]HTML representation and a user interface (UI) preview as defined by UI Preview Guidance. [promcode-24]

For HTTP PUT/POST requests for the PROMCODE resources that support the requests:

- A PROMCODE server **MUST** accept Turtle and JSON-LD representations and **SHOULD** accept RDF/XML and XML representations. A PROMCODE server accepting RDF/XML **SHOULD** be prepared for any valid RDF/XML document. For XML A PROMCODE server **SHOULD** be prepared for representations that follow the guidelines outlined in the OSLC Core Representations Guidance. [promcode-25]

For HTTP GET response formats for Query requests,

- A PROMCODE server **MUST** provide Turtle and JSON-LD, **SHOULD** provide RDF/XML and XML, and **MAY** provide Atom Syndication Format XML representations. [promcode-26]

When A PROMCODE client request:

- `text/turtle` A PROMCODE server **MUST** respond with Turtle representation.
- `application/ld+json` A PROMCODE server **MUST** respond with JSON-LD representation.
- `application/rdf+xml` A PROMCODE server **SHOULD** respond with RDF/XML representation without restrictions.
- `application/xml` A PROMCODE server **SHOULD** respond with OSLC-defined abbreviated XML representation as defined in the OSLC Core Representations Guidance.
- `application/atom+xml` A PROMCODE server **SHOULD** respond with Atom Syndication FOrmat XML representation as defined in the OSLC Core Representations Guidance. The Atom Syndication Format XML representation **SHOULD** use RDF/XML representation without restrictions for the atom:content entries representing the resource representations.

[promcode-27]

5.3.1 Content Negotiation

OSLC Core 3.0 [OSLCCore3] specifies RDF representations (and specifically Turtle and JSON-LD) as a convention that all OSLC server implementations minimally provide and accept. OSLC PROMCODE server implementations are strongly encouraged to adopt this convention. Future versions of this specification are expected to require RDF representations for all operations and relax requirements for specialized XML representations.

XML Representation - identified by the `application/xml` content type. Format representation rules are outlined in Core OSLC Core Resource Formats section.

RDF/XML Representation - identified by the `application/rdf+xml` content type. No additional guidance is given.

JSON-LD Representation - identified by the `application/ld+json` content type. Format representation rules are specified in JSON-LD 1.0.

Atom Syndication Format XML Representation - identified by the `application/atom+xml` content type. Format representation rules are outlined in Core OSLC Core Resource Formats section.

5.4 Authentication

[OSLCCore3] specifies the recommended OSLC authentication mechanisms. The PROMCODE specification introduces no additional constraints on error responses.

5.5 Error Responses

[OSLCCore3] specifies the OSLC Core error responses. The PROMCODE specification introduces no additional constraints on error responses.

5.6 Pagination

[OSLCCore3] specifies the support of pagination of query results. The PROMCODE specification introduces no additional constraints on error responses.

5.7 Requesting and Updating Properties

5.7.1 Requesting a Subset of Properties

A client **MAY** request a subset of a resource's properties as well as properties from a referenced resource. In order to support this behavior, a server **MAY** support the `oslc.properties` and `oslc.prefix` URL parameter on a HTTP

GET request on individual resource request or a collection of resources by query. If the oslc.properties parameter is omitted on the request, then all resource properties **MUST** be provided in the response. [promcode-28]

5.7.2 Updating a Subset of Properties

A client **MAY** request that a subset of a resource's properties be updated by using the PATCH method. [promcode-29]

For compatibility with [OSLCCore2], A PROMCODE server **MAY** also support partial update by identifying those properties to be modified using the `oslc.properties` URL parameter on a HTTP PUT request. [promcode-30]

If the parameter `oslc.properties` contains a valid resource property on the request that is not provided in the content, the server **MUST** set the resource's property to a null or empty value. If the parameter `oslc.properties` contains an invalid resource property, then a 409 Conflict **MUST** be returned. [promcode-31]

5.7.3 Updating Multi-Valued Properties

For multi-valued properties that contain a large number of values, it may be difficult and inefficient to add or remove property values. A PROMCODE server **MAY** provide support for a partial update of the multi-valued properties as defined by OSLC Core Partial Update. [promcode-32]

5.8 Resource Operations

This section describes frequently used resource operations.

5.8.1 Create Resources

A PROMCODE client **MAY** request the creation of a PROMCODE domain resource via HTTP/POST operation to a PROMCODE server that supports creation of a resource using Creation Factories as defined in [OSLCCore3]. [promcode-33]

Alternatively, creation of a domain resource **MAY** be done outside of the PROMCODE framework, i.e., by using an existing project management tool that creates a new resource and makes the resource visible to the PROMCODE framework. [promcode-34]

5.8.2 Get Resources

A PROMCODE server **MUST** support the HTTP GET method to obtain the representation of a resource. The detailed behavior of the GET method is described in [LDP]. [promcode-35]

5.8.3 Update Resources

A PROMCODE server **MAY** support the HTTP PUT method or the HTTP PATCH method to update resources. The detailed behavior of the PUT and PATCH methods is described in [LDP]. [promcode-36]

5.8.4 Delete Resources

A PROMCODE server **MAY** support the HTTP DELETE method to delete resources. The detailed behavior of the DELETE method is described in [LDP]. [promcode-37]

5.9 PROMCODE Server Capabilities

5.9.1 Server Resources

A PROMCODE server **MUST** support OSLC Discovery capabilities defined by [OSLCCore3]. Specifically, they **SHOULD** support LDPC Creation Factories, **MAY** provide a ServiceProviderCatalog and a ServiceProvider resource for Core v2 compatibility. [promcode-38]

A PROMCODE server **MAY** provide a ServiceProvider Resource that can be retrieved at a implementation dependent URI. [promcode-39]

A PROMCODE server **MAY** provide a ServiceProviderCatalog Resource that can be retrieved at a implementation dependent URI. [promcode-40]

A PROMCODE server **MAY** provide a `oslc:serviceProvider` property for their defined resources that will be the URI to a ServiceProvider Resource. [promcode-41]

If a PROMCODE server supports `oslc:Service` and/or `oslc:ServiceProviderCatalog` resources for the compatibility with [OSLCCore2], the server **MUST** supply a value of `http://open-services.net/ns/promcode` for the property `oslc:domain` on either `oslc:Service` or `oslc:ServiceProviderCatalog` resources. [promcode-42]

5.9.2 Creation Capabilities

A PROMCODE server **SHOULD** support Creation Factories and list them in the Service Provider Resource as defined by OSLC Core. A PROMCODE server **SHOULD** support Resource Shapes for Creation Factories as defined in [OSLCCore3] [promcode-43]

5.9.3 Query Capabilities

A PROMCODE Server **SHOULD** support query capabilities, as defined by [OSLCCore3]. Servers **MAY** also provide `oslc:ResourceShape` on `oslc:QueryCapability` resources as defined by [OSLC-Shapes]. [promcode-44]

6. Vocabulary Terms and Constraints

OSLC PROMCODE Version 1.0. Part 2: Vocabulary defines the vocabulary terms and constraints for OSLC PROMCODE resources. These terms and constraints are specified according to [OSLC-Vocabulary]. OSLC PROMCODE Version 1.0. Part 3: Constraints defines the constraints for OSLC resources based on [OSLC-Shapes].

7. Conformance

7.1 Conformance Targets

The PROMCODE specification is based on [OSLCCore3], and therefore both PROMCODE servers and clients **MUST** conform to OSLC servers and clients defined in [OSLCCore3].

7.2 Conformance Clauses

The following table summarizes the conformance clauses.

Clause Number	Requirement
promcode-1	The PROMCODE specification is based on OSLC Core 3.0 [OSLCCore3]. PROMCODE servers and clients **MUST** be compliant with both the [OSLCCore3] specification and the PROMCODE specification, and **SHOULD** follow all the guidelines and recommendations in both of these specifications.
promcode-2	A PROMCODE server **MUST** implement the vocabulary defined in 6. Vocabulary Terms and Constraints of this specification with the exception of two abstract classes `ManagedItem` and `ManagedItemCollection`.
promcode-3	OSLC services **MAY** ignore unknown content and OSLC clients **MUST** preserve unknown content.
promcode-4	OSLC service **MUST** support resource operations via standard HTTP operations.
promcode-5	OSLC services **MAY** provide paging for resources but only when specifically requested by client.
promcode-6	OSLC services **SHOULD** support request for a subset of a resource's properties via the oslc.properties URL parameter retrieval via HTTP GET.
promcode-7	OSLC services **MAY** support partial update of resources using patch semantics and **MAY** support via HTTP PUT.
promcode-8	OSLC servers **SHOULD** support OSLC Core 3.0 Discovery, **MAY** provide a ServiceProviderCatalog and **SHOULD** provide a ServiceProvider resource for Core v2 compatibility.
promcode-9	OSLC servers **MAY** provide LDPC creation factories to enable resource creation of PROMCODE resources via HTTP POST.
promcode-10	OSLC servers **SHOULD** provide query capabilities to enable clients to query for resources.
promcode-11	OSLC query capabilities **SHOULD** support the OSLC Core Query Syntax and **MAY** use other query syntax.
promcode-12	OSLC Services **MAY** offer delegated UI dialogs (creation and selections) specified via OSLC Core 3.0 Delegated Dialogs and **SHOULD** include discovery through a ServiceProvider resource for OSLC v2 compatibility.
promcode-13	OSLC Services **MAY** offer UI previews for resources that may be referenced by other resources specified via OSLC Core 3.0 Preview and **SHOULD** include discovery through a server resource for OSLC v2 compatibility.
promcode-14	OSLC Services **MAY** support Basic Auth and should do so only over HTTPS.
promcode-15	OSLC Services **MAY** support OAuth and can indicate the required OAuth URLs via the ServiceProviderCatalog or ServiceProvider resources.
promcode-16	OSLC Services **MAY** provide error responses using OSLC Core 3.0 defined error formats.

Clause Number	Requirement
promcode-17	OSLC services **MUST** provide a Turtle representation for HTTP GET requests and **SHOULD** support Turtle representations on POST and PUT requests.
promcode-18	OSLC services **SHOULD** provide an RDF/XML representation for HTTP GET requests and **SHOULD** support RDF/XML representations on POST and PUT requests.
promcode-19	OSLC services **SHOULD** provide a XML representation for HTTP GET, POST and PUT requests that conform to the Core 2.0 Guidelines for XML.
promcode-20	OSLC services **MUST** provide JSON-LD representations for HTTP GET, POST and PUT requests that conform to the Core Guidelines for JSON-LD.
promcode-21	OSLC services **MAY** provide HTML representations for HTTP GET requests.
promcode-22	A PROMCODE server **MUST** provide Turtle and JSON-LD, and **SHOULD** provide RDF/XML and XML representations. The XML and JSON representations **SHOULD** follow the guidelines outlined in the OSLC Core Representations Guidance to maintain compatibility with [OSLCCore2].
promcode-23	A PROMCODE client requesting RDF/XML **SHOULD** be prepared for any valid RDF/XML document. A PROMCODE client requesting XML **SHOULD** be prepared for representations that follow the guidelines outlined in the OSLC Core Representations Guidance.
promcode-24	A PROMCODE server **SHOULD** support an [X]HTML representation and a user interface (UI) preview as defined by UI Preview Guidance.
promcode-25	A PROMCODE server **MUST** accept Turtle and JSON-LD representations and **SHOULD** accept RDF/XML and XML representations. A PROMCODE server accepting RDF/XML **SHOULD** be prepared for any valid RDF/XML document. For XML A PROMCODE server **SHOULD** be prepared for representations that follow the guidelines outlined in the OSLC Core Representations Guidance.
promcode-26	A PROMCODE server **MUST** provide Turtle and JSON-LD, **SHOULD** provide RDF/XML and XML, and **MAY** provide Atom Syndication Format XML representations.
promcode-27	When A PROMCODE client request: • `text/turtle` A PROMCODE server **MUST** respond with Turtle representation. • `application/ld+json` A PROMCODE server **MUST** respond with JSON-LD representation. • `application/rdf+xml` A PROMCODE server **SHOULD** respond with RDF/XML representation without restrictions. • `application/xml` A PROMCODE server **SHOULD** respond with OSLC-defined abbreviated XML representation as defined in the OSLC Core Representations Guidance. • `application/atom+xml` A PROMCODE server **SHOULD** respond with Atom Syndication FOrmat XML representation as defined in the OSLC Core Representations Guidance. The Atom Syndication Format XML representation **SHOULD** use RDF/XML representation without restrictions for the atom:content entries representing the resource representations.
promcode-28	A client **MAY** request a subset of a resource's properties as well as properties from a referenced resource. In order to support this behavior, a server **MAY** support the `oslc.properties` and `oslc.prefix` URL parameter on a HTTP GET request on individual resource request or a collection of resources by query. If the oslc.properties parameter is omitted on the request, then all resource properties **MUST** be provided in the response.
promcode-29	A client **MAY** request that a subset of a resource's properties be updated by using the PATCH method.
promcode-30	For compatibility with [OSLCCore2], A PROMCODE server **MAY** also support partial update by identifying those properties to be modified using the `oslc.properties` URL parameter on a HTTP PUT request.

Clause Number	Requirement
promcode-31	If the parameter `oslc.properties` contains a valid resource property on the request that is not provided in the content, the server **MUST** set the resource's property to a null or empty value. If the parameter `oslc.properties` contains an invalid resource property, then a 409 Conflict **MUST** be returned.
promcode-32	For multi-valued properties that contain a large number of values, it may be difficult and inefficient to add or remove property values. A PROMCODE server **MAY** provide support for a partial update of the multi-valued properties as defined by OSLC Core Partial Update.
promcode-33	A PROMCODE client **MAY** request the creation of a PROMCODE domain resource via HTTP/POST operation to a PROMCODE server that supports creation of a resource using Creation Factories as defined in [OSLCCore3].
promcode-34	Alternatively, creation of a domain resource **MAY** be done outside of the PROMCODE framework, i.e., by using an existing project management tool that creates a new resource and makes the resource visible to the PROMCODE framework.
promcode-35	A PROMCODE server **MUST** support the HTTP GET method to obtain the representation of a resource. The detailed behavior of the GET method is described in [LDP].
promcode-36	A PROMCODE server **MAY** support the HTTP PUT method or the HTTP PATCH method to update resources. The detailed behavior of the PUT and PATCH methods is described in [LDP].
promcode-37	A PROMCODE server **MAY** support the HTTP DELETE method to delete resources. The detailed behavior of the DELETE method is described in [LDP].
promcode-38	A PROMCODE server **MUST** support OSLC Discovery capabilities defined by [OSLCCore3]. Specifically, they **SHOULD** support LDPC Creation Factories, **MAY** provide a ServiceProviderCatalog and a ServiceProvider resource for Core v2 compatibility.
promcode-39	A PROMCODE server **MAY** provide a ServiceProvider Resource that can be retrieved at a implementation dependent URI.
promcode-40	A PROMCODE server **MAY** provide a ServiceProviderCatalog Resource that can be retrieved at a implementation dependent URI.
promcode-41	A PROMCODE server **MAY** provide a `oslc:serviceProvider` property for their defined resources that will be the URI to a ServiceProvider Resource.
promcode-42	If a PROMCODE server supports `oslc:Service` and/or `oslc:ServiceProviderCatalog` resources for the compatibility with [OSLCCore2], the server **MUST** supply a value of `http://open-services.net/ns/promcode` for the property `oslc:domain` on either `oslc:Service` or `oslc:ServiceProviderCatalog` resources.
promcode-43	A PROMCODE server **SHOULD** support Creation Factories and list them in the Service Provider Resource as defined by OSLC Core. A PROMCODE server **SHOULD** support Resource Shapes for Creation Factories as defined in [OSLCCore3]
promcode-44	A PROMCODE Server **SHOULD** support query capabilities, as defined by [OSLCCore3]. Servers **MAY** also provide `oslc:ResourceShape` on `oslc:QueryCapability` resources as defined by [OSLC-Shapes].

Appendix A. Change History

This section is non-normative.

Below is a summary of some of the changes in this draft.

- 2017-12-08 Working Draft was newly created

Appendix B. Acknowledgments

This section is non-normative.

Members of the OSLC PROMCODE TC:
Amsden, Mr. James - IBM
Horiuchi, Mr. Yoshio - IBM
Kamimura, Dr. Tom - Nanzan University
LaRochelle, Mr. Robert - IBM
Matsumoto, Mr. Shigeaki - NEC Corporation
Matsuoka, Mr. Shinji - Fujitsu Limited
Speicher, Steve - IBM
Wakao, Mr. Masaki - IBM
Watanabe, Mr. Takeshi - IBM
Yabuta, Mr. Kazuo - Nanzan University
Yoshida, Dr. Hiroyuki - Nanzan University

The PROMCODE TC would like to thank the members of the PROMCODE Consortium for their constructive discussions during the development of the specification. Thanks also go to Arthur Ryman and Kazuhiko Funakoshi who helped in creating technical content at an early stage of the project.

Appendix C. References

C.1 Normative references

[HTTP11]
R. Fielding, Ed.; J. Reschke, Ed.. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7230.html

[LDP]
Steve Speicher; John Arwe; Ashok Malhotra. *Linked Data Platform 1.0*. 26 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/ldp/

[OSLC-Shapes]
Arthur Ryman; Jim Amsden. *OSLC Core Version 3.0. Part6: Resource Shapes*. URL: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/core-shapes.html

[OSLC-Vocabulary]
Jim Amsden; Martin Sarabura. *OSLC Core Version 3.0. Part7: Vocabulary*. URL: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/core-vocab.html

[OSLCCore2]
Dave Johnson; Steve Speicher. *Open Services for Lifecycle Collaboration Core Specification Version 2.0*. URL: https://archive.open-services.net/bin/view/Main/OslcCoreSpecification

[OSLCCore3]
Jim Amsden; Martin Sarabura. *OSLC Core 3.0*. URL: https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/oslc-core.html

[RFC2119]
S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice. URL: https://datatracker.ietf.org/doc/html/rfc2119

[rdf11-concepts]
Richard Cyganiak; David Wood; Markus Lanthaler. *RDF 1.1 Concepts and Abstract Syntax*. URL: http://www.w3.org/TR/rdf11-concepts/

C.2 Informative references

[PMBOK5]
A Guide to Project Management Body of Knowledge, 5th ed.. URL: http://www.pmi.org/

[PROMCODE13]
PROMCODE (PROject Management of COntracted Delivery), Interface Specification, Version 1. URL: http://www.promcode.org/

[PROMCODE14]
M. Aoyama; K. Yabuta; T. Kamimura; S. Inomata; T. Chiba; T. Niwa; K Sakata. *A Resource-Oriented Services Platform for Managing Software Supply Chains and its Experience*. URL: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6928949&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6928949

[ROA]
Roy T. Fielding; Richard N. Taylor. *Principled Design of the Modern Web Architecture*. May 2002. URL: https://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf

[SPMC]
G. Ruhe; C. Wohlin (eds.). *Software Project Management in a Changing World*. 2014. URL: https://dl.acm.org/citation.cfm?id=2677111

[WEBARCH]
Ian Jacobs; Norman Walsh. *Architecture of the World Wide Web, Volume One*. 15 December 2004. W3C Recommendation. URL: https://www.w3.org/TR/webarch/

