PROMCODE

Project Management of COntracted Delivery for software supply chain

Interface Specification

(Version 1)

October 22, 2013

PROMCODE Consortium

Nanzan University
IBM Corporation
FUJITSU LIMITED
NEC Corporation
NTT DATA CORPORATION
Hitachi, Ltd.
Nomura Research Institute, Ltd.

http://www.promcode.org
Copyright © Nanzan University 2013 All rights reserved.
Copyright © IBM Corporation 2013 All rights reserved.
Copyright © FUJITSU LIMITED 2013 All rights reserved.
Copyright © NEC Corporation 2013 All rights reserved.
Copyright © NTT DATA CORPORATION 2013 All rights reserved
Copyright © Hitachi, Ltd. 2013 All rights reserved.
Copyright © Nomura Research Institute, Ltd. 2013 All rights reserved.

This version is only for limited distribution.
# Table of Contents

1. **INTRODUCTION** .......................................................................................................................... 5
   1.1 Motivation................................................................................................................................. 5  
   1.2 Approach .................................................................................................................................. 6

2. **PROMCODE MODELING FRAMEWORK** .................................................................................... 7

3. **PROMCODE DOMAIN MODEL SPECIFICATION** ......................................................................... 8
   3.1 Domain Model.......................................................................................................................... 8  
   3.1.1 ScopeItem............................................................................................................................... 8  
   3.1.2 WorkItem .............................................................................................................................. 9  
   3.1.3 Artifact .................................................................................................................................. 10  
   3.1.4 ManagedItem ......................................................................................................................... 11  
   3.1.5 Change .................................................................................................................................. 11  
   3.1.6 Issue ...................................................................................................................................... 12  
   3.1.7 Measure ................................................................................................................................. 12  
   3.1.8 Measurement ......................................................................................................................... 12  
   3.2 Examples of Project Models......................................................................................................... 13  
   3.2.1 Applying to Progress Management...................................................................................... 13  
   3.2.2 Applying Quality Management............................................................................................ 14

4. **PROMCODE SERVICE SPECIFICATION** .................................................................................. 15
   4.1 Overview..................................................................................................................................... 15  
   4.2 Compliance ............................................................................................................................... 15  
   4.2.1 Specification Versioning ........................................................................................................ 16  
   4.2.2 Namespaces ........................................................................................................................... 16  
   4.2.3 Resource Formats .................................................................................................................. 16  
   4.2.4 Authentication ....................................................................................................................... 17  
   4.2.5 Error Responses .................................................................................................................... 17  
   4.3 PROMCODE Resource Definitions ............................................................................................ 17  
   4.3.1 Resource Definitions .............................................................................................................. 17  
   4.3.2 OSLC Core Properties .......................................................................................................... 18  
   4.3.3 PROMCODE ManagedItemProperties .................................................................................. 19  
   4.3.4 Resource ScopeItem .............................................................................................................. 20  
   4.3.5 Resource WorkItem ................................................................................................................ 20  
   4.3.6 Resource Artifact .................................................................................................................. 21  
   4.3.7 Resource Measurement ......................................................................................................... 21  
   4.3.8 Resource Measure ............................................................................................................... 22
4.3.9 Resource Issue ................................................................................................................................22
Issue Properties .............................................................................................................................................22
4.3.10 Resource Change ............................................................................................................................22
4.4 SERVICE PROVIDER CAPABILITIES ...........................................................................................................23
4.4.1 Service Provider Resources ............................................................................................................23
4.4.2 Query Capabilities ..........................................................................................................................23
4.4.3 Delegated UIs ......................................................................................................................................23
4.5 COMMON PRACTICES FOR ADOPTION .......................................................................................................23
4.5.1 Define Concrete Class ....................................................................................................................24
4.5.2 Extended Properties ........................................................................................................................24

5 REFERENCES ................................................................................................................................................. 25
1 Introduction

This document defines the PROMCODE Domain Model Specification and PRMCODE Service Specification based on the PROMCODE architecture proposed by Next Generation Project Management Data Exchange Architecture Consortium, or PROMCODE consortium.

1.1 Motivation

One of the major obstacles for real-time exchange of management data in collaborative project management is a large diversity of management data models used by different projects.

Figure 1 illustrates a typical scenario where Supplier X and Supplier Y are involved in the software development Project A managed by Acquirer A. Supplier Y and Z are also involved in Project B managed by Acquirer B. It is not easy for Supplier Y to efficiently use a common project management tool since it needs to process both of different management data models of Project A and B. This situation leads to extensive use of general purpose management system such as spreadsheet. A spreadsheet manages a set of management data for each project. It is common that a project manager has to handle the management manually without using any sophisticated tools. Although Figure 1 shows only two layers of contract relationship, a large-scale project typically forms a multiple layers of supply chain consisting of a number of organizations. It causes an error prone and time-consuming procedure in exchanging management data across the organizations.

There are two major differences of management data models across the organizations. The first is difference of data used for managing each project. For example, the name, granularity, and structure of development activities may be different between organizations, and hence, the name, content, and purpose of the artifact produced by different activities may be different.

The second is difference of data formats. Although data may be represented in a two dimensional
table format, usages of rows and columns may be different. For example, some project places actual data on a column next to that of planned data while another project may place actual data on the next row of planned data. Some project may place all management data into only one large table while another may use several tables to represent whole project data. In addition, even in a single software development organization, there may be no standard schema and each department or each project may use its own schema of management data.

In such situation, if a project management tool is developed for the data model of Project A in Figure 1, it is not easy to apply it to the data model of Project B. This is because, firstly, the semantic structures of management data depends on physical structure of a table. It is difficult to specify the correspondence between different table format. Secondly, each project uses its own development process. It is also difficult to specify the correspondence between data of different schema.

1.2 Approach

Our approach attempts to introduces a common interface to use the data for project management, independent of physical formats of management data that can support any data used for. Essentially, project management data is considered to exist in multi-dimensional information space, consisting of many types of information represented, while table data formats is two-dimensional projections of the space. Because difference of formats is made by different methods of projection, it is difficult to find common structure between different formats. If we can use a format which can appropriately represent the essential semantic structures, it will be easy to design a common interface to access data.

Using Linked Data technology, we can flexibly define semantic structures as resources and links between resources and, then we can define a common interface for exchanging data of any semantic structures. For example in Figure 1, if a project management tool of Supplier Y implements the Linked Data interface, it can be used to exchange management data with both Acquirer A and Acquirer B whose data models have different semantic structures.

In addition, defining a domain model at an appropriate abstraction level, we can use it to deal with a variety of data models of projects and to define a standard schema of resources. Supplier Y in Figure 1 can use this standard schema to convert its management data for both Acquirer A and Acquirer B.
2 PROMCODE Modeling Framework

Figure 2 illustrates PROMIS PROMCODE Modeling Framework which represents the relationship of the domain model and the resource definition, and their usages in actual projects.

![PROMCODE Modeling Framework](image)

In this framework, PROMCODE Domain Model is an abstract data model of the collaborative project management domain.

PROMCODE Resource Definition defines a schema for exchanging project management data as linked data resources. Each resource is defined by mapping an abstract class in the domain model to a corresponding resource.

Project Model is a concrete data model of each project. Each project should define its own project model as a set of subclasses of abstract classes defined in the domain model.

Project Data is an instance data of the project model. Since it is also instance data of abstract classes defined in the domain model, it is represented as a resource defined by the resource definition.

The left half of the framework in Figure 2 shows models independent of implementation technology for data exchange while the right half shows their representations based on Linked Data as a particular implementation technology. By separating models and their representations, it is easy to adopt any implementation technology other than Linked Data.

Moreover, the upper half of the framework shows the abstract model and its representation which is used to address differences between projects at an abstract level, while the lower half shows own model and its data of each project. This document defines the PROMCODE Domain Model and PROMCODE Resource Definition in the upper half. Each project can use this to define its own project model and then the representation of concrete project management data is automatically defined as linked data resources. Once a project management tool implements PROMCODE Resource Definition, it can be used for exchanging actual project management data. Especially when used for providing management data for mulcarriersuirers, the same tool can be used without customizaton because no specific data conversion is required.
3 PROMCODE Domain Model Specification

3.1 Domain Model

Figure 3 illustrates PROMIS Domain Model.

![PROMCODE Domain Model](image)

**Figure 3** 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 following part of this section.

Each of the following subsections describes each class in the domain model.

3.1.1 ScopeItem

A ScopeItem is an entity that represents a scope of work from an acquirer’s view of a software development contract. It represents a unit of value to be accomplished by the software a supplier in the contract will develop. For example, it may represent a function required, a use case in which the
software will be used, a requirement which an acquirer expects, or a screen which will provide some concrete function to the user of the software.

Because a ScopeItem is not an activity, it cannot be started nor ended. Because it is not a produced artifact, it does not have any measurable entity.

An acquirer can use a set of ScopeItems as managed units to manage a whole scope of development. Both an acquirer and a supplier also use a set of ScopeItems to estimate the scale of development and so the size of each ScopeItem should be estimated. There should be agreement between an acquirer and a supplier on what kind of ScopeItem should be used and on how large each ScopeItem is. A change of some ScopeItem or its estimated size needs another agreement.

A ScopeItem can be decomposed into finer grain ScopeItems to be used in detailed management. In that case, a coarse grain ScopeItems may be used to aggregate a set of finer grain ScopeItems.

(1) Super Class
   ManagedItem

(2) Attributes
   1) plannedSize: Decimal [0.. 1]
      Estimated size agreed by both acquirer and supplier.
   2) actualSize: Decimal [0.. 1]
      Actual size agreed by both acquirer and supplier.

(3) Links
   1) composedBy: ScopeItem [0.. 1]
      Ancestors of this ScopeItem.
   2) Consists Of: ScopeItem [*]
      Descendants of this ScopeItem.
   3) Requires: WorkItem [*]
      WorkItems required to implement this ScopeItem.
   4) Produces: Artifact [*]
      Artifacts produced when implementing this ScopeItem.

3.1.2 WorkItem

A WorkItem is an entity to represent 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 grain work such as document writing, reviewing, or coding.

WorkItem is managed unit of activity required to implement a ScopeItem or to produce an Artifact.

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

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

(1) Super Class
ManagedItem

(2) Attributes
1) Phase: String [0.. 1]
   Name of development phase such as Analysis, Design, or Implementation. While these
development phases can be modeled as three subclasses of WorkItem, it may also be modeled as
three kinds of Phase which is a subclass of WorkItem. Using the phase attribute may lead a
simple hierarchy structure of WorkItem.
2) plannedStartDate: DateTime [0.. 1]
   Planned date to start this workItem.
3) actualStartDate: DateTime [0..1]
   Actual date to start this workItem.
4) plannedEndDate: DateTime [0..1]
   Planned date to end this workItem.
5) actualEndDate: DateTime [0..1]
   Actual date to end this victim.

(3) Links
1) representedBy: Person [0.. 1]
   The person responsible for the progress of this WorkItem who may or may not actually do this
   WorkItem.
2) composedBy: WorkItem [0.. 1]
   Ancestors of this WorkItem.
3) Consists Of: WorkItem [*]
   Descendants of this WorkItem.
4) requiredBy: ScopeItem [0.. 1]
   ScopeItem which requires this WorkItem.
5) Produces: Artifact [*]
   Artifacts produced by this WorkItem.

3.1.3 Artifact
An Artifact is an entity to represent an output of the development project such as design
documents, source code, test report, and so on.

An Artifact is produced by a Work Item to implement a ScopeItem.

An Artifact can be measured using some measure, and their measured values may vary at each
point of time on a project. The quality of an Artifact is managed by comparing planned and actual
measures.

An Artifact can be decomposed into finer grain Artifacts to be used in detailed management. A
coarse grained Artifacts may be used to aggregate a set of finer grain Artifacts.
1) composedBy: Artifact [0.. 1]
   Ancestors of this Artifact.
2) Consists Of: Artifact [*]
   Descendants of this Artifact.
3) Produced By: WorkItem [*]
   WorkItems required to produce this Artifact.
4) Produced By: ScopeItem [*]
   Scopeltem implemented by producing this Artifact.
5) measuredBy: Measurement [*]
   A measurement which measures this Artifact.

3.1.4 ManagedItem

ManagedItem is a super class which abstracts four kinds of managed entities such as ScopeItem, WorkItem, Artifact, and Issue.

(1) Attributes
1) Identifier: String [1]
   Identifier.
2) Title: String [1]
   Name.
3) Description: String [0.. 1]
   Text which describes of this ManagedItem.

(2) Links
1) relatedBy: Issue [*]
   Issue related to this ManagedItem.
2) Supersedes: Change [0.. 1]
   A change which links a ManagedItem superseded by this ManagedItem.
3) supersededBy: Change [0.. 1]
   A change which links a ManagedItem superseding this ManagedItem. Change. next should be a ManagedItem of the same type as this ManagedItem.

3.1.5 Change

Change manages change history of ManagedItems.

(1) Attributes
1) Created: DateTime [0.. 1]
   The date on which ManagedItems are changed.
2) Authorized: DateTime [0.. 1]
   The date on which change of ManagedItems is authorized.

(2) Links
1) **Previous**: ManagedItem [0.. 1]
   ManagedItem superseded. It should be of the same type as the next ManagedItem. When omitted, this Change represents the creation of a ManagedItem.

2) **Next**: ManagedItem [0.. 1]
   ManagedItem superseding. It should be of the same type as the previous ManagedItem. When omitted, this Change represents the deletion of a ManagedItem.

### 3.1.6 Issue
An Issue is ManagedItems to represent an item such as a to dolem, risk, todo, and so on. Currently, PROMCODE has not yet used this class. It is reserved to be used for Issue Management.

1) **Super Class**
   ManagedItem

2) **Links**
   - Relates: ManagedItem [*]
     ManagedItems related to this Issue.

### 3.1.7 Measure
A Measure is a numbered value representing some quality aspect of Artifacts.

1) **Attributes**
   1) plannedValue: Decimal [0.. 1]
      Planned value.
   2) actualValue: Decimal [0.. 1]
      Actual Value.

2) **Links**
   - Measurement: Measurement [1]
     A measurement which measures this measure.

### 3.1.8 Measurement
A Measurement represents a date on which some measures of an Artifact are measured.

1) **Attributes**
   - Date: DateTime [1]
     The date on which an Artifact is measured.

2) **Links**
   1) Measure: Measure [*]
      Measures measured by this Measurement. One Measurement can measure zero or more Measures.
   2) Measures: Artifact [1]
      Artifact measured by this Measurement.
3.2 Examples of Project Models

3.2.1 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 subfunctions. Each Subfunction has phases of Analysis, Design, and Coding activities. Note that the real management tables are more complex than shown in in real cases. Function-Sub Function forms a tree structure with several levels. There are more activities required to implement SubFunctions.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Sub Function</th>
<th>Analysis</th>
<th>Design</th>
<th>Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Start</td>
<td>End</td>
<td>Start</td>
</tr>
</tbody>
</table>

Figure 4 illustrates the corresponding project 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 SubFunctions 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.

Figure 4 Example of Project Model for Progress Management
3.2.2 Applying Quality Management

Table 2 shows a typical quality management table. Main managed entity is Module which is grouped under Requirement. Each Module is measured using several KPIs including lines of code, number of test cases, and number of defects found.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Module</th>
<th>Line of Code</th>
<th>#Test Case</th>
<th>#Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Planned</td>
<td>Actual</td>
<td>Planned</td>
</tr>
<tr>
<td>R1</td>
<td>M1-1</td>
<td>2,000</td>
<td>2,130</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>M1-2</td>
<td>1,500</td>
<td>1,450</td>
<td>45</td>
</tr>
<tr>
<td>R2</td>
<td>M2-1</td>
<td>2,000</td>
<td>1,980</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>M2-2</td>
<td>1,000</td>
<td>950</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 5 illustrates the corresponding project model. Requirement is a subclass of ScopeItem and Module is a subclass of Artifact. There are three subclasses of Measure. A Requirement is used to group Modules that implement it. 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.

![Figure 5 Example of Project Model for Quality Management](image-url)
4 PROMCODE Service Specification

4.1 Overview

PROMCODE service is a service to exchange project management data defined by the PROMCODE domain model specification.

PROMCODE service is based on OSLC Core Specification Version 2.0 [2] and defines PROMCODE Resource Definition which is mapped from the PROMCODE domain model and PROMCODE Service Provider which allows a client program to access the resources through HTTP protocol.

4.2 Compliance

This specification is based on OSLC Core Specification Version 2.0. PROMCODE consumers and service providers MUST be compliant with both the core specification and this PROMCODE specification, and SHOULD follow all the guidelines and recommendations in both these specifications.

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

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Level</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown properties and content</td>
<td>MAY/MUST</td>
<td>OSLC services MAY ignore unknown content and OSLC clients MUST preserve unknown content</td>
</tr>
<tr>
<td>Resource Operations</td>
<td>MUST</td>
<td>OSLC service MUST support resource operations via standard HTTP operations</td>
</tr>
<tr>
<td>Resource Paging</td>
<td>MAY</td>
<td>OSLC services MAY provide paging for resources but only when specifically requested by service consumer.</td>
</tr>
<tr>
<td>Partial Resource Representations</td>
<td>MUST/MAY</td>
<td>OSLC services MUST support request for a subset of a resource's properties via the oslc.properties URL parameter retrieval via HTTP GET and MAY support via HTTP PUT</td>
</tr>
<tr>
<td>Partial Update</td>
<td>MAY</td>
<td>OSLC services MAY support partial update of resources using patch semantics</td>
</tr>
<tr>
<td>Service Provider Resources</td>
<td>MUST/MAY</td>
<td>OSLC service providers MAY provide a Service Provider Catalog and MUST provide a Service Provider resource</td>
</tr>
<tr>
<td>Creation Factories</td>
<td>MAY</td>
<td>OSLC service providers MAY provide creation factory resource for PROMCODE resource.</td>
</tr>
<tr>
<td>Query Capabilities</td>
<td>MUST</td>
<td>OSLC service providers MUST provide query capabilities to enable clients to query for resources</td>
</tr>
<tr>
<td>Query Syntax</td>
<td>MUST</td>
<td>OSLC query capabilities MUST support the OSLC Core Query Syntax</td>
</tr>
<tr>
<td>Delegated UI Dialogs</td>
<td>MAY</td>
<td>OSLC Services MAY offer delegated UI dialogs (for both creation and selection)</td>
</tr>
<tr>
<td>Requirement</td>
<td>Level</td>
<td>Meaning</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UI Preview</td>
<td>MAY</td>
<td>OSLC Services MAY offer UI previews for resources that may be referenced by other resources</td>
</tr>
<tr>
<td>HTTP Basic</td>
<td>MAY</td>
<td>OSLC Services MAY support Basic Authentication and SHOULD only do so only over HTTPS</td>
</tr>
<tr>
<td>OAuth Authentication</td>
<td>MAY</td>
<td>OSLC Services MAY support OAuth and MAY indicate the required OAuth URLs via the service provider resource.</td>
</tr>
<tr>
<td>Error Responses</td>
<td>MAY</td>
<td>OSLC Services MAY provide error responses using Core defined error formats.</td>
</tr>
<tr>
<td>RDF/XML Representations</td>
<td>MUST</td>
<td>OSLC services MUST support RDF/XML representations for OSLC Defined Resources</td>
</tr>
<tr>
<td>XML Representations</td>
<td>MAY</td>
<td>OSLC services MAY support XML representations that conform to the OSLC Core Guidelines for XML</td>
</tr>
<tr>
<td>JSON Representations</td>
<td>MAY</td>
<td>OSLC services MAY support JSON representations; those which do MUST conform to the OSLC Core Guidelines for JSON</td>
</tr>
<tr>
<td>HTML Representations</td>
<td>MAY</td>
<td>OSLC services MAY provide HTML representations for GET requests</td>
</tr>
</tbody>
</table>

4.2.1 Specification Versioning

See [Core Specification Version 2.0 - Specification Versioning](#).

4.2.2 Namespaces

In addition to the namespace URIs and namespace prefixes oslc, rdf, dcterms and foaf defined in the [Core Specification Version 2.0](#), PROMCODE defines the namespace URI of [http://promcode.org/ns/pm#](http://promcode.org/ns/pm#) with a preferred namespace prefix promcode_pm.

4.2.3 Resource Formats

In addition to the requirements for [Core Specification Version 2.0 - OSLC Defined Resource Representations](#), this section outlines further refinements and restrictions.

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

(1) PROMCODE Providers MUST support RDF/XML representations with media-type application/rdf+xml. PROMCODE Consumers MUST be prepared to deal with any valid RDF/XML document.

For HTTP POST/PUT requests on all PROMCODE and OSLC Core defined resource types,

(2) PROMCODE Providers MAY support RDF/XML representations with media-type application/rdf+xml. PROMCODE Consumers MUST be prepared to deal with any valid RDF/XML document.

For HTTP GET response formats for Query requests,

(3) PROMCODE Providers MUST support RDF/XML representations with media-type application/rdf+xml.
4.2.4 Authentication

See Core Specification Version 2.0 - Authentication. PROMCODE places no additional constraints on authentication.

4.2.5 Error Responses

See Core Specification Version 2.0 - Error Responses. PROMCODE places no additional constraints on error responses.

4.3 PROMCODE Resource Definitions

4.3.1 Resource Definitions

Fig 6 describes the overview of PROMCODE resource definition mapped from PROMCODE domain model.

![Diagram of PROMCODE Resource Definitions]

Figure 6  PROMCODE Resource Definitions
In Figure 6, the hierarchical structure expresses that the child resource type includes properties defined in the parent resource type. For example, ScopeItem includes properties defined in ManagedItem and OSLC Core properties.

The following sections describe the detailed definition of each resource. In the sections, the namespace prefix is omitted for properties in PROMCODE namespace.

### 4.3.2 OSLC Core Properties

This section describes properties defined in OSLC Core which are used for the following PROMCODE resources:

1. ScopeItem
2. WorkItem
3. Artifact
4. Issue

<table>
<thead>
<tr>
<th>Table 4</th>
<th>CSLC Core Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Occurs</td>
</tr>
<tr>
<td>oslc:shortTitle</td>
<td>zero-or-one</td>
</tr>
<tr>
<td>dcterms:description</td>
<td>zero-or-one</td>
</tr>
<tr>
<td>dcterms:title</td>
<td>exactly-one</td>
</tr>
<tr>
<td>dcterms:identifier</td>
<td>exactly-one</td>
</tr>
<tr>
<td>dcterms:subject</td>
<td>zero-or-many</td>
</tr>
</tbody>
</table>
4.3.3 PROMCODE ManagedItemProperties

This section describes common properties among the following PROMCODE resources:

1. ScopeItem
2. WorkItem
3. Artifact
4. Issue

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcterms:type</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>String</td>
<td></td>
<td>A short string representation for the concrete type.</td>
</tr>
<tr>
<td>Name</td>
<td>Occurs</td>
<td>Read-Only</td>
<td>Value-type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>Supersedes</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>Change</td>
<td></td>
</tr>
<tr>
<td>supersededBy</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>Change</td>
<td></td>
</tr>
</tbody>
</table>
4.3.4 Resource ScopeItem

- Name: ScopeItem
- URI: http://promcode.org/ns/pm#ScopeItem

ScopeItem Properties

ScopeItem has the following properties in addition to the common properties of OSLC Core and PROMCODE ManagedItem.

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>plannedSize</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Decimal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actualSize</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Decimal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Relationship Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>consistsOf</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>ScopeItem</td>
<td>Reference to a child of nested ScopeItem</td>
</tr>
<tr>
<td>composedBy</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>ScopeItem</td>
<td>Reference to a parent of nested ScopeItem</td>
</tr>
<tr>
<td>Requires</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>WorkItem</td>
<td></td>
</tr>
<tr>
<td>Produces</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>Artifact</td>
<td></td>
</tr>
</tbody>
</table>

4.3.5 Resource WorkItem

- Name: WorkItem
- URI: http://promcode.org/ns/pm#WorkItem

WorkItem Properties

WorkItem has the following properties in addition to the common properties of OSLC Core and PROMCODE ManagedItem.

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>plannedStartDate</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>DateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actualStartDate</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>DateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>plannedEndDate</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>DateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actualEndDate</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>DateTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Occurs</td>
<td>Read-Only</td>
<td>Value-type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-----------</td>
<td>------------</td>
<td>-------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>Relationship Properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>consistsOf</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>WorkItem</td>
<td>Reference to a child of nested Work Item</td>
</tr>
<tr>
<td>composedBy</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>WorkItem</td>
<td>Reference to a parent of nested Work Items</td>
</tr>
<tr>
<td>requiredBy</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>ScopeItem</td>
<td></td>
</tr>
<tr>
<td>produces</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>Artifact</td>
<td></td>
</tr>
<tr>
<td>representedBy</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>Any</td>
<td>It is likely that the target resource will be a foaf:Person but that is not necessarily the case.</td>
</tr>
</tbody>
</table>

4.3.6 Resource Artifact

- Name: Artifact
- URI: [http://promcode.org/ns/pm#Artifact](http://promcode.org/ns/pm#Artifact)

Artifact Properties

Artifact has the following properties in addition to the common properties of OSLC Core and PROMCODE ManagedItem.

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Relationship Properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>consistsOf</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>Artifact</td>
<td>Reference to a child of nested Artifacts</td>
</tr>
<tr>
<td>composedBy</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>Artifact</td>
<td>Reference to a parent of nested Artifacts</td>
</tr>
<tr>
<td>producedBy</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>ScopeItem or WorkItem</td>
<td></td>
</tr>
<tr>
<td>measuredBy</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>Measurement</td>
<td></td>
</tr>
</tbody>
</table>

4.3.7 Resource Measurement

- Name: Measurement
- URI: [http://promcode.org/ns/pm#Measurement](http://promcode.org/ns/pm#Measurement)

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Measurement: Start of additional properties</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>zero-or-one</td>
<td>FALSE</td>
<td>DateTime</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 4.3.8 Resource Measure
- Name: Measure
- URI: [http://promcode.org/ns/pm#Measure](http://promcode.org/ns/pm#Measure)

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcterms:type</td>
<td>Zero-or-many</td>
<td>FALSE</td>
<td>String</td>
<td></td>
<td>A short string representation for the concrete type.</td>
</tr>
<tr>
<td>plannedValue</td>
<td>Zero-or-one</td>
<td>FALSE</td>
<td>Decimal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actualValue</td>
<td>Zero-or-one</td>
<td>FALSE</td>
<td>Decimal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Occurs</td>
<td>Read-Only</td>
<td>Value-type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>measurement</td>
<td>Zero-or-one</td>
<td>FALSE</td>
<td>Resource</td>
<td>Measurement</td>
<td></td>
</tr>
</tbody>
</table>

### 4.3.9 Resource Issue
- Name: Issue
- URI: [http://promcode.org/ns/pm#Issue](http://promcode.org/ns/pm#Issue)

### Issue Properties
Issue has the following properties in addition to the common properties of OSLC Core and PROMCODE ManagedItem.

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>relates</td>
<td>zero-or-many</td>
<td>FALSE</td>
<td>Resource</td>
<td>Scopeltem, WorkItem, Artifact or Issue</td>
<td></td>
</tr>
</tbody>
</table>

### 4.3.10 Resource Change
- Name: Change
- URI: [http://promcode.org/ns/pm#Change](http://promcode.org/ns/pm#Change)

<table>
<thead>
<tr>
<th>Name</th>
<th>Occurs</th>
<th>Read-Only</th>
<th>Value-type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
</table>
### 4.4 Service Provider Capabilities

#### 4.4.1 Service Provider Resources

Service providers MUST provide one or more oslc:ServiceProvider resources as defined by Core Specification Version 2.0 - Service Provider Resource. Discovery of OSLC Service Provider Resources MAY be via one or more OSLC Service Provider Catalog Resources, or may be discovered by some other and/or additional Provider-specific means out with the scope of this specification. The oslc:Service resources referenced by this oslc:ServiceProvider MUST have an oslc:domain of http://promcode.org/ns/pm#.

Service providers MAY provide one more more oslc:ServiceProviderCatalog resources as defined by Core Specification Version 2.0 - Service Provider Resources. Any such catalog resources MUST include at least one oslc:domain of http://promcode.org/ns/pm#. Discovery of top-level OSLC Service Provider Catalog Resources is out with the scope of this specification.

Service providers MUST give an oslc:serviceProvider property on all OSLC Defined Resources. This property MUST refer to an appropriate oslc:ServiceProvider resource.

#### 4.4.2 Query Capabilities

Service providers MUST support query capabilities, as defined by Core Specification Version 2.0 - Query Capabilities.

#### 4.4.3 Delegated UIs

PROMCODE service providers MAY support the selection and creation of resources by delegated web-based user interface dialogs Delegated UIs as defined by OSLC Core.

### 4.5 Common Practices for Adoption

As described in Section 1.3, the Project Data is the data which is exchanged among organizations. The Project Data is an RDF instance mapped from Project Model which is a concrete model based on the PROMCODE Domain Model.
This section describes the common practice to define the Project Data for the Project Model.

4.5.1 Define Concrete Class

As described in Section 3.2, usually each project defines their Project Model by declaring subclasses of classes in the Domain Model. To define subclasses in RDF resource, the “dcterms:type” property of ManagedItem can be used.

(Note: In OSLC 3.0, dcterms:type will be deprecated, so we will move to the new method)

For example, Function class which is a subclass of the ScopeItem can be defined by setting the type property “FunctionItem” as follows:

```xml
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:promis_pm="http://promis.jp/ns/pm/"
  <promis_pm:ScopeItem rdf:about="http://localhost:8080/oslc-excel/rest/services/projectA/data/SII_1"
    <dcterms:type>FunctionItem</dcterms:type>
    <dcterms:title>システム</dcterms:title>
    <dcterms:identifier>SII_1</dcterms:identifier>
  </promis_pm:ScopeItem>
</rdf:RDF>
```

Figure 7 An Example of Resource Definition

4.5.2 Extended Properties

Extended properties can be defined by declaring extended properties in own namespace. For example, to add an extended property “ownerGroup” for WorkItem:

- Namespace : http://my.bbb.com/prj/
- Property : ownerGroup
- Example:

```xml
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:promis_pm="http://promis.jp/ns/pm/"
  xmlns:myext="http://my.bbb.com/prj/"
  <promis_pm:WorkItem rdf:about="http://localhost:8080/oslc-excel/rest/services/projectA/data/WI3_1.1.2"
    <dcterms:title></dcterms:title>
    <dcterms:identifier>WI3_1.1.2</dcterms:identifier>
    <myext:ownerGroup>グループ1</myext:ownerGroup>
    ...
  </promis_pm:WorkItem>
</rdf:RDF>
```

Figure 8 An Example of Adding an Extended Property
5 References