Date: June 2021

Systems Modeling Application Programming Interface (API) and Services

Version 1.0
Release 2021-05

Submitted in response to Systems Modeling Language (SysML®) v2 API and Services RFP (ad/2018-06-03) by:

88Solutions Corporation	InterCax LLC
Dassault Systèmes	Lockheed Martin Corporation
GfSE e.V.	Model Driven Solutions, Inc.
IBM	PTC
INCOSE	Simula Research Laboratory AS

Table of Contents

List of Tables

List of Figures

0 Submission Introduction

0.1 Submission Overview

This proposed specification is submitted in response to the Systems Modeling Language (SysML®) v2 API and Services RFP (ad/2018-16-03). It is being submitted along with two other specifications that are being submitted together in response to the Systems Modeling Language (SysML®) v2 Request for Proposals (RFP):

- Kernel Modeling Language (KerML), Version 1.0
- OMG Systems Modeling Language (SysML), Version 2.0

These specifications are all being submitted together because the APIs and services defined in this specification are intended to work with the language metamodels defined in the other two specification. Together, these three specifications respond to the full SysML v2 vision, as captured jointly by the two RFPs.

Release note. The present document is an update to the initial submission document submitted to OMG in August 2020.

0.2 Submission Submitters

The following OMG member organizations are jointly submitting this proposed specification:

- 88Solutions Corporation
- GfSE e.V.
- IBM
- INCOSE
- InterCax LLC
- Lockheed Martin Corporation
- Model Driven Solutions, Inc.
- Dassault Système
- PTC
- Simula Research Laboratory AS

The submitters also thankfully acknowledge the support of over 60 other organizations that participated in the SysML v2 Submission Team.

0.3 Submission - Issues to be discussed

The SysML v2 API and Services RFP does not request that any issue be discussed by submitters.

0.4 API and Services Requirements Tables

0.4.1 Mandatory API & Services Requirements Table

Table 1. Mandatory API & Services Requirements Table

Reqt. ID	Reqt. Name	Text
API 1	API and Services Architecture	This group includes general requirements related to the architecture and design of the API.

Reqt. ID	Reqt. Name	Text
API 1.1	API Architecture	Proposals for SysML v2 API and Services shall specify: 1. Platform-independent model (PIM) that specifies the services and the operations provided by the API. Services are collections of operations. Inputs and outputs of each operation shall be specified and be conformant to the SysML v2 meta-model and resulting UML profile [SysML v2]. The platform-independent model shall be defined using an open standard, such as UML2 or IDL, so that it can be used to auto-generate platform-specific bindings. 2. Mappings to bind the platform-independent model (PIM) to each of platform-specific models (PSMs). The mappings shall be defined using an open standard, such as QVT. 3. Platform-specific models (PSMs) as bindings of the PIM to OSLC 3.0 and one or more commonly used technology platforms, such as, but not limited to, REST/HTTP, Java, C#, Javascript, or GraphQL. The platform-specific models shall also include API documentation for each of the services and their operations. Proposals are also encouraged to leverage the latest industry standards and technologies for API specification and cross-language code generation (bindings), such as Apache Thrift, OpenAPI, and Swagger Codegen.
API 1.2	API Infrastructure Services	Proposals for SysML v2 API and Services shall provide the following infrastructure-level service specifications. • Service discovery to get all the available services • Service access to get the handle of a service **Supporting Information:** Refer to DDS [DDS] and CORBA [CORBA] as examples
API 1.3	Design Constraints	Proposals for SysML v2 API and Services shall allow extensions of the platform-independent model (PIM) and platform-specific bindings (PSMs). **Supporting Information:** Refer to DDS [DDS] and CORBA [CORBA] as examples
API 2	API and Services Conformance	The requirements in this group are related to measuring the conformance level of SysML v2 modeling environments to the platform-specific models (PSMs). SysML v2 modeling environments may implement one or more platform-specific models in the SysML v2 API and Service specification, as described in section 6.2 and API 1.1 requirement above.
API 2.1	API and Services Conformance Criteria	Proposals for SysML v2 API and Services shall provide measurable conformance criteria for all services in the proposal.

Reqt. ID	Reqt. Name	Text
API 2.2	API and Services Conformance Evaluation Method and Test Cases	Proposals for SysML v2 API and Services shall provide an evaluation method and associated test cases to assess conformance for all the services in the proposal.
API 2.3	Functional Thread Conformance	Proposals for SysML v2 API and Services shall provide test cases to assess conformance of SysML v2 modeling environments to functional threads that use a combination of services. Four functional threads are specified below. Proposals can specify test cases for additional functional threads relevant to model-based systems engineering. • FT 1 - Assessing the impact of a requirement change during system development • FT 2 - Investigation and review of system design and relevant analyses after a failure during system operation is reported • FT 3 – Coordination and concurrent development of systems involving multiple disciplines, such as systems, hardware, software, simulation, project management, manufacturing, and operations • FT 4 Tracing system artifacts downstream and upstream during system development and during system operations and maintenance
SV 1	Service Scope, Conditions, and Response Requirements	The requirements in this group are applicable to all the services in the scope of this RFP.
SV 1.1	Service Scope	Proposals for SysML v2 API and Services shall provide a mechanism to specify the model and its version, or collection of models and their versions, that are in scope for a service request. **Supporting Information:** Example: If the Model Navigation Service is used by a consumer to get all elements of a specific type, then the consumer must have a mechanism to specify a specific model or collection of models in which elements of that type will be searched.

Reqt. ID	Reqt. Name	Text
SV 1.2	Pre- and Post-Conditions	Proposals for SysML v2 API and Services shall provide a model-based specification for the pre-conditions and post-conditions for each service. Pre-conditions include the list of conditions that must be true for the service to initiate, and post-conditions include the list of conditions that must be true after the service has responded, whether successfully or unsuccessfully. **Supporting Information:** **E**xample: An example pre-condition for the Model Update Service may be that the model element being updated must exist and not be checked-out (or being modified). An example post-condition for the same service may be that the model must be in a valid state after the service has responded.
SV 1.3	Standard Response Model	Proposals for SysML v2 API and Services shall provide a Standard Response Model that represents the response structure for all the services in the proposal. A Standard Response Model will make it feasible for service consumers to process service responses in a standard manner. **Supporting Information:** REST/HTTP APIs provide common response status codes specified by the HTTP protocol [HTTP/1.1].

Reqt. ID	Reqt. Name	Text
SV 2	Model Navigation Service Requirement	Proposals for SysML v2 API and Services shall specify a service to navigate a SysML v2 model, as described in detail below. (a) Define starting point(s) for navigation i. Get model (top-level/root element) ii. Get an element by its unique identifier iii. Get elements by their type in a given scope using direct or recursive strategy, e.g. get elements of type Block directly in a given package, or get elements of type Block recursively in the context of a given package. (b) Get all properties and their values for a given element, including the meta-data for each property such as its name, type, and multiplicity. (c) Get relationships given one or more of the following criteria, including meta-data for each relationship such as its name and type. i. Source element of the relationship ii. Target element of the relationship iii. Type of the relationship **Supporting Information:** The terms element, property, relationship, type, and package used here are based on the latest version of SysML/UML standard.
SV 3	Model Creation Service Requirement	Proposals for SysML v2 API and Services shall specify a service to create a model element with a unique identifier given the name, type, and owning element for the model element.
SV 4	Model Update Service Requirement	Proposals for SysML v2 API and Services shall specify a service to update model elements, such as updating the name, type, property values, or the owning element.
SV 5	Model Deletion Service Requirement	Proposals for SysML v2 API and Services shall specify a service to delete model elements according to deletion semantics. The deletion semantics will specify other model elements that may need to be deleted or updated to ensure that the model is valid. **Supporting Information:** Example: The deletion semantics for deleting an element may specify deletion of all owned elements and all outgoing/incoming relationships from/to the given element.

Reqt. ID	Reqt. Name	Text
SV 6	External Relationship Management Services Requirements	The requirements in this group are related to creating, reading, updating, and deleting relationships between elements in a SysML v2 model and elements in non-SysML models and other SysML v2 models. These relations are termed as “external relationships” and are labeled as 2 in Figure 2. Assumption: These set of SysML v2 services assume that there is a service available to access and reference model elements in non-SysML v2 models.
SV 6.1	Get External Relationships	Proposals for SysML v2 API and Services shall specify a service to get all external relationships for a given SysML v2 model element. The type of external relationship may be specified as a filter. **Supporting Information:** Refer to SV 6.5 below for types of external relationships.
SV 6.2	Create External Relationships	Proposals for SysML v2 API and Services shall specify a service to create an external relationship given its type and two model elements, one of which must be a SysML v2 model element. **Supporting Information:** Refer to SV 6.5 below for types of external relationships.
SV 6.3	Update External Relationships	Proposals for SysML v2 API and Services shall specify a service to update an external relationship, such as updating the name or type of the relationships, or updating the ends of the relationship. **Supporting Information:** Refer to SV 6.5 below for types of external relationships.
SV 6.4	Delete External Relationships	Proposals for SysML v2 API and Services shall specify a service to delete an external relationship.

Reqt. ID	Reqt. Name	Text
SV 6.5	Types and Behaviors of External Relationships	Proposals for SysML v2 API and Services shall specify the types of external relationships and their semantics that includes at least the following: 1. Reference / Trace: Provide navigation between related elements 2. Version tracking: Track versions of the related elements over time 3. Executable model wrapping: Transfer inputs from the source element (at one end) to execute a model (at the other end) and to transfer results from the model execution back to the source element 4. Data transfer: Bi-directionally transfer attribute values from model element at one end to the model element at the other end of the relationship 5. Surrogate: Represent a model element in one domain as a surrogate of a model element in another domain, e.g. represent a physical part in a SysML model as a surrogate of a part in a PLM system 6. Model generation: Generate/update model (or model element) at one end of the relationship from the model (or model element) at the other end, e.g. generating code from behavior elements in a SysML v2 model, or generating a simulation model from system ports, connectors, and item flows in a SysML v2 model.

0.4.2 Non-Mandatory API & Services Requirements Table

Table 2. Non-Mandatory API & Services Requirements Table

Reqt. ID	Reqt. Name	Text
SVA 1	Model Analysis Services Requirements	The requirements in this group are related to services for creating, querying, updating, deleting, and executing analysis-related model elements in SysML v2 models.
SVA 1.1	Analysis Creation, Query, Update, and Deletion Services	Proposals for SysML v2 API and Services may specify services for creating, querying, updating, and deleting analysis-related model elements by reusing the model construction services (sections 6.5.2 and 6.6.2) and model navigation and query services (sections 6.5.2 and 6.6.1). **Supporting Information:** Section 6.5.2.8 (requirement group ANL 1) of the SysML v2 language RFP specifies requirements for representing analysis-related concepts in SysML v2 language, such as Analysis, Analysis Model, Analysis Objective, Analysis Scenarios, Analysis Result, and others. The generic services for model navigation, model query, and model construction can be used for providing CRUD operations on analysis-related model elements in a SysML v2 model.

Reqt. ID	Reqt. Name	Text
SVA 1.2	Analysis Execution Service	Proposals for SysML v2 API and Services may specify a service to execute analysis models. The service shall consume analysis inputs and produce analysis results conformant to the SysML v2 language. **Supporting Information:** Section 6.5.2.8 (ANL 1.09) of SysML v2 RFP states: Analysis models can be defined natively in SysML (e.g. parametric model or behavior model) or externally (e.g. equation-based math models, finite element analysis models, or computational fluid dynamics models). Example: Consider the following two scenarios for an analysis model to compute the energy collected by an array of solar panels on a spacecraft in a 24-hour period. In the first scenario, the analysis model for energy collection is defined entirely using parametric and activity-related concepts in a SysML v2 model. In the second scenario, the analysis model in a SysML v2 model is only a specification that is linked using an external relationship to a MATLAB function for energy calculation. A SysML v2 modeling environment may support one or both the scenarios and make them available via the Analysis Execution Service API. For the first scenario, the Analysis Execution Service implementation may take the inputs specified in the SysML v2 model, invoke an execution engine provided by the SysML v2 modeling environment to execute the parametric/activity model for energy collection, and return the results as a service response. For the second scenario, the Analysis Execution Service implementation may invoke the MATLAB code for energy calculation with the given inputs in the SysML v2 model and return the output of the MATLAB code execution as a service response.
SVC 1	Advanced Model Construction Services Requirements	The requirements in this group are related to advanced services for creating, updating, and deleting elements in the SysML v2 model beyond the basic model creation services (see requirements SV 3 to SV 6 in section 6.5.2 under Mandatory Requirements.
SVC 1.1	Bulk Model Creation Service	Proposals for SysML v2 API and Services may specify a service to create multiple model elements (in bulk), each with their unique identifier, given the name, type, and owning element for each model element.
SVC 1.2	Bulk Model Update Service	Proposals for SysML v2 API and Services may specify a service to update multiple model elements (in bulk), such as updating the name, type, property values, or the owning element for each model element.

Reqt. ID	Reqt. Name	Text
SVC 1.3	Bulk Model Deletion Service	Proposals for SysML v2 API and Services may specify a service to delete multiple model elements (in bulk) according to deletion semantics. The deletion semantics will specify other model elements that may need to be deleted or updated to ensure that the model is valid. **Supporting Information:** Example: The deletion semantics for deleting an element may specify deletion of all owned elements and all outgoing/incoming relationships from/to the given element.
SVC 1.4	Bulk External Relationship Management Service	The requirements in this group are related to services for bulk creation, update, and deletion of external relationships.
SVC 1.4.1	Bulk External Relationship Creation Service	Proposals for SysML v2 API and Services may specify a service to create multiple external relationships (in bulk), given the relationship type and two model elements (one of which must be a SysML v2 model element) for each external relationship that needs to be created. **Supporting Information:** Refer to SV 6.5 (section 6.5.2 under Mandatory Requirements) for types of external relationships.
SVC 1.4.2	Bulk External Relationship Update Service	Proposals for SysML v2 API and Services may specify a service to update multiple external relationships (in bulk), such as updating the name or type or participant model elements for each relationship. **Supporting Information:** Refer to SV 6.5 (section 6.5.2 under Mandatory Requirements) for types of external relationships.
SVC 1.4.3	Bulk External Relationship Deletion Service	Proposals for SysML v2 API and Services may specify a service to delete multiple external relationships (in bulk).
SVG 1	General Services Requirements	The requirements in this group are related to general services, such as timestamp and UUID generation, and API call back.

Reqt. ID	Reqt. Name	Text
SVG 1.1	Timestamp Generation	Proposals for SysML v2 API and Services may specify a service to read and process standard-formatted timestamps on existing model elements and to provide standard-formatted timestamps for new model elements, or new versions of existing model elements. **Supporting Information:** An example of a timestamp, including both date and time (with time zone) is [2009-06-15T13:45:30-04:00]. Refer to ISO 8601 for a standard representation of date and time. The service can create a new timestamp from the current clock time and time zone information, or provide year, month, day, time, and time zone details given a timestamp. Requirement CRC 1.6.2 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that SysML v2 language provide a way to represent timestamp meta-data for model elements.
SVG 1.2	UUID Generation	Proposals for SysML v2 API and Services may specify a service to generate a universally unique identifier (UUID). **Supporting Information:** Requirement CRC 1.2.2 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that SysML v2 language provide a way to represent a universally unique identifier for each model element that cannot be changed.
SVG 1.3	API Call Back	Proposals for SysML v2 API and Services may specify a service to create or remove callbacks. **Supporting Information:** A callback is an asynchronous, out-of-band request that a service can send to some other services in response to certain events [OpenAPI]. For example, a subscription functionality is a form of a callback where service consumers can subscribe to certain events of a service and receive notification when that event occurs. Most modern APIs provide a way to define callbacks, especially for other services and user-interfaces to respond to events. Callbacks are defined in the OpenAPI specification that submitters are encouraged to leverage for SysML 2 API and Services (see section 6.3).
SVM 1	Model Management Services Requirements	The requirements in this group are related to services for version and configuration management of SysML v2 models, and data control services.
SVM 1.1	Model Versioning Services	The requirements in this group are related to services for version management of SysML v2 models and model elements.

Reqt. ID	Reqt. Name	Text
SVM 1.1.1	Define MCI Definition Default Rules	Proposals for SysML v2 API and Services may specify services to define the rules to determine the type of model elements that will be versioned, which is referred to as a Model Configuration Item (MCI), and types of changes in a MCI that qualify as a version update. **Supporting Information:** A Model Configuration Item is a specific portion of the system model that is maintained in a controlled fashion, i.e. has a unique ID and version history. MCI can be defined in different granularities, from an individual fine-grained Model Element, a set of Model Elements, a set of Elements, to the entire Model. Any MCI can contain another MCI. Examples include Container (i.e. Package), Block, Model Element, and View Definition. Refer to section A.2 (Glossary) for more details. Requirement CRC 1.6.1 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that the SysML v2 language provide a way to represent version meta-data for model elements.
SVM 1.1.2	Create and Delete Versions of a MCI	Proposals for SysML v2 API and Services may specify services that can create new versions or delete a specific version of a Model Configuration Item. **Supporting Information:** Services for model construction will use this service to create new versions or delete existing versions of model elements. For example, Model Creation Service (SV 3, section 6.5.2) will require this service to create the first version of a model element if it is a MCI. Model Update Service (SV 4, section 6.5.2) will require this service to create a new version of an existing model element if it is a MCI.
SVM 1.1.3	Get Versions of a MCI	Proposals for SysML v2 API and Services may specify services to get the version history of a MCI. The version history provides a chronological list of all the versions of the given MCI.
SVM 1.1.4	Compare Versions of a MCI	Proposals for SysML v2 API and Services may specify services to compare two or more versions of a MCI and provide a list of differences in a standard way.
SVM 1.2	Model Configuration Services	Proposals for SysML v2 API and Services may specify services to create baseline configurations with MCIs, create branch configurations with MCIs from a given baseline configuration, and merge branch configurations with each other or with the originating baseline configuration. **Supporting Information:** The intent of this service (or set of services) is to provide a standard way to define stable releases of a SysML v2 model as baselines, and allow multiple parallel branches of concurrent development on a SysML v2 model.

Reqt. ID	Reqt. Name	Text
SVM 1.3	Model Data Control Services	Proposals for SysML v2 API and Services may specify services to create, read, update, and delete data protection control markings on model elements. **Supporting Information:** Requirement CRC 1.6.3 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that SysML v2 language provide a way to represent data protection control markings on model elements, such as ITAR, security, and proprietary classifications.
SVQ 1	Model Query Services Requirements	The requirements in this group are related to services for creating, executing, and managing queries for SysML v2 models, beyond basic model navigation services covered in SV 2 (section 6.5.2 under Mandatory Requirements). **Supporting Information:** A query is a precise request for information retrieval from a SysML v2 model (see section A.2 - Glossary).
SVQ 1.1	Standard Query Model	Proposals for SysML v2 API and Services may provide a Standard Query Model to specify the scope, input criteria, and the outputs requested in a query. The elements used to specify the input criteria and the outputs shall conform to the SysML v2 meta-model and the resulting UML profile. The Standard Query Model shall serve as a language- and platform-independent interface definition for a query to a SysML v2 model, and can be used to generate language- or platform-specific queries for SysML v2 modeling environments. **Supporting Information:** **E**xample: Consider an example query to get all electrical interfaces in system S, where electrical interfaces are identified as ports typed by standard value types AC or DC. A query can be created conforming to the Standard Query Model—scope of the query is System S (immediate level or recursive), the input criteria is port type is AC or port type is DC, and the requested output element type is Port. Now consider two different SysML v2 modeling environments, one that provides a REST/HTTP binding (PSM implementation) and one that provides a SQL binding. The Standard Query Model can be mapped to both the bindings, e.g. GET calls to one or more REST API endpoints in one SysML v2 modeling environment and SQL-based query in the other SysML v2 modeling environment.

Reqt. ID	Reqt. Name	Text
SVQ 1.2	Query Execution Service	Proposals for SysML v2 API and Services may specify a service to execute queries specified using the Standard Query Model. **Supporting Information:** Example: Consider two different SysML v2 modeling environments that implement the query execution service based on their specific platform bindings (REST/HTTP versus SQL/JDBC). In one implementation, the query execution service is implemented as a REST endpoint that accepts a query specified using the Standard Query Model in JSON format and returns result conforming to the Standard Response Model (SV 1.1, section 6.5.2) in JSON format. In the other implementation, the query execution service is implemented using a Java-based JDBC call that accepts a query specified using the Standard Query Model, converts it to a SQL-based query expression, runs the SQL query, and returns the query result conforming to the Standard Response Model as a Java object.
SVT 1	Model Transformation Services Requirements	The requirements in this group are related to services for transforming SysML models.
SVT 1.1	Model Transformation Service	Proposals for SysML v2 may specify services to create, read, update, delete, and execute model transformations specified as SysML models.
SVT 1.2	SysML Version to Version Transformation	Proposals for SysML v2 may provide services to transform a model in a previous version of SysML to a model in the next version of SysML, beginning with the transformation from the current version of SysML v1 to SysML v2. Proposals may provide services to transform models in earlier versions of SysML v1 (e.g. 1.3 and 1.4 if 1.5 is the current version) to SysML v2 models. Proposals must state the specific SysML v1 versions supported beyond the current version. **Supporting Information:** This includes the ability to transform the abstract syntax, concrete syntax and semantics. Some of the SysML v2 execution semantics are specified in other specifications including fUML, PSCS, and PSSM.
SVT 1.3	Model to Textual Syntax Generation Service	Proposals for SysML v2 API and Services may provide a service to generate the textual representation of a SysML v2 model (or model elements) conforming to the concrete textual syntax of the SysML v2 language. The scope of model elements shall be specified as an input to the service. **Supporting Information:** The concrete textual syntax for SysML v2 language is presented in requirement LNG 1.4.4 of the SysML v2 RFP under Non-Mandatory Features. The concrete textual syntax provides a textual human-readable representation of a SysML v2 model.

Reqt. ID	Reqt. Name	Text
SVT 1.4	Textual Syntax to Model Generation Service	Proposals for SysML v2 API and Services may provide a service to generate SysML v2 models (or model elements) from a textual representation conforming to the concrete textual syntax of the SysML v2 language. **Supporting Information:** The concrete textual syntax for SysML v2 language is presented in requirement LNG 1.4.4 of the SysML v2 RFP under Non-Mandatory Features. The concrete textual syntax provides a textual human-readable representation of a SysML v2 model.
SVV 1	Model View and Viewpoint Management Services Requirements	The requirements in this group are related to services for creating, querying, updating, and deleting viewpoints and views in SysML v2 models.
SVV 1.1	Viewpoint Creation, Query, Update, and Deletion Services	Proposals for SysML v2 API and Services may specify services for creating, querying, updating, and deleting viewpoints by reusing the model construction services (sections 6.5.2 and 6.6.2) and model navigation and query services (sections 6.5.2 and 6.6.1). **Supporting Information:** Section 6.5.2.1 (requirement group CRC 1.5) of the SysML v2 language RFP specifies requirements for representing view and viewpoint-related concepts in SysML v2 language. The generic services for model navigation, model query, and model construction can be used for providing CRUD operations on viewpoint-related model elements in a SysML v2 model.
SVV 1.2	View Creation, Query, Update, and Deletion Services	Proposals for SysML v2 API and Services may specify services for creating, querying, updating, and deleting views by reusing the model construction services (sections 6.5.2 and 6.6.2) and model navigation and query services (sections 6.5.2 and 6.6.1). Services to create, query, update, and delete views shall provide a mechanism to specify the viewpoint—to which the subject views conform—as an input. **Supporting Information:** Section 6.5.2.1 (requirement group CRC 1.5) of the SysML v2 language RFP specifies requirements for representing view and viewpoint-related concepts in SysML v2 language. The generic services for model navigation, model query, and model construction can be used for providing CRUD operations on view-related model elements in a SysML v2 model.

0.4.3 Mandatory API and Services Requirements - Satisfied-by Table

Table 3. Mandatory API and Services Requirements - Satisfied-by Table

ID	Name	Satisfied?	Satisfied-by	Comment
API 1	API and Services Architecture			

ID	Name	Satisfied?	Satisfied-by	Comment
API 1.1	API Architecture	Yes	1 Platform Independent Model (PIM) 2 Platform Specific Models	
API 1.2	API Infrastructure Services	Partial	1 Platform Independent Model (PIM)	
API 1.3	Design Constraints			
API 2	API and Services Conformance			
API 2.1	API and Services Conformance Criteria	Yes		
API 2.2	API and Services Conformance Evaluation Method and Test Cases	Yes		
API 2.3	Functional Thread Conformance	Yes		
SV 1	Service Scope, Conditions, and Response Requirements			
SV 1.1	Service Scope	Yes	1 Platform Independent Model (PIM)	
SV 1.2	Pre- and Post-Conditions	Yes	1 Platform Independent Model (PIM)	
SV 1.3	Standard Response Model	Yes	1 Platform Independent Model (PIM)	
SV 2	Model Navigation Service Requirement	Yes	Element Navigation Service	
SV 3	Model Creation Service Requirement	Yes	Element Creation Service	
SV 4	Model Update Service Requirement	Yes	cf name does not exist	
SV 5	Model Deletion Service Requirement	Yes	cf name does not exist	
SV 6	External Relationship Management Services Requirements	Yes	External Relationship Service	
SV 6.1	Get External Relationships	Yes	External Relationship Service	
SV 6.2	Create External Relationships	Yes	External Relationship Service	
SV 6.3	Update External Relationships	Yes	cf name does not exist	
SV 6.4	Delete External Relationships	Yes	cf name does not exist	
SV 6.5	Types and Behaviors of External Relationships	Partial	cf name does not exist	

0.4.4 Non-Mandatory API and Services Requirements - Satisfied-by Table

Table 4. Non-Mandatory API and Services Requirements - Satisfied-by Table

ID	Name	Satisfied?	Satisfied-by	Comment
SVA 1	Model Analysis Services Requirements	No		
SVA 1.1	Analysis Creation, Query, Update, and Deletion Services	No		
SVA 1.2	Analysis Execution Service	No		
SVC 1	Advanced Model Construction Services Requirements	Partial		
SVC 1.1	Bulk Model Creation Service	Partial		
SVC 1.2	Bulk Model Update Service	Partial		
SVC 1.3	Bulk Model Deletion Service	Partial		
SVC 1.4	Bulk External Relationship Management Service	Partial		
SVC 1.4.1	Bulk External Relationship Creation Service	Partial		
SVC 1.4.2	Bulk External Relationship Update Service	Partial		
SVC 1.4.3	Bulk External Relationship Deletion Service	Partial		
SVG 1	General Services Requirements			
SVG 1.1	Timestamp Generation	Yes	1 Platform Independent Model (PIM) 2 Platform Specific Models	
SVG 1.2	UUID Generation	Yes	1 Platform Independent Model (PIM) 2 Platform Specific Models	
SVG 1.3	API Call Back	No		
SVM 1	Model Management Services Requirements	Partial		
SVM 1.1	Model Versioning Services	Partial		
SVM 1.1.1	Define MCI Definition Default Rules	No		
SVM 1.1.2	Create and Delete Versions of a MCI	Partial	cf name does not exist	

ID	Name	Satisfied?	Satisfied-by	Comment
SVM 1.1.3	Get Versions of a MCI	Partial	cf name does not exist	
SVM 1.1.4	Compare Versions of a MCI	No		
SVM 1.2	Model Configuration Services	Partial		
SVM 1.3	Model Data Control Services	No		
SVQ 1	Model Query Services Requirements	Yes		
SVQ 1.1	Standard Query Model	Yes	Query Service Query	
SVQ 1.2	Query Execution Service	Yes	Query Service Query	
SVT 1	Model Transformation Services Requirements	No		
SVT 1.1	Model Transformation Service	No		
SVT 1.2	SysML Version to Version Transformation	No		
SVT 1.3	Model to Textual Syntax Generation Service	No		
SVT 1.4	Textual Syntax to Model Generation Service	No		
SVV 1	Model View and Viewpoint Management Services Requirements	No		
SVV 1.1	Viewpoint Creation, Query, Update, and Deletion Services	No		
SVV 1.2	View Creation, Query, Update, and Deletion Services	No		

0.4.5 Changed API and Services Requirements Table

Table 5. Changed API and Services Requirements Table

ID	Name	Requirement Text	Change Status	Change Description
SVA 1	Model Analysis Services Requirements	The requirements in this group are related to services for creating, querying, updating, deleting, and executing analysis-related model elements in SysML v2 models.	Modified	30 June 2018 - The original RFP requirement number was SVA

ID	Name	Requirement Text	Change Status	Change Description
SVA 1.1	Analysis Creation, Query, Update, and Deletion Services	Proposals for SysML v2 API and Services may specify services for creating, querying, updating, and deleting analysis-related model elements by reusing the model construction services (sections 6.5.2 and 6.6.2) and model navigation and query services (sections 6.5.2 and 6.6.1). **Supporting Information:** Section 6.5.2.8 (requirement group ANL 1) of the SysML v2 language RFP specifies requirements for representing analysis-related concepts in SysML v2 language, such as Analysis, Analysis Model, Analysis Objective, Analysis Scenarios, Analysis Result, and others. The generic services for model navigation, model query, and model construction can be used for providing CRUD operations on analysis-related model elements in a SysML v2 model.	Modified	30 June 2018 - The original RFP requirement number was SVA1
SVA 1.2	Analysis Execution Service	Proposals for SysML v2 API and Services may specify a service to execute analysis models. The service shall consume analysis inputs and produce analysis results conformant to the SysML v2 language. **Supporting Information:** Section 6.5.2.8 (ANL 1.09) of SysML v2 RFP states: Analysis models can be defined natively in SysML (e.g. parametric model or behavior model) or externally (e.g. equation-based math models, finite element analysis models, or computational fluid dynamics models). Example: Consider the following two scenarios for an analysis model to compute the energy collected by an array of solar panels on a spacecraft in a 24-hour period. In the first scenario, the analysis model for energy collection is defined entirely using parametric and activity-related concepts in a SysML v2 model. In the second scenario, the analysis model in a SysML v2 model is only a specification that is linked using an external relationship to a MATLAB function for energy calculation. A SysML v2 modeling environment may support one or both the scenarios and make them available via the Analysis Execution Service API. For the first scenario, the Analysis Execution Service implementation may take the inputs specified in the SysML v2 model, invoke an execution engine provided by the SysML v2 modeling environment to execute the parametric/activity model for energy collection, and return the results as a service response. For the second scenario, the Analysis Execution Service implementation may invoke the MATLAB code for energy calculation with the given inputs in the SysML v2 model and return the output of the MATLAB code execution as a service response.	Modified	30 June 2018 - The original RFP requirement number was SVA2

ID	Name	Requirement Text	Change Status	Change Description
SVC 1	Advanced Model Construction Services Requirements	The requirements in this group are related to advanced services for creating, updating, and deleting elements in the SysML v2 model beyond the basic model creation services (see requirements SV 3 to SV 6 in section 6.5.2 under Mandatory Requirements.	Modified	30 June 2018 - The original RFP requirement number was SVC
SVC 1.1	Bulk Model Creation Service	Proposals for SysML v2 API and Services may specify a service to create multiple model elements (in bulk), each with their unique identifier, given the name, type, and owning element for each model element.	Modified	30 June 2018 - The original RFP requirement number was SVC1
SVC 1.2	Bulk Model Update Service	Proposals for SysML v2 API and Services may specify a service to update multiple model elements (in bulk), such as updating the name, type, property values, or the owning element for each model element.	Modified	30 June 2018 - The original RFP requirement number was SVC2
SVC 1.3	Bulk Model Deletion Service	Proposals for SysML v2 API and Services may specify a service to delete multiple model elements (in bulk) according to deletion semantics. The deletion semantics will specify other model elements that may need to be deleted or updated to ensure that the model is valid. **Supporting Information:** Example: The deletion semantics for deleting an element may specify deletion of all owned elements and all outgoing/ incoming relationships from/to the given element.	Modified	30 June 2018 - The original RFP requirement number was SVC3
SVC 1.4	Bulk External Relationship Management Service	The requirements in this group are related to services for bulk creation, update, and deletion of external relationships.	Modified	30 June 2018 - The original RFP requirement number was SVC4

ID	Name	Requirement Text	Change Status	Change Description
SVC 1.4.1	Bulk External Relationship Creation Service	Proposals for SysML v2 API and Services may specify a service to create multiple external relationships (in bulk), given the relationship type and two model elements (one of which must be a SysML v2 model element) for each external relationship that needs to be created. **Supporting Information:** Refer to SV 6.5 (section 6.5.2 under Mandatory Requirements) for types of external relationships.	Modified	30 June 2018 - The original RFP requirement number was SVC4.1
SVC 1.4.2	Bulk External Relationship Update Service	Proposals for SysML v2 API and Services may specify a service to update multiple external relationships (in bulk), such as updating the name or type or participant model elements for each relationship. **Supporting Information:** Refer to SV 6.5 (section 6.5.2 under Mandatory Requirements) for types of external relationships.	Modified	30 June 2018 - The original RFP requirement number was SVC4.2
SVC 1.4.3	Bulk External Relationship Deletion Service	Proposals for SysML v2 API and Services may specify a service to delete multiple external relationships (in bulk).	Modified	30 June 2018 - The original RFP requirement number was SVC4.3
SVG 1	General Services Requirements	The requirements in this group are related to general services, such as timestamp and UUID generation, and API call back.	Modified	30 June 2018 - The original RFP requirement number was SVG

ID	Name	Requirement Text	Change Status	Change Description
SVG 1.1	Timestamp Generation	Proposals for SysML v2 API and Services may specify a service to read and process standard-formatted timestamps on existing model elements and to provide standard-formatted timestamps for new model elements, or new versions of existing model elements. **Supporting Information:** An example of a timestamp, including both date and time (with time zone) is [2009-06-15T13:45:30-04:00]. Refer to ISO 8601 for a standard representation of date and time. The service can create a new timestamp from the current clock time and time zone information, or provide year, month, day, time, and time zone details given a timestamp. Requirement CRC 1.6.2 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that SysML v2 language provide a way to represent timestamp meta-data for model elements.	Modified	30 June 2018 - The original RFP requirement number was SVG1
SVG 1.2	UUID Generation	Proposals for SysML v2 API and Services may specify a service to generate a universally unique identifier (UUID). **Supporting Information:** Requirement CRC 1.2.2 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that SysML v2 language provide a way to represent a universally unique identifier for each model element that cannot be changed.	Modified	30 June 2018 - The original RFP requirement number was SVG2
SVG 1.3	API Call Back	Proposals for SysML v2 API and Services may specify a service to create or remove callbacks. **Supporting Information:** A callback is an asynchronous, out-of-band request that a service can send to some other services in response to certain events [OpenAPI]. For example, a subscription functionality is a form of a callback where service consumers can subscribe to certain events of a service and receive notification when that event occurs. Most modern APIs provide a way to define callbacks, especially for other services and user-interfaces to respond to events. Callbacks are defined in the OpenAPI specification that submitters are encouraged to leverage for SysML 2 API and Services (see section 6.3).	Modified	30 June 2018 - The original RFP requirement number was SVG3

ID	Name	Requirement Text	Change Status	Change Description
SVM 1	Model Management Services Requirements	The requirements in this group are related to services for version and configuration management of SysML v2 models, and data control services.	Modified	30 June 2018 - The original RFP requirement number was SVM
SVM 1.1	Model Versioning Services	The requirements in this group are related to services for version management of SysML v2 models and model elements.	Modified	30 June 2018 - The original RFP requirement number was SVM1
SVM 1.1.1	Define MCI Definition Default Rules	Proposals for SysML v2 API and Services may specify services to define the rules to determine the type of model elements that will be versioned, which is referred to as a Model Configuration Item (MCI), and types of changes in a MCI that qualify as a version update. **Supporting Information:** A Model Configuration Item is a specific portion of the system model that is maintained in a controlled fashion, i.e. has a unique ID and version history. MCI can be defined in different granularities, from an individual fine-grained Model Element, a set of Model Elements, a set of Elements, to the entire Model. Any MCI can contain another MCI. Examples include Container (i.e. Package), Block, Model Element, and View Definition. Refer to section A.2 (Glossary) for more details. Requirement CRC 1.6.1 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that the SysML v2 language provide a way to represent version meta-data for model elements.	Modified	30 June 2018 - The original RFP requirement number was SVM1.1

ID	Name	Requirement Text	Change Status	Change Description
SVM 1.1.2	Create and Delete Versions of a MCI	Proposals for SysML v2 API and Services may specify services that can create new versions or delete a specific version of a Model Configuration Item. **Supporting Information:** Services for model construction will use this service to create new versions or delete existing versions of model elements. For example, Model Creation Service (SV 3, section 6.5.2) will require this service to create the first version of a model element if it is a MCI. Model Update Service (SV 4, section 6.5.2) will require this service to create a new version of an existing model element if it is a MCI.	Modified	30 June 2018 - The original RFP requirement number was SVM1.2
SVM 1.1.3	Get Versions of a MCI	Proposals for SysML v2 API and Services may specify services to get the version history of a MCI. The version history provides a chronological list of all the versions of the given MCI.	Modified	30 June 2018 - The original RFP requirement number was SVM1.3
SVM 1.1.4	Compare Versions of a MCI	Proposals for SysML v2 API and Services may specify services to compare two or more versions of a MCI and provide a list of differences in a standard way.	Modified	30 June 2018 - The original RFP requirement number was SVM1.4
SVM 1.2	Model Configuration Services	Proposals for SysML v2 API and Services may specify services to create baseline configurations with MCIs, create branch configurations with MCIs from a given baseline configuration, and merge branch configurations with each other or with the originating baseline configuration. **Supporting Information:** The intent of this service (or set of services) is to provide a standard way to define stable releases of a SysML v2 model as baselines, and allow multiple parallel branches of concurrent development on a SysML v2 model.	Modified	30 June 2018 - The original RFP requirement number was SVM2

ID	Name	Requirement Text	Change Status	Change Description
SVM 1.3	Model Data Control Services	Proposals for SysML v2 API and Services may specify services to create, read, update, and delete data protection control markings on model elements. **Supporting Information:** Requirement CRC 1.6.3 (section 6.5.2 under Mandatory Requirements) of the SysML v2 language RFP requires that SysML v2 language provide a way to represent data protection control markings on model elements, such as ITAR, security, and proprietary classifications.	Modified	30 June 2018 - The original RFP requirement number was SVM3
SVQ 1	Model Query Services Requirements	The requirements in this group are related to services for creating, executing, and managing queries for SysML v2 models, beyond basic model navigation services covered in SV 2 (section 6.5.2 under Mandatory Requirements). **Supporting Information:** A query is a precise request for information retrieval from a SysML v2 model (see section A.2 - Glossary).	Modified	30 June 2018 - The original RFP requirement number was SVQ

ID	Name	Requirement Text	Change Status	Change Description
SVQ 1.1	Standard Query Model	Proposals for SysML v2 API and Services may provide a Standard Query Model to specify the scope, input criteria, and the outputs requested in a query. The elements used to specify the input criteria and the outputs shall conform to the SysML v2 meta-model and the resulting UML profile. The Standard Query Model shall serve as a language- and platform-independent interface definition for a query to a SysML v2 model, and can be used to generate language- or platform-specific queries for SysML v2 modeling environments. **Supporting Information:** **E**xample: Consider an example query to get all electrical interfaces in system S, where electrical interfaces are identified as ports typed by standard value types AC or DC. A query can be created conforming to the Standard Query Model—scope of the query is System S (immediate level or recursive), the input criteria is port type is AC or port type is DC, and the requested output element type is Port. Now consider two different SysML v2 modeling environments, one that provides a REST/HTTP binding (PSM implementation) and one that provides a SQL binding. The Standard Query Model can be mapped to both the bindings, e.g. GET calls to one or more REST API endpoints in one SysML v2 modeling environment and SQL-based query in the other SysML v2 modeling environment.	Modified	30 June 2018 - The original RFP requirement number was SVQ1

ID	Name	Requirement Text	Change Status	Change Description
SVQ 1.2	Query Execution Service	Proposals for SysML v2 API and Services may specify a service to execute queries specified using the Standard Query Model. **Supporting Information:** Example: Consider two different SysML v2 modeling environments that implement the query execution service based on their specific platform bindings (REST/HTTP versus SQL/JDBC). In one implementation, the query execution service is implemented as a REST endpoint that accepts a query specified using the Standard Query Model in JSON format and returns result conforming to the Standard Response Model (SV 1.1, section 6.5.2) in JSON format. In the other implementation, the query execution service is implemented using a Java-based JDBC call that accepts a query specified using the Standard Query Model, converts it to a SQL-based query expression, runs the SQL query, and returns the query result conforming to the Standard Response Model as a Java object.	Modified	30 June 2018 - The original RFP requirement number was SVQ2
SVT 1	Model Transformation Services Requirements	The requirements in this group are related to services for transforming SysML models.	Modified	30 June 2018 - The original RFP requirement number was SVT
SVT 1.1	Model Transformation Service	Proposals for SysML v2 may specify services to create, read, update, delete, and execute model transformations specified as SysML models.	Modified	30 June 2018 - The original RFP requirement number was SVT1

ID	Name	Requirement Text	Change Status	Change Description
SVT 1.2	SysML Version to Version Transformation	Proposals for SysML v2 may provide services to transform a model in a previous version of SysML to a model in the next version of SysML, beginning with the transformation from the current version of SysML v1 to SysML v2. Proposals may provide services to transform models in earlier versions of SysML v1 (e.g. 1.3 and 1.4 if 1.5 is the current version) to SysML v2 models. Proposals must state the specific SysML v1 versions supported beyond the current version. **Supporting Information:** This includes the ability to transform the abstract syntax, concrete syntax and semantics. Some of the SysML v2 execution semantics are specified in other specifications including fUML, PSCS, and PSSM.	Modified	30 June 2018 - The original RFP requirement number was SVT2
SVT 1.3	Model to Textual Syntax Generation Service	Proposals for SysML v2 API and Services may provide a service to generate the textual representation of a SysML v2 model (or model elements) conforming to the concrete textual syntax of the SysML v2 language. The scope of model elements shall be specified as an input to the service. **Supporting Information:** The concrete textual syntax for SysML v2 language is presented in requirement LNG 1.4.4 of the SysML v2 RFP under Non-Mandatory Features. The concrete textual syntax provides a textual human-readable representation of a SysML v2 model.	Modified	30 June 2018 - The original RFP requirement number was SVT3
SVT 1.4	Textual Syntax to Model Generation Service	Proposals for SysML v2 API and Services may provide a service to generate SysML v2 models (or model elements) from a textual representation conforming to the concrete textual syntax of the SysML v2 language. **Supporting Information:** The concrete textual syntax for SysML v2 language is presented in requirement LNG 1.4.4 of the SysML v2 RFP under Non-Mandatory Features. The concrete textual syntax provides a textual human-readable representation of a SysML v2 model.	Modified	30 June 2018 - The original RFP requirement number was SVT4
SVV 1	Model View and Viewpoint Management Services Requirements	The requirements in this group are related to services for creating, querying, updating, and deleting viewpoints and views in SysML v2 models.	Modified	30 June 2018 - The original RFP requirement number was SVV

ID	Name	Requirement Text	Change Status	Change Description
SVV 1.1	Viewpoint Creation, Query, Update, and Deletion Services	Proposals for SysML v2 API and Services may specify services for creating, querying, updating, and deleting viewpoints by reusing the model construction services (sections 6.5.2 and 6.6.2) and model navigation and query services (sections 6.5.2 and 6.6.1). **Supporting Information:** Section 6.5.2.1 (requirement group CRC 1.5) of the SysML v2 language RFP specifies requirements for representing view and viewpoint-related concepts in SysML v2 language. The generic services for model navigation, model query, and model construction can be used for providing CRUD operations on viewpoint-related model elements in a SysML v2 model.	Modified	30 June 2018 - The original RFP requirement number was SVV1
SVV 1.2	View Creation, Query, Update, and Deletion Services	Proposals for SysML v2 API and Services may specify services for creating, querying, updating, and deleting views by reusing the model construction services (sections 6.5.2 and 6.6.2) and model navigation and query services (sections 6.5.2 and 6.6.1). Services to create, query, update, and delete views shall provide a mechanism to specify the viewpoint—to which the subject views conform—as an input. **Supporting Information:** Section 6.5.2.1 (requirement group CRC 1.5) of the SysML v2 language RFP specifies requirements for representing view and viewpoint-related concepts in SysML v2 language. The generic services for model navigation, model query, and model construction can be used for providing CRUD operations on view-related model elements in a SysML v2 model.	Modified	30 June 2018 - The original RFP requirement number was SVV2

1 Scope

The purpose of this standard is to specify the Systems Modeling Application Programming Interface (API) and Services that provide standard services to access, navigate, and operate on KerML-based models, and in particular SysML models. The standard services facilitate interoperability both across SysML modeling environments and between SysML modeling environments and other engineering tools and enterprise services.

The Systems Modeling API and Services specifies the types and details of the requests that can be made and responses that can be received by software applications that are consuming the services to software applications that are providing the services.

The Systems Modeling API and Services specification includes the Platform Independent Model (PIM) - see Clause 7 - and two Platform Specific Models (PSMs) - see Clause 8: REST/HTTP PSM and OSLC PSM.

2 Conformance

This specification defines the Systems Modeling API and Services that provide standard services to access, navigate, and operate on KerML-based models, and in particular SysML models. The specification comprises this document together with the content of the machine-readable files listed on the cover page. If there are any conflicts between this document and the machine-readable files, the machine-readable files take precedence.

A Systems Modeling API and Services Provider tool is a software application that provides the services defined in this specification. A Systems Modeling API and Services Consumer tool is a software application that consumes the services defined in this specification and provided by the Service Provider tool.

A Service Provider tool can conform to this specification at the PSM or PIM level.

1. *PSM-level Conformance* - A Service Provider tool demonstrating PSM-level Conformance implements one or more of the Systems Modeling API and Services PSMs defined in this specification. For example, a Provider tool can implement the REST/HTTP PSM, the OSLC PSM, or both. A Service Provider tool demonstrating PSM-level Conformance shall implement the mandatory services listed below. PSM-level conformance of Service Provider tools ensures interoperability of Service Consumer tools using the PSM across the different Service Providers. See Clause 8.
2. *PIM-level Conformance* - A Service Provider tool demonstrating PIM-level Conformance implements a PSM that is not defined in this specification but is based on Systems Modeling API and Services PIM defined in this specification. The Service Provider tool shall define the PSM and the mapping from PIM to PSM with the goal that the new PSM may become part of future versions of this specification. See Clause 7.

Mandatory Service Conformance - A Service Provider tool *shall* demonstrate the implementation of the following services:

1. Project Service
2. Element Navigation Service
3. Element Creation Service

A Service Provider tool *may* demonstrate the implementation of the following additional services:

1. Element Versioning Service
2. External Relationship Service
3. Query Service

For a tool to demonstrate any of the above forms of conformance, it is sufficient that the tool passes the relevant tests from the Conformance Test Suite specified in Annex A.

3 Normative References

[GraphQL] GraphQL
https://graphql.org/

[Gremlin] Gremlin Graph Traversal Machine and Language
https://tinkerpop.apache.org/gremlin.html

[IRI] *Internationalized Resource Identifiers* (IRI)
https://www.w3.org/International/articles/idn-and-iri/

[KerML] *Kernel Modeling Language (KerML),* Version 1.0
(as submitted with this proposed specification)

[MOFVD] *MOF2 Versioning and Development Lifecycle (MOFVD™),* Version 2.0
https://www.omg.org/spec/MOFVD/2.0

[OpenAPI] *OpenAPI Specification*
https://www.openapis.org/

[OSLC] *Open Services for Lifecycle Collaboration (OSLC)*
http://open-services.net/

[QVT] *MOF Query/View/Transformation (QVT™)*, Version 1.3
https://www.omg.org/spec/QVT/1.3

[SEBoK] *Systems Engineering Body of Knowledge (SEBoK)*
www.sebokwiki.org

[SE Handbook] *INCOSE Systems Engineering Handbook*
https://www.incose.org/products-and-publications/se-handbook

[SMOF] *MOF Support for Semantic Structures (SMOF™)*, Version 1.0
https://www.omg.org/spec/SMOF/1.0

[SPARQL] *SPARQL Query Language for RDF*
https://www.w3.org/TR/rdf-sparql-query/

[SQL] *ISO/IEC 9075:2016, Information technology — Database languages — SQL*
https://www.iso.org/standard/63555.html

[STEP] *ISO 10303-233:2012 (STEP)*
https://www.iso.org/standard/55257.html

[SysML] *OMG Systems Modeling Language (SysML®), Version 2.0*
(as submitted with this proposed specification)

[UML] *Unified Modeling Language (UML),* Version 2.5.1
https://www.omg.org/spec/UML/2.5.1

[UUID] *Universally Unique IDentifier (UUID) URN Namespace*
https://tools.ietf.org/html/rfc4122

[XMI] *XML Metadata Interchange (XMI®)*, Version 2.5.1
https://www.omg.org/spec/XMI/2.5.1

4 Terms and Definitions

To be provided in a future release.

5 Symbols

There are no symbols defined in this specification.

6 Introduction

6.1 API and Services Architecture

The Systems Modeling API and Services includes the following (see Fig. 1):

(1) Platform-Independent Model (PIM) provides a service specification that is consistent with KerML and SysML. The PIM serves as the logical API model and provides a specification of services independent of the platform or technology.

(2) Platform-Specific Models (PSMs) are bindings of the PIM using a particular technology, such as REST/HTTP, SOAP, Java, and .NET. Multiple platform-specific models can exist for a given PIM. Two PSMs are provided in this specification:

- REST / HTTP PSM - a binding of the PIM as a REST / HTTP API using OpenAPI specification.
- OSLC PSM - a binding of the PIM as services based on the OSLC standard.

For each PSM, a mapping is defined. This mapping is used to generate the PSM from the PIM

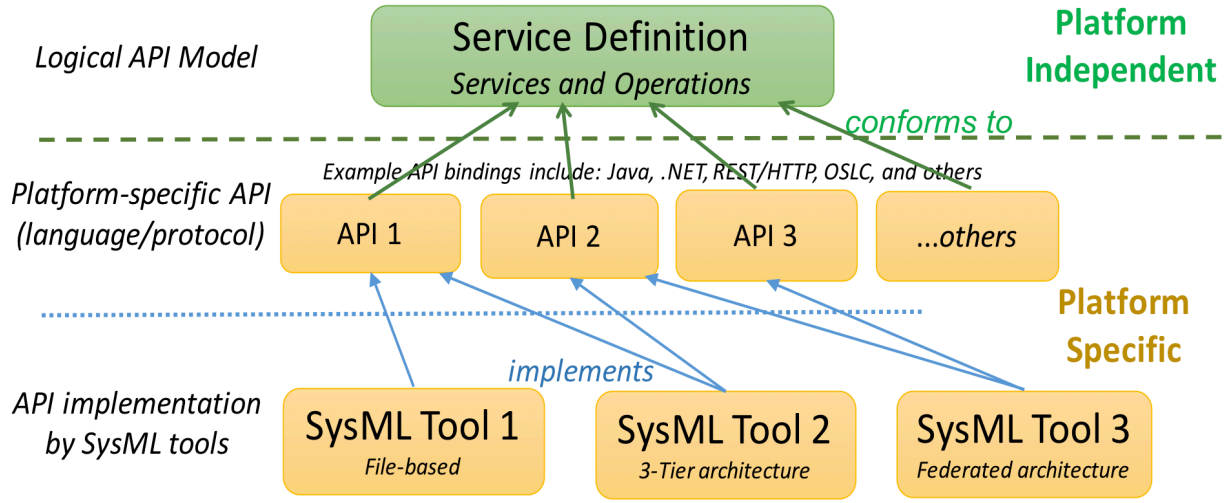

Figure 1. API and Services Architecture

Service specifications in the PIM do not prescribe or constrain the architecture of the SysML modeling environments. For example, SysML modeling environments with file-based, traditional 3-tier database based, or federated microservice-based architectures can all implement one or more PSMs derived from the same service specifications (PIM).

Applications written using a platform-specific binding (PSM) should be interoperable across multiple SysML modeling environments that implement the binding, without requiring any modification to the application.

6.2 Document Conventions

The following stylistic conventions are applied in the presentation of the Platform Independent Model (PIM) of the Systems Modeling API and Services.

Service definitions

1. Names of classes representing services start with an uppercase letter, such as Element Navigation Service.

2. Names of operations representing the API calls available for each service start with a lowercase letter, such as get_element_by_id

Input / output data

1. Names of classes representing data that is the input or output of services start with an uppercase letter, such as Project and Element
2. Names of attributes representing the details of the data that is the input or output of services start with a lowercase letter, such as identifier

The services and operations in the PIM are presented using class diagrams and tables with description of each operation.

The input and output data for services in the PIM are presented using class diagrams followed by detailed descriptions. In the description, the attributes of the input and output data are *italicized.*

6.3 Document Organization

The rest of this document is organized into two major clauses.

- Clause 7. Platform Independent Model (PIM) of the Systems Modeling API and Services
- Clause 8. Platform Specific Models (PSMs) of the Systems Modeling API and Services

  8.1 REST/HTTP PSM
  8.2 OSLC PSM

These clauses are followed by an annex.

- Annex A defines the suite of conformance tests that may be used to demonstrate the conformance of systems modeling environments to this specification - see Clause 2.

6.4 Acknowledgements

The primary authors of this specification document, the PIM, and the REST/HTTP PSM are:

- Manas Bajaj, InterCax LLC
- Ivan Gomes, NASA Jet Propulsion Laboratory

The primary authors of the OSLC PSM are:

- Jim Amsden, IBM
- Jad El-Khoury, KTH Royal Institute of Technology

The specification was formally submitted for standardization by the following organizations:

- 88Solutions Corporation
- GfSE e.V.
- IBM
- INCOSE
- InterCax LLC
- Lockheed Martin Corporation
- Model Driven Solutions, Inc.
- Dassault Système

- PTC
- Simula Research Laboratory AS

However, work on the specification was also supported by over 120 people in over 60 other organizations that participated in the SysML v2 Submission Team (SST). The following individuals had leadership roles in the SST:

- Manas Bajaj, InterCax LLC (API and services development lead)
- Yves Bernard, Airbus (profile development co-lead)
- Bjorn Cole, Lockheed Martin Corporation (metamodel development co-lead)
- Sanford Friedenthal, SAF Consulting (SST co-lead, requirements V&V lead)
- Charles Galey, Lockheed Martin Corporation (metamodel development co-lead)
- Karen Ryan, Siemens (metamodel development co-lead)
- Ed Seidewitz, Model Driven Solutions (SST co-lead, pilot implementation lead)
- Tim Weilkiens, oose (profile development co-lead)

The specification was prepared using CATIA No Magic modeling tools and the OpenMBEE system for model publication (http://www.openmbee.org), with the invaluable support of the following individuals:

- Tyler Anderson, No Magic/Dassault Systèmes
- Christopher Delp, Jet Propulsion Laboratory
- Ivan Gomes, Jet Propulsion Laboratory
- Robert Karban, Jet Propulsion Laboratory
- Christopher Klotz, No Magic/Dassault Systèmes
- John Watson, Lightstreet consulting

7 Platform Independent Model (PIM)

7.1 Project Service

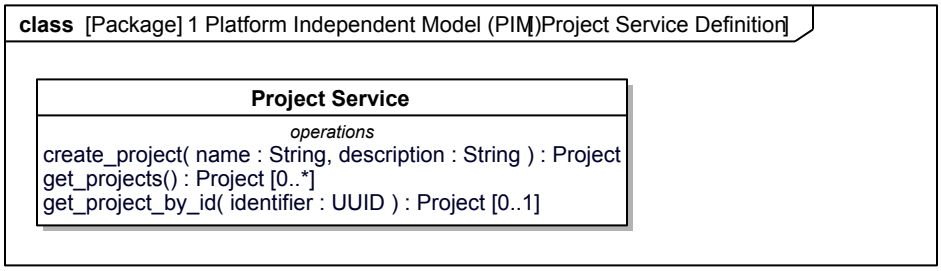

Figure 2. Project Service Definition

Table 6. Operations

Name	Documentation
create_project	Creates a new project with the given name and description. Precondition: Name and description must be provided and valid.
get_project_by_id	Get project with the given identifier. Precondition: Identifier must be specified and valid.
get_projects	Gets all projects.

7.2 Element Navigation Service

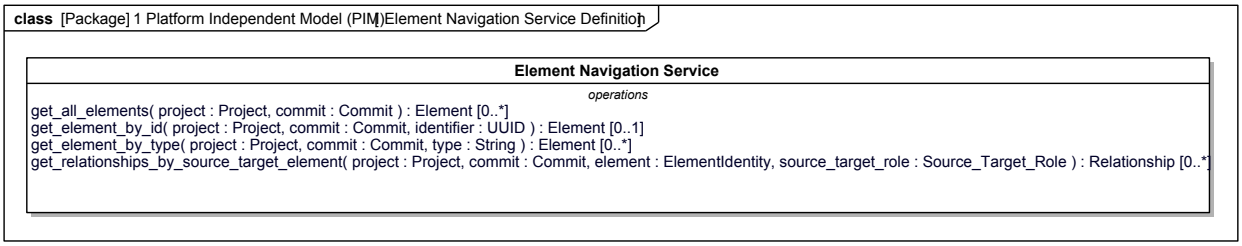

Figure 3. Element Navigation Service Definition

Table 7. Operations

Name	Documentation
get_all_elements	Get all the elements in a given project at a given commit. Precondition: Project and Commit must be specified and valid.
get_element_by_id	Gets the element with the given identifier. Precondition: Identifier (UUID) for the element must be provided.
get_element_by_type	Gets elements with the given type. If project is specified, gets elements with the given type in that project. Precondition: Element type must be provided and valid. Project must be valid.
get_relationships_by_source_target_element	Gets relationships by source element. If project is specified, gets relationships with the given source element in that project. Precondition: Source element must be specified and valid. Project must be valid.

7.3 External Relationship Service

7.3.1 External Relationship Service Definition

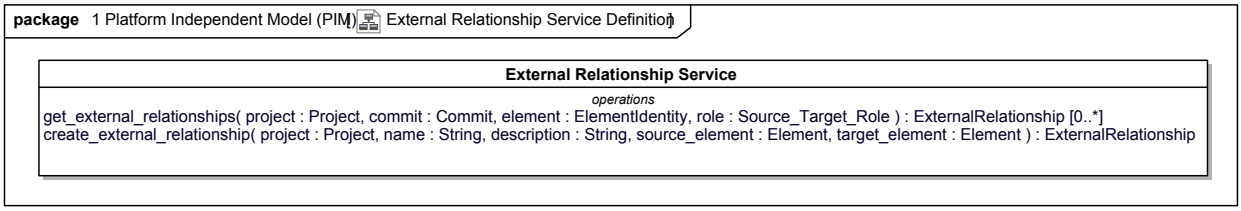

Figure 4. External Relationship Service Definition

Table 8. Operations

Name	Documentation
get_external_relationships	Gets external relationships in the given project and commit where the given element has the role of source, target, or both (see parameter source_target_role) by source element. Precondition: Project, commit, element must be specified and valid. If the role of the given element is not specified, the default value of Source_Target_Role.Both will be used.
create_external_relationship	Creates an external relationship with the given name and description, between the given source and target elements in the given project. Precondition: Project, source element, and target element must be provided and valid. Either the source or the target element, but not both, should be an ExternalElement.

7.3.2 External Relationship API Model

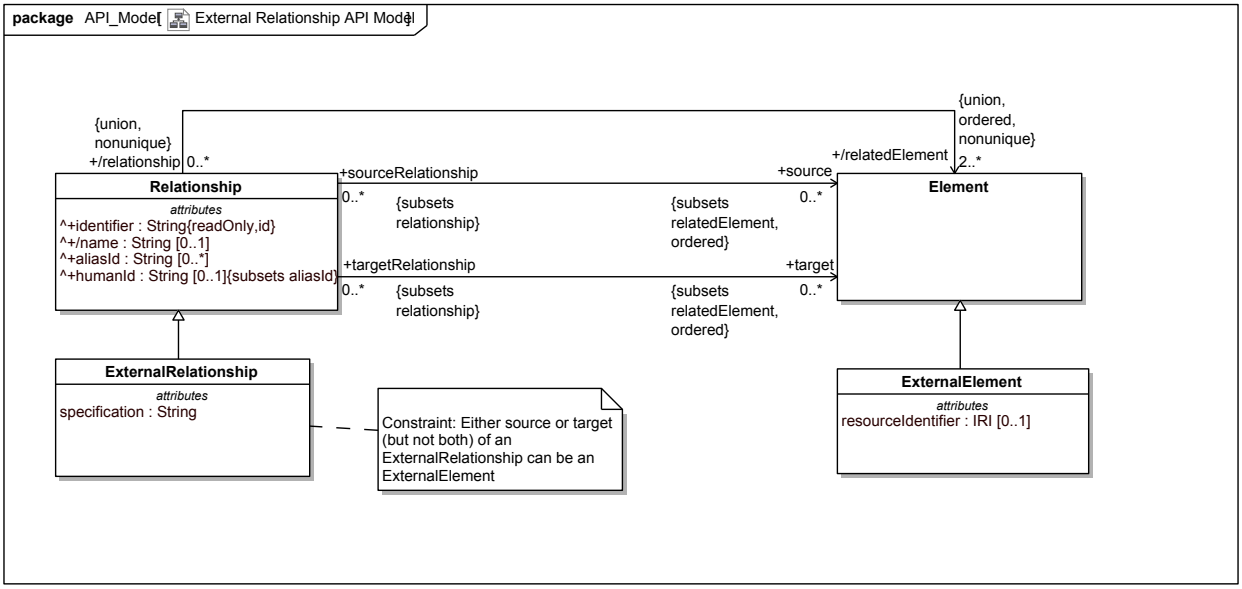

Figure 5. External Relationship API Model

The class diagram above presents concepts related to External Relationship Service. ExternalRelationship and ExternalElement are defined as sub-types of Relationship and Element concepts in the SysML v2 meta-model.

ExternalRelationship - ExternalRelationship is a subclass of Relationship that represents the relationship between a SysML element in a tool or repository to an external element in another tool or repository. The external element may

be a SysML element or a non-SysML element. A hyperlink between a SysML element to a web resource is the most primitive example of an external relationship. The source or target, but not both, of an ExternalRelationship must be an ExternalElement. An ExternalRelationship has the following additional attributes:

- *specification* is the description of the semantics of the external relationship. The specification can be a collection of mathematical expressions. For example, an ExternalRelationship can be defined to map the attributes of a SysML element to the attributes of an ExternalElement. In this case, the specification would contain the equations representing the mapping.

ExternalElement - ExternalElement is a subclass of Element that represents a resource external to a given tool or repository. An ExternalElement is defined only for the purpose of defining an ExternalRelationship. An ExternalElement has the following additional attributes.

- *resourceIdentifier* is the IRI of the resource represented by the ExternalElement

7.4 Element Creation Service

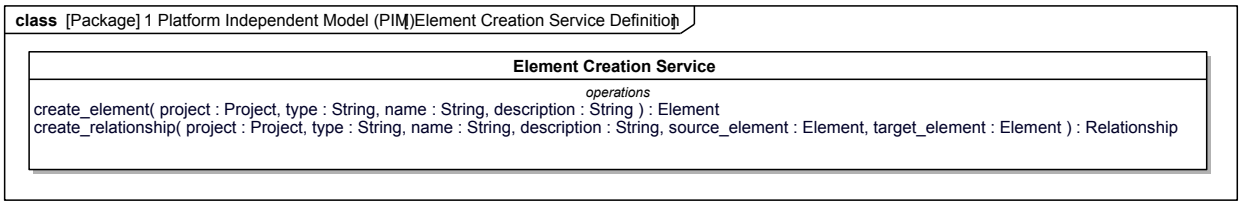

Figure 6. Element Creation Service Definition

Table 9. Operations

Name	Documentation
create_relationship	Creates a relationship element given the parent project, element type, name, and description. Precondition: Project, relationship type, source element, and target element must be provided and valid.
create_element	Creates an element given the parent project, element type, name, and description. Precondition: Project and element type must be provided and valid.

7.5 Project Commit and Branch Service

7.5.1 Project Commit and Branch Service Definition

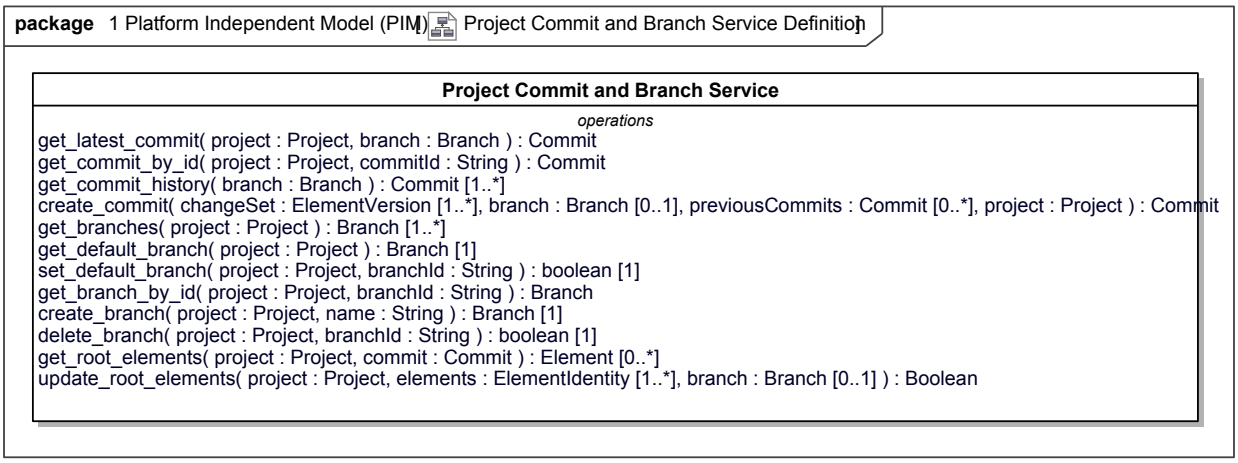

Figure 7. Project Commit and Branch Service Definition

Table 10. Operations

Name	Documentation
get_branch_by_id	Gets the branch with the given ID in the given project. Precondition: Project and branch ID must be specified and valid.
update_root_elements	Updates the root elements of the given project at the latest commit on the given branch and creates a new commit on the branch. Precondition: Project, branch, and elements to be set as root elements must be specified and valid.
get_branches	Gets all the branches in a given project. Precondition: Project must be specified and valid.
create_commit	Creates a new commit with the given change set (collection of all elements that have updated) in the given branch of the project. If the branch is not specified, the default branch of the project is used. Precondition: Project and change set with one or more elements with updates must be specified and valid.
get_root_elements	Gets all the root elements in the given commit for the given project. Precondition: Project and commit must be specified and valid.
get_commit_by_id	Gets the commit with the given ID in the given project. Precondition: Project and commit ID must be specified and valid.
delete_branch	Deletes the branch with the given ID in the given project. Precondition: Project and branch ID must be specified and valid.
get_default_branch	Gets the default branch of the given project. Precondition: Project must be specified and valid.
set_default_branch	Sets the branch with the given ID as the default branch of the given project. Precondition: Project and branch ID must be specified and valid.
get_latest_commit	Gets the latest commit in the given branch of the given project. If the branch is not specified, the default branch of the project is used. Precondition: Project must be specified and valid.
get_commit_history	Gets all the commits on a given branch in the given project in a reverse chronological order, starting with the latest commit. Precondition: Project and branch must be specified and valid.

Name	Documentation
create_branch	Creates a new branch with the given name in the given project. Precondition: Project and branch name must be specified and valid.

7.5.2 Project Commit and Branch API Model

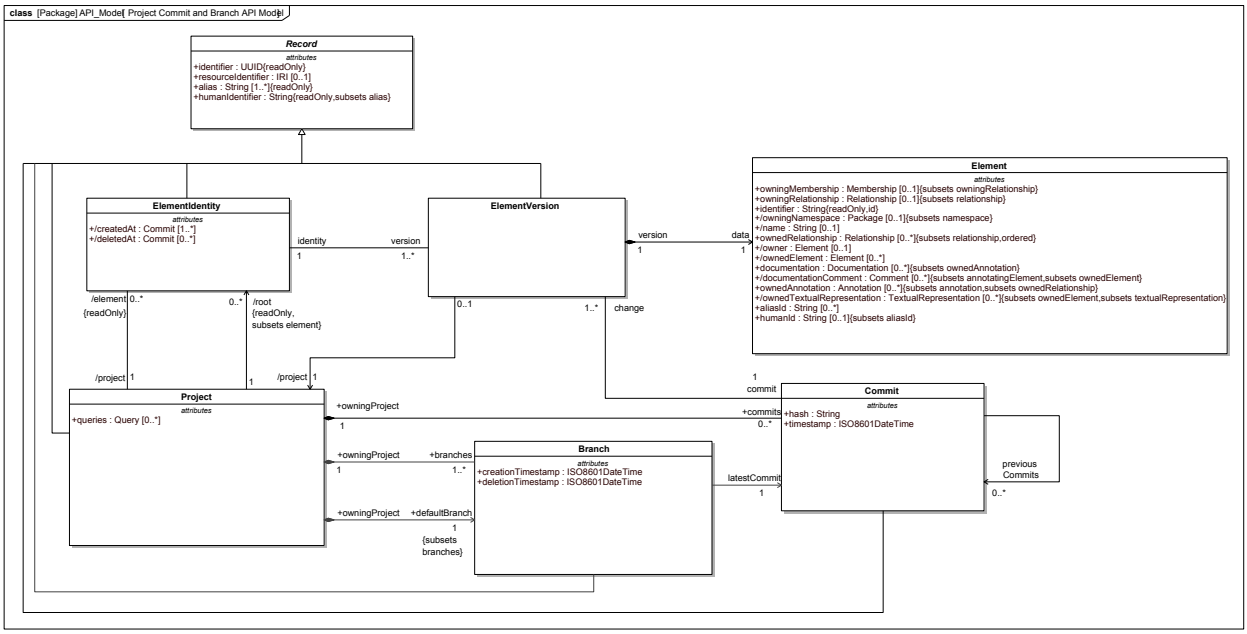

Figure 8. Project Commit and Branch API Model

The class diagram above presents concepts related to Element Versioning Service.

Record - A Record represents any data that is consumed (input) or produced (output) by the Systems Modeling API and Services. A Record is an abstract concept from which other concrete concepts inherit. A Record has the following attributes:

- *identifier* is the UUID assigned to the record
- *resourceIdentifier* is an IRI for the record
- *alias* is a collection of other identifiers for this record, especially if the record was created or represented in other software applications and systems
- *humanIdentifier* is a human-friendly unique identifier for this record

ElementIdentity - ElementIdentity is a subclass of Record that represents a unique, version-independent representation of an element through its lifecycle. An ElementIdentity is associated with 1 or more ElementVersion records that represent different versions of the same element. An ElementIdentity record has the following additional attributes:

- *createdAt* is a derived attribute that references the Commit in a project in which the given element was created
- *deletedAt* is a derived attribute that references the Commit in a project in which the given element was deleted

ElementVersion - ElementVersion is a subclass of Record that represents an element at a specific version in its lifecycle. An ElementVersion is associated with only 1 ElementIdentity. An ElementVersion relates the versioning concepts to the abstract syntax for the SysML v2 language. This is represented by the *data* property typed by **Element** which is the root metaclass in the SysML v2 language metamodel. All versions of an Element, i.e. model

elements with the same identifier, would each be associated with a unique ElementVersion and the set of those ElementVersions would be associated (*identity* property) to a single ElementIdentity whose *identifier* property matches the *identifier* property of the Element.

Project - Project is a subclass of Record that represents a container for other Records and an entry point for version and model navigation. An element in a project can reference and use elements in other projects. A project can have 0 or more root elements. A root element is defined as an element without an *owner*, i.e. is not owned by any other model element. Root elements serve as starting points of navigating the elements and relationships in a model.

A project also represents a permission target at which access and authorization control may be applied to teams associated with a project.

Commit - Commit is a subclass of Record that represents the changes made to a project at a specific point in time in its lifecycle, such as creation, update, or deletion of elements in a project. A project has 1 or more commits. A commit has the following additional attributes:

- *hash* is the unique hash generated from the changes of the commit
- *timestamp* is the timestamp of the commit

A Commit has an associated set of changes, represented by the *change* property, that includes 1 or more ElementVersions, each of which includes the element data that is being created, updated, or deleted in that commit.

A Commit is associated with 0 or more previous commits. In a project with a single branch with no merges, every commit except for the initial commit would have only 1 previous commit. However, the 0 or more cardinality on the *previous* property is defined to support the merging of branches.

Branch - Branch is a subclass of Record that represents an independent line of development in a project. A project can have 1 or more branches. When a project is created, a default branch is also created. The default branch of a project can be changed, and a project can have only 1 default branch.

Branches build upon the concept of Commits. A Branch is a pointer to a commit (*Branch.latestCommit*), also known as the head of the Branch. The commit history of project on a given branch can be computed by recursively navigating *Commit.previousCommits*, starting from the latest commit of the given branch (*Branch.latestCommit*). Project commits are immutable but branches are mutable. Branches can be created, deleted, and merged with other branches. A Branch has the following additional attributes:

- *creationTimestamp* is the timestamp at which the branch was created
- *deletionTimestamp* is the timestamp at which the branch was deleted
- *latestCommit* is the commit to which the branch is currently pointing. It represents the latest state of the project on the given branch.
- *owningProject* is the project that owns the given branch

7.6 Query Service

7.6.1 Query Service Definition

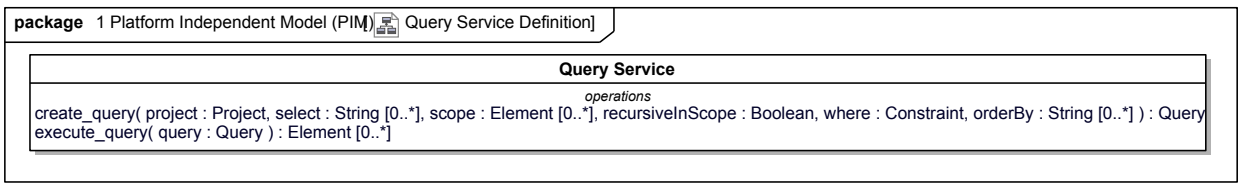

Figure 9. Query Service Definition

Table 11. Operations

Name	Documentation
execute_query	Executes the given query and returns a collection of Elements. Precondition: Query must be provided and valid.
create_query	Creates a query in a project, given the query inputs. Precondition: Project and where constraint must be specified and valid.

7.6.2 Query API Model

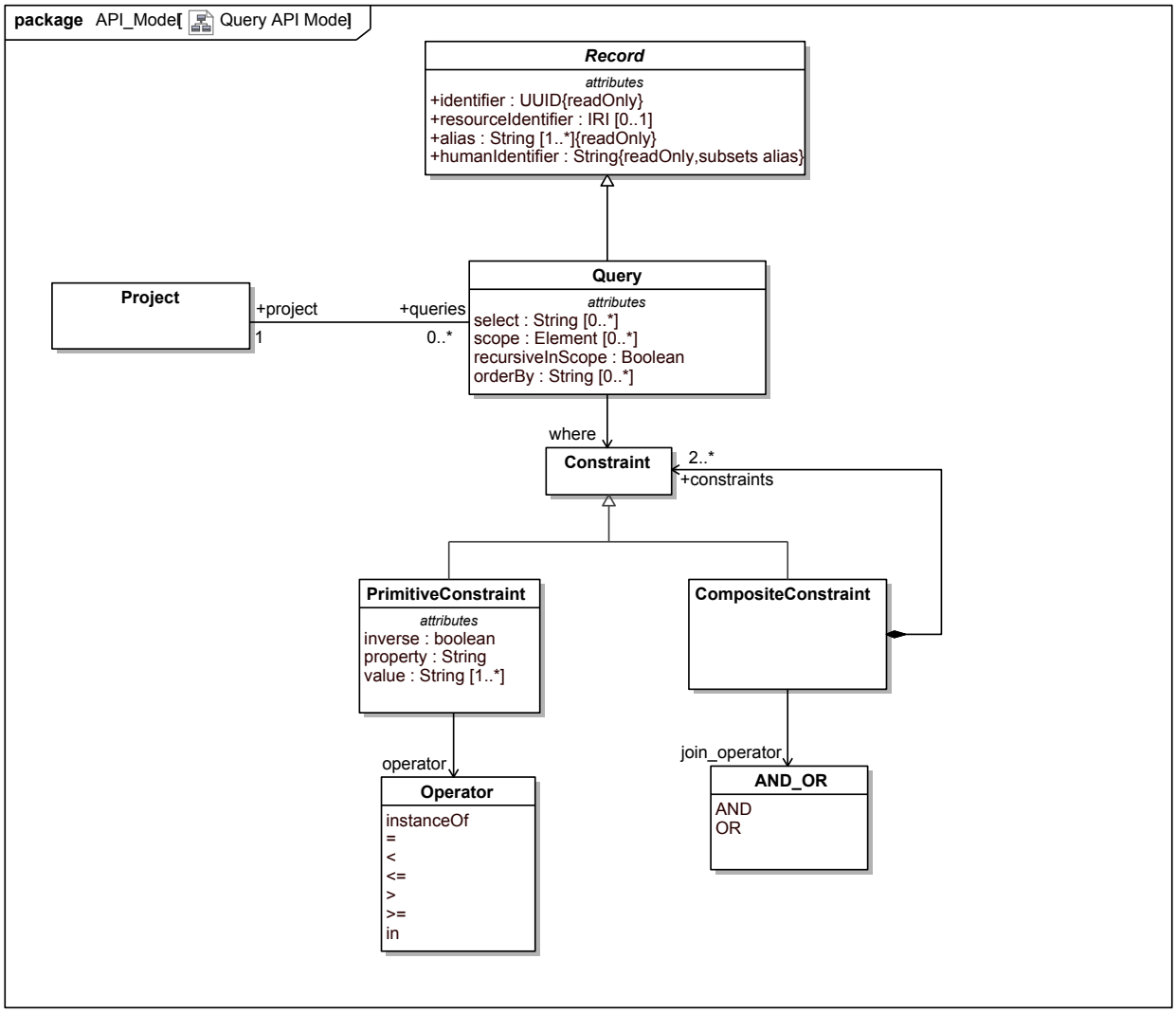

Figure 10. Query API Model

The class diagram above presents concepts related to the Query service.

Query - Query is a subclass of Record that represents a precise and language-independent request for information retrieval using the Systems Modeling API and Services. Query can be mapped to commonly used query languages, such as SQL, Gremlin, GraphQL, and SPARQL.

A Query record has the following additional attributes:

- *select* is a list of properties of Element or its subtypes that will be included for each Element object in the query response. Element is the root-metaclass in KerML. If no properties are specified, then all the properties will be included for each Element in the query response.
- *scope* is a list of Element objects that define the scope for query execution. The default scope of a Query is the owning project.
- *recursiveInScope* is of type Boolean and is relevant if one or more Element objects are specified in the *scope*. If true, then the query will be executed across all Element objects that are directly or indirectly (recursively) owned by the Element objects in scope. If false, then the query will be executed across all Element objects that are directly owned by the Element objects in scope.
- *where* is of type Constraint and represents the conditions that Elements in the query response must satisfy.
- *orderBy* is a list of properties of Element or its subtypes that are used for sorting the Element objects in the query response. The order of properties in the list governs the sorting order.

Constraint - Constraint represents conditions that must be satisfied by Element objects in the query response.

PrimitiveConstraint is a subtype of Constraint that represents simple conditions that be modeled using the *property-operator-value* tuple, e.g. *mass <= 4 kg*, or *type instanceOf Generalization.* A PrimitiveConstraint has the following additional attributes:

- *property* is a property of Element or its subtypes that is being constrained
- *operator* is of type Enumeration whose literals are mathematical operators, as listed above
- *value* is a list of primitive objects, such as String, Boolean, Integer, Double, and UUID
- *inverse* is of type Boolean. If true, a logical NOT operator is applied to the PrimitiveConstraint.

CompositeConstraint is a subtype of Constraint that represents complex conditions composed of two or more CompositeConstraints or PrimitiveConstraints using logical AND or OR operators.

8 Platform Specific Models (PSMs)

8.1 REST/HTTP PSM

The REST/HTTP Platform-Specific Model (PSM) for the Systems Modeling API and Services is described using OpenAPI Specification (OAS) 2.0 and is included with this specification.

The table below shows the mapping between Systems Modeling API and Services PIM to the REST / HTTP PSM.

Table 12. PIM to REST / HTTP PSM Mapping

PIM	REST / HTTP PSM
Project Service	
create_project	POST /projects
get_projects	GET /projects
get_project_by_id	GET /projects/{projectId}
Element Navigation Service	
get_all_elements	GET/projects/{projectId}/commits/{commitId}/elements
get_element_by_id	GET /projects/{projectId}/commits/{commitId}/elements/{elementId}
get_elements_by_type	*Not available yet*
get_relationships_by_source_target_element	GET /projects/{projectId}/commits/{commitId}/elements/{relatedElementId}/relationships • *direction* query parameter with allowable values {in, out, both}
Element Creation Service	
create_element	POST /projects/{projectId}/commits
create_relationship	POST /projects/{projectId}/commits
Project Commit and Branch Service	
get_latest_commit	• GET /projects/{projectId}/branches/{branchId} returns the branch with the given ID. • Branch.head is the latest commit on the branch
	GET /projects/{projectId}/commits/{commitId}
get_commit_history	Not available yet
create_commit	POST /projects/{projectId}/commits
get_branches	GET /projects/{projectId}/branches

PIM	REST / HTTP PSM
get_default_branch	• GET /projects/{projectId} returns a Project • Project.defaultBranch gives the default branch
get_branch_by_id	GET /projects/{projectId}/branches/{branchId}
set_default_branch	Not available yet
create_branch	POST /projects/{projectId}/branches
delete_branch	Not available yet
get_root_elements	GET /projects/{projectId}/commits/{commitId}/roots
update_root_elements	• POST /projects/{projectId}/commits • Provide the updated set of root elements in the change set of the Commit sent in the body of the POST request

Pagination

The REST/HTTP PSM uses a Cursor-based pagination strategy for the responses received from the GET requests. The following 3 query parameters can be specified in any GET request that returns a collection of records.

1. *page[size]* specifies the maximum number of records that will be returned per page in the response
2. *page[before]* specifies the URL of the page succeeding the page being requested
3. *and page[after]* specifies the URL of a page preceding the page being requested

If neither *page[before]* nor *page[after]* is specified, the first page is returned with the same number of records as specified in the *page[size]* query parameter. If the *page[size]* parameter is not specified, then a default page size is used, which can be set by the API provider.

The *Link* header in the response includes links (URLs) to the previous page and the next page, if any, for the given page in the response. The specification of these links is conformant to the IETF Web Linking standard. As an example, the value of the *Link* response header is shown below. The *rel* value associated with each page link specifies the type of relationship the linked page has with the page returned in the response. Page link specified with *rel* value as *next* is the link for the next (or succeeding) page to the page returned in the response, and the page link specified with *rel* value as *prev* is the link for the previous (or preceding) page to the page returned in the response.

```
<http://sysml2-api-host:9000/projects?
    page[after]=MTYxODg2MTQ5NjYzMnwyMDEwOWY0MC00ODI1LTQxNmEtODZmNi03NTA4YWM0MmEwMjE&
    page[size]=3>; rel="next",
<http://sysml2-api-host:9000/projects?
    page[before]=MTYxODg2MTQ5NjYzMnwxMDg2MDFjMS1iNzk1LTRkMGEtYTFiYy1lZjEyYmMwNTU5ZjI&
    page[size]=3>; rel="prev"
```

Example

An example demonstrating the Cursor-based paginated responses received from GET requests to the */projects* endpoint is presented here. The term "User" in the example scenario presented below refers to an API user that could be a human user or a software program.

Step 1 - User makes a GET request to the */projects* endpoint with *page[size]* query parameter set to 3. If successful, this request will return the first page with a maximum of 3 project records. The URL for this GET request is shown below.

```
http://sysml2-api-host:9000/projects?page[size]=3
```

Step 2 - If there are more than 3 projects in the provider repository, the *Link* header in the response will provide the URL for the next page with *rel* value equal to *next.* The User gathers the link to the next page.

```
<http://sysml2-api-host:9000/projects?
    page[after]=MTYxODg2MjE2NTMxNXwwOGY0MzNkYi1iNmQ0LTQxYjgtOTAyMC1lODIwZWJjNDE3YmU&
    page[size]=3>; rel="next"
```

Step 3 - User makes a GET request to the URL for the next page gathered from Step 2. The *Link* header in the response will provide the URL for the next page with *rel* value equal to *next*. Additionally, the *Link* header will include the URL for the previous page with *rel* value equal to *prev*.

```
<http://sysml2-dev.intercax.com:9000/projects?
    page[after]=MTYxODg2MjY4OTYxNHwyMDEwOWY0MC00ODI1LTQxNmEtODZmNi03NTA4YWM0MmEwMjE&
    page[size]=3>; rel="next",
<http://sysml2-dev.intercax.com:9000/projects?
    page[before]=MTYxODg2MjY4OTYxNHwxMDg2MDFjMS1iNzk1LTRkMGEtYTFiYy1lZjEyYmMwNTU5ZjI&
    page[size]=3>; rel="prev"
```

Step 4 - User continues Step 3 until the *Link* header in the response does not include the URL for the next page (*rel* value as *next*).

8.2 OSLC PSM

The OSLC Platform-Specific Model (PSM) for the Systems Modeling API and Services is described using OpenAPI Specification (OAS) 2.0 and is included with this specification. The table below shows the mapping between Systems Modeling API and Services PIM to the OSLC PSM.

An OSLC implementation may typically need to realize a broader set of services than those defined by the PIM for a full integration with other OSLC-compliant systems. Services such as Delegated UI for Selection and Creation, resource UI Preview, authentication, and support for arbitrary queries (beyond those defined in the PIM).

The table shows services relating to the Element class, which will implicitly include services to any of the Element subclasses. An OSLC implementation may also provide specific services relating to a specific subset of the Element subclasses.

Note that the URLs listed in the tables below are provided as examples only. With OSLC, all URLs are implementation-specific. A OSLC client typically relies on the OSLC discovery mechanism (https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/discovery.html), to determine what services are provided by an OSLC server, as well as the necessary information (such as a service URL) to be able to consume any such service. At the least, an OSLC client requires a discovery URL to bootstrap this discovery for any particular server. The various approaches for bootstrapping and discovery are further detailed in the OSLC standard.

Typically, services (such as element creation) are performed in the context of a specific project within the application. In OSLC, this is reflected by typically mapping each project to a Service Provider or LDP Container in the OSLC discovery chain. That is, for a client to request any service, it first needs to identify the appropriate service for the desired project, through the discovery mechanism.

Specifically for the OSLC capabilities of relevance for the PSM:

- **Resource Creation**: A client requests a resource creation by sending a HTTP POST request to the URI defined in the oslc:creation property of a Create Factory resource, as defined under https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/discovery.html#creationFactoryShape. The resource to be created is provided in the request body, as an RDF resource. https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/discovery.html#creation details how to discover such a service.
- **Query Capability**: A client requests a query by sending a HTTP GET or POST to the URL defined in the oslc:queryBase property of a Query Capability resource, as defined under https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/discovery.html#queryCapabilityShape. Search parameters (such as oslc.where and oslc.searchTerms) are sent as HTTP query parameters, or in the `application/x-www-form-urlencoded` body.

- **Read, Update, and Delete (CRUD) Operations:** on a specific OSLC resource are performed using the URL of the resource itself. A client obtains the resource URL through any OSLC capability such as the response from a creation request, a end-user selection through a DelegatedUI, a filtering through a query request, etc. See https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/oslc-core.html#resourceShapes - subsection "Resource Operations" - for details.

Table 13. PIM to OSLC PSM Mapping

PIM	OSLC PSM
Project Service	Each Project is mapped to correspond to an OSLC Service Provider. All Projects (Service Providers) are listed under the Server's Service Provider Catalog.
create_project	POST request on the Service Provider Catalog URL (alternatively, a server may provide a dedicated Service Provider - with corresponding OSLC capabilities - to manipulate Project resources through typical OSLC capabilities, such as a Creation Factory.) For example, POST /services/catalog/singleton
get_projects	GET request on the Service Provider Catalog URL For example, GET /services/catalog/singleton
get_project_by_id	GET request on the Project's corresponding Service Provider URL GET /projects/{projectId}
get_root_elements	GET/POST request on the OSLC Query capability. The Query capability is identified under the Project's corresponding Service Provider. *<TODO: Provide the expected request parameters (a) oslc.where and (b) oslc.prefix>* For example, GET projects/{projectId}/service/elements/query
update_root_elements	OSLC does not support the atomic updating of multiple elements. OSLC has no specific support for updating root elements, which differs from the standard mechanism to update any other element.
Element Navigation Service	
get_all_elements	GET/POST request on the OSLC Query capability. The Query capability is identified under the Project's corresponding Service Provider. No request parameters are expected to obtain all elements of a project. For example, GET projects/{projectId}/service/elements/query

PIM	OSLC PSM
get_element_by_id	GET request on the resource URL. For example, GET projects/{projectId}/elements/{elementId}
get_elements_by_type	GET/POST request on the OSLC Query capability. The Query capability is identified under the Project's corresponding Service Provider. The following request parameters are expected: • oslc.prefix: rdf=<http://www.w3.org/1999/02/22-rdf-syntax-ns#> • oslc.where: rdf:type=<http://omg.org/ns/sysml#Subsetting> (The request paramters **MUST** be properly encoded when transmitted in HTTP requests.) For example, GET projects/{projectId}/service/elements/query
get_relationships_by_source_target_element	GET/POST request on the OSLC Query capability. The Query capability is identified under the Project's corresponding Service Provider. The following request parameters are expected: <TODO: Provide the expected request parameters (a) oslc.where and (b) oslc.prefix> For example, GET projects/{projectId}/service/elements/query
External Relationship Service	
get_external_relationships	GET/POST request on the OSLC Query capability. The Query capability is identified under the Project's corresponding Service Provider. The following request parameters are expected: <TODO: Provide the expected request parameters (a) oslc.where and (b) oslc.prefix. OSLC does/can not differentiate between properties/relationships that are external or otherwise, but we formulate the query to filter on that specific attribute> For example, GET projects/{projectId}/service/elements/query
create_external_relationship	PUT request on the resource URL. (See https://docs.oasis-open-projects.org/oslc-op/core/v3.0/ps01/oslc-core.html#resourceShapes - subsection "Resource Operations" - for details.) or example, PUT projects/{projectId}/elements/{elementId}
Element Creation Service	
create_element	POST request on the OSLC Creation Factory. The Creation Factory is identified under the Project's corresponding Service Provider. For example, POST projects/{projectId}/service/elements/create
create_relationship	Same as create_element Implementations can choose to define dedicated creation factories for any specific Element subclasses.
Element Versioning Service	Versioning is to be according to the OSLC Configuration Management 1.0 specification

PIM	OSLC PSM
get_commits	
get_commit_by_id	
get_latest_commit	
create_commit	
get_previous_commits	
get_versions	
Query Service	Not available. OSLC does not support the ability to create and store queries. Systems may choose to use the REST/HTTP PSM to realize such services. However, the query being created and/or executed is defined using the OSLC Query Capability.
execute_query	
create_query	

A Annex: Conformance Test Suite

Project Service Conformance Tests

The table below is a list of conformance tests for the Project Service.

Test ID	PIM Service operation	Input	Expected Output
PS-CT1	Project Service.*create_project*	• Project name • Project description	Project
PS-CT2	Project Service.*get_projects*		Set of Projects
PS-CT3	Project Service.*get_project_by_id*	• identifier	Project if a project with the given identifier exists
PS-CT4	Project Service.*get_root_elements*	• Project	Set of elements
PS-CT5	Project Service.*update_root_elements*	• Project • Set of elements	True if the given elements were set as root elements of the project, false otherwise

Element Navigation Service Conformance Tests

The table below is a list of conformance tests for the Element Navigation Service.

Test ID	PIM Service operation	Input	Expected Output
NS-CT1	Element Navigation Service.*get_all_elements*	• Project • Commit	Set of elements
NS-CT2	Element Navigation Service.*get_element_by_id*	• Project • Commit • identifier	Element if an element with the given identifier exists

NS-CT3	Element Navigation Service.*get_element_by_type*	• Project • Commit • Type	Set of elements
NS-CT4	Element Navigation Service. *get_relationships_by_source_target_element*	• Project • Commit • ElementIdentity • Source_Target_Role (source, target, both)	Set of relationships

Element Creation Service Conformance Tests

The table below is a list of conformance tests for the Element Creation Service.

Test ID	PIM Service operation	Input	Expected Output
CS-CT1	Element Creation Service.*create_element*	• Project • Name • Description	Element with the given name and description in the given project
CS-CT2	Element Creation Service.*create_relationship*	• Project • Name • Description • Element (as source) • Element (as target)	Relationship with the given name and description in the given project. The source and target elements of the relationship should be the same as specified.

Element Versioning Service Conformance Tests

The table below is a list of conformance tests for the Element Versioning Service.

Test ID	PIM Service operation	Input	Expected Output
VS-CT1	Element Versioning Service.*get_commits*	• Project	Set of commits
VS-CT2	Element Versioning Service.*get_commit_by_id*	• Project • Commit id	Commit if a commit with the given id exists for the given project

VS-CT3	Element Versioning Service.*get_latest_commit*	• Project	Commit (latest commit in the project)
VS-CT4	Element Versioning Service.*create_commit*	• Change-set	Commit
VS-CT5	Element Versioning Service.*get_versions*	• ElementIdentity	Set of ElementVersion

External Relationship Service Conformance Tests

The table below is a list of conformance tests for the External Relationship Service.

Test ID	PIM Service operation	Input	Expected Output
ES-CT1	External Relationship Service. *get_external_relationships*	• Project • Commit • ElementIdentity • Source_Target_Role	Set of ExternalRelationship records in the given project and commit that have the given ElementIdentity in the role specified by Source_Target_Role (source/target/both).
CS-CT2	Element Creation Service. *create_external_relationship*	• Project • Name • Description • Element (as source) • Element (as target)	External Relationship with the given name and description in the given project. The source and target elements of the relationship should be the same as specified.

Query Service Conformance Tests

The table below is a list of conformance tests for the Query Service.

Test ID	PIM Service operation	Input	Expected Output
QS-CT1	Query Service.*create_query*	• Project • Element as scope • recursiveInScope (true) • where constraint ◦ CompositeConstraint with two or more PrimitiveConstraint	Query object
QS-CT2	Element Creation Service.*execute_query*	• Query	Collection of Element objects in the query response